



FOCUS

FinOps Open Cost and Usage Specification

Version

Publication version 1.3

Copyright © 2023-2025 - FinOps Open Cost and Usage Specification (FOCUS) a Series of the Joint Development Foundation Projects, LLC. Joint Development Foundation [trademark](#), and document use rules apply.

Status of This Document

This section describes the status of this document at the time of its publication.

This is a published release of the FinOps Open Cost and Usage Specification.

This document was produced by a group operating under the Joint Development Foundation Projects agreement. FOCUS maintains a public list of any patent disclosures made in connection with the deliverables of the group; [that page](#) also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) must disclose the information.

Document Use License

Copyright (c) Joint Development Foundation Projects, LLC, FinOps Open Cost and Usage Specification (FOCUS) Series and its contributors. The materials in this repository are made available under the Creative Commons Attribution 4.0 International license (CC-BY-4.0), available at <https://creativecommons.org/licenses/by/4.0/legalcode>.

Shield: License CC BY 4.0

This work is made available under: [Creative Commons Attribution 4.0 International License](https://creativecommons.org/licenses/by/4.0/legalcode).



THESE MATERIALS ARE PROVIDED "AS IS." The parties expressly disclaim any warranties (express, implied, or otherwise), including implied warranties of merchantability, non-infringement, fitness for a particular purpose, or title, related to the materials. The entire risk as to implementing or otherwise using the materials is assumed by the implementer and user. IN NO EVENT WILL THE PARTIES BE LIABLE TO ANY OTHER PARTY FOR LOST PROFITS OR ANY FORM OF INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER FROM ANY CAUSES OF ACTION OF ANY KIND WITH RESPECT TO THIS DELIVERABLE OR ITS GOVERNING AGREEMENT, WHETHER BASED ON BREACH OF CONTRACT, TORT (INCLUDING NEGLIGENCE), OR OTHERWISE, AND WHETHER OR NOT THE OTHER MEMBER HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This document is governed by the Patent Policy Option 4: W3C Mode: See [project charter](#).

Abstract

FOCUS is an open specification for billing data. It defines a common schema for billing data, aligns terminology with the FinOps Framework and defines a minimum set of requirements for billing data. The specification provides clear guidelines for billing data generators to produce FinOps-serviceable data. The specification enables FinOps practitioners to perform common FinOps capabilities such as chargeback, cost allocation, budgeting and forecasting etc. using a generic set of instructions, regardless of the origin of the FOCUS compatible dataset.

Working Group

Maintainers

Thanks to the following FOCUS Maintainers for their leadership and contributions to the FOCUS Release **v1.3** specification.

- Christopher Harris (Datadog)
- Irena Jurica (Neos)
- Joaquin Prado (FinOps Foundation)
- Karl Kraft (Walmart)
- Larry Advey (Twilio / CloudZero)
- Michael Flanakin (Microsoft / Salesforce)
- Mike Fuller (FinOps Foundation)
- Riley Jenkins (Domo)
- Shawn Alpay (FinOps Foundation)
- Udam Dewaraja (StitcherAI)

Contributors

Thanks to the following FOCUS members for their contributions to the FOCUS Release **v1.3** specification.

- Alexandra McCoy (A.M. Tech)
- Andrew Qu (Everest)
- Andrew Quigley (Northwestern Mutual)
- Beau Nelford (Anglepoint)
- David Dinh (Google)
- David Earney (American Express)
- Deeja Cruz (Datadog)
- Erik Norman (Caligo)
- George Parker (Salesforce)
- Greg Kroleski (Databricks)
- James Deloid (Oracle)
- Jason Wu (Amazon Web Services)
- Justin Marks (Amazon Web Services)
- Marc Perreaut (Amadeus)
- Matt Cowsert (FinOps Foundation)
- Nan Braun (Thavron Solutions)
- Ramkumar Narla (Adobe)
- Rich Kreitz (Grafana)
- Rob Martin (FinOps Foundation)
- Robert Reeves (Mavvrik)
- Satya Vandrangi (Amazon Web Services)
- Sanjna Srivatsa (Broadcom)
- Tim Wright (Google)

Steering Committee Members

Thanks to the following FOCUS Steering Committee members for their leadership on the FOCUS specification.

- Ben Olson (Adobe)
- Christopher Harris (Datadog)
- Jerzy Grzywinski (Capital One)
- Letian Feng (Amazon Web Services)
- Michael Flanakin (Microsoft) (term ended 2025/08/01)
- Mike Fuller (FinOps Foundation)
- Richard Steck (Adobe) (term ended 2025/10/03)

- Sarah McMullin (Google)
- Tim O'Brien (Walmart)
- Vikram Desai (Microsoft)

Table of Contents

1. Introduction

- 1.1. Background and History
- 1.2. Intended Audience
- 1.3. Scope
- 1.4. Design Principles
- 1.5. Design Notes
- 1.6. Typographic Conventions
- 1.7. FOCUS Feature Level
- 1.8. Conformance Checkers and Validators

2. Supported Features

- 2.1. Account Structures
- 2.2. Billed Cost and Invoice Alignment
- 2.3. Charge Categorization
- 2.4. Commit Usage and Under Usage
- 2.5. Contract Commitments
- 2.6. Cost and Usage Attribution
- 2.7. Cost Comparison
- 2.8. Custom Columns
- 2.9. Data Generator-Calculated Split Cost Allocation
- 2.10. Data Granularity
- 2.11. Dataset Instance Metadata
- 2.12. Effective Cost Analysis
- 2.13. Location
- 2.14. Marketplace Purchases
- 2.15. Recency Metadata
- 2.16. Resource Usage
- 2.17. Schema Metadata
- 2.18. Service Provider Services
- 2.19. Service Categorization
- 2.20. Participating Entity Identification
- 2.21. Verification, Comparison, and Fluctuation Tracking of Unit Prices

3. Datasets

- 3.1. Cost and Usage
 - 3.1.1. Allocated Method ID
 - 3.1.2. Allocated Method Details
 - 3.1.3. Allocated Resource ID
 - 3.1.4. Allocated Resource Name
 - 3.1.5. Allocated Tags
 - 3.1.6. Availability Zone
 - 3.1.7. Billed Cost
 - 3.1.8. Billing Account ID
 - 3.1.9. Billing Account Name
 - 3.1.10. Billing Account Type
 - 3.1.11. Billing Currency
 - 3.1.12. Billing Period End
 - 3.1.13. Billing Period Start
 - 3.1.14. Capacity Reservation ID
 - 3.1.15. Capacity Reservation Status
 - 3.1.16. Charge Category
 - 3.1.17. Charge Class
 - 3.1.18. Charge Description
 - 3.1.19. Charge Frequency
 - 3.1.20. Charge Period End
 - 3.1.21. Charge Period Start
 - 3.1.22. Commitment Discount Category

- 3.1.23. Commitment Discount ID
- 3.1.24. Commitment Discount Name
- 3.1.25. Commitment Discount Quantity
- 3.1.26. Commitment Discount Status
- 3.1.27. Commitment Discount Type
- 3.1.28. Commitment Discount Unit
- 3.1.29. Consumed Quantity
- 3.1.30. Consumed Unit
- 3.1.31. Contract Applied
- 3.1.32. Contracted Cost
- 3.1.33. Contracted Unit Price
- 3.1.34. Effective Cost
- 3.1.35. Host Provider Name
- 3.1.36. Invoice ID
- 3.1.37. Invoice Issuer Name
- 3.1.38. List Cost
- 3.1.39. List Unit Price
- 3.1.40. Pricing Category
- 3.1.41. Pricing Currency
- 3.1.42. Pricing Currency Contracted Unit Price
- 3.1.43. Pricing Currency Effective Cost
- 3.1.44. Pricing Currency List Unit Price
- 3.1.45. Pricing Quantity
- 3.1.46. Pricing Unit
- 3.1.47. Provider - DEPRECATED
- 3.1.48. Publisher - DEPRECATED
- 3.1.49. Region ID
- 3.1.50. Region Name
- 3.1.51. Resource ID
- 3.1.52. Resource Name
- 3.1.53. Resource Type
- 3.1.54. Service Provider Name
- 3.1.55. Service Category
- 3.1.56. Service Name
- 3.1.57. Service Subcategory
- 3.1.58. SKU ID
- 3.1.59. SKU Meter
- 3.1.60. SKU Price Details
- 3.1.61. SKU Price ID
- 3.1.62. Sub Account ID
- 3.1.63. Sub Account Name
- 3.1.64. Sub Account Type
- 3.1.65. Tags
- 3.2. Contract Commitment
 - 3.2.1. Billing Currency
 - 3.2.2. Contract Commitment Cost
 - 3.2.3. Contract Commitment ID
 - 3.2.4. Contract Commitment Category
 - 3.2.5. Contract Commitment Description
 - 3.2.6. Contract Commitment Period End
 - 3.2.7. Contract Commitment Period Start
 - 3.2.8. Contract Commitment Quantity
 - 3.2.9. Contract Commitment Type
 - 3.2.10. Contract Commitment Unit
 - 3.2.11. Contract ID
 - 3.2.12. Contract Period End
 - 3.2.13. Contract Period Start

4. Attributes

- 4.1. Column Handling
- 4.2. Currency Format
- 4.3. Data Generator-Calculated Split Cost Allocation Handling
- 4.4. Date/Time Format
- 4.5. Discount Handling
- 4.6. JSON Object Format
- 4.7. Invoice Handling
- 4.8. Key-Value Format
- 4.9. Null Handling
- 4.10. Numeric Format
- 4.11. String Handling
- 4.12. Unit Format

5. Metadata

- 5.1. Data Generator
- 5.2. Dataset Instance
- 5.3. Recency
- 5.4. Schema

6. Use Case Library

7. Glossary

8. Appendix

- 8.1. Commitment Discounts
- 8.2. Examples: Commitment Discount Flexibility
- 8.3. Examples: Metadata
- 8.4. Examples: Participating Entity Identification
- 8.5. Examples: SaaS
- 8.6. Grouping Constructs for Resources or Services

1. Introduction

This section is non-normative.

FOCUS is a standards development organization (SDO) formed to establish an open, consensus-driven standard for billing data. In the absence of a broadly adopted standard, infrastructure and service [*service providers*](#glossary:service provider) have relied on proprietary billing schemas and inconsistent terminology, making cost data difficult to normalize and act upon across environments. This lack of conformance has forced FinOps *practitioners* to develop best-effort custom normalization schemes for each provider, in order to perform essential FinOps capabilities such as chargeback, cost allocation, budgeting and forecasting.

The FOCUS Specification, developed by a global community of practitioners and vendors, defines a consistent, vendor-neutral approach to billing data. It is designed to improve interoperability between service providers, reduce operational complexity, and enable greater transparency in cloud and SaaS cost management.

1.1. Background and History

This project is supported by the [FinOps Foundation](#). This work initially started under the Open Billing working group under the FinOps Foundation. The decision was made in Jan 2023 to begin to migrate the work to a newly formed project under the Linux Foundation called the FinOps Open Cost and Usage Specification (FOCUS) to better support the creation of a specification.

1.2. Intended Audience

This specification is designed to be used by three major groups:

- Billing data generators: Entities that present consumption-based billing information related to infrastructure and *service providers*, such as (but not limited to):
 - [Cloud Service Providers \(CSPs\)](#)
 - Software as a Service (SaaS) platforms
 - [Managed Service Providers \(MSPs\)](#)
 - Internal infrastructure and service platforms
- FinOps tool *providers*: Organizations that provide tools to assist with FinOps
- FinOps practitioners: Organizations and individuals consuming billing data for doing FinOps

1.3. Scope

The FOCUS working group will develop an open-source specification for billing data. The schema will define data [dimensions](#), [metrics](#), a set of attributes about billing data, and a common lexicon for describing billing data.

1.4. Design Principles

The following principles were considered while building the specification.

1.4.1. FOCUS is an iterative, living specification

- Incremental iterations of the specification released regularly will provide higher value to practitioners and allow feedback as the specification develops. The goal is not to get to a complete, finished specification in one pass.

1.4.2. Working backward with ease of adoption

- Aim to work backward from essential FinOps capabilities that practitioners need to perform to prioritize the dimensions, metrics and attributes of the cost and usage data that should be defined in the specification to fulfill that capability.
- Be FinOps scenario-driven. Define columns that answer scenario questions; don't look for scenarios to fit a column, each column must have a use case.
- Don't add dimensions or metrics to the specification just because it can be added.
- When defining the specification, consideration should be made to existing data already in the major cloud service providers' (AWS, GCP, Azure, OCI) datasets.
- As long as it solves the FinOps use case, there should be a preference to align with data that is already present in a majority of the major data generators.
- Strive for simplicity. However, prioritize accuracy, clarity, and consistency.
- Strive to build columns that serve a single purpose, with clear and concise names and values.
- The specification should allow data to be presented free from jargon, using simple understandable terms, and be approachable.
- Naming and terms used should be carefully considered to avoid using terms for which the definition could be confused by the reader. If a term must be used which has either an unclear or multiple definitions, it should be clarified in the [glossary](#).
- The specification should provide all of the data elements necessary for the [Capabilities](#).

1.4.3. Provider-neutral approach by default

- While the schema, naming, terminology, and attributes of many service providers are reviewed during development, this specification aims to be service-provider-neutral.
- Contributors must take care to ensure the specification examines how each decision relates to each of the major cloud service providers and SaaS vendors, not favoring any single one.
- In some cases, the approach may closely resemble one or more service provider's implementations, while in other cases, the approach might be new. In all cases, the FOCUS group (community composed of FinOps practitioners, Cloud and SaaS providers and FinOps vendors) will attempt to prioritize enabling FinOps [Capabilities](#) and alignment with the FinOps [Framework](#).

1.4.4. Extensibility

The FOCUS Specification is designed to support evolving FinOps needs across diverse billing models and service provider types.

While the initial focus was on billing data from Cloud Service Providers (CSPs), version 1.2 introduces foundational support for Software as a Service (SaaS) platforms, including normative columns for pricing currencies, effective cost, and contracted pricing in non-monetary units such as credits or tokens.

The specification supports extensibility through structured naming conventions (e.g., x_ custom columns), conditional requirements, and a version-aware schema approach.

Future versions of FOCUS will consider including additional FinOps capabilities such as forecasting, exchange rate modeling, and anomaly detection, while continuing to support a broader range of

billing and cost datasets — including internal infrastructure platforms and marketplace offerings.

1.5. Design Notes

1.5.1. Optimize for data analysis

- Optimize columns for data analysis at scale and avoid the requirement of splitting or parsing values.
- Avoid complex JSON structures when an alternative columnar structure is possible.
- Facilitate the inclusion of data necessary for a system of record for cost and usage data to consume.

1.5.2. Consistency helps with clarity

- Where possible, use consistent names that will naturally create associations between related columns in the specification.
- Column naming must strictly follow the [column handling](#) requirements.
- Use established standards (e.g., ISO8601 for dates, ISO4217 for currency).

1.6. Typographic Conventions

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be interpreted as described in [BCP14 \[RFC2119\]\[RFC8174\]](#) when, and only when, they appear in all capitals, as shown here.

1.7. FOCUS Feature Level

Under each column defined in the FOCUS specification, there exists a 'Feature level' designation that describes the column as 'Mandatory', 'Conditional', or 'Optional'. Feature level is designated based on the following criteria described in the normative requirements in each column definition:

- If the existence of a column is described with MUST with no conditions of when it applies, then the feature level is designated as 'Mandatory'.
- If the existence of a column is described as MUST with conditions of when it applies, then the feature level is designated as 'Conditional'.
- If the existence of a column is described as RECOMMENDED, then the feature level is designated as 'Recommended'.
- If the existence of a column is described as MAY, then the feature level is designated as 'Optional'.

1.8. Conformance Checkers and Validators

Validation tools may be employed to determine conformance of data and implementations per this specification.

The FinOps Foundation maintains a validator called the [FOCUS Validator](#) which it uses for its own conformance assessments and serves as a reference implementation to support validation activities.

Other validation tools may be developed and made available by third parties. The FOCUS specification does not mandate the use of any particular tool, nor does it maintain a registry of available validators.

2. Supported Features

The FOCUS specification is designed to meet the needs of FinOps practitioners in numerous scenarios. The following section contains features supported by the FOCUS specification. This list does not represent all possible combinations or use of FOCUS data but does represent core capabilities that the FOCUS specification supports.

2.1. Account Structures

2.1.1. Description

Different service providers have different account constructs that FinOps practitioners use for allocation, reporting, and more. Organizations may have one or many accounts within one or more service providers and FinOps practitioners may need to review the cost broken down by each account. FOCUS has two types of accounts: a billing account and a sub account.

A billing account is the account where invoices are generated. Each billing account can have one or more sub accounts, which can be used for deploying and managing resources and services. Billing and sub accounts are often used to facilitate allocation strategies and FinOps practitioners must be able to break costs down by billing and sub account to facilitate FinOps scenarios like chargeback and budgeting.

2.1.2. Directly Dependent Columns

- BilledCost
- BillingAccountId
- BillingAccountName
- BillingAccountType
- SubAccountId
- SubAccountName
- SubAccountType

2.1.3. Supporting Columns

- InvoiceId

2.1.4. Example SQL Query

SELECT

BillingAccountId,
BillingAccountName,
BillingAccountType,
SubAccountId,
SubAccountName,
SubAccountType,
SUM(BilledCost)

FROM focus_data_table**WHERE** BillingPeriodStart >= ? **AND** BillingPeriodEnd < ?**GROUP BY**

BillingAccountId,
SubAccountId

2.1.5. Introduced (Version)

0.5

2.2. Billed Cost and Invoice Alignment**2.2.1. Description**

FOCUS data should be consistent with the costs indicated on payable invoices. This is relevant to the total cost of the invoice, as well as the period of time the invoice covers.

2.2.2. Directly Dependent Columns

- BilledCost
- BillingCurrency
- BillingPeriodEnd
- BillingPeriodStart
- InvoiceId

2.2.3. Supporting Columns

- ServiceName

2.2.4. Example SQL Query

SELECT

BillingPeriodStart,
BillingPeriodEnd,
InvoiceId,
SUM(BilledCost)

FROM focus_data_table

GROUP BY

BillingPeriodStart,
BillingPeriodEnd,
InvoiceId

2.2.5. Introduced (Version)

0.5

2.3. Charge Categorization

2.3.1. Description

FOCUS supports the categorization of charges including purchases, usage, tax, credits and adjustments. It includes classification on frequency. It includes classification on correction vs normal entries.

2.3.2. Directly Dependent Columns

- ChargeCategory
- ChargeClass
- ChargeFrequency

2.3.3. Supporting Columns

- BilledCost
- BillingAccountId
- BillingPeriodEnd
- BillingPeriodStart
- CommitmentDiscountId
- CommitmentDiscountType
- ServiceProviderName
- ServiceCategory

2.3.4. Example SQL Query

2.3.4.1. Report on Commitment Discount Purchases

SELECT

```
MIN(ChargePeriodStart) AS ChargePeriodStart,  
MAX(ChargePeriodEnd) AS ChargePeriodEnd,  
ServiceProviderName,  
BillingAccountId,  
CommitmentDiscountId,  
CommitmentDiscountType,  
CommitmentDiscountUnit,  
CommitmentDiscountQuantity,  
ChargeFrequency,  
SUM(BilledCost) AS TotalBilledCost
```

```
FROM focus_data_table
```

```
WHERE ChargePeriodStart >= ? AND ChargePeriodEnd < ?
```

```
AND ChargeCategory = 'Purchase'
```

```
AND CommitmentDiscountId IS NOT NULL
```

GROUP BY

```
ServiceProviderName,  
BillingAccountId,  
CommitmentDiscountId,  
CommitmentDiscountType,  
CommitmentDiscountUnit,  
CommitmentDiscountQuantity,  
ChargeFrequency
```

2.3.4.2. Report on Corrections**SELECT**

```
ServiceProviderName,  
BillingAccountId,  
ChargeCategory,  
ServiceCategory,  
ServiceName,  
SUM(BilledCost) AS TotalBilledCost
```

```
FROM focus_data_table
```

```
WHERE BillingPeriodStart >= ? AND BillingPeriodEnd < ?
```

```
AND ChargeClass = 'Correction'
```

GROUP BY

```
ServiceProviderName,  
BillingAccountId,  
ChargeCategory,  
ServiceCategory,  
ServiceName
```

2.3.4.3. Report Recurring Charges

```

SELECT
BillingPeriodStart,
CommitmentDiscountId,
CommitmentDiscountName,
CommitmentDiscountType,
ChargeFrequency,
SUM(BilledCost) AS TotalBilledCost
FROM focus_data_table
WHERE BillingPeriodStart >= ? AND BillingPeriodStart < ?
AND ChargeFrequency = 'Recurring'
AND CommitmentDiscountId IS NOT NULL
GROUP BY
BillingPeriodStart,
CommitmentDiscountId,
CommitmentDiscountName,
CommitmentDiscountType,
ChargeFrequency

```

2.3.5. Introduced (Version)

1.0

2.4. Commit Usage and Under Usage

2.4.1. Description

FOCUS supports the tracking of commitment discounts usage and under usage, which can come in the form of commitment discounts or capacity reservations.

2.4.2. Directly Dependent Columns

- CommitmentDiscountID
- CommitmentDiscountStatus
- CommitmentDiscountType
- CapacityReservationID
- CapacityReservationStatus
- CapacityReservationType

2.4.3. Supporting Columns

- BilledCost
- ChargePeriodStart
- ChargePeriodEnd
- EffectiveCost
- ServiceCategory

2.4.4. Example SQL Query for Commitment Discounts

```

SELECT
  ServiceProviderName,
  BillingAccountId,
  CommitmentDiscountId,
  CommitmentDiscountType,
  CommitmentDiscountStatus,
  SUM(BilledCost) AS TotalBilledCost,
  SUM(EffectiveCost) AS TotalEffectiveCost
FROM focus_data_table
WHERE ChargePeriodStart >= ? AND ChargePeriodEnd < ?
AND CommitmentDiscountStatus = 'Unused'
GROUP BY
  ServiceProviderName,
  BillingAccountId,
  CommitmentDiscountId,
  CommitmentDiscountType

```

2.4.5. Example SQL Query for Capacity Reservations

```

SELECT
  ServiceProviderName,
  BillingAccountId,
  CapacityReservationId,
  CapacityReservationStatus,
  SUM(BilledCost) AS TotalBilledCost,
  SUM(EffectiveCost) AS TotalEffectiveCost
FROM focus_data_table
WHERE ChargePeriodStart >= ? AND ChargePeriodEnd < ?
AND CapacityReservationStatus = 'Unused'
GROUP BY
  ServiceProviderName,
  BillingAccountId,
  CapacityReservationId,
  CapacityReservationStatus

```

2.4.6. Introduced (Version)

1.0

2.5. Contract Commitments

2.5.1. Description

FOCUS supports the tracking of commitments made via contractual agreements between a service provider and a customer. Each row in the Cost and Usage dataset is associated with one or more unique identifiers representing those contracts and contract commitments, stored in a JSON column called Contract Applied. A richer amount of detail that describes those commitments is carried in a separate Contract Commitment dataset, which can be joined to the Cost and Usage dataset to facilitate various queries involving filtering and aggregation.

The Contract Applied column contains several FOCUS-defined properties. For more information, see the definition of Contract Applied [here](#).

2.5.2. Directly Dependent Columns

- CostAndUsage
 - ContractApplied

2.5.3. Supporting Columns

- ContractCommitment
 - BillingCurrency
 - ContractCommitmentCategory
 - ContractCommitmentCost
 - ContractCommitmentDescription
 - ContractCommitmentId
 - ContractCommitmentPeriodEnd
 - ContractCommitmentPeriodStart
 - ContractCommitmentQuantity
 - ContractCommitmentType
 - ContractCommitmentUnit
 - ContractId
 - ContractPeriodEnd
 - ContractPeriodStart

2.5.4. Example SQL Queries

The FOCUS specification implements the application of contract commitments to cost and usage via the [ContractApplied](#) column, which is defined in [JSON object format](#).

Because ANSI SQL does not inherently support the parsing of JSON, the following queries leverage the JSON functions found in BigQuery Standard SQL in order to demonstrate this feature's functionality. Similar JSON functions are available in all major SQL engines; thus, the below examples can be slightly modified to accommodate any particular database instance.

2.5.4.1. Report on Initial Contract Commitment

This query takes inputs of a time range via ChargePeriodStart and ChargePeriodEnd, then presents the aggregation of initial contract commitments from the CostAndUsage dataset per ServiceProviderName and ContractCommitmentID by filtering on the specified time range, along with ChargeCategory of Purchase.

SELECT

```

MIN(CU.ChargePeriodStart) AS ChargePeriodStart,
MAX(CU.ChargePeriodEnd) AS ChargePeriodEnd,
CU.ServiceProviderName,
JSON_VALUE(CA, '$.ContractCommitmentID') AS ContractCommitmentId,
SUM(CAST(JSON_VALUE(CA, '$.ContractCommitmentAppliedCost') AS FLOAT64)) AS ContractComm

```

FROM CostAndUsage CU

CROSS JOIN

```

UNNEST(JSON_EXTRACT_ARRAY(CU.ContractApplied, '$.Elements')) AS CA

```

WHERE JSON_VALUE(CA, '\$.ContractCommitmentAppliedCost') IS NOT NULL

```

AND ChargePeriodStart >= ? AND ChargePeriodEnd < ?

```

```

AND ChargeCategory = 'Purchase'

```

GROUP BY ServiceProviderName, ContractCommitmentId**ORDER BY** ServiceProviderName, ContractCommitmentId**2.5.4.2. Report on Usage Against Contract Commitment**

This query takes inputs of a time range via ChargePeriodStart and ChargePeriodEnd, then presents the aggregation of the application of contract commitments from the CostAndUsage dataset per ServiceProviderName and ContractCommitmentID by filtering on the specified time range, along with ChargeCategory of Usage.

SELECT

```

MIN(CU.ChargePeriodStart) AS ChargePeriodStart,
MAX(CU.ChargePeriodEnd) AS ChargePeriodEnd,
CU.ServiceProviderName,
JSON_VALUE(CA, '$.ContractCommitmentID') AS ContractCommitmentId,
SUM(CAST(JSON_VALUE(CA, '$.ContractCommitmentAppliedCost') AS FLOAT64)) AS ContractComm

```

FROM CostAndUsage CU

CROSS JOIN

```

UNNEST(JSON_EXTRACT_ARRAY(CU.ContractApplied, '$.Elements')) AS CA

```

WHERE JSON_VALUE(CA, '\$.ContractCommitmentAppliedCost') IS NOT NULL

```

AND ChargePeriodStart >= ? AND ChargePeriodEnd < ?

```

```

AND ChargeCategory = 'Usage'

```

GROUP BY ServiceProviderName, ContractCommitmentId**ORDER BY** ServiceProviderName, ContractCommitmentId**2.5.4.3. Report on Usage Against Contract Commitment by Category**

This query takes inputs of a time range via ChargePeriodStart and ChargePeriodEnd, then presents the aggregation of the application of contract commitments from the CostAndUsage dataset per ServiceProviderName and ContractCommitmentID by filtering on the specified time range, along with ChargeCategory of Usage. It also joins in the ContractCommitment dataset to provide further information about each contract commitment (in this case, the start and end date/time).

```

SELECT
  MIN(CU.ChargePeriodStart) AS ChargePeriodStart,
  MAX(CU.ChargePeriodEnd) AS ChargePeriodEnd,
  CU.ServiceProviderName,
  JSON_VALUE(CA, '$.ContractCommitmentID') AS ContractCommitmentId,
  CC.ContractCommitmentPeriodStart,
  CC.ContractCommitmentPeriodEnd,
  SUM(CAST(JSON_VALUE(CA, '$.ContractCommitmentAppliedCost') AS FLOAT64)) AS ContractCommitt
FROM CostAndUsage CU
CROSS JOIN
  UNNEST(JSON_EXTRACT_ARRAY(CU.ContractApplied, '$.Elements')) AS CA
INNER JOIN
  ContractCommitment CC
ON
  JSON_VALUE(CA, '$.ContractCommitmentID') = CC.ContractCommitmentID
WHERE JSON_VALUE(CA, '$.ContractCommitmentAppliedCost') IS NOT NULL
  AND ChargePeriodStart >= ? AND ChargePeriodEnd < ?
  AND ChargeCategory = 'Usage'
GROUP BY ServiceProviderName, ContractCommitmentId, ContractCommitmentPeriodStart, Contract
ORDER BY ServiceProviderName, ContractCommitmentId, ContractCommitmentPeriodStart, Contract

```

2.5.5. Introduced (Version)

1.3

2.6. Cost and Usage Attribution

2.6.1. Description

Many service providers have features that allow FinOps practitioners to enrich cost and usage data with metadata that is in addition to service provider defined data, in order to analyze FinOps data using organizational, deployment, or other structures. These features may take the form of directly applied metadata or inherited metadata. FOCUS facilitates the inclusion of this metadata at a row level.

2.6.2. Directly Dependent Columns

- Tags

2.6.3. Supporting Columns

- BilledCost
- ConsumedQuantity
- ConsumedUnit
- EffectiveCost

2.6.4. Example SQL Query

```

SELECT
tags,
ConsumedUnit,
SUM(BilledCost),
SUM(EffectiveCost),
SUM(ConsumedQuantity)
FROM focus_data_table
WHERE BillingPeriodStart >= ? AND BillingPeriodEnd < ?
GROUP BY
tags,
ConsumedUnit

```

2.6.5. Introduced (Version)

1.0

2.7. Cost Comparison

2.7.1. Description

FOCUS supports the comparison of cost columns in order to identify savings, amortization, or other constructs.

2.7.2. Directly Dependent Columns

- BilledCost
- ContractedCost
- EffectiveCost
- ListCost

2.7.3. Supporting Columns

- BillingAccountId
- BillingAccountName
- BillingCurrency
- BillingPeriodEnd
- BillingPeriodStart
- ChargePeriodEnd
- ChargePeriodStart
- ServiceName

2.7.4. Example SQL Query

```

WITH AggregatedData AS (
SELECT
    ServiceProviderName,
    BillingAccountId,
    BillingAccountName,
    BillingCurrency,
    ServiceName,
    SUM(EffectiveCost) AS TotalEffectiveCost,
    SUM(BilledCost) AS TotalBilledCost,
    SUM(CASE
        WHEN ChargeCategory = 'Usage' AND BilledCost = 0 AND EffectiveCost != 0
        THEN 0
        ELSE ContractedCost
    END) AS TotalContractedCost,
    SUM(CASE
        WHEN ChargeCategory = 'Usage' AND BilledCost = 0 AND EffectiveCost != 0
        THEN 0
        ELSE ListCost
    END) AS TotalListCost
FROM focus_data_table
WHERE BillingPeriodStart >= ?
    AND BillingPeriodEnd < ?
    AND ChargeClass IS NULL
GROUP BY
    ServiceProviderName,
    BillingAccountId,
    BillingAccountName,
    BillingCurrency,
    ServiceName
)
SELECT ServiceProviderName,
    BillingAccountId,
    BillingAccountName,
    BillingCurrency,
    ServiceName,
    TotalEffectiveCost,
    TotalBilledCost,
    TotalListCost,
    1 - (TotalContractedCost / NULLIF(TotalListCost, 0)) * 100 AS ContractedDiscount,
    1 - (TotalEffectiveCost / NULLIF(TotalListCost, 0)) * 100 AS EffectiveDiscount
FROM AggregatedData

```

2.7.5. Introduced (Version)

0.5

2.8. Custom Columns

2.8.1. Description

FOCUS supports the inclusion of custom columns to facilitate reporting capability that is not covered by the columns included in the specification.

2.8.2. Directly Dependent Columns

- x_CustomColumn

2.8.3. Example SQL Query

SELECT

```
BillingPeriodStart,  
x_CustomColumn,  
SUM(BilledCost) AS TotalBilledCost,
```

FROM focus_data_table

WHERE ServiceName = ?

AND BillingPeriodStart >= ? **AND** BillingPeriodStart < ?

GROUP BY

```
BillingPeriodStart,  
x_CustomColumn
```

ORDER BY MonthlyCost **DESC**

2.8.4. Introduced (Version)

0.5

2.9. Data Generator-Calculated Split Cost Allocation

2.9.1. Description

FOCUS enables tracking of resources split by some internal consumption metrics. This is most common for resources supporting shared usage like compute nodes in a shared cluster (Kubernetes, databases) or storage engines that can share capacity between workloads.

2.9.2. Directly Dependent Columns

- ResourceId
- EffectiveCost
- BilledCost
- AllocatedResourceId
- AllocatedResourceName
- AllocatedMethodDetails
- AllocatedMethodId

2.9.3. Supporting Columns

- ChargeCategory
- ChargePeriodEnd
- ChargePeriodStart
- ServiceProviderName
- ServiceName

2.9.4. Example SQL Query (Find resources with a shared cost)

```
SELECT  
  DISTINCT ResourceId  
FROM focus_data_table  
WHERE ChargeCategory='Usage'  
  AND ChargePeriodStart >= ? AND ChargePeriodEnd <= ?  
  AND AllocatedMethodId IS NOT NULL
```

2.9.5. Example SQL Query (Get total effective cost by ResourceId (ignore shared cost))

```
SELECT  
  ResourceId,  
  SUM(EffectiveCost) AS TotalEffectiveCost  
FROM focus_data_table  
WHERE ChargeCategory='Usage'  
  AND ChargePeriodStart >= ? AND ChargePeriodEnd <= ?  
  AND AllocatedMethodId IS NOT NULL  
GROUP BY  
  ResourceId
```

2.9.6. Example SQL Query (Get total effective cost by AllocatedResourceId)

```
SELECT  
  AllocatedResourceId,  
  SUM(EffectiveCost) AS TotalEffectiveCost  
FROM focus_data_table  
WHERE ChargeCategory='Usage'  
  AND ChargePeriodStart >= ? AND ChargePeriodEnd <= ?  
  AND AllocatedMethodId IS NOT NULL  
GROUP BY  
  AllocatedResourceId
```

2.9.7. Example SQL Query (Find total unallocated split costs by resourceId)

```
SELECT  
  ResourceId,  
  SUM(EffectiveCost) AS TotalEffectiveCost  
FROM focus_data_table  
WHERE ChargeCategory='Usage'  
  AND ChargePeriodStart >= ? AND ChargePeriodEnd <= ?  
  AND AllocatedMethodId IS NOT NULL AND AllocatedResourceId IS NULL  
GROUP BY  
  ResourceId
```

2.9.8. Example SQL Query (Find how a single resource has been split)

```
SELECT
  ResourceId,
  COALESCE(AllocatedResourceId, 'Unallocated') AS AllocatedResourceId,
  SUM(EffectiveCost) AS TotalEffectiveCost
FROM focus_data_table
WHERE ChargeCategory='Usage'
AND ChargePeriodStart >= ? AND ChargePeriodEnd <= ?
AND AllocatedResourceId = ?
GROUP BY
  ResourceId,
  COALESCE(AllocatedResourceId, 'Unallocated')
```

2.9.9. Example SQL Query (Extract JSON from AllocatedMethodDetails)

```
SELECT
  resource_id,
  elements.allocated_ratio,
  elements.usage_unit,
  elements.usage_quantity
FROM
  focus_data_table,
  JSON_TABLE(
    AllocatedMethodDetails,
    '$.Elements[*]' COLUMNS (
      allocated_ratio DECIMAL(10, 2) PATH '$.AllocatedRatio',
      usage_unit VARCHAR(50) PATH '$.UsageUnit',
      usage_quantity DECIMAL(10, 2) PATH '$.UsageQuantity'
    )
  ) AS elements
```

2.9.10. Introduced (Version)

1.3

2.10. Data Granularity

2.10.1. Description

FOCUS supports multiple levels of cost and usage data granularity. This includes the ability to report on a daily, hourly, or other time period basis. FOCUS also supports the ability for cost and usage data to be provided for high granularity scenarios, such as down to the individual resources. It also supports high level granularity cost and usage data, such as account level, or service level charges.

2.10.2. Directly Dependent Columns

- ResourceId

- ResourceName
- ChargePeriodEnd
- ChargePeriodStart

2.10.3. Supporting Columns

- BilledCost
- ConsumedQuantity
- ConsumedUnit
- EffectiveCost
- ListCost
- PricingCurrency
- PricingUnit

2.10.4. Example SQL Query

SELECT

```
ChargePeriodStart,  
ChargePeriodEnd,  
ResourceId,  
SUM(EffectiveCost)
```

FROM focus_data_table

Group by

```
ChargePeriodStart,  
ChargePeriodEnd,  
ResourceId
```

2.10.5. Introduced (Version)

0.5

2.11. Dataset Instance Metadata

2.11.1. Description

FOCUS supports the ability for data generators to provide metadata that describes information about the [dataset artifacts](#) they provide. This includes properties such as the name of the [dataset instance](#), the unique identifier of the *dataset instance*, and the [FOCUS dataset](#) that it aligns with. This metadata can be used by consumers to understand the context of the data they are receiving, and to ensure that they are working with the correct dataset instance to execute their particular FinOps use cases.

2.11.2. Applicable Metadata

- Dataset Instance
 - Dataset Instance ID
 - Dataset Instance Name
 - FOCUS Dataset ID

2.11.3. Introduced (Version)

1.3

2.12. Effective Cost Analysis

2.12.1. Description

FOCUS enables practitioners to analyze costs without having to distribute upfront fees and discounts, taking discounts and the amortization of upfront fees paid for services into account. The EffectiveCost column represents cost after negotiated discounts, commitment discounts, and the applicable portion of relevant, prepaid purchases (one-time or recurring) that covered this charge. EffectiveCost is commonly utilized to track and analyze spending trends.

2.12.2. Directly Dependent Columns

- EffectiveCost

2.12.3. Supporting Columns

- BillingPeriodEnd
- BillingPeriodStart
- ChargeCategory
- ChargePeriodEnd
- ChargePeriodStart
- ConsumedQuantity
- ConsumedUnit
- PricingQuantity
- ServiceProviderName
- RegionName
- ServiceName

2.12.4. Example SQL Query

```

SELECT
  ServiceProviderName,
  BillingPeriodStart,
  BillingPeriodEnd,
  ServiceCategory,
  ServiceName,
  RegionId,
  RegionName,
  PricingUnit,
  SUM(EffectiveCost) AS TotalEffectiveCost,
  SUM(PricingQuantity) AS TotalPricingQuantity
FROM focus_data_table
WHERE BillingPeriodStart >= ? AND BillingPeriodEnd <= ?
GROUP BY
  ServiceProviderName,
  BillingPeriodStart,
  BillingPeriodEnd,
  ServiceCategory,
  ServiceName,
  RegionId,
  RegionName,
  PricingUnit

```

2.12.5. Introduced (Version)

0.5

2.13. Location

2.13.1. Description

FOCUS provides structured location data through region and availability zone information. By documenting geographic deployment locations, practitioners can organize and analyze costs based on where resources and services are deployed. This standardized location data helps practitioners understand the geographical distribution of infrastructure across host providers.

2.13.2. Directly Dependent Columns

- AvailabilityZone
- RegionId
- RegionName

2.13.3. Supporting Columns

- BilledCost
- ChargePeriodEnd
- ChargePeriodStart

2.13.4. Example SQL Query

SELECT

```
RegionId,  
RegionName,  
AvailabilityZone,  
SUM(BilledCost) AS TotalBilledCost
```

FROM focus_data_table

```
WHERE ChargePeriodStart >= ? AND ChargePeriodEnd <= ?
```

GROUP BY

```
RegionId,  
RegionName,  
AvailabilityZone
```

2.13.5. Introduced (Version)

1.0

2.14. Marketplace Purchases

2.14.1. Description

FOCUS supports the analysis of cost and usage data for marketplace purchases and their associated costs. It also supports the reporting of EffectiveCost for usage from the service provider.

2.14.2. Directly Dependent Columns

- InvoiceIssuerName
- ServiceProviderName

2.14.3. Supporting Columns

- BilledCost
- EffectiveCost

2.14.4. Example SQL Query on a CSP Marketplace using the Cost and Usage FOCUS Dataset

```

SELECT
  ServiceProviderName,
  InvoiceIssuerName,
  BillingPeriodStart,
  BillingPeriodEnd,
  SUM(BilledCost) AS TotalBilledCost
FROM focus_data_table
WHERE ServiceProviderName = '<Example SaaS Provider>'
  AND InvoiceIssuerName = '<Example CSP Marketplace>'
GROUP BY
  ServiceProviderName,
  InvoiceIssuerName,
  BillingPeriodStart,
  BillingPeriodEnd

```

2.14.5. Example SQL Query on a Provider using the Cost and Usage FOCUS Dataset

```

SELECT
  ChargePeriodStart,
  ChargePeriodEnd,
  ResourceId,
  SUM(EffectiveCost) AS TotalEffectiveCost
FROM focus_data_table
WHERE InvoiceIssuerName = '<Example CSP Marketplace>'
GROUP BY
  ChargePeriodStart,
  ChargePeriodEnd,
  ResourceId

```

2.14.6. Introduced (Version)

1.0

2.15. Recency Metadata

2.15.1. Description

FOCUS supports the ability for data generators to provide metadata indicating 1) what portion of a [FOCUS dataset artifact](#) is complete (either in total, or per time sector), and 2) how recently it has been updated. This metadata allows practitioners to understand whether a given subset of FOCUS data is subject to further change, which informs when and whether they can perform various FinOps functions such as chargeback.

2.15.2. Applicable Metadata

- Recency
 - Dataset Instance Complete
 - Dataset Instance Last Updated
 - Dataset Instance ID

- Recency Last Updated
- Time Sectors
 - Time Sector Start
 - Time Sector End
 - Time Sector Complete
 - Time Sector Last Updated

2.15.3. Introduced (Version)

1.3

2.16. Resource Usage

2.16.1. Description

FOCUS enables tracking of resource consumption by providing information about which resources were used, in what quantities, and with what units of measure.

2.16.2. Directly Dependent Columns

- ConsumedQuantity
- ConsumedUnit
- ResourceId
- Skuld

2.16.3. Supporting Columns

- ChargeCategory
- ChargePeriodEnd
- ChargePeriodStart
- ServiceProviderName
- ServiceName

2.16.4. Example SQL Query

```

SELECT
  ServiceProviderName,
  ServiceName,
  ResourceId,
  Skuld,
  ConsumedUnit,
  SUM(ConsumedQuantity) AS TotalQuantity
FROM focus_data_table
WHERE ChargeCategory='Usage'
  AND ChargePeriodStart >= ? AND ChargePeriodEnd <= ?
GROUP BY
  ServiceProviderName,
  ServiceName,
  ResourceId,
  Skuld,
  ConsumedUnit

```

2.16.5. Introduced (Version)

1.0

2.17. Schema Metadata

2.17.1. Description

FOCUS' schema metadata supports communication of important attributes about the data, facilitating notifications about changing structure and database table creation between data generator and consumer. This includes column names, data types, and any other relevant information about the data schema. It also includes information as to the version of FOCUS and Data Generator versioning that the data uses.

2.17.2. Applicable Metadata

- Schema
 - Column Definition

2.17.3. Introduced (Version)

1.1

2.18. Service Provider Services

2.18.1. Description

FOCUS supports service providers specifying the services and product offerings that they provide their customers that align with the names practitioners are familiar with. This empowers practitioners to analyze cost by service, report service costs by subaccount, forecast based on

historical trends by service, and verify accuracy of services charged across service providers.

2.18.2. Directly Dependent Columns

- ServiceCategory
- ServiceName
- ServiceSubcategory

2.18.3. Supporting Columns

- ServiceProviderName
- Skuld

2.18.4. Example SQL Query

SELECT

```
BillingPeriodStart,  
ServiceProviderName,  
SubAccountId,  
SubAccountName,  
ServiceName,  
SUM(BilledCost) AS TotalBilledCost,  
SUM(EffectiveCost) AS TotalEffectiveCost
```

FROM focus_data_table

WHERE ServiceName = ?

AND BillingPeriodStart >= ? **AND** BillingPeriodStart < ?

GROUP BY

```
BillingPeriodStart,  
ServiceProviderName,  
SubAccountId,  
SubAccountName,  
ServiceName
```

ORDER BY MonthlyCost **DESC**

2.18.5. Introduced (Version)

0.5

2.19. Service Categorization

2.19.1. Description

FOCUS provides a structure for categorizing services based on their core functions. By classifying services into high-level categories and more granular subcategories, practitioners can organize costs according to functional areas. This standardized categorization provides data that practitioners can use in their cost management processes and decision making.

2.19.2. Directly Dependent Columns

- ServiceCategory
- ServiceName
- ServiceSubcategory

2.19.3. Supporting Columns

- BilledCost
- BillingCurrency
- BillingPeriodEnd
- BillingPeriodStart
- ServiceProviderName

2.19.4. Example SQL Query

SELECT

```
BillingPeriodStart,  
BillingPeriodEnd,  
ServiceProviderName,  
ServiceCategory,  
ServiceSubcategory,  
ServiceName,  
BillingCurrency,  
SUM(BilledCost) AS TotalBilledCost
```

FROM focus_data_table

WHERE BillingPeriodStart >= ? **and** BillingPeriodEnd < ?

GROUP BY

```
BillingPeriodStart,  
BillingPeriodEnd,  
ServiceProviderName,  
ServiceCategory,  
ServiceSubcategory,  
ServiceName,  
BillingCurrency
```

2.19.5. Introduced (Version)

1.1

2.20. Participating Entity Identification

2.20.1. Description

FOCUS allows practitioners to identify the several participating entities involved in resource or service hosting, invoicing, and data generation. The FOCUS Specification includes multiple columns to identify key participating entities, including Service Provider Name, Invoice Issuer Name, Host Provider Name, and Data Generator.

2.20.2. Directly Dependent Columns

- ServiceProviderName
- InvoiceIssuerName
- HostProviderName

2.20.3. Applicable Metadata

- DataGenerator

2.20.4. Example SQL Query

SELECT

```
BillingPeriodStart,  
BillingPeriodEnd,  
ServiceProviderName,  
InvoiceIssuerName,  
HostProviderName,  
ServiceName,  
BillingCurrency,  
SUM(BilledCost) AS TotalBilledCost
```

FROM

```
focus_data_table
```

WHERE

```
BillingPeriodStart >= ? and BillingPeriodEnd < ?
```

GROUP BY

```
BillingPeriodStart,  
BillingPeriodEnd,  
ServiceProviderName,  
InvoiceIssuerName,  
HostProviderName,  
ServiceName,  
BillingCurrency
```

2.20.5. Introduced (Version)

1.1

2.20.6. Updated (Version)

1.3

2.21. Verification, Comparison, and Fluctuation Tracking of Unit Prices

2.21.1. Description

When a service provider supports unit pricing concepts, FOCUS allows practitioners to:

- Verify that the correct List Unit Prices and Contracted Unit Prices are applied.

- Compare applied Contracted Unit Prices across different billing accounts and with applied List Unit Prices at specific points in time.
- Track fluctuations in unit prices over time.

2.21.2. Directly Dependent Columns

- ContractedUnitPrice
- ListUnitPrice
- Skuld
- SkuPriceDetails
- SkuPriceId

2.21.3. Supporting Columns

- BillingCurrency
- BillingPeriodId
- ChargePeriodEnd
- ChargePeriodStart

2.21.4. Example SQL Query

SELECT DISTINCT

Skuld,
SkuPriceId,
SkuPriceDetails,
BillingPeriodId,
ChargePeriodStart,
ChargePeriodEnd,
BillingCurrency,
ListUnitPrice,
ContractedUnitPrice

FROM focus_data_table

WHERE

SkuPriceId = ?

AND ChargePeriodStart >= ?

AND ChargePeriodEnd < ?

2.21.5. Introduced (Version)

1.0

3. Datasets

FOCUS defines many individual datasets made up of a selected set of columns which abide by the attributes outlined in this FOCUS Specification.

3.1. Cost and Usage

The Cost and Usage dataset is the primary dataset for FOCUS cost and usage data.

The specification for the Cost and Usage dataset defines a group of columns that provide qualitative values (such as dates, resource, and service provider information) categorized as "dimensions" and quantitative values (numeric values) categorized as "metrics" that can be used for performing various [FinOps capabilities](#). Metrics are commonly used for aggregations (sum, multiplication, averaging etc.) and statistical operations within the dataset. Dimensions are commonly used to categorize, filter, and reveal details in your data when combined with metrics. The columns are presented in alphabetical order.

Columns

Column	Column Type	Feature Level	Allows Nulls	Data Type
Allocated Method Details	Dimension	Recommended	True	JSON
Allocated Method ID	Dimension	Conditional	True	String
Allocated Resource ID	Dimension	Conditional	True	String
Allocated Resource Name	Dimension	Conditional	True	String
Allocated Tags	Dimension	Conditional	True	JSON
Availability Zone	Dimension	Recommended	True	String
Billed Cost	Metric	Mandatory	False	Decimal
Billing Account ID	Dimension	Mandatory	False	String
Billing Account Name	Dimension	Mandatory	True	String
Billing Account Type	Dimension	Conditional	False	String
Billing Currency	Dimension	Mandatory	False	String
Billing Period End	Dimension	Mandatory	False	Date/Time
Billing Period Start	Dimension	Mandatory	False	Date/Time
Capacity Reservation ID	Dimension	Conditional	True	String
Capacity Reservation Status	Dimension	Conditional	True	String
Charge Category	Dimension	Mandatory	False	String
Charge Class	Dimension	Mandatory	True	String
Charge Description	Dimension	Mandatory	True	String
Charge Frequency	Dimension	Recommended	False	String
Charge Period End	Dimension	Mandatory	False	Date/Time
Charge Period Start	Dimension	Mandatory	False	Date/Time
Commitment Discount Category	Dimension	Conditional	True	String
Commitment Discount ID	Dimension	Conditional	True	String
Commitment Discount Name	Dimension	Conditional	True	String
Commitment Discount Quantity	Metric	Conditional	True	Decimal

Column	Column Type	Feature Level	Allows Nulls	Data Type
Commitment Discount Status	Dimension	Conditional	True	String
Commitment Discount Type	Dimension	Conditional	True	String
Commitment Discount Unit	Dimension	Conditional	True	String
Consumed Quantity	Metric	Conditional	True	Decimal
Consumed Unit	Dimension	Conditional	True	String
Contract Applied	Dimension / Metric	Conditional	True	JSON
Contracted Cost	Metric	Mandatory	False	Decimal
Contracted Unit Price	Metric	Conditional	True	Decimal
Effective Cost	Metric	Mandatory	False	Decimal
Host Provider Name	Dimension	Mandatory	False	String
Invoice ID	Dimension	Recommended	True	String
Invoice Issuer Name	Dimension	Mandatory	False	String
List Cost	Metric	Mandatory	False	Decimal
List Unit Price	Metric	Conditional	True	Decimal
Pricing Category	Dimension	Conditional	True	String
Pricing Currency	Dimension	Conditional	True	String
Pricing Currency Contracted Unit Price	Metric	Conditional	True	Decimal
Pricing Currency Effective Cost	Metric	Conditional	True	Decimal
Pricing Currency List Unit Price	Metric	Conditional	True	Decimal
Pricing Quantity	Metric	Mandatory	True	Decimal
Pricing Unit	Dimension	Mandatory	True	String
Provider - DEPRECATED	Dimension	Mandatory	False	String
Publisher - DEPRECATED	Dimension	Mandatory	False	String
Region ID	Dimension	Conditional	True	String
Region Name	Dimension	Conditional	True	String
Resource ID	Dimension	Conditional	True	String
Resource Name	Dimension	Conditional	True	String
Resource Type	Dimension	Conditional	True	String
Service Category	Dimension	Mandatory	False	String
Service Name	Dimension	Mandatory	False	String
Service Provider Name	Dimension	Mandatory	False	String
Service Subcategory	Dimension	Recommended	False	String
SKU ID	Dimension	Conditional	True	String
SKU Meter	Dimension	Conditional	True	String
SKU Price Details	Dimension	Conditional	True	JSON
SKU Price ID	Dimension	Conditional	True	String
Sub Account ID	Dimension	Conditional	True	String
Sub Account Name	Dimension	Conditional	True	String
Sub Account Type	Dimension	Conditional	True	String
Tags	Dimension	Conditional	True	JSON

Relationships

The Cost and Usage dataset can be joined to the Contract Commitment dataset through the use of the Contract Commitment ID.

- In the Cost and Usage dataset, Contract Commitment ID is a property within a JSON object array provided in Contract Applied column.
- In the Contract Commitment dataset, Contract Commitment ID is a column.

Dataset A	Dataset A Column	Dataset B	Dataset B Column
Cost and Usage	Contract Applied	Contract Commitment	Contract Commitment ID

Requirements

CostAndUsage adheres to the following requirements:

- CostAndUsage MUST be present.
- CostAndUsage MUST conform to [ColumnHandling](#) requirements.
- CostAndUsage MUST conform to [NullHandling](#) requirements.
- CostAndUsage MUST conform to [DiscountHandling](#) requirements.
- CostAndUsage MUST conform to [InvoiceHandling](#) requirements.
- CostAndUsage MUST conform to [DataGeneratorCalculatedSplitCostAllocationHandling](#) requirements.

Dataset ID

CostAndUsage

Display Name

Cost and Usage

Description

Describes the cost and usage incurred through using or purchasing a service provider's [resources](#) or [services](#).

Introduced (version)

0.5

3.1.1. Allocated Method ID

Allocated Method ID is the unique identifier for the [allocated method](#) defined by the service provider which was used for the [Data Generator-Calculated Split Cost Allocation](#). This unique identifier can be used to find how the [allocated charge](#) was calculated in the provider's documentation.

3.1.1.1. Requirements

AllocatedMethodId adheres to the following requirements:

- AllocatedMethodId MUST be present in a Cost and Usage [FOCUS dataset](#) when the data generator supports data generator-calculated split cost allocation.
- AllocatedMethodId MUST be of type String.
- AllocatedMethodId MUST conform to [StringHandling](#) requirements.
- AllocatedMethodId nullability is defined as follows:
 - AllocatedMethodId MUST be null when a [charge](#) is not related to a data generator-calculated split cost allocation.
 - AllocatedMethodId MUST NOT be null when a [charge](#) is related to a data generator-

- Data generator documentation of a split cost allocation method MUST make reference to a single AllocatedMethodId value.

3.1.1.2. Column ID

AllocatedMethodId

3.1.1.3. Display Name

Allocated Method ID

3.1.1.4. Description

A unique identifier defining the method of data generator-calculated split cost allocation.

3.1.1.5. Content constraints

Constraint	Value
Column type	Dimension
Feature level	Conditional
Allows nulls	True
Data type	String
Value format	<not specified>

3.1.1.6. Introduced (version)

1.3

3.1.2. Allocated Method Details

Allocated Method Details provides information about how resources are allocated when usage records are split to support cost allocation requirements.

Allocated Method Details consists of a valid JSON object which contains an array consisting of key-value objects describing the one or more factors that determined the split cost allocation. Each object consists of FOCUS-defined keys but can be extended to provide additional details about the allocation.

The FOCUS-defined properties are:

- Allocated Ratio: The ratio of a [charge](#) that this allocation represents.
- Usage Unit: Unit being measured used to calculate this allocation.
- Usage Quantity: The quantity of units used denominated by the defined usage unit.

In addition to these, a data generator may include one or more custom properties, also denoted as key-value pairs.

3.1.2.1. Requirements

3.1.2.1.1. Column Requirements

The AllocatedMethodDetails column adheres to the following requirements:

- AllocatedMethodDetails SHOULD be present in a Cost and Usage [FOCUS dataset](#) when the data generator supports [Data Generator-Calculated Split Cost Allocation](#).
- AllocatedMethodDetails MUST be of type String.
- AllocatedMethodDetails MUST conform to [StringHandling](#) requirements.
- AllocatedMethodDetails MUST conform to [JsonObjectFormat](#) requirements.
- AllocatedMethodDetails nullability is defined as follows:
 - AllocatedMethodDetails MUST be null when a charge is not related to a data generator-calculated split cost allocation.
 - AllocatedMethodDetails SHOULD NOT be null when a charge is related to a data generator-calculated split cost allocation.

3.1.2.1.2. Object Schema Requirements

Allocated Method Details consists of a valid JSON object which contains an array of key-value objects describing the one or more factors (allocation properties) that determined the split cost allocation. Each object consists of FOCUS-defined keys but can be extended to provide additional details about the allocation.

When AllocatedMethodDetails is not null, the JsonObjectFormat for AllocatedMethodDetails adheres to the following requirements:

- AllocatedMethodDetails MUST have a top-level key "Elements" which contains an array.
- Each item in "Elements" MUST be an object.
 - Objects inside "Elements" MUST conform to [KeyValueFormat](#) requirements.
 - FOCUS-defined allocation properties adhere to the following additional requirements:
 - Allocation property key MUST match the spelling and casing specified for the FOCUS-defined property.
 - Allocation property value MUST be of the type specified for that property.
 - Allocation properties MUST adhere to additional normative requirements specific to that property.
 - Data generator-defined allocation properties MAY be included in "Elements".
 - Allocation property keys MUST begin with the string "x_" unless it is a FOCUS-defined allocation property.
- AllocatedMethodDetails root object MAY contain additional data generator-defined items, in addition to "Elements".

3.1.2.1.3. Content Requirements

The following keys are used for allocation properties to facilitate querying data across allocations and across data generators. Focus-defined keys will appear in the list below and data generator-defined keys will be prefixed with "x_" to make them easy to identify as well as prevent collisions.

Allocated Ratio

Allocated Ratio communicates the percentage of the [Origin Charge](#) that this [Allocated Charge](#) derived

from the corresponding [Allocated Method Id](#) and Usage Unit property.

The "AllocatedRatio" property adheres to the following requirements:

- "AllocatedRatio" MUST be included inside each "Elements" object.
- Values for "AllocatedRatio" MUST be a decimal value compatible with [NumericFormat](#) representing the allocated charge's percentage of the origin charge.
- Values for all "AllocatedRatio" properties across all allocated charges related to a single origin charge MUST sum up to 1 (100%).

Usage Unit

Usage Unit communicates the aspect of the documented Allocation Method Id being used to calculate the Allocated Ratio property and what is being measured by Usage Quantity property.

The "UsageUnit" property adheres to the following requirements:

- "UsageUnit" MUST be included inside an "Elements" object if "UsageQuantity" allocation property is included in that "Elements" object, otherwise "UsageUnit" MAY be included in each "Elements" object.
- Values for "UsageUnit" MUST capture the unit or component of data generator's documented [AllocationMethod](#) that was used to determine the "AllocatedRatio" value.
- Values for "UsageUnit" SHOULD conform to [UnitFormat](#) requirements.

Usage Quantity

Usage Quantity communicates the volume that was consumed or used, denominated in the Usage Unit property value.

The "UsageQuantity" property adheres to the following requirements:

- "UsageQuantity" MAY be included inside an "Elements" object when that "Elements" object contains a "UsageUnit" allocation property.
- Values for "UsageQuantity" MUST be compatible with NumericFormat.
- Values for "UsageQuantity" SHOULD capture the quantity or volume of the "UsageUnit" measured by the data generator that was used to determine the "AllocatedRatio" value.

3.1.2.2. Overview

3.1.2.2.1. Array of Objects

The parent array is called Elements and contains one or more objects which communicate information about how an allocated record was calculated.

Key	ValueType	Required	Description
Elements	Array	True	The parent array containing one or more objects which communicate information about how an allocated record was calculated.

3.1.2.2.2. Object Entries

The Elements array contains one or more objects, each of which contains the following entries:

Key	ValueType	Required	Description
AllocatedRatio	Numeric	True	Percentage of overall cost derived from corresponding method and metric.

Key	ValueType	Required	Description
UsageUnit	String	Conditional	Unit being measured used to calculate allocation.
UsageQuantity	Numeric	False	Volume of UsageUnit consumed or used.

3.1.2.2.3. Example

```
{
  "Elements" : [ {
    "AllocatedRatio" : 0.05,
    "UsageUnit" : "CPU",
    "UsageQuantity" : 0.5
  }, {
    "AllocatedRatio" : 0.1,
    "UsageUnit" : "Memory",
    "UsageQuantity" : 4
  } ]
}
```

3.1.2.2.4. JSON Type Definition

```
{
  "properties": {
    "Elements": {
      "elements": {
        "properties": {
          "AllocatedRatio": { "type": "float64" }
        },
        "optionalProperties": {
          "UsageUnit": { "type": "string" },
          "UsageQuantity": { "type": "float64" }
        },
        "additionalProperties": true
      }
    }
  },
  "additionalProperties": true
}
```

NOTE: The above JSON Type Definition (JTD) is an approximation of the expected contents of this column, but it should not be considered normative because it cannot accurately describe the normative requirements (above) for AllocatedMethodDetails. Where there are discrepancies, deference will be given to the normative requirements. For example, [NumericFormat](#) allows for multiple numeric data types and precisions, but JTD requires both to be specified; other numeric data types and precisions allowable under NumericFormat are considered valid.

3.1.2.3. Example Scenarios

The JSON samples in the scenarios below each represent a single allocated record out of the multiple records derived from an origin record for that scenario. The sum AllocatedRatio will add up to 1 (100%) across all allocated records for an origin record, with the AllocatedRatio (or sum of AllocatedRatio) representing the allocated record's portion of the overall origin record.

3.1.2.3.1. Scenario 1: Single "UsageUnit" value used for allocation

When only a single "UsageUnit" is used to calculate the allocation.

```
{
  "Elements" : [ {
    "AllocatedRatio" : 0.1,
    "UsageUnit" : "Hours",
    "UsageQuantity" : 300
  }
]
```


3.1.2.3.2. Scenario 2: Multiple "UsageUnit" values used for allocation

When multiple "UsageUnit" values are used to calculate the allocation, another object is added to the "Elements" collection.

```
{
  "Elements": [
    {
      "AllocatedRatio": 0.05,
      "UsageUnit": "CPU",
      "UsageQuantity": 0.5
    },
    {
      "AllocatedRatio": 0.1,
      "UsageUnit": "Memory",
      "UsageQuantity": 4
    }
  ]
}
```

3.1.2.3.3. Scenario 3: Data generator omits keys that are not required

This data generator does not wish to supply the "UsageUnit" or "UsageQuantity" keys but still provides cost allocation with some additional allocation method details. In this case, "UsageUnit" and "UsageQuantity" are omitted, and only the "AllocatedRatio" is supplied.

```
{
  "Elements" : [ {
    "AllocatedRatio" : 0.45
  }
]
```

3.1.2.3.4. Scenario 4: Additional non-FOCUS specified properties

A data generator can add additional properties if they feel more context is helpful or necessary to the practitioner. In this scenario, the data generator is supplying additional context that shows only 0.5 of a unit was used. However, since 1 unit was requested by the service this allocation represents, the

allocation is being charged at 1 regardless.

```
{
  "Elements": [
    {
      "AllocatedRatio": 0.6,
      "UsageUnit": "vCPU",
      "UsageQuantity": 1,
      "x_ReservedVCPU": 1,
      "x_UsedVCPU": 0.5,
      "x_AllocatedVCPU": 1
    }
  ]
}
```

3.1.2.4. Column ID

AllocatedMethodDetails

3.1.2.5. Display Name

Allocated Method Details

3.1.2.6. Description

A set of properties describing how resources are allocated in data generator-defined split cost allocation.

3.1.2.7. Content Constraints

Constraint	Value
Column type	Dimension
Feature level	Recommended
Allows nulls	True
Data type	JSON
Value format	JSON Object Format

3.1.2.8. Introduced (version)

1.3

3.1.3. Allocated Resource ID

An Allocated Resource ID is an identifier assigned by the data generator which cost is being allocated to in a [Data Generator-Calculated Split Cost Allocation](#). The Allocated Resource ID is used to understand what the cost is being allocated to in [charges](#) where the data generator is allocating

costs to something other than the *charge's* [ResourceID](#), as is the case for [allocated charges](#).

3.1.3.1. Requirements

AllocatedResourceid adheres to the following requirements:

- AllocatedResourceid MUST be present in a Cost and Usage [FOCUS dataset](#) when the service provider supports data generator-calculated split cost allocation.
- AllocatedResourceid MUST be of type String.
- AllocatedResourceid MUST conform to [StringHandling](#) requirements.
- AllocatedResourceid nullability is defined as follows:
 - AllocatedResourceid MUST be null when a *charge* is not related to a data generator-calculated split cost allocation.
 - AllocatedResourceid MUST be null when a *charge* represents the unallocated portion of the origin *charge* after split cost allocation.
 - AllocatedResourceid MUST NOT be null when a *charge* represents the allocated portion of the origin *charge*.
- When AllocatedResourceid is not null, AllocatedResourceid adheres to the following additional requirements:
 - AllocatedResourceid SHOULD be a locally unique identifier within the associated Resourceid and ChargePeriod.
 - AllocatedResourceid MAY NOT be unique across Resourceid or ChargePeriod values.

3.1.3.2. Column ID

AllocatedResourceid

3.1.3.3. Display Name

Allocated Resource ID

3.1.3.4. Description

The identifier of the object to which cost is allocated in data generator-calculated split cost allocation.

3.1.3.5. Content Constraints

Constraint	Value
Column type	Dimension
Feature level	Conditional
Allows nulls	True
Data type	String
Value format	<not specified>

3.1.3.6. Introduced (version)

3.1.4. Allocated Resource Name

The Allocated Resource Name is a display name which cost is being allocated to in a [Data Generator-Calculated Split Cost Allocation](#). The Allocated Resource Name is used to understand what the cost is being allocated to in [charges](#) where the service provider is allocating costs to something other than the charge's [ResourceID](#), as is the case for [allocated charges](#).

3.1.4.1. Requirements

AllocatedResourceName adheres to the following requirements:

- AllocatedResourceName MUST be present in a Cost and Usage [FOCUS dataset](#) when the service provider supports data generator-calculated split cost allocation.
- AllocatedResourceName MUST be of type String.
- AllocatedResourceName MUST conform to [StringHandling](#) requirements.
- AllocatedResourceName nullability is defined as follows:
 - AllocatedResourceName MUST be null when [AllocatedResourceID](#) is null.
 - AllocatedResourceName MUST NOT be null when AllocatedResourceID is not null.
- AllocatedResourceName MAY duplicate AllocatedResourceID when a separate display name is not applicable.

3.1.4.2. Column ID

AllocatedResourceName

3.1.4.3. Display Name

Allocated Resource Name

3.1.4.4. Description

The display name of the object to which cost is allocated in data generator-calculated split cost allocation.

3.1.4.5. Content Constraints

Constraint	Value
Column type	Dimension
Feature level	Conditional
Allows nulls	True
Data type	String
Value format	<not specified>

3.1.4.6. Introduced (version)

1.3

3.1.5. Allocated Tags

The Allocated Tags column represents the set of [tags](#) assigned to [tag sources](#) which are specifically applicable to [allocated charges](#) resulting from a data generator-calculated split cost allocation.

3.1.5.1. Requirements

AllocatedTags adheres to the following requirements:

- AllocatedTags MUST be present in a Cost and Usage [FOCUS dataset](#) when the service provider supports [Data Generator-Calculated Split Cost Allocation](#).
- AllocatedTags MUST conform to [KeyValueFormat](#) requirements.
- AllocatedTags nullability is defined as follows:
 - AllocatedTags MUST be null when a *charge* is not related to a data generator-calculated split cost allocation.
 - AllocatedTags MAY be null in all other cases.
- When AllocatedTags is not null, AllocatedTags adheres to the following additional requirements:
 - AllocatedTags MUST NOT include resource tags already present in [Tags](#).
 - AllocatedTags MUST include all applicable user-defined and data generator-defined tags for the [AllocatedResourceId](#).
 - Tag keys that do not support corresponding values MUST have a corresponding true (boolean) value set.
 - Data generator MUST NOT alter tag values unless applying true (boolean) to valueless tags.
- Data generator-defined tags adhere to the following additional requirements:
 - Data generator-defined tag keys MUST be prefixed with a predetermined, data generator-specified tag key prefix that is unique to each corresponding provider-specified tag scheme.
 - Data generator SHOULD publish all data generator-specified tag key prefixes within their respective documentation.
- User-defined tags adhere to the following additional requirements:
 - Data generator MUST prefix all user-defined tags scheme with a predetermined, data generator-specified tag key prefix that is unique to each corresponding user-defined tag scheme when the data generator has more than one user-defined tag scheme.

3.1.5.2. Data Generator-Defined vs. User-Defined Tags

This example illustrates various tags produced from multiple user-defined and data generator-defined tag schemes. The first two tags illustrate examples from two different, user-defined tag schemes. The second tag is produced from a valueless, user-defined tag scheme, so the data generator also applies true as its default value.

The last two tags illustrate examples from two different, data generator-defined tag schemes.

```
{
  "userDefinedTagScheme1/foo": "bar",
  "userDefinedTagScheme2/foo": true,
  "providerDefinedTagScheme1/foo": "bar",
  "providerDefinedTagScheme2/foo": "bar"
}
```

3.1.5.3. Column ID

AllocatedTags

3.1.5.4. Display Name

Allocated Tags

3.1.5.5. Description

A set of tags assigned to tag sources that are applicable to *allocated charges* in data generator-calculated split cost allocation.

3.1.5.6. Content Constraints

Constraint	Value
Column type	Dimension
Feature level	Conditional
Allows nulls	True
Data type	JSON
Value format	Key-Value Format

3.1.5.7. Introduced (version)

1.3

3.1.6. Availability Zone

An [availability zone](#) is a host-provider-assigned identifier for a physically separated and isolated area within a Region that provides high availability and fault tolerance. Availability Zone is commonly used for scenarios like analyzing cross-zone data transfer usage and the corresponding cost based on where [resources](#) are deployed.

3.1.6.1. Requirements

AvailabilityZone adheres to the following requirements:

- AvailabilityZone is RECOMMENDED to be present in a Cost and Usage [FOCUS dataset](#) when the host provider supports deploying resources or services within an *availability zone*.
- AvailabilityZone MUST be of type String.
- AvailabilityZone MUST conform to [StringHandling](#) requirements.
- AvailabilityZone MUST be null when a [charge](#) is not specific to an *availability zone*.

3.1.6.2. Column ID

AvailabilityZone

3.1.6.3. Display Name

Availability Zone

3.1.6.4. Description

A host-provider-assigned identifier for a physically separated and isolated area within a Region that provides high availability and fault tolerance.

3.1.6.5. Content constraints

Constraint	Value
Column type	Dimension
Feature level	Recommended
Allows nulls	True
Data type	String
Value format	<not specified>

3.1.6.6. Introduced (version)

0.5

3.1.7. Billed Cost

The [billed cost](#) represents a [charge](#) serving as the basis for invoicing, inclusive of the impacts of all reduced rates and discounts while excluding the [amortization](#) of relevant purchases (one-time or recurring) paid to cover future eligible [charges](#). This cost is denominated in the [Billing Currency](#). The Billed Cost is commonly used to perform FinOps capabilities that require cash-basis accounting such as cost allocation, budgeting, and invoice reconciliation.

3.1.7.1. Requirements

BilledCost adheres to the following requirements:

- BilledCost MUST be present in a Cost and Usage [FOCUS dataset](#).
- BilledCost MUST be of type Decimal.
- BilledCost MUST conform to [NumericFormat](#) requirements.
- BilledCost MUST NOT be null.
- BilledCost MUST be a valid decimal value.
- BilledCost MUST be 0 for [charges](#) where payments are received by a third party (e.g., marketplace transactions).

- BilledCost MUST be denominated in the BillingCurrency.
- The sum of the BilledCost for a given [InvoiceId](#) MUST match the sum of the payable amount provided in the corresponding invoice with the same id generated by the [InvoiceIssuerName](#).

3.1.7.2. Column ID

BilledCost

3.1.7.3. Display Name

Billed Cost

3.1.7.4. Description

A *charge* serving as the basis for invoicing, inclusive of all reduced rates and discounts while excluding the *amortization* of upfront *charges* (one-time or recurring).

3.1.7.5. Content constraints

Constraint	Value
Column type	Metric
Feature level	Mandatory
Allows nulls	False
Data type	Decimal
Value format	Numeric Format
Number range	Any valid decimal value

3.1.7.6. Introduced (version)

0.5

3.1.8. Billing Account ID

A Billing Account ID is an invoice-issuer-assigned identifier for a [billing account](#). *Billing accounts* are commonly used for scenarios like grouping based on organizational constructs, invoice reconciliation and cost allocation strategies.

3.1.8.1. Requirements

BillingAccountId adheres to the following requirements:

- BillingAccountId MUST be present in a Cost and Usage [FOCUS dataset](#).
- BillingAccountId MUST be of type String.

- BillingAccountId MUST conform to [StringHandling](#) requirements.
- BillingAccountId MUST NOT be null.
- BillingAccountId MUST be a unique identifier within an invoice issuer.
- BillingAccountId SHOULD be a fully-qualified identifier.

See [Appendix: Grouping constructs for resources or services](#) for details and examples of the different grouping constructs supported by FOCUS.

3.1.8.2. Column ID

BillingAccountId

3.1.8.3. Display Name

Billing Account ID

3.1.8.4. Description

The identifier assigned to a *billing account* by the invoice issuer.

3.1.8.5. Content constraints

Constraint	Value
Column type	Dimension
Feature level	Mandatory
Allows nulls	False
Data type	String
Value format	<not specified>

3.1.8.6. Introduced (version)

0.5

3.1.9. Billing Account Name

A Billing Account Name is a display name assigned to a [billing account](#). *Billing accounts* are commonly used for scenarios like grouping based on organizational constructs, invoice reconciliation and cost allocation strategies.

3.1.9.1. Requirements

BillingAccountName adheres to the following requirements:

- BillingAccountName MUST be present in a Cost and Usage [FOCUS dataset](#).
- BillingAccountName MUST be of type String.

- BillingAccountName MUST conform to [StringHandling](#) requirements.
- BillingAccountName MUST NOT be null when the invoice issuer supports assigning a display name for the *billing account*.

See [Appendix: Grouping constructs for resources or services](#) for details and examples of the different grouping constructs supported by FOCUS.

3.1.9.2. Column ID

BillingAccountName

3.1.9.3. Display Name

Billing Account Name

3.1.9.4. Description

The display name assigned to a *billing account*.

3.1.9.5. Content constraints

Constraint	Value
Column type	Dimension
Feature level	Mandatory
Allows nulls	True
Data type	String
Value format	<not specified>

3.1.9.6. Introduced (version)

0.5

3.1.10. Billing Account Type

Billing Account Type is an invoice-issuer-assigned name to identify the type of *billing account*. Billing Account Type is a readable display name and not a code. Billing Account Type is commonly used for scenarios like mapping FOCUS and provider constructs, summarizing costs across providers, or invoicing and chargeback.

3.1.10.1. Requirements

BillingAccountType adheres to the following requirements:

- BillingAccountType MUST be present in a Cost and Usage [FOCUS dataset](#) when the invoice issuer supports more than one possible BillingAccountType value.

- BillingAccountType MUST be of type String.
- BillingAccountType MUST conform to [StringHandling](#) requirements.
- BillingAccountType nullability is defined as follows:
 - BillingAccountType MUST be null when [BillingAccountId](#) is null.
 - BillingAccountType MUST NOT be null when BillingAccountId is not null.
- BillingAccountType MUST be a consistent, readable display value.

3.1.10.2. Column ID

BillingAccountType

3.1.10.3. Display Name

Billing Account Type

3.1.10.4. Description

An invoice-issuer-assigned name to identify the type of *billing account*.

3.1.10.5. Content Constraints

Constraint	Value
Column type	Dimension
Feature level	Conditional
Allows nulls	False
Data type	String
Value format	<not specified>

3.1.10.6. Introduced (version)

1.2

3.1.11. Billing Currency

[Billing currency](#) is an identifier that represents the currency that a [charge](#) for [resources](#) or [services](#) was billed in. Billing Currency is commonly used in scenarios where costs need to be grouped or aggregated.

3.1.11.1. Requirements

BillingCurrency adheres to the following requirements:

- BillingCurrency MUST be present in a Cost and Usage [FOCUS dataset](#).
- BillingCurrency MUST be of type String.
- BillingCurrency MUST conform to [StringHandling](#) requirements.

- BillingCurrency MUST conform to [CurrencyFormat](#) requirements.
- BillingCurrency MUST NOT be null.
- BillingCurrency MUST match the currency used in the invoice generated by the invoice issuer.
- BillingCurrency MUST be expressed in [national currency](#) (e.g., USD, EUR).

3.1.11.2. Column ID

BillingCurrency

3.1.11.3. Display Name

Billing Currency

3.1.11.4. Description

Represents the currency that a *charge* was billed in.

3.1.11.5. Content Constraints

Constraint	Value
Column type	Dimension
Feature level	Mandatory
Allows nulls	False
Data type	String
Value format	Currency Format

3.1.11.6. Introduced (version)

0.5

3.1.12. Billing Period End

Billing Period End represents the [exclusive end bound](#) of a [billing period](#). For example, a time period where [Billing Period Start](#) is '2024-01-01T00:00:00Z' and Billing Period End is '2024-02-01T00:00:00Z' includes [charges](#) for January since Billing Period Start represents the [inclusive start bound](#), but does not include *charges* for February since Billing Period End represents the [exclusive end bound](#).

3.1.12.1. Requirements

BillingPeriodEnd adheres to the following requirements:

- BillingPeriodEnd MUST be present in a Cost and Usage [FOCUS dataset](#).
- BillingPeriodEnd MUST be of type Date/Time.

- BillingPeriodEnd MUST conform to [DateTimeFormat](#) requirements.
- BillingPeriodEnd MUST NOT be null.
- BillingPeriodEnd MUST be the *exclusive end bound* of the *billing period*.

3.1.12.2. Column ID

BillingPeriodEnd

3.1.12.3. Display Name

Billing Period End

3.1.12.4. Description

The *exclusive end bound* of a *billing period*.

3.1.12.5. Content Constraints

Constraint	Value
Column type	Dimension
Feature level	Mandatory
Allows nulls	False
Data type	Date/Time
Value format	Date/Time Format

3.1.12.6. Introduced (version)

0.5

3.1.13. Billing Period Start

Billing Period Start represents the *inclusive start bound* of a *billing period*. For example, a time period where Billing Period Start is '2024-01-01T00:00:00Z' and [Billing Period End](#) is '2024-02-01T00:00:00Z' includes [charges](#) for January since Billing Period Start represents the *inclusive start bound*, but does not include [charges](#) for February since BillingPeriodEnd represents the *exclusive end bound*.

3.1.13.1. Requirements

BillingPeriodStart adheres to the following requirements:

- BillingPeriodStart MUST be present in a Cost and Usage [FOCUS dataset](#).
- BillingPeriodStart MUST be of type Date/Time.
- BillingPeriodStart MUST conform to [DateTimeFormat](#) requirements.
- BillingPeriodStart MUST NOT be null.

- BillingPeriodStart MUST be the *inclusive start bound* of the *billing period*.

3.1.13.2. Column ID

BillingPeriodStart

3.1.13.3. Display Name

Billing Period Start

3.1.13.4. Description

The *inclusive start bound* of a *billing period*.

3.1.13.5. Content Constraints

Constraint	Value
Column type	Dimension
Feature level	Mandatory
Allows nulls	False
Data type	Date/Time
Value format	Date/Time Format

3.1.13.6. Introduced (version)

0.5

3.1.14. Capacity Reservation ID

A Capacity Reservation ID is the identifier assigned to a [capacity reservation](#) by the service provider. Capacity Reservation ID is commonly used for scenarios to allocate [charges](#) for capacity reservation usage.

3.1.14.1. Requirements

CapacityReservationId adheres to the following requirements:

- CapacityReservationId MUST be present in a Cost and Usage [FOCUS dataset](#) when the service provider supports *capacity reservations*.
- CapacityReservationId MUST be of type String.
- CapacityReservationId MUST conform to [StringHandling](#) requirements.
- CapacityReservationId nullability is defined as follows:
 - CapacityReservationId MUST be null when a *charge* is not related to a *capacity*

reservation.

- CapacityReservationId MUST NOT be null when a *charge* represents the unused portion of a *capacity reservation*.
- CapacityReservationId SHOULD NOT be null when a *charge* is related to a capacity reservation.
- When CapacityReservationId is not null, CapacityReservationId adheres to the following additional requirements:
 - CapacityReservationId MUST be a unique identifier within the service provider.
 - CapacityReservationId SHOULD be a fully-qualified identifier.

3.1.14.2. Column ID

CapacityReservationId

3.1.14.3. Display Name

Capacity Reservation ID

3.1.14.4. Description

The identifier assigned to a *capacity reservation* by the service provider.

3.1.14.5. Content constraints

Constraint	Value
Column type	Dimension
Feature level	Conditional
Allows nulls	True
Data type	String
Value format	<not specified>

3.1.14.6. Introduced (version)

1.1

3.1.15. Capacity Reservation Status

Capacity Reservation Status indicates whether the [charge](#) represents either the consumption of the [capacity reservation](#) identified in the CapacityReservationId column or when the *capacity reservation* is unused.

3.1.15.1. Requirements

CapacityReservationStatus adheres to the following requirements:

- CapacityReservationStatus MUST be present in a Cost and Usage [FOCUS dataset](#) when the service provider supports *capacity reservations*.
- CapacityReservationStatus MUST be of type String.
- CapacityReservationStatus nullability is defined as follows:
 - CapacityReservationStatus MUST be null when CapacityReservationId is null.
 - CapacityReservationStatus MUST NOT be null when CapacityReservationId is not null and [ChargeCategory](#) is "Usage".
- When CapacityReservationStatus is not null, CapacityReservationStatus adheres to the following additional requirements:
 - CapacityReservationStatus MUST be one of the allowed values.
 - CapacityReservationStatus MUST be "Unused" when the *charge* represents the unused portion of a *capacity reservation*.
 - CapacityReservationStatus MUST be "Used" when the *charge* represents the used portion of a *capacity reservation*.

3.1.15.2. Column ID

CapacityReservationStatus

3.1.15.3. Display Name

Capacity Reservation Status

3.1.15.4. Description

Indicates whether the *charge* represents either the consumption of a *capacity reservation* or when a *capacity reservation* is unused.

3.1.15.5. Content constraints

Constraint	Value
Column type	Dimension
Feature level	Conditional
Allows nulls	True
Data type	String
Value format	Allowed Values

Allowed values:

Value	Description
Used	<i>Charges</i> that utilized a specific amount of a <i>capacity reservation</i> .
Unused	<i>Charges</i> that represent the unused portion of a <i>capacity reservation</i> .

3.1.15.6. Introduced (version)

3.1.16. Charge Category

Charge Category represents the highest-level classification of a [charge](#) based on the nature of how it is billed. Charge Category is commonly used to identify and distinguish between types of [charges](#) that may require different handling.

3.1.16.1. Requirements

ChargeCategory adheres to the following requirements:

- ChargeCategory MUST be present in a Cost and Usage [FOCUS dataset](#).
- ChargeCategory MUST be of type String.
- ChargeCategory MUST NOT be null.
- ChargeCategory MUST be one of the allowed values.

3.1.16.2. Column ID

ChargeCategory

3.1.16.3. Display Name

Charge Category

3.1.16.4. Description

Represents the highest-level classification of a *charge* based on the nature of how it is billed.

3.1.16.5. Content Constraints

Constraint	Value
Column type	Dimension
Feature level	Mandatory
Allows nulls	False
Data type	String
Value format	Allowed values

Allowed values:

Value	Description
Usage	Positive or negative <i>charges</i> based on the quantity of a service or resource that was consumed over a given period of time including refunds.
Purchase	Positive or negative <i>charges</i> for the acquisition of a service or resource bought upfront or on a recurring basis including refunds.

Value	Description
Tax	Positive or negative applicable taxes that are levied by the relevant authorities including refunds. Tax <i>charges</i> may vary depending on factors such as the location, jurisdiction, and local or federal regulations.
Credit	Positive or negative <i>charges</i> granted by the service provider for various scenarios e.g promotional credits or corrections to promotional credits.
Adjustment	Positive or negative <i>charges</i> the service provider applies that do not fall into other category values.

3.1.16.6. Introduced (version)

0.5

3.1.17. Charge Class

Charge Class indicates whether the [row](#) represents a correction to a previously invoiced [billing period](#). Charge Class is commonly used to differentiate [corrections](#) from regularly incurred [charges](#).

3.1.17.1. Requirements

ChargeClass adheres to the following requirements:

- ChargeClass MUST be present in a Cost and Usage [FOCUS dataset](#).
- ChargeClass MUST be of type String.
- ChargeClass nullability is defined as follows:
 - ChargeClass MUST be null when the *row* does not represent a correction or when it represents a correction within the current *billing period*.
 - ChargeClass MUST NOT be null when the *row* represents a correction to a previously invoiced *billing period*.
- ChargeClass MUST be "Correction" when ChargeClass is not null.

3.1.17.2. Column ID

ChargeClass

3.1.17.3. Display Name

Charge Class

3.1.17.4. Description

Indicates whether the *row* represents a correction to a previously invoiced *billing period*.

3.1.17.5. Content Constraints

Constraint	Value
Column type	Dimension
Feature level	Mandatory
Allows nulls	True
Data type	String
Value format	Allowed values

Allowed values:

Value	Description
Correction	Correction to a previously invoiced <i>billing period</i> (e.g., refunds and credit modifications).

3.1.17.6. Introduced (version)

1.0

3.1.18. Charge Description

A Charge Description provides a high-level context of a [row](#) without requiring additional discovery. This column is a self-contained summary of the [charge's](#) purpose and price. It typically covers a select group of corresponding details across a billing dataset or provides information not otherwise available.

3.1.18.1. Requirements

ChargeDescription adheres to the following requirements:

- ChargeDescription MUST be present in a Cost and Usage [FOCUS dataset](#).
- ChargeDescription MUST be of type String.
- ChargeDescription MUST conform to [StringHandling](#) requirements.
- ChargeDescription SHOULD NOT be null.
- ChargeDescription maximum length SHOULD be provided in the corresponding FOCUS Metadata Schema.

3.1.18.2. Column ID

ChargeDescription

3.1.18.3. Display Name

Charge Description

3.1.18.4. Description

Self-contained summary of the *charge's* purpose and price.

3.1.18.5. Content Constraints

Constraint	Value
Column type	Dimension
Feature level	Mandatory
Allows nulls	True
Data type	String
Value format	<not specified>

3.1.18.6. Introduced (version)

1.0-preview

3.1.19. Charge Frequency

Charge Frequency indicates how often a [charge](#) will occur. Along with the [charge period](#) related columns, the Charge Frequency is commonly used to understand recurrence periods (e.g., monthly, yearly), forecast upcoming *charges*, and differentiate between one-time and recurring fees for purchases.

3.1.19.1. Requirements

ChargeFrequency adheres to the following requirements:

- ChargeFrequency is RECOMMENDED to be present in a Cost and Usage [FOCUS dataset](#).
- ChargeFrequency MUST be of type String.
- ChargeFrequency MUST NOT be null.
- ChargeFrequency MUST be one of the allowed values.
- ChargeFrequency MUST NOT be "Usage-Based" when [ChargeCategory](#) is "Purchase".

3.1.19.2. Column ID

ChargeFrequency

3.1.19.3. Display Name

Charge Frequency

3.1.19.4. Description

Indicates how often a *charge* will occur.

3.1.19.5. Content Constraints

Constraint	Value
Column type	Dimension
Feature level	Recommended
Allows nulls	False
Data type	String
Value format	Allowed values

Allowed values:

Value	Description
One-Time	<i>Charges</i> that only happen once and will not repeat. One-time <i>charges</i> are typically recorded on the hour or day when the cost was incurred.
Recurring	<i>Charges</i> that repeat on a periodic cadence (e.g., weekly, monthly) regardless of whether the product or service was used. Recurring <i>charges</i> typically happen on the same day or point within every period. The charge date does not change based on how or when the <i>service</i> is used.
Usage-Based	<i>Charges</i> that repeat every time the <i>service</i> is used. Usage-based <i>charges</i> are typically recorded hourly or daily, based on the granularity of the cost data for the period when the <i>service</i> was used (referred to as <i>charge period</i>). Usage-based <i>charges</i> are not recorded when the <i>service</i> is not used.

3.1.19.6. Introduced (version)

1.0-preview

3.1.20. Charge Period End

Charge Period End represents the [exclusive end bound](#) of a [charge period](#). For example, a time period where [Charge Period Start](#) is '2024-01-01T00:00:00Z' and Charge Period End is '2024-01-02T00:00:00Z' includes [charges](#) for January 1 since Charge Period Start represents the [inclusive start bound](#), but does not include *charges* for January 2 since Charge Period End represents the [exclusive end bound](#).

3.1.20.1. Requirements

ChargePeriodEnd adheres to the following requirements:

- ChargePeriodEnd MUST be present in a Cost and Usage [FOCUS dataset](#).
- ChargePeriodEnd MUST be of type Date/Time.
- ChargePeriodEnd MUST conform to [DateTimeFormat](#) requirements.
- ChargePeriodEnd MUST NOT be null.
- ChargePeriodEnd MUST be the *exclusive end bound* of the effective period of the *charge*.

3.1.20.2. Column ID

ChargePeriodEnd

3.1.20.3. Display Name

Charge Period End

3.1.20.4. Description

The *exclusive end bound* of a *charge period*.

3.1.20.5. Content constraints

Constraint	Value
Column type	Dimension
Feature level	Mandatory
Allows nulls	False
Data type	Date/Time
Value format	Date/Time Format

3.1.20.6. Introduced (version)

0.5

3.1.21. Charge Period Start

Charge Period Start represents the *inclusive start bound* of a *charge period*. For example, a time period where Charge Period Start is '2024-01-01T00:00:00Z' and [Charge Period End](#) is '2024-01-02T00:00:00Z' includes [charges](#) for January 1 since Charge Period Start represents the *inclusive start bound*, but does not include *charges* for January 2 since Charge Period End represents the *exclusive end bound*.

3.1.21.1. Requirements

ChargePeriodStart adheres to the following requirements:

- ChargePeriodStart MUST be present in a Cost and Usage [FOCUS dataset](#).
- ChargePeriodStart MUST be of type Date/Time.
- ChargePeriodStart MUST conform to [DateTimeFormat](#) requirements.
- ChargePeriodStart MUST NOT be null.
- ChargePeriodStart MUST be the *inclusive start bound* of the effective period of the *charge*.

3.1.21.2. Column ID

ChargePeriodStart

3.1.21.3. Display Name

Charge Period Start

3.1.21.4. Description

The *inclusive start bound* of a *charge period*.

3.1.21.5. Content constraints

Constraint	Value
Column type	Dimension
Feature level	Mandatory
Allows nulls	False
Data type	Date/Time
Value format	Date/Time Format

3.1.21.6. Introduced (version)

0.5

3.1.22. Commitment Discount Category

Commitment Discount Category indicates whether the [commitment discount](#) identified in the `CommitmentDiscountId` column is based on usage quantity or cost (aka "spend"). The `CommitmentDiscountCategory` column is only applicable to *commitment discounts* and not [negotiated discounts](#).

3.1.22.1. Requirements

`CommitmentDiscountCategory` adheres to the following requirements:

- `CommitmentDiscountCategory` MUST be present in a Cost and Usage [FOCUS dataset](#) when the service provider supports *commitment discounts*.
- `CommitmentDiscountCategory` MUST be of type String.
- `CommitmentDiscountCategory` nullability is defined as follows:
 - `CommitmentDiscountCategory` MUST be null when [CommitmentDiscountId](#) is null.
 - `CommitmentDiscountCategory` MUST NOT be null when `CommitmentDiscountId` is not null.
- `CommitmentDiscountCategory` MUST be one of the allowed values.

3.1.22.2. Column ID

`CommitmentDiscountCategory`

3.1.22.3. Display Name

Commitment Discount Category

3.1.22.4. Description

Indicates whether the *commitment discount* identified in the CommitmentDiscountId column is based on usage quantity or cost (aka "spend").

3.1.22.5. Content constraints

Constraint	Value
Column type	Dimension
Feature level	Conditional
Allows nulls	True
Data type	String
Value format	Allowed Values

Allowed values:

Value	Description
Spend	Commitment discounts that require a predetermined amount of spend.
Usage	Commitment discounts that require a predetermined amount of usage.

3.1.22.6. Introduced (version)

1.0-preview

3.1.23. Commitment Discount ID

A Commitment Discount ID is the identifier assigned to a [commitment discount](#) by the service provider. Commitment Discount ID is commonly used for scenarios like chargeback for *commitments* and savings per *commitment discount*. The CommitmentDiscountId column is only applicable to *commitment discounts* and not [negotiated discounts](#).

3.1.23.1. Requirements

CommitmentDiscountId adheres to the following requirements:

- CommitmentDiscountId MUST be present in a Cost and Usage [FOCUS dataset](#) when the service provider supports *commitment discounts*.
- CommitmentDiscountId MUST be of type String.
- CommitmentDiscountId MUST conform to [StringHandling](#) requirements.
- CommitmentDiscountId nullability is defined as follows:
 - CommitmentDiscountId MUST be null when a [charge](#) is not related to a *commitment discount*.

- CommitmentDiscountId MUST NOT be null when a *charge* is related to a *commitment discount*.
- When CommitmentDiscountId is not null, CommitmentDiscountId adheres to the following additional requirements:
 - CommitmentDiscountId MUST be a unique identifier within the service provider.
 - CommitmentDiscountId SHOULD be a fully-qualified identifier.

3.1.23.2. Column ID

CommitmentDiscountId

3.1.23.3. Display Name

Commitment Discount ID

3.1.23.4. Description

The identifier assigned to a *commitment discount* by the service provider.

3.1.23.5. Content constraints

Constraint	Value
Column type	Dimension
Feature level	Conditional
Allows nulls	True
Data type	String
Value format	<not specified>

3.1.23.6. Introduced (version)

1.0-preview

3.1.24. Commitment Discount Name

A Commitment Discount Name is the display name assigned to a [commitment discount](#). The CommitmentDiscountName column is only applicable to *commitment discounts* and not [negotiated discounts](#).

3.1.24.1. Requirements

CommitmentDiscountName adheres to the following requirements:

- CommitmentDiscountName MUST be present in a Cost and Usage [FOCUS dataset](#) when the service provider supports *commitment discounts*.
- CommitmentDiscountName MUST be of type String.

- CommitmentDiscountName MUST conform to [StringHandling](#) requirements.
- CommitmentDiscountName nullability is defined as follows:
 - CommitmentDiscountName MUST be null when [CommitmentDiscountId](#) is null.
 - When CommitmentDiscountId is not null, CommitmentDiscountName adheres to the following additional requirements:
 - CommitmentDiscountName MUST NOT be null when a display name can be assigned to a *commitment discount*.
 - CommitmentDiscountName MAY be null when a display name cannot be assigned to a *commitment discount*.

3.1.24.2. Column ID

CommitmentDiscountName

3.1.24.3. Display Name

Commitment Discount Name

3.1.24.4. Description

The display name assigned to a *commitment discount*.

3.1.24.5. Content constraints

Constraint	Value
Column type	Dimension
Feature level	Conditional
Allows nulls	True
Data type	String
Value format	<not specified>

3.1.24.6. Introduced (version)

1.0-preview

3.1.25. Commitment Discount Quantity

Commitment Discount Quantity is the amount of a [commitment discount](#) purchased or accounted for in *commitment discount* related [rows](#) that is denominated in [Commitment Discount Units](#). The aggregated Commitment Discount Quantity across purchase records, pertaining to a particular [Commitment Discount ID](#) during its commitment [period](#), represents the total Commitment Discount Units acquired with that commitment discount. For committed usage, the Commitment Discount Quantity is either the number of Commitment Discount Units consumed by a *row* that is covered by a *commitment discount* or is the unused portion of a *commitment discount* over a [charge period](#). Commitment Discount Quantity is commonly used in *commitment discount* analysis and optimization use cases and only applies to *commitment discounts*, not [negotiated discounts](#).

When [CommitmentDiscountCategory](#) is "Usage" (usage-based *commitment discounts*), the Commitment Discount Quantity reflects the predefined amount of usage purchased or consumed. If [commitment discount flexibility](#) is applicable, this value may be further transformed based on additional, service-provider-specific requirements. When CommitmentDiscountCategory is "Spend" (spend-based *commitment discounts*), the Commitment Discount Quantity reflects the predefined amount of spend purchased or consumed. See [Appendix: Commitment Discount Flexibility](#) for more details around *commitment discount flexibility*.

3.1.25.1. Requirements

CommitmentDiscountQuantity adheres to the following requirements:

- CommitmentDiscountQuantity MUST be present in a Cost and Usage [FOCUS dataset](#) when the service provider supports *commitment discounts*.
- CommitmentDiscountQuantity MUST be of type Decimal.
- CommitmentDiscountQuantity MUST conform to [NumericFormat](#) requirements.
- CommitmentDiscountQuantity nullability is defined as follows:
 - CommitmentDiscountQuantity MUST be null when [SkuPriceld](#) is null.
 - When ChargeCategory is "Usage" or "Purchase" and CommitmentDiscountId is not null, CommitmentDiscountQuantity adheres to the following additional requirements:
 - CommitmentDiscountQuantity MUST NOT be null when [ChargeClass](#) is not "Correction".
 - CommitmentDiscountQuantity MAY be null when ChargeClass is "Correction".
 - CommitmentDiscountQuantity MUST be null in all other cases.
- CommitmentDiscountQuantity MUST be a valid decimal value when not null.
- When CommitmentDiscountQuantity is not null and ChargeCategory is "Purchase", CommitmentDiscountQuantity adheres to the following additional requirements:
 - CommitmentDiscountQuantity MUST be the quantity of CommitmentDiscountUnit, paid fully or partially upfront, that is eligible for consumption over the *commitment discount's term* when [ChargeFrequency](#) is "One-Time".
 - CommitmentDiscountQuantity MUST be the quantity of CommitmentDiscountUnit that is eligible for consumption for each *charge period* that corresponds with the purchase when ChargeFrequency is "Recurring".
- When CommitmentDiscountQuantity is not null and ChargeCategory is "Usage", CommitmentDiscountQuantity adheres to the following additional requirements:
 - CommitmentDiscountQuantity MUST be the metered quantity of CommitmentDiscountUnit that is consumed in a given *charge period* when [CommitmentDiscountStatus](#) is "Used".
 - CommitmentDiscountQuantity MUST be the remaining, unused quantity of CommitmentDiscountUnit in a given *charge period* when CommitmentDiscountStatus is "Unused".

3.1.25.2. Column ID

CommitmentDiscountQuantity

3.1.25.3. Display Name

Commitment Discount Quantity

3.1.25.4. Description

The amount of a *commitment discount* purchased or accounted for in *commitment discount* related rows that is denominated in Commitment Discount Units.

3.1.25.5. Usability Constraints

Aggregation: When aggregating Commitment Discount Quantity for commitment utilization calculations, it's important to exclude [commitment discount](#) purchases (i.e. when Charge Category is "Purchase") that are paid to cover future eligible [charges](#) (e.g., *commitment discount*). Otherwise, when accounting for all upfront or accrued purchases, it's important to exclude *commitment discount* usage (i.e. when Charge Category is "Usage"). This exclusion helps prevent double counting of these quantities in the aggregation.

3.1.25.6. Content constraints

Constraint	Value
Column type	Metric
Feature level	Conditional
Allows nulls	True
Data type	Decimal
Value format	Numeric Format
Number range	Any valid decimal value

3.1.25.7. Introduced (version)

1.1

3.1.26. Commitment Discount Status

Commitment Discount Status indicates whether the [charge](#) corresponds with the consumption of a [commitment discount](#) identified in the CommitmentDiscountId column or the unused portion of the committed amount. The CommitmentDiscountStatus column is only applicable to *commitment discounts* and not [negotiated discounts](#).

3.1.26.1. Requirements

CommitmentDiscountStatus adheres to the following requirements:

- CommitmentDiscountStatus MUST be present in a Cost and Usage [FOCUS dataset](#) when the service provider supports *commitment discounts*.
- CommitmentDiscountStatus MUST be of type String.
- CommitmentDiscountStatus nullability is defined as follows:
 - CommitmentDiscountStatus MUST be null when [CommitmentDiscountId](#) is null.
 - CommitmentDiscountStatus MUST NOT be null when CommitmentDiscountId is not null and [Charge Category](#) is "Usage".
- CommitmentDiscountStatus MUST be one of the allowed values.

3.1.26.2. Column ID

CommitmentDiscountStatus

3.1.26.3. Display name

Commitment Discount Status

3.1.26.4. Description

Indicates whether the *charge* corresponds with the consumption of a *commitment discount* or the unused portion of the committed amount.

3.1.26.5. Content constraints

Constraint	Value
Column type	Dimension
Feature level	Conditional
Allows nulls	True
Data type	String
Value format	Allowed Values

Allowed values:

Value	Description
Used	<i>Charges</i> that utilized a specific amount of a commitment discount.
Unused	<i>Charges</i> that represent the unused portion of the commitment discount.

3.1.26.6. Introduced (version)

1.0

3.1.27. Commitment Discount Type

Commitment Discount Type is a service-provider-assigned name to identify the type of [commitment discount](#) applied to the [row](#). The CommitmentDiscountType column is only applicable to *commitment discounts* and not [negotiated discounts](#).

3.1.27.1. Requirements

CommitmentDiscountType adheres to the following requirements:

- CommitmentDiscountType MUST be present in a Cost and Usage [FOCUS dataset](#) when the service provider supports *commitment discounts*.

- CommitmentDiscountType MUST be of type String.
- CommitmentDiscountType MUST conform to [StringHandling](#) requirements.
- CommitmentDiscountType nullability is defined as follows:
 - CommitmentDiscountType MUST be null when [CommitmentDiscountId](#) is null.
 - CommitmentDiscountType MUST NOT be null when CommitmentDiscountId is not null.

3.1.27.2. Column ID

CommitmentDiscountType

3.1.27.3. Display Name

Commitment Discount Type

3.1.27.4. Description

A service-provider-assigned identifier for the type of *commitment discount* applied to the *row*.

3.1.27.5. Content Constraints

Constraint	Value
Column type	Dimension
Feature level	Conditional
Allows nulls	True
Data type	String
Value format	<not specified>

3.1.27.6. Introduced (version)

1.0-preview

3.1.28. Commitment Discount Unit

Commitment Discount Unit represents the service-provider-specified measurement unit indicating how a service provider measures the [Commitment Discount Quantity](#) of a *commitment discount*. The CommitmentDiscountUnit column is only applicable to *commitment discounts* and not [negotiated discounts](#).

3.1.28.1. Requirements

CommitmentDiscountUnit adheres to the following requirements:

- CommitmentDiscountUnit MUST be present in a Cost and Usage [FOCUS dataset](#) when the service provider supports *commitment discounts*.
- CommitmentDiscountUnit MUST be of type String.

- CommitmentDiscountUnit MUST conform to [StringHandling](#) requirements.
- CommitmentDiscountUnit SHOULD conform to [UnitFormat](#) requirements.
- CommitmentDiscountUnit nullability is defined as follows:
 - CommitmentDiscountUnit MUST be null when CommitmentDiscountQuantity is null.
 - CommitmentDiscountUnit MUST NOT be null when CommitmentDiscountQuantity is not null.
- When CommitmentDiscountUnit is not null, CommitmentDiscountUnit adheres to the following additional requirements:
 - CommitmentDiscountUnit MUST remain consistent over time for a given CommitmentDiscountId.
 - CommitmentDiscountUnit MUST represent the unit used to measure the *commitment discount*.
 - When accounting for [commitment discount flexibility](#), the CommitmentDiscountUnit value SHOULD reflect this consideration.

See [Examples: Commitment Discount Flexibility](#) for more details around *commitment discount flexibility*.

3.1.28.2. Column ID

CommitmentDiscountUnit

3.1.28.3. Display Name

Commitment Discount Unit

3.1.28.4. Description

The service-provider-specified measurement unit indicating how a service provider measures the Commitment Discount Quantity of a *commitment discount*.

3.1.28.5. Content constraints

Constraint	Value
Column type	Dimension
Feature level	Conditional
Allows nulls	True
Data type	String
Value format	Unit Format

3.1.28.6. Introduced (version)

1.1

3.1.29. Consumed Quantity

The Consumed Quantity represents the volume of a metered SKU associated with a [resource](#) or [service](#) used, based on the [Consumed Unit](#). Consumed Quantity is often derived at a finer granularity or over a different time interval when compared to the [Pricing Quantity](#) (complementary to [Pricing Unit](#)) and focuses on *resource* and *service* consumption, not pricing and cost.

3.1.29.1. Requirements

ConsumedQuantity adheres to the following requirements:

- ConsumedQuantity MUST be present in a Cost and Usage [FOCUS dataset](#) when the service provider supports the measurement of usage.
- ConsumedQuantity MUST be of type Decimal.
- ConsumedQuantity MUST conform to [NumericFormat](#) requirements.
- ConsumedQuantity nullability is defined as follows:
 - ConsumedQuantity MUST be null when [SkuPriceId](#) is null.
 - ConsumedQuantity MUST be null when [ChargeCategory](#) is not "Usage", or when ChargeCategory is "Usage" and [CommitmentDiscountStatus](#) is "Unused".
 - When ChargeCategory is "Usage" and CommitmentDiscountStatus is not "Unused", ConsumedQuantity adheres to the following additional requirements:
 - ConsumedQuantity MUST NOT be null when [ChargeClass](#) is not "Correction".
 - ConsumedQuantity MAY be null when ChargeClass is "Correction".
- ConsumedQuantity MUST be a valid decimal value when not null.

3.1.29.2. Column ID

ConsumedQuantity

3.1.29.3. Display Name

Consumed Quantity

3.1.29.4. Description

The volume of a metered SKU associated with a *resource* or *service* used, based on the Consumed Unit.

3.1.29.5. Content constraints

Constraint	Value
Column type	Metric
Feature level	Conditional
Allows nulls	True
Data type	Decimal
Value format	Numeric Format
Number range	Any valid decimal value

3.1.29.6. Introduced (version)

1.0

3.1.30. Consumed Unit

The Consumed Unit represents a service-provider-specified measurement unit indicating how a service provider measures usage of a metered SKU associated with a [resource](#) or [service](#). Consumed Unit complements the [Consumed Quantity](#) metric. It is often listed at a finer granularity or over a different time interval when compared to [Pricing Unit](#) (complementary to [Pricing Quantity](#)), and focuses on *resource* and *service* consumption, not pricing and cost.

3.1.30.1. Requirements

ConsumedUnit adheres to the following requirements:

- ConsumedUnit MUST be present in a Cost and Usage [FOCUS dataset](#) when the service provider supports the measurement of usage.
- ConsumedUnit MUST be of type String.
- ConsumedUnit MUST conform to [StringHandling](#) requirements.
- ConsumedUnit SHOULD conform to [UnitFormat](#) requirements.
- ConsumedUnit nullability is defined as follows:
 - ConsumedUnit MUST be null when ConsumedQuantity is null.
 - ConsumedUnit MUST NOT be null when ConsumedQuantity is not null.

3.1.30.2. Column ID

ConsumedUnit

3.1.30.3. Display Name

Consumed Unit

3.1.30.4. Description

Service-provider-specified measurement unit indicating how a service provider measures usage of a metered SKU associated with a *resource* or *service*.

3.1.30.5. Content constraints

Constraint	Value
Column type	Dimension
Feature level	Conditional
Allows nulls	True

Constraint	Value
Data type	String
Value format	Unit Format recommended

3.1.30.6. Introduced (version)

1.0

3.1.31. Contract Applied

Contract Applied is a set of properties that associate a charge with one or more [contract commitments](#), denoted as key-value pairs in a JSON object. Contract Applied allows the practitioner to track the progress of the commitments to which they have agreed with a service provider.

The FOCUS-defined properties are:

- Contract ID: The unique identifier representing a single contract.
- Contract Commitment ID: The unique identifier representing a single contract term.
- Contract Commitment Applied Cost: The value of the charge applied to a single contract term.
- Contract Commitment Applied Quantity: The usage of the charge applied to a single contract term.
- Contract Commitment Applied Unit: The unit of measure for the usage of the charge applied to a single contract term.

In addition to these, a data generator may include one or more custom properties, also denoted as key-value pairs.

3.1.31.1. Requirements

3.1.31.1.1. Column Requirements

The ContractApplied column adheres to the following requirements:

- ContractApplied MUST be present in a Cost and Usage [FOCUS dataset](#) when the service provider supports *contract commitments*.
- ContractApplied MUST conform to [JsonObjectFormat](#) requirements.
- ContractApplied MUST NOT be null when one or more *contract commitments* are applied to the *charge*.

3.1.31.1.2. Object Schema Requirements

Contract Applied consists of a valid JSON object which contains an array of key-value objects describing the one or more contract commitments applied to the charge. Each object consists of FOCUS-defined keys but can be extended to provide additional details about the contract application.

- If ContractApplied is not null, ContractApplied adheres to the following requirements:
 - ContractApplied MUST have a top-level key "Elements" which contains an array.
 - ContractApplied root object MAY contain custom objects, in addition to "Elements".
 - Each item in "Elements" MUST be an object.
 - "Elements" objects MUST conform to [KeyValueFormat](#) requirements.

- "Elements" objects MUST contain key-value pairs (contract application properties).
- Contract application property keys SHOULD conform to [PascalCase](#) format.
- "Elements" objects MUST contain four key-value pairs, representing "ContractCommitmentID", "ContractCommitmentAppliedCost", "ContractCommitmentAppliedQuantity", and "ContractCommitmentAppliedUnit".
- "Elements" objects MAY contain custom key-value pairs, representing additional datapoints provided by the data generator.
- When custom key-value pairs within "Elements" objects are present:
 - Contract application property custom key-value pairs MUST be prefixed with a consistent x_ prefix to identify them as external, custom columns and distinguish them from FOCUS columns to avoid conflicts in future releases.
 - Contract application property custom key-value pairs MUST be documented by the data generator.
 - Contract application property custom key-value pairs MUST NOT be nested.
- FOCUS-defined contract application properties adhere to the following additional requirements:
 - Contract application property key MUST match the spelling and casing specified for the FOCUS-defined property.
 - Contract application property value MUST be of the type specified for that property.
 - Contract application property MUST adhere to additional normative requirements specific to that property.
- Contract application property keys MUST begin with the string "x_" unless it is a FOCUS-defined allocation property.

3.1.31.1.3. Content Requirements

The following keys are used for contract application properties to facilitate querying data across allocations and across service providers. FOCUS-defined keys will appear in the list below, and custom keys will be prefixed with "x_" to make them easy to identify as well as prevent collisions.

Contract ID

Contract ID is a service-provider-assigned identifier for a contract describing the agreed terms between a service provider and a customer. Contracts can include commitment to a certain amount of spend or usage over an agreed period of time.

The "ContractId" property adheres to the following requirements:

- "ContractId" MUST be present in a Cost and Usage [FOCUS dataset](#) when the service provider supports *contract commitments*.
- "ContractId" MUST be of type String.
- "ContractId" MUST conform to [StringHandling](#) requirements.
- "ContractId" nullability is defined as follows:
 - "ContractId" MUST be null when a [charge](#) is not related to a *contract commitment*.
 - "ContractId" MUST NOT be null when a *charge* is related to a *contract commitment*.
- When "ContractId" is not null, "ContractId" adheres to the following additional requirements:
 - "ContractId" MUST be a unique identifier within the service provider.
 - "ContractId" SHOULD be a fully-qualified identifier.

Contract Commitment ID

A Contract Commitment ID is a service-provider-assigned identifier describing an agreement agreed between a service provider and a customer. Contracts can include commitment to a certain amount of spend or usage over an agreed period of time.

The "ContractCommitmentID" property adheres to the following requirements:

- "ContractCommitmentID" MUST be present in a Cost and Usage [FOCUS dataset](#) when the service provider supports *contract commitments*.

- "ContractCommitmentID" MUST be of type String.
- "ContractCommitmentID" MUST conform to [StringHandling](#) requirements.
- "ContractCommitmentID" nullability is defined as follows:
 - "ContractCommitmentID" MUST be null when a [charge](#) is not related to a *contract commitment*.
 - "ContractCommitmentID" MUST NOT be null when a *charge* is related to a *contract commitment*.
- When "ContractCommitmentID" is not null, "ContractCommitmentID" adheres to the following additional requirements:
 - "ContractCommitmentID" MUST be a unique identifier within the service provider.
 - "ContractCommitmentID" SHOULD be a fully-qualified identifier.
 - "ContractCommitmentID" MUST have one and only one parent "ContractID".
 - "ContractCommitmentID" MUST be equal to ResourceID when ChargeCategory is "Purchase".
 - "ContractCommitmentID" MAY be equal to "ContractID".

Contract Commitment Applied Cost

Contract Commitment Applied Cost represents the cost of the charge applied to the contract line item. Contract Commitment Applied Cost is associated with the contract line item via Contract Commitment ID. Contract Commitment Applied Cost is commonly used for monitoring the progress towards fulfilling contractual commitments that may facilitate discounts for [resources](#) or [services](#) as agreed between a service provider and a customer.

The "ContractCommitmentAppliedCost" property adheres to the following requirements:

- "ContractCommitmentAppliedCost" MUST be present in a Cost and Usage [FOCUS dataset](#) when the service provider associates the *charge's* value with one or more *contract commitments*.
- "ContractCommitmentAppliedCost" MUST be of type Decimal.
- "ContractCommitmentAppliedCost" MUST conform to [NumericFormat](#) requirements.
- "ContractCommitmentAppliedCost" nullability is defined as follows:
 - "ContractCommitmentAppliedCost" MUST NOT be null when "ContractCommitmentAppliedQuantity" is null.
 - "ContractCommitmentAppliedCost" MAY be null in all other cases.
- "ContractCommitmentAppliedCost" MUST be a valid decimal value.
- "ContractCommitmentAppliedCost" MUST be denominated in the BillingCurrency.

Contract Commitment Applied Quantity

Contract Commitment Applied Quantity represents the quantity of the charge applied to the contract line item. Contract Commitment Applied Quantity is associated with the contract line item via Contract Commitment ID. Contract Commitment Applied Quantity is commonly used for monitoring the progress towards fulfilling contractual commitments that may facilitate discounts for [resources](#) or [services](#) as agreed between a service provider and a customer.

The "ContractCommitmentAppliedQuantity" property adheres to the following requirements:

- "ContractCommitmentAppliedQuantity" MUST be present in a Cost and Usage [FOCUS dataset](#) when the service provider associates the *charge's* quantity with one or more *contract commitments*.
- "ContractCommitmentAppliedQuantity" MUST be of type Decimal.
- "ContractCommitmentAppliedQuantity" MUST conform to [NumericFormat](#) requirements.
- "ContractCommitmentAppliedQuantity" nullability is defined as follows:
 - "ContractCommitmentAppliedQuantity" MUST NOT be null when "ContractCommitmentAppliedCost" is null.
 - "ContractCommitmentAppliedQuantity" MAY be null in all other cases.
- "ContractCommitmentAppliedQuantity" MUST be a valid decimal value.
- "ContractCommitmentAppliedQuantity" MUST be denominated in the "ContractCommitmentAppliedUnit".

Contract Commitment Applied Unit

The Contract Commitment Applied Unit represents a service-provider-specified measurement unit for

the usage declared in Contract Commitment Applied Quantity. Contract Commitment Applied Unit complements the Contract Commitment Applied Quantity metric.

The "ContractCommitmentAppliedUnit" property adheres to the following requirements:

- "ContractCommitmentAppliedUnit" MUST be present in a Cost and Usage [FOCUS dataset](#) when the service provider associates the *charge's* quantity with one or more *contract commitments*.
- "ContractCommitmentAppliedUnit" MUST be of type String.
- "ContractCommitmentAppliedUnit" MUST conform to [StringHandling](#) requirements.
- "ContractCommitmentAppliedUnit" SHOULD conform to [UnitFormat](#) requirements.
- "ContractCommitmentAppliedUnit" nullability is defined as follows:
 - "ContractCommitmentAppliedUnit" MUST be null when "ContractCommitmentAppliedQuantity" is null.
 - "ContractCommitmentAppliedUnit" MUST NOT be null when "ContractCommitmentAppliedQuantity" is not null.

3.1.31.2. Overview

3.1.31.2.1. Array of Objects

The parent array is called Elements and contains one or more objects which communicate information about how an allocated record was calculated.

Key	ValueType	Required	Description
Elements	Array	True	The parent array containing one or more objects which communicate information about how contract commitments were applied to the charge.

3.1.31.2.2. Object Entries

The Elements array contains one or more objects, each of which contains the following entries:

Key	Key Type	Feature Level	Allows Nulls	Data Type
ContractID	Dimension	Conditional	False	String
ContractCommitmentID	Dimension	Conditional	False	String
ContractCommitmentAppliedCost	Dimension	Conditional	True	Numeric
ContractCommitmentAppliedQuantity	Dimension	Conditional	True	Numeric
ContractCommitmentAppliedUnit	Dimension	Conditional	True	String

3.1.31.2.3. Example

```

{
  "Elements" : [ {
    "ContractID" : "12345",
    "ContractCommitmentID" : "23456",
    "ContractCommitmentAppliedCost" : 500000.00
  }, {
    "ContractID" : "12345",
    "ContractCommitmentID" : "34567",
    "ContractCommitmentAppliedQuantity" : 10000.00,
    "ContractCommitmentAppliedUnit" : "compute_hours"
  } ]
}

```

3.1.31.2.4. JSON Type Definition

```

{
  "properties": {
    "Elements": {
      "elements": {
        "properties": {
          "ContractID": { "type": "string" },
          "ContractCommitmentID": { "type": "string" }
        },
        "optionalProperties": {
          "ContractCommitmentAppliedCost": { "type": "float64" },
          "ContractCommitmentAppliedQuantity": { "type": "float64" },
          "ContractCommitmentAppliedUnit": { "type": "float64" }
        },
        "additionalProperties": true
      }
    }
  },
  "additionalProperties": true
}

```

NOTE: The above JSON Type Definition (JTD) is an approximation of the expected contents of this column, but it should not be considered normative because it cannot accurately describe the normative requirements (above) for ContractApplied. Where there are discrepancies, deference will be given to the normative requirements. For example, [NumericFormat](#) allows for multiple numeric data types and precisions, but JTD requires both to be specified; other numeric data types and precisions allowable under NumericFormat are considered valid.

3.1.31.3. Example Scenarios

3.1.31.3.1. Scenario 1: Initial contract commitment

A single Cost and Usage charge represents the values stated on a contract and its three contract commitments agreed between a service provider and a customer:

1. 12345: Spend \$500k overall. (This is the value of the contract, and thus ContractID = ContractCommitmentID.)
2. 23456: Spend \$25k on a particular service.
3. 34567: Consume 100k compute hours on a particular resource type.

The Charge Category is denoted as Purchase, and the Contract ID, Resource ID, and Contract Commitment ID are all denoted as 12345.

```
{
  "ResourceID": "12345",
  "ChargeCategory": "Purchase",
  "BilledCost": 500000.00,
  "EffectiveCost": 0.00,
  "ContractApplied":
  {
    "Elements": [ {
      "ContractID": "12345",
      "ContractCommitmentID": "12345",
      "ContractCommitmentAppliedCost": 500000.00
    }, {
      "ContractID": "12345",
      "ContractCommitmentID": "23456",
      "ContractCommitmentAppliedCost": 25000.00
    }, {
      "ContractID": "12345",
      "ContractCommitmentID": "34567",
      "ContractCommitmentAppliedQuantity": 100000.00,
      "ContractCommitmentAppliedUnit": "compute_hours"
    } ]
  }
}
```

3.1.31.3.2. Scenario 2: Contract commitment usage with no custom columns

Assume the contract commitment as described in Scenario 1. Assume that only 50% of cost and usage gets applied to the contract commitments, per the contract terms.

A single Cost and Usage charge for myResource1 carries Effective Cost of 30 (denominated in USD) and Consumed Quantity of 1 (denominated in compute hours). The Charge Category is denoted as Usage.

This applies to the contract commitments in the following manner:

```

{
  "ResourceID": "myResource1",
  "ChargeCategory": "Usage",
  "BilledCost": 0.00,
  "EffectiveCost": 30.00,
  "ConsumedQuantity": 1,
  "ContractApplied":
  {
    "Elements": [ {
      "ContractID": "12345",
      "ContractCommitmentID": "12345",
      "ContractCommitmentAppliedCost": 15.00
    }, {
      "ContractID": "12345",
      "ContractCommitmentID": "23456",
      "ContractCommitmentAppliedCost": 15.00
    }, {
      "ContractID": "12345",
      "ContractCommitmentID": "34567",
      "ContractCommitmentAppliedQuantity": 0.50,
      "ContractCommitmentAppliedUnit": "compute_hours"
    } ]
  }
}

```

3.1.31.3.3. Scenario 3: Contract commitment usage with custom columns

The same as Scenario 2, except a custom key-value pair `x_ContractCommitmentCostBalance` is provided by the data generator. This datapoint represents the value remaining on a given contract commitment.

```

{
  "ResourceID": "myResource1",
  "ChargeCategory": "Usage",
  "BilledCost": 0.00,
  "EffectiveCost": 30.00,
  "ConsumedQuantity": 1,
  "ContractApplied":
  {
    "Elements": [ {
      "ContractID": "12345",
      "ContractCommitmentID": "12345",
      "ContractCommitmentAppliedCost": 15.00,
      "x_ContractCommitmentCostBalance": 499985.00
    }, {
      "ContractID": "12345",
      "ContractCommitmentID": "23456",
      "ContractCommitmentAppliedCost": 15.00,
      "x_ContractCommitmentCostBalance": 24985.00
    }, {
      "ContractID": "12345",
      "ContractCommitmentID": "34567",
      "ContractCommitmentAppliedQuantity": 0.50,
      "ContractCommitmentAppliedUnit": "compute_hours"
    } ]
  }
}

```

3.1.31.4. Column ID

ContractApplied

3.1.31.5. Display Name

Contract Applied

3.1.31.6. Description

A set of properties that associate a charge with one or more [contract commitments](#).

3.1.31.7. Content Constraints

Constraint	Value
Column type	Dimension and Metric
Feature level	Conditional
Allows nulls	True
Data type	JSON
Value format	JSON Object Format

3.1.31.8. Introduced (version)

1.3

3.1.32. Contracted Cost

Contracted Cost represents the cost calculated by multiplying [contracted unit price](#) and the corresponding [Pricing Quantity](#). Contracted Cost is denominated in the [Billing Currency](#) and is commonly used for calculating savings based on negotiation activities, by comparing it with [List Cost](#). If [negotiated discounts](#) are not applicable, the Contracted Cost defaults to the List Cost.

3.1.32.1. Requirements

ContractedCost adheres to the following requirements:

- ContractedCost MUST be present in a Cost and Usage [FOCUS dataset](#).
- ContractedCost MUST be of type Decimal.
- ContractedCost MUST conform to [NumericFormat](#) requirements.
- ContractedCost MUST NOT be null.
- ContractedCost MUST be a valid decimal value.
- ContractedCost MUST be denominated in the BillingCurrency.
- When [ContractedUnitPrice](#) is null, ContractedCost adheres to the following additional requirements:
 - ContractedCost of a [charge](#) calculated based on other *charges* (e.g., when the [ChargeCategory](#) is "Tax") MUST be calculated based on the ContractedCost of those

related *charges*.

- ContractedCost of a *charge* unrelated to other *charges* (e.g., when the ChargeCategory is "Credit") MUST match the [BilledCost](#).
- ContractedCost MUST equal the product of ContractedUnitPrice and PricingQuantity when ContractedUnitPrice is not null and PricingQuantity is not null.

3.1.32.2. Column ID

ContractedCost

3.1.32.3. Display Name

Contracted Cost

3.1.32.4. Description

Cost calculated by multiplying *contracted unit price* and the corresponding Pricing Quantity.

3.1.32.5. Usability Constraints

Aggregation: When aggregating Contracted Cost for savings calculations, it's important to exclude either [Charge Category](#) "Purchase" *charges* (one-time and recurring) that are paid to cover future eligible *charges* (e.g., [commitment discount](#)) or the covered [Charge Category](#) "Usage" *charges* themselves. This exclusion helps prevent double counting of these *charges* in the aggregation. Which set of *charges* to exclude depends on whether cost are aggregated on a billed basis (exclude covered *charges*) or accrual basis (exclude Purchases for future *charges*). For instance, *charges* categorized as [Charge Category](#) "Purchase" and their related [Charge Category](#) "Tax" *charges* for a Commitment Discount might be excluded from an accrual basis cost aggregation of Contracted Cost. This is because the "Usage" and "Tax" charge records provided during the commitment *period* already specify the Contracted Cost. Purchase *charges* that cover future eligible *charges* can be identified by filtering for [Charge Category](#) "Purchase" records with a [Billed Cost](#) greater than 0 and an [Effective Cost](#) equal to 0.

3.1.32.6. Content Constraints

Constraint	Value
Column type	Metric
Feature level	Mandatory
Allows nulls	False
Data type	Decimal
Value format	Numeric Format
Number range	Any valid decimal value

3.1.32.7. Introduced (version)

1.0

3.1.33. Contracted Unit Price

The Contracted Unit Price represents the agreed-upon unit price for a single [Pricing Unit](#) of the associated SKU, inclusive of [negotiated discounts](#), if present, while excluding negotiated [commitment discounts](#) or any other discounts. This price is denominated in the [Billing Currency](#). The Contracted Unit Price is commonly used for calculating savings based on negotiation activities. If negotiated discounts are not applicable, the Contracted Unit Price defaults to the [List Unit Price](#).

3.1.33.1. Requirements

ContractedUnitPrice adheres to the following requirements:

- ContractedUnitPrice MUST be present in a Cost and Usage [FOCUS dataset](#) when the service provider supports negotiated pricing concepts.
- ContractedUnitPrice adheres to the following additional requirements:
- ContractedUnitPrice MUST be of type Decimal.
- ContractedUnitPrice MUST conform to [NumericFormat](#) requirements.
- ContractedUnitPrice nullability is defined as follows:
 - ContractedUnitPrice MUST be null when [SkuPriceld](#) is null.
 - ContractedUnitPrice MUST be null when [ChargeCategory](#) is "Tax".
 - ContractedUnitPrice MUST NOT be null when [SkuPriceld](#) is not null.
 - ContractedUnitPrice MUST NOT be null when [ChargeCategory](#) is "Usage" or "Purchase" and [ChargeClass](#) is not "Correction".
 - ContractedUnitPrice MAY be null in all other cases.
- When ContractedUnitPrice is not null, ContractedUnitPrice adheres to the following additional requirements:
 - ContractedUnitPrice MUST be a non-negative decimal value.
 - ContractedUnitPrice MUST be denominated in the [BillingCurrency](#).
- [ContractedCost](#) MUST equal the product of ContractedUnitPrice and [PricingQuantity](#) when ContractedUnitPrice is not null and PricingQuantity is not null.

3.1.33.2. Column ID

ContractedUnitPrice

3.1.33.3. Display Name

Contracted Unit Price

3.1.33.4. Description

The agreed-upon unit price for a single Pricing Unit of the associated SKU, inclusive of negotiated discounts, if present, while excluding negotiated commitment discounts or any other discounts.

3.1.33.5. Usability Constraints

Aggregation: Column values should only be viewed in the context of their row and not aggregated to produce a total.

3.1.33.6. Content Constraints

Constraint	Value
Column type	Metric
Feature level	Conditional
Allows nulls	True
Data type	Decimal
Value format	Numeric Format
Number range	Any valid non-negative decimal value

3.1.33.7. Introduced (version)

1.0

3.1.34. Effective Cost

Effective Cost represents the [amortized](#) cost of the [charge](#) after applying all reduced rates, discounts, and the applicable portion of relevant, prepaid purchases (one-time or recurring) that covered this [charge](#). The [amortized](#) portion included should be proportional to the [Pricing Quantity](#) and the time granularity of the data. Since amortization breaks down and spreads the cost of a prepaid purchase, to subsequent eligible [charges](#), the Effective Cost of the original prepaid [charge](#) is set to 0. Effective Cost does not mix or "blend" costs across multiple [charges](#) of the same [service](#). This cost is denominated in the [Billing Currency](#). The Effective Cost is commonly utilized to track and analyze spending trends.

This column resolves two challenges that are faced by practitioners:

1. Practitioners need to [amortize](#) relevant purchases, such as upfront fees, throughout the [commitment](#) and distribute them to the appropriate reporting groups (e.g., [tags](#), [resources](#)).
2. Many [commitment discount](#) constructs include a recurring expense for the [commitment](#) for every [billing period](#) and must distribute this cost to the [resources](#) using the [commitment](#). This forces reconciliation between the initial [commitment row](#) per period and the actual usage [rows](#).

3.1.34.1. Requirements

EffectiveCost adheres to the following requirements:

- EffectiveCost MUST be present in a Cost and Usage [FOCUS dataset](#).
- EffectiveCost MUST be of type Decimal.
- EffectiveCost MUST conform to [NumericFormat](#) requirements.
- EffectiveCost MUST NOT be null.
- EffectiveCost MUST be a valid decimal value.
- EffectiveCost MUST be 0 when [ChargeCategory](#) is "Purchase" and the purchase is intended to cover future eligible [charges](#).
- EffectiveCost MUST be denominated in the BillingCurrency.
- The sum of EffectiveCost in a given [billing period](#) MAY differ from the sum of the invoices received for the same [billing period](#) for a [billing account](#).

- When ChargeCategory is not "Usage" or "Purchase", EffectiveCost adheres to the following additional requirements:
 - EffectiveCost of a *charge* calculated based on other *charges* (e.g., when the ChargeCategory is "Tax") MUST be calculated based on the EffectiveCost of those related *charges*.
 - EffectiveCost of a *charge* unrelated to other *charges* (e.g., when the ChargeCategory is "Credit") MUST match the [BilledCost](#).
- *Charges* for a given [CommitmentDiscountId](#) adhere to the following additional requirements:
 - The sum of EffectiveCost where ChargeCategory is "Usage" MUST equal the sum of BilledCost where ChargeCategory is "Purchase".
 - The sum of EffectiveCost where ChargeCategory is "Usage" MUST equal the sum of EffectiveCost where ChargeCategory is "Usage" and [CommitmentDiscountStatus](#) is "Used", plus the sum of EffectiveCost where ChargeCategory is "Usage" and CommitmentDiscountStatus is "Unused".

3.1.34.2. Column ID

EffectiveCost

3.1.34.3. Display Name

Effective Cost

3.1.34.4. Description

The *amortized* cost of the *charge* after applying all reduced rates, discounts, and the applicable portion of relevant, prepaid purchases (one-time or recurring) that covered this *charge*.

3.1.34.4.1. Concerning Granularity and Distribution of Recurring Fee

Service providers should distribute the *commitment* purchase amount instead of including a *row* at the beginning of a period so practitioners do not need to manually distribute the fee themselves.

3.1.34.4.2. Concerning Amortization Approaches

Eligible purchases should be *amortized* using a methodology determined by the service provider that reflects the needs of their customer base and is proportional to the Pricing Quantity and the time granularity of the *row*. Should a practitioner desire to *amortize* relevant purchases using a different approach, the practitioner can do so using the [Billed Cost](#) for the line item representing the initial purchase.

3.1.34.5. Content constraints

Constraint	Value
Column type	Metric
Feature level	Mandatory

Constraint	Value
Allows nulls	False
Data type	Decimal
Value format	Numeric Format
Number range	Any valid decimal value

3.1.34.6. Introduced (version)

0.5

3.1.35. Host Provider Name

Host Provider Name is the name of the entity that provides the underlying infrastructure on which the [resources](#) or [services](#) of the [Service Provider](#) are deployed.

In some instances, the host provider and the service provider are the same entity: the provider hosts their own services. In other instances, the host provider and the service provider are separate entities, though the service provider may or may not expose the host provider and/or allow the customer to select the host provider.

3.1.35.1. Requirements

HostProviderName adheres to the following requirements:

- HostProviderName MUST be present in a Cost and Usage [FOCUS dataset](#).
- HostProviderName MUST be of type String.
- HostProviderName MUST conform to [StringHandling](#) requirements.
- HostProviderName nullability is defined as follows:
 - HostProviderName MAY be NULL when the associated [ServiceName](#) does not involve deployment on any underlying infrastructure (e.g., professional services, software licenses).
 - HostProviderName MAY be NULL when the information about the entity providing the underlying infrastructure cannot be uniquely determined (e.g., when the [ChargeCategory](#) is "Tax" or "Adjustment").
 - HostProviderName MUST NOT be null in all other cases.
- When HostProviderName is not null, HostProviderName values are defined as follows:
 - HostProviderName MUST reflect the name of the host provider when explicitly selected by the customer.
 - HostProviderName MUST reflect the name of the host provider when the service provider exposes the underlying hosting provider.
 - HostProviderName MUST equal [ServiceProviderName](#) in all other cases.

See [Appendix: Participating Entity Identification Examples](#) section for examples of Host Provider values across various use case scenarios.

3.1.35.2. Column ID

HostProviderName

3.1.35.3. Display Name

Host Provider Name

3.1.35.4. Description

The name of the entity whose *resources* are used by the Service Provider to make their [resources](#) or [services](#) available.

3.1.35.5. Content Constraints

Constraint	Value
Column type	Dimension
Feature level	Mandatory
Allows nulls	True
Data type	String
Value format	<not specified>

3.1.35.6. Introduced (version)

1.3

3.1.36. Invoice ID

An Invoice ID is an invoice-issuer-assigned identifier for an invoice encapsulating [charges](#) in the corresponding [billing period](#) for a given [billing account](#). Invoices are commonly used for scenarios like tracking billing transactions, facilitating payment processes and for performing invoice reconciliation between *charges* and billing periods.

3.1.36.1. Requirements

InvoiceId adheres to the following requirements:

- InvoiceId is RECOMMENDED to be present in a Cost and Usage [FOCUS dataset](#).
- InvoiceId MUST be of type String.
- InvoiceId MUST conform to [StringHandling](#) requirements.
- The sum of BilledCost for a given [InvoiceId](#) MUST match the sum of the payable amount provided in the corresponding invoice with the same id generated by the [InvoiceIssuerName](#).
- InvoiceId nullability is defined as follows:
 - InvoiceId MUST be null when the *charge* is not associated either with an invoice or with a pre-generated provisional invoice.
 - InvoiceId MUST NOT be null when the *charge* is associated with either an issued invoice or a pre-generated provisional invoice.
- InvoiceId MAY be generated prior to an invoice being issued.
- InvoiceId MUST be associated with the related *charge* and BillingAccountId when a pre-generated invoice or provisional invoice exists.

See [Appendix: Grouping constructs for resources or services](#) for details and examples of the different grouping constructs supported by FOCUS.

3.1.36.2. Column ID

InvoiceId

3.1.36.3. Display Name

Invoice ID

3.1.36.4. Description

The invoice-issuer-assigned identifier for an invoice encapsulating *charges* in the corresponding billing period for a given billing account.

3.1.36.5. Content constraints

Constraint	Value
Column type	Dimension
Feature level	Recommended
Allows nulls	True
Data type	String
Value format	<not specified>

3.1.36.6. Introduced (version)

1.2

3.1.37. Invoice Issuer Name

Invoice Issuer Name is the name of the entity responsible for issuing payable invoices for the [resources](#) or [services](#) consumed. It is commonly used for cost analysis and reporting scenarios.

3.1.37.1. Requirements

InvoiceIssuerName adheres to the following requirements:

- InvoiceIssuerName MUST be present in a Cost and Usage [FOCUS dataset](#).
- InvoiceIssuerName MUST be of type String.
- InvoiceIssuerName MUST conform to [StringHandling](#) requirements.
- InvoiceIssuerName MUST NOT be null.

See [Appendix: Participating Entity Identification Examples](#) section for examples of Invoice Issuer Name values across various use case scenarios.

3.1.37.2. Column ID

InvoiceIssuerName

3.1.37.3. Display Name

Invoice Issuer Name

3.1.37.4. Description

The name of the entity responsible for invoicing for the *resources* or *services* consumed.

3.1.37.5. Content Constraints

Constraint	Value
Column type	Dimension
Feature level	Mandatory
Allows nulls	False
Data type	String
Value format	<not specified>

3.1.37.6. Introduced (version)

0.5

3.1.38. List Cost

List Cost represents the cost calculated by multiplying the [list unit price](#) and the corresponding [Pricing Quantity](#). List Cost is denominated in the [Billing Currency](#) and is commonly used for calculating savings based on various rate optimization activities by comparing it with [Contracted Cost](#), [Billed Cost](#) and [Effective Cost](#).

3.1.38.1. Requirements

ListCost adheres to the following requirements:

- ListCost MUST be present in a Cost and Usage [FOCUS dataset](#).
- ListCost MUST be of type Decimal.
- ListCost MUST conform to [NumericFormat](#) requirements.
- ListCost MUST NOT be null.
- ListCost MUST be a valid decimal value.
- ListCost MUST be denominated in the BillingCurrency.
- When [ListUnitPrice](#) is null, ListCost adheres to the following additional requirements:
 - ListCost of a *charge* calculated based on other *charges* (e.g., when the [ChargeCategory](#) is "Tax") MUST be calculated based on the ListCost of those related *charges*.
 - ListCost of a *charge* unrelated to other *charges* (e.g., when the ChargeCategory is "Credit") MUST match the [BilledCost](#).

- ListCost MUST equal the product of ListUnitPrice and PricingQuantity when ListUnitPrice is not null and PricingQuantity is not null.

3.1.38.2. Column ID

ListCost

3.1.38.3. Display Name

List Cost

3.1.38.4. Description

Cost calculated by multiplying List Unit Price and the corresponding Pricing Quantity.

3.1.38.5. Usability Constraints

Aggregation: When aggregating List Cost for savings calculations, it's important to exclude either [Charge Category](#) "Purchase" *charges* (one-time and recurring) that are paid to cover future eligible *charges* (e.g., [commitment discount](#)) or the covered [Charge Category](#) "Usage" *charges* themselves. This exclusion helps prevent double counting of these *charges* in the aggregation. Which set of *charges* to exclude depends on whether cost are aggregated on a billed basis (exclude covered *charges*) or accrual basis (exclude Purchases for future *charges*). For instance, *charges* categorized as [Charge Category](#) "Purchase" and their related [Charge Category](#) "Tax" *charges* for a Commitment Discount might be excluded from an accrual basis cost aggregation of List Cost. This is because the "Usage" and "Tax" charge records provided during the term of the commitment discount already specify the List Cost. Purchase *charges* that cover future eligible *charges* can be identified by filtering for [Charge Category](#) "Purchase" records with a [Billed Cost](#) greater than 0 and an [Effective Cost](#) equal to 0.

3.1.38.6. Content Constraints

Constraint	Value
Column type	Metric
Feature level	Mandatory
Allows nulls	False
Data type	Decimal
Value format	Numeric Format
Number range	Any valid decimal value

3.1.38.7. Introduced (version)

1.0-preview

3.1.39. List Unit Price

The List Unit Price represents the suggested service-provider-published unit price for a single [Pricing Unit](#) of the associated SKU, exclusive of any discounts. This price is denominated in the [Billing Currency](#). The List Unit Price is commonly used for calculating savings based on various rate optimization activities.

3.1.39.1. Requirements

ListUnitPrice adheres to the following requirements:

- ListUnitPrice MUST be present in a Cost and Usage [FOCUS dataset](#) when the service provider publishes unit prices exclusive of discounts.
- ListUnitPrice MUST be of type Decimal.
- ListUnitPrice MUST conform to [NumericFormat](#) requirements.
- ListUnitPrice nullability is defined as follows:
 - ListUnitPrice MUST be null when [SkuPriceld](#) is null.
 - ListUnitPrice MUST be null when [ChargeCategory](#) is "Tax".
 - ListUnitPrice MUST NOT be null when [SkuPriceld](#) is not null.
 - ListUnitPrice MUST NOT be null when ChargeCategory is "Usage" or "Purchase" and [ChargeClass](#) is not "Correction".
 - ListUnitPrice MAY be null in all other cases.
- When ListUnitPrice is not null, ListUnitPrice adheres to the following additional requirements:
 - ListUnitPrice MUST be a non-negative decimal value.
 - ListUnitPrice MUST be denominated in the BillingCurrency.
- [ListCost](#) MUST equal the product of ListUnitPrice and [PricingQuantity](#) when ListUnitPrice is not null and PricingQuantity is not null.

3.1.39.2. Column ID

ListUnitPrice

3.1.39.3. Display Name

List Unit Price

3.1.39.4. Description

The suggested service-provider-published unit price for a single Pricing Unit of the associated SKU, exclusive of any discounts.

3.1.39.5. Usability Constraints

Aggregation: Column values should only be viewed in the context of their row and not aggregated to produce a total.

3.1.39.6. Content Constraints

Constraint	Value
Column type	Metric
Feature level	Conditional
Allows nulls	True
Data type	Decimal
Value format	Numeric Format
Number range	Any valid non-negative decimal value

3.1.39.7. Introduced (version)

1.0-preview

3.1.40. Pricing Category

Pricing Category describes the pricing model used for a [charge](#) at the time of use or purchase. It can be useful for distinguishing between *charges* incurred at the [list unit price](#) or a reduced price and exposing optimization opportunities, like increasing [commitment discount](#) coverage.

3.1.40.1. Requirements

PricingCategory adheres to the following requirements:

- PricingCategory MUST be present in a Cost and Usage [FOCUS dataset](#) when the service provider supports more than one pricing category across all [SKUs](#).
- PricingCategory MUST be of type String.
- PricingCategory nullability is defined as follows:
 - PricingCategory MUST be null when [SkuPriceld](#) is null.
 - PricingCategory MUST be null when [ChargeCategory](#) is "Tax".
 - PricingCategory MUST NOT be null when ChargeCategory is "Usage" or "Purchase" and [ChargeClass](#) is not "Correction".
 - PricingCategory MAY be null in all other cases.
- When PricingCategory is not null, PricingCategory adheres to the following additional requirements:
 - PricingCategory MUST be one of the allowed values.
 - PricingCategory MUST be "Standard" when pricing is predetermined at the agreed upon rate for the [billing account](#).
 - PricingCategory MUST be "Committed" when the *charge* is subject to an existing [commitment discount](#) and is not the purchase of the *commitment discount*.
 - PricingCategory MUST be "Dynamic" when pricing is determined by the service provider and may change over time, regardless of predetermined agreement pricing.
 - PricingCategory MUST be "Other" when there is a pricing model but none of the allowed values apply.

3.1.40.2. Column ID

PricingCategory

3.1.40.3. Display Name

3.1.40.4. Description

Describes the pricing model used for a *charge* at the time of use or purchase.

3.1.40.5. Content constraints

Constraint	Value
Column type	Dimension
Feature level	Conditional
Allows nulls	True
Data type	String
Value format	Allowed values

Allowed values:

Value	Description
Standard	<i>Charges</i> priced at the agreed upon rate for the billing account, including negotiated discounts . This pricing includes any flat rate and volume/tiered pricing but does not include dynamic pricing or reduced pricing due to the application of a <i>commitment discount</i> . This does include the purchase of a commitment discount at agreed upon rates.
Dynamic	<i>Charges</i> priced at a variable rate determined by the service provider. This includes any product or service with a unit price the service provider can change without notice, like interruptible or low priority resources .
Committed	<i>Charges</i> with reduced pricing due to the application of the <i>commitment discount</i> specified by the Commitment Discount ID.
Other	<i>Charges</i> priced in a way not covered by another pricing category.

3.1.40.6. Introduced (version)

1.0-preview

3.1.41. Pricing Currency

[Pricing Currency](#) is the national or virtual currency denomination that a [resource](#) or [service](#) was priced in. Pricing Currency is commonly used in scenarios where different currencies are used for pricing and billing.

3.1.41.1. Requirements

PricingCurrency adheres to the following requirements:

- PricingCurrency MUST be present in a Cost and Usage [FOCUS dataset](#) when the service provider supports pricing and billing in different currencies.
- PricingCurrency MUST be of type String.
- PricingCurrency MUST conform to [StringHandling](#) requirements.

- PricingCurrency MUST conform to [CurrencyFormat](#) requirements.
- PricingCurrency MUST NOT be null.

3.1.41.2. Column ID

PricingCurrency

3.1.41.3. Display Name

Pricing Currency

3.1.41.4. Description

The national or virtual currency denomination that a *resource* or *service* was priced in.

3.1.41.5. Content Constraints

Constraint	Value
Column type	Dimension
Feature level	Conditional
Allows nulls	True
Data type	String
Value format	Currency Format

3.1.41.6. Introduced (version)

1.2

3.1.42. Pricing Currency Contracted Unit Price

The Pricing Currency Contracted Unit Price represents the agreed-upon unit price for a single [Pricing Unit](#) of the associated [SKU](#), inclusive of [negotiated discounts](#), if present, while excluding negotiated [commitment discounts](#) or any other discounts. This price is denominated in the [Pricing Currency](#). When negotiated discounts do not apply to unit prices and instead are applied to exchange rates, the Pricing Currency Contracted Unit Price defaults to the [Pricing Currency List Unit Price](#). The Pricing Currency Contracted Unit Price is commonly used to calculate savings based on negotiation activities.

3.1.42.1. Requirements

PricingCurrencyContractedUnitPrice adheres to the following requirements:

- PricingCurrencyContractedUnitPrice presence in a Cost and Usage [FOCUS dataset](#) is defined as follows:

- PricingCurrencyContractedUnitPrice MUST be present in a Cost and Usage *FOCUS dataset* when the service provider supports prices in virtual currency and publishes unit prices exclusive of discounts.
- PricingCurrencyContractedUnitPrice is RECOMMENDED to be present in a Cost and Usage *FOCUS dataset* when the service provider supports pricing and billing in different currencies and publishes unit prices exclusive of discounts.
- PricingCurrencyContractedUnitPrice MAY be present in a Cost and Usage *FOCUS dataset* in all other cases.
- PricingCurrencyContractedUnitPrice MUST be of type Decimal.
- PricingCurrencyContractedUnitPrice MUST conform to [NumericFormat](#) requirements.
- PricingCurrencyContractedUnitPrice nullability is defined as follows:
 - PricingCurrencyContractedUnitPrice MUST be null when [SkuPriceld](#) is null.
 - PricingCurrencyContractedUnitPrice MUST be null when [ChargeCategory](#) is "Tax".
 - PricingCurrencyContractedUnitPrice MUST NOT be null when [SkuPriceld](#) is not null.
 - PricingCurrencyContractedUnitPrice MUST NOT be null when ChargeCategory is "Usage" or "Purchase" and [ChargeClass](#) is not "Correction".
 - PricingCurrencyContractedUnitPrice MAY be null in all other cases.
- When PricingCurrencyContractedUnitPrice is not null, PricingCurrencyContractedUnitPrice adheres to the following additional requirements:
 - PricingCurrencyContractedUnitPrice MUST be a non-negative decimal value.
 - PricingCurrencyContractedUnitPrice MUST be denominated in the PricingCurrency.

3.1.42.2. Column ID

PricingCurrencyContractedUnitPrice

3.1.42.3. Display Name

Pricing Currency Contracted Unit Price

3.1.42.4. Description

The agreed-upon unit price for a single Pricing Unit of the associated SKU, inclusive of *negotiated discounts*, if present, while excluding negotiated *commitment discounts* or any other discounts, and expressed in Pricing Currency.

3.1.42.5. Usability Constraints

Aggregation: Column values should only be viewed in the context of their row and not aggregated to produce a total.

3.1.42.6. Content Constraints

Constraint	Value
Column type	Metric
Feature level	Conditional
Allows nulls	True

Constraint	Value
Data type	Decimal
Value format	Numeric Format
Number range	Any valid non-negative decimal value

3.1.42.7. Introduced (version)

1.2

3.1.43. Pricing Currency Effective Cost

The Pricing Currency Effective Cost represents the cost of the [charge](#) after applying all reduced rates, discounts, and the applicable portion of relevant, prepaid purchases (one-time or recurring) that covered this *charge*, as denominated in [Pricing Currency](#). This allows the practitioner to perform a conversion from either 1) a [national currency](#) to a [virtual currency](#) (e.g., tokens to USD), or 2) one national currency to another (e.g., EUR to USD).

3.1.43.1. Requirements

PricingCurrencyEffectiveCost adheres to the following requirements:

- PricingCurrencyEffectiveCost presence in a Cost and Usage [FOCUS dataset](#) is defined as follows:
 - PricingCurrencyEffectiveCost MUST be present in a Cost and Usage [FOCUS dataset](#) when the service provider supports prices in virtual currency and publishes unit prices exclusive of discounts.
 - PricingCurrencyEffectiveCost is RECOMMENDED to be present in a Cost and Usage [FOCUS dataset](#) when the service provider supports pricing and billing in different currencies and publishes unit prices exclusive of discounts.
 - PricingCurrencyEffectiveCost MAY be present in a Cost and Usage [FOCUS dataset](#) in all other cases.
- PricingCurrencyEffectiveCost MUST be of type Decimal.
- PricingCurrencyEffectiveCost MUST conform to [NumericFormat](#) requirements.
- PricingCurrencyEffectiveCost MUST NOT be null.
- PricingCurrencyEffectiveCost MUST be a valid decimal value.
- PricingCurrencyEffectiveCost MUST be 0 in the event of prepaid purchases or purchases that are applicable to previous usage.
- PricingCurrencyEffectiveCost MUST be denominated in the [PricingCurrency](#).

3.1.43.2. Column ID

PricingCurrencyEffectiveCost

3.1.43.3. Display Name

Pricing Currency Effective Cost

3.1.43.4. Description

The cost of the *charge* after applying all reduced rates, discounts, and the applicable portion of relevant, prepaid purchases (one-time or recurring) that covered this *charge*, as denominated in Pricing Currency.

3.1.43.5. Content Constraints

Constraint	Value
Column type	Metric
Feature level	Conditional
Allows nulls	True
Data type	Decimal
Value format	Numeric Format
Number range	Any valid decimal value

3.1.43.6. Introduced (version)

1.2

3.1.44. Pricing Currency List Unit Price

The Pricing Currency List Unit Price represents the suggested service-provider-published unit price for a single [Pricing Unit](#) of the associated [SKU](#), exclusive of any discounts. This price is denominated in the [Pricing Currency](#). The Pricing Currency List Unit Price is commonly used for calculating savings based on various rate optimization activities.

3.1.44.1. Requirements

PricingCurrencyListUnitPrice adheres to the following requirements:

- PricingCurrencyListUnitPrice presence in a Cost and Usage [FOCUS dataset](#) is defined as follows:
 - PricingCurrencyListUnitPrice MUST be present in a Cost and Usage [FOCUS dataset](#) when the service provider supports prices in virtual currency and publishes unit prices exclusive of discounts.
 - PricingCurrencyListUnitPrice is RECOMMENDED to be present in a Cost and Usage [FOCUS dataset](#) when the service provider supports pricing and billing in different currencies and publishes unit prices exclusive of discounts.
 - PricingCurrencyListUnitPrice MAY be present in a Cost and Usage [FOCUS dataset](#) in all other cases.
- PricingCurrencyListUnitPrice MUST be of type Decimal.
- PricingCurrencyListUnitPrice MUST conform to [NumericFormat](#) requirements.
- PricingCurrencyListUnitPrice nullability is defined as follows:
 - PricingCurrencyListUnitPrice MUST be null when [SkuPriceld](#) is null.
 - PricingCurrencyListUnitPrice MUST be null when [ChargeCategory](#) is "Tax".
 - PricingCurrencyListUnitPrice MUST NOT be null when [SkuPriceld](#) is not null.
 - PricingCurrencyListUnitPrice MUST NOT be null when ChargeCategory is "Usage" or "Purchase" and [ChargeClass](#) is not "Correction".

- PricingCurrencyListUnitPrice MAY be null in all other cases.
- When PricingCurrencyListUnitPrice is not null, PricingCurrencyListUnitPrice adheres to the following additional requirements:
 - PricingCurrencyListUnitPrice MUST be a non-negative decimal value.
 - PricingCurrencyListUnitPrice MUST be denominated in the PricingCurrency.

3.1.44.2. Column ID

PricingCurrencyListUnitPrice

3.1.44.3. Display Name

Pricing Currency List Unit Price

3.1.44.4. Description

The suggested service-provider-published unit price for a single Pricing Unit of the associated *SKU*, exclusive of any discounts and expressed in Pricing Currency.

3.1.44.5. Usability Constraints

Aggregation: Column values should only be viewed in the context of their row and not aggregated to produce a total.

3.1.44.6. Content Constraints

Constraint	Value
Column type	Metric
Feature level	Conditional
Allows nulls	True
Data type	Decimal
Value format	Numeric Format
Number range	Any valid non-negative decimal value

3.1.44.7. Introduced (version)

1.2

3.1.45. Pricing Quantity

The Pricing Quantity represents the volume of a given [SKU](#) associated with a [resource](#) or [service](#) used or purchased, based on the [Pricing Unit](#). Distinct from [Consumed Quantity](#) (complementary to [Consumed Unit](#)), it focuses on pricing and cost, not *resource* and *service* consumption.

3.1.45.1. Requirements

PricingQuantity adheres to the following requirements:

- PricingQuantity MUST be present in a Cost and Usage [FOCUS dataset](#).
- PricingQuantity MUST be of type Decimal.
- PricingQuantity MUST conform to [NumericFormat](#) requirements.
- PricingQuantity nullability is defined as follows:
 - PricingQuantity MUST be null when [SkuPriceld](#) is null.
 - PricingQuantity MUST be null when [ChargeCategory](#) is "Tax".
 - PricingQuantity MUST NOT be null when ChargeCategory is "Usage" or "Purchase" and [ChargeClass](#) is not "Correction".
 - PricingQuantity MAY be null in all other cases.
- PricingQuantity MUST be a valid decimal value when not null.
- Cost metric (e.g., [ContractedCost](#)) MUST equal the product of the corresponding unit price (e.g., [ContractedUnitPrice](#)) and PricingQuantity when the unit price is not null and PricingQuantity is not null.

3.1.45.2. Column ID

PricingQuantity

3.1.45.3. Display Name

Pricing Quantity

3.1.45.4. Description

The volume of a given *SKU* associated with a *resource* or *service* used or purchased, based on the Pricing Unit.

3.1.45.5. Usability Constraints

Aggregation: When aggregating Pricing Quantity for commitment utilization calculations, it's important to exclude [commitment discount](#) purchases (i.e. when Charge Category is "Purchase") that are paid to cover future eligible [charges](#) (e.g., [commitment discount](#)). Otherwise, when accounting for all upfront or accrued purchases, it's important to exclude [commitment discount](#) usage (i.e. when Charge Category is "Usage"). This exclusion helps prevent double counting of these quantities in the aggregation.

3.1.45.6. Content Constraints

Constraint	Value
Column type	Metric
Feature level	Mandatory
Allows nulls	True

Constraint	Value
Data type	Decimal
Value format	Numeric Format
Number Range	Any valid decimal value

3.1.45.7. Introduced (version)

1.0-preview

3.1.46. Pricing Unit

The Pricing Unit represents a service-provider-specified measurement unit for determining unit prices, indicating how the service provider rates measured usage and purchase quantities after applying pricing rules like [block pricing](#). Common examples include the number of hours for compute appliance runtime (e.g., Hours), gigabyte-hours for a storage appliance (e.g., GB-Hours), or an accumulated count of requests for a network appliance or API service (e.g., 1000 Requests). Pricing Unit complements the [Pricing Quantity](#) metric. Distinct from the [Consumed Unit](#), it focuses on pricing and cost, not [resource](#) and [service](#) consumption, often at a coarser granularity.

3.1.46.1. Requirements

PricingUnit adheres to the following requirements:

- PricingUnit MUST be present in a Cost and Usage [FOCUS dataset](#).
- PricingUnit MUST be of type String.
- PricingUnit MUST conform to [StringHandling](#) requirements.
- PricingUnit SHOULD conform to [UnitFormat](#) requirements.
- PricingUnit nullability is defined as follows:
 - PricingUnit MUST be null when PricingQuantity is null.
 - PricingUnit MUST NOT be null when PricingQuantity is not null.
- When PricingUnit is not null, PricingUnit adheres to the following additional requirements:
 - PricingUnit MUST be semantically equal to the corresponding pricing measurement unit provided in service-provider-published [price list](#).
 - PricingUnit MUST be semantically equal to the corresponding pricing measurement unit provided in invoice, when the invoice includes a pricing measurement unit.

3.1.46.2. Column ID

PricingUnit

3.1.46.3. Display Name

Pricing Unit

3.1.46.4. Description

Service-provider-specified measurement unit for determining unit prices, indicating how the service provider rates measured usage and purchase quantities after applying pricing rules like *block pricing*.

3.1.46.5. Content constraints

Constraint	Value
Column type	Dimension
Feature level	Mandatory
Allows nulls	True
Data type	String
Value format	Unit Format

3.1.46.6. Introduced (version)

1.0-preview

3.1.47. Provider - DEPRECATED

Provider is the name of the entity that makes the [resources](#) or [services](#) available for purchase. It is commonly used for cost analysis and reporting scenarios.

3.1.47.1. Requirements

ProviderName adheres to the following requirements:

- ProviderName MUST be present in a Cost and Usage [FOCUS dataset](#).
- ProviderName MUST be of type String.
- ProviderName MUST conform to [StringHandling](#) requirements.
- ProviderName MUST NOT be null.

See [Appendix: Participating Entity Identification Examples](#) section for examples of [Service Provider Name](#), [Host Provider Name](#) and [Invoice Issuer Name](#) values across various use case scenarios. For entity identification examples that include the deprecated Provider column, please see [here](#) (external GitHub link to FOCUS v1.2).

3.1.47.2. Column ID

ProviderName

3.1.47.3. Display Name

Provider Name

3.1.47.4. Description

The name of the entity that made the *resources* or *services* available for purchase.

3.1.47.5. Content Constraints

Constraint	Value
Column type	Dimension
Feature level	Mandatory
Allows nulls	False
Data type	String
Value format	<not specified>

3.1.47.6. Introduced (version)

0.5

3.1.47.7. Deprecated (version)

1.3 Replaced by [ServiceProviderName](#)

3.1.48. Publisher - DEPRECATED

Publisher is the name of the entity that produces the *resources* or *services* that were purchased. It is commonly used for cost analysis and reporting scenarios.

3.1.48.1. Requirements

PublisherName adheres to the following requirements:

- PublisherName MUST be present in a Cost and Usage [FOCUS dataset](#).
- PublisherName MUST be of type String.
- PublisherName MUST conform to [StringHandling](#) requirements.
- PublisherName MUST NOT be null.

See [Appendix: Participating Entity Identification Examples](#) section for examples of [Service Provider Name](#), [Host Provider Name](#) and [Invoice Issuer Name](#) values across various use case scenarios. For entity identification examples that include the deprecated Publisher column, please see [here](#) (external GitHub link to FOCUS v1.2).

3.1.48.2. Column ID

PublisherName

3.1.48.3. Display Name

Publisher Name

3.1.48.4. Description

The name of the entity that produced the *resources* or *services* that were purchased.

3.1.48.5. Content Constraints

Constraint	Value
Column type	Dimension
Feature level	Mandatory
Allows nulls	False
Data type	String
Value format	<not specified>

3.1.48.6. Introduced (version)

0.5

3.1.48.7. Deprecated (version)

1.3

3.1.49. Region ID

A Region ID is a host-provider-assigned identifier for an isolated geographic area where a [resource](#) is provisioned or a [service](#) is provided. The region is commonly used for scenarios like analyzing cost and unit prices based on where *resources* are deployed.

3.1.49.1. Requirements

RegionId adheres to the following requirements:

- RegionId MUST be present in a Cost and Usage [FOCUS dataset](#) when the host provider supports deploying resources or services within a region.
- RegionId MUST be of type String.
- RegionId MUST conform to [StringHandling](#) requirements.
- RegionId nullability is defined as follows:
 - RegionId MUST NOT be null when a *resource* or *service* is operated in or managed from a distinct region.
 - RegionId MAY be null when a *resource* or *service* is not operated in or managed from a distinct region.

3.1.49.2. Column ID

RegionId

3.1.49.3. Display Name

Region ID

3.1.49.4. Description

Host-provider-assigned identifier for an isolated geographic area where a *resource* is provisioned or a *service* is provided.

3.1.49.5. Content constraints

Constraint	Value
Column type	Dimension
Feature level	Conditional
Allows nulls	True
Data type	String
Value format	<not specified>

3.1.49.6. Introduced (version)

1.0

3.1.50. Region Name

Region Name is a host-provider-assigned display name for an isolated geographic area where a [resource](#) is provisioned or a [service](#) is provided. Region Name is commonly used for scenarios like analyzing cost and unit prices based on where *resources* are deployed.

3.1.50.1. Requirements

RegionName adheres to the following requirements:

- RegionName MUST be present in a Cost and Usage [FOCUS dataset](#) when the host provider supports deploying resources or services within a region.
- RegionName MUST be of type String.
- RegionName MUST conform to [StringHandling](#) requirements.
- RegionName nullability is defined as follows:
 - RegionName MUST be null when [RegionId](#) is null.
 - RegionName MUST NOT be null when RegionId is not null.

3.1.50.2. Column ID

RegionName

3.1.50.3. Display Name

Region Name

3.1.50.4. Description

The name of an isolated geographic area where a *resource* is provisioned or a *service* is provided.

3.1.50.5. Content constraints

Constraint	Value
Column type	Dimension
Feature level	Conditional
Allows nulls	True
Data type	String
Value format	<not specified>

3.1.50.6. Introduced (version)

1.0

3.1.51. Resource ID

A Resource ID is an identifier assigned to a [resource](#) by the service provider. The Resource ID is commonly used for cost reporting, analysis, and allocation scenarios.

3.1.51.1. Requirements

Resourceid adheres to the following requirements:

- Resourceid MUST be present in a Cost and Usage [FOCUS dataset](#) when the service provider supports billing based on provisioned *resources*.
- Resourceid MUST be of type String.
- Resourceid MUST conform to [StringHandling](#) requirements.
- Resourceid nullability is defined as follows:
 - Resourceid MUST be null when a [charge](#) is not related to a *resource*.
 - Resourceid MUST NOT be null when a [charge](#) is related to a *resource*.
- When Resourceid is not null, Resourceid adheres to the following additional requirements:
 - Resourceid MUST be a unique identifier within the service provider.
 - Resourceid SHOULD be a fully-qualified identifier.

3.1.51.2. Column ID

Resourceid

3.1.51.3. Display Name

Resource ID

3.1.51.4. Description

Identifier assigned to a *resource* by the service provider.

3.1.51.5. Content Constraints

Constraint	Value
Column type	Dimension
Feature level	Conditional
Allows nulls	True
Data type	String
Value format	<not specified>

3.1.51.6. Introduced (version)

0.5

3.1.52. Resource Name

The Resource Name is a display name assigned to a [resource](#). It is commonly used for cost analysis, reporting, and allocation scenarios.

3.1.52.1. Requirements

ResourceName adheres to the following requirements:

- ResourceName MUST be present in a Cost and Usage [FOCUS dataset](#) when the service provider supports billing based on provisioned resources.
- ResourceName MUST be of type String.
- ResourceName MUST conform to [StringHandling](#) requirements.
- ResourceName nullability is defined as follows:
 - ResourceName MUST be null when [ResourceID](#) is null or when the *resource* does not have an assigned display name.
 - ResourceName MUST NOT be null when ResourceID is not null and the *resource* has an assigned display name.
- ResourceName MUST NOT duplicate ResourceID when the *resource* is not provisioned interactively or only has a system-generated ResourceID.

3.1.52.2. Column ID

ResourceName

3.1.52.3. Display Name

Resource Name

3.1.52.4. Description

Display name assigned to a *resource*.

3.1.52.5. Content Constraints

Constraint	Value
Column type	Dimension
Feature level	Conditional
Allows nulls	True
Data type	String
Value format	<not specified>

3.1.52.6. Introduced (version)

0.5

3.1.53. Resource Type

Resource Type describes the kind of [resource](#) the [charge](#) applies to. A Resource Type is commonly used for scenarios like identifying cost changes in groups of similar *resources* and may include values like Virtual Machine, Data Warehouse, and Load Balancer.

3.1.53.1. Requirements

ResourceType adheres to the following requirements:

- ResourceType MUST be present in a Cost and Usage [FOCUS dataset](#) when the service provider supports billing based on provisioned *resources* and supports assigning types to *resources*.
- ResourceType MUST be of type String.
- ResourceType MUST conform to [StringHandling](#) requirements.
- ResourceType nullability is defined as follows:
 - ResourceType MUST be null when [ResourceId](#) is null.
 - ResourceType MUST NOT be null when ResourceId is not null.

3.1.53.2. Column ID

ResourceType

3.1.53.3. Display Name

Resource Type

3.1.53.4. Description

The kind of *resource* the *charge* applies to.

3.1.53.5. Content Constraints

Constraint	Value
Column type	Dimension
Feature level	Conditional
Allows nulls	True
Data type	String
Value format	<not specified>

3.1.53.6. Introduced (version)

1.0-preview

3.1.54. Service Provider Name

Service Provider Name is the name of the entity that provides the [resources](#) or [services](#) available for usage or purchase. These services can be built on top of infrastructure provided by a [Host Provider](#), offered as fully integrated solutions, or include complementary offerings such as support, licensing, or consulting. It is commonly used for cost analysis and reporting scenarios.

Notes:

- In marketplace scenarios, the Service Provider represents the seller rather than the marketplace operator, as the marketplace operator merely provides a purchasing mechanism and does not itself provide the *resources* or *services* available for usage or purchase.
- In reseller scenarios, if the reseller is selling resource or services that are white-labeled from another provider, the Service Provider is the reseller. In all other cases the Service Provider is the entity that produced the resources or services.

3.1.54.1. Requirements

ServiceProviderName adheres to the following requirements:

- ServiceProviderName MUST be present in a Cost and Usage [FOCUS dataset](#).
- ServiceProviderName MUST be of type String.
- ServiceProviderName MUST conform to [StringHandling](#) requirements.
- ServiceProviderName MUST NOT be null.

See [Appendix: Participating Entity Identification Examples](#) section for examples of Service Provider

Name values across various use case scenarios.

3.1.54.2. Column ID

ServiceProviderName

3.1.54.3. Display Name

Service Provider Name

3.1.54.4. Description

The name of the entity that made the *resources* or *services* available for purchase or consumption.

3.1.54.5. Content Constraints

Constraint	Value
Column type	Dimension
Feature level	Mandatory
Allows nulls	False
Data type	String
Value format	<not specified>

3.1.54.6. Introduced (version)

1.3 Introduced as a replacement for [ProviderName](#)

3.1.55. Service Category

The Service Category is the highest-level classification of a [service](#) based on the core function of the *service*. Each *service* should have one and only one category that best aligns with its primary purpose. The Service Category is commonly used for scenarios like analyzing costs across service providers and tracking the migration of workloads across fundamentally different architectures.

3.1.55.1. Requirements

ServiceCategory adheres to the following requirements:

- ServiceCategory MUST be present in a Cost and Usage [FOCUS dataset](#).
- ServiceCategory MUST be of type String.
- ServiceCategory MUST NOT be null.
- ServiceCategory MUST be one of the allowed values.

3.1.55.2. Column ID

ServiceCategory

3.1.55.3. Display Name

Service Category

3.1.55.4. Description

Highest-level classification of a *service* based on the core function of the *service*.

3.1.55.5. Content Constraints

Constraint	Value
Column type	Dimension
Feature level	Mandatory
Allows nulls	False
Data type	String
Value format	Allowed Values

Allowed values:

Service Category	Description
AI and Machine Learning	Artificial Intelligence and Machine Learning related technologies.
Analytics	Data processing, analytics, and visualization capabilities.
Business Applications	Business and productivity applications and services.
Compute	Virtual, containerized, serverless, or high-performance computing infrastructure and services.
Databases	Database platforms and services that allow for storage and querying of data.
Developer Tools	Software development and delivery tools and services.
Multicloud	Support for interworking of multiple cloud and/or on-premises environments.
Identity	Identity and access management services.
Integration	Services that allow applications to interact with one another.
Internet of Things	Development and management of IoT devices and networks.
Management and Governance	Management, logging, and observability of a customer's use of cloud.
Media	Media and entertainment streaming and processing services.
Migration	Moving applications and data to the cloud.
Mobile	Services enabling cloud applications to interact via mobile technologies.
Networking	Network connectivity and management.
Security	Security monitoring and compliance services.

Service Category	Description
Storage	Storage services for structured or unstructured data.
Web	Services enabling cloud applications to interact via the Internet.
Other	New or emerging services that do not align with an existing category.

3.1.55.6. Introduced (version)

0.5

3.1.56. Service Name

A [service](#) represents an offering that can be purchased from a service provider (e.g., cloud virtual machine, SaaS database, professional services from a systems integrator). A *service* offering can include various types of usage or other [charges](#). For example, a cloud database *service* may include compute, storage, and networking *charges*.

The Service Name is a display name for the offering that was purchased. The Service Name is commonly used for scenarios like analyzing aggregate cost trends over time and filtering data to investigate anomalies.

3.1.56.1. Requirements

ServiceName adheres to the following requirements:

- ServiceName MUST be present in a Cost and Usage [FOCUS dataset](#).
- ServiceName MUST be of type String.
- ServiceName MUST conform to [StringHandling](#) requirements.
- ServiceName MUST NOT be null.
- The relationship between ServiceName and [ServiceCategory](#) is defined as follows:
 - ServiceName MUST have one and only one ServiceCategory that best aligns with its primary purpose, except when no suitable ServiceCategory is available.
 - ServiceName MUST be associated with the ServiceCategory "Other" when no suitable ServiceCategory is available.
- The relationship between ServiceName and [ServiceSubcategory](#) is defined as follows:
 - ServiceName SHOULD have one and only one ServiceSubcategory that best aligns with its primary purpose, except when no suitable ServiceSubcategory is available.
 - ServiceName SHOULD be associated with the ServiceSubcategory "Other" when no suitable ServiceSubcategory is available.

3.1.56.2. Column ID

ServiceName

3.1.56.3. Display Name

Service Name

3.1.56.4. Description

An offering that can be purchased from a service provider (e.g., cloud virtual machine, SaaS database, professional *services* from a systems integrator).

3.1.56.5. Content Constraints

Constraint	Value
Column type	Dimension
Feature level	Mandatory
Allows nulls	False
Data type	String
Value format	<not specified>

3.1.56.6. Introduced (version)

0.5

3.1.57. Service Subcategory

The Service Subcategory is a secondary classification of the [Service Category](#) for a [service](#) based on its core function. The Service Subcategory (in conjunction with the Service Category) is commonly used for scenarios like analyzing spend and usage for specific workload types across service providers and tracking the migration of workloads across fundamentally different architectures.

3.1.57.1. Requirements

ServiceSubcategory adheres to the following requirements:

- ServiceSubcategory is RECOMMENDED to be present in a Cost and Usage [FOCUS dataset](#).
- ServiceSubcategory MUST be of type String.
- ServiceSubcategory MUST NOT be null.
- ServiceSubcategory MUST be one of the allowed values.
- ServiceSubcategory MUST have one and only one parent ServiceCategory as specified in the allowed values below.

3.1.57.2. Column ID

ServiceSubcategory

3.1.57.3. Display Name

Service Subcategory

3.1.57.4. Description

Secondary classification of the Service Category for a *service* based on its core function.

3.1.57.5. Content Constraints

Constraint	Value
Column type	Dimension
Feature level	Recommended
Allows nulls	False
Data type	String
Value format	Allowed Values

Allowed values:

Service Category	Service Subcategory	Service Subcategory Description
AI and Machine Learning	AI Platforms	Unified solution that combines artificial intelligence and machine learning technologies.
AI and Machine Learning	Bots	Automated performance of tasks such as customer service, data collection, and content moderation.
AI and Machine Learning	Generative AI	Creation of content like text, images, and music by learning patterns from existing data.
AI and Machine Learning	Machine Learning	Creation, training, and deployment of statistical algorithms that learn from and perform tasks based on data.
AI and Machine Learning	Natural Language Processing	Generation of human language, handling tasks like translation, sentiment analysis, and text summarization.
AI and Machine Learning	Other (AI and Machine Learning)	AI and Machine Learning services that do not fall into one of the defined subcategories.
Analytics	Analytics Platforms	Unified solution that combines technologies across the entire analytics lifecycle.
Analytics	Business Intelligence	Semantic models, dashboards, reports, and data visualizations to track performance and identify trends.
Analytics	Data Processing	Integration and transformation tasks to prepare data for analysis.
Analytics	Search	Discovery of information by indexing and retrieving data from various sources.
Analytics	Streaming Analytics	Real-time data stream processes to detect patterns, trends, and anomalies as they occur.
Analytics	Other (Analytics)	Analytics services that do not fall into one of the defined subcategories.
Business Applications	Productivity and Collaboration	Tools that facilitate individuals managing tasks and working together.
Business Applications	Other (Business Applications)	Business Applications services that do not fall into one of the defined subcategories.
Compute	Containers	Management and orchestration of containerized compute platforms.

Service Category	Service Subcategory	Service Subcategory Description
Compute	End User Computing	Virtualized desktop infrastructure and device / endpoint management.
Compute	Quantum Compute	Resources and simulators that leverage the principles of quantum mechanics.
Compute	Serverless Compute	Enablement of compute capabilities without provisioning or managing servers.
Compute	Virtual Machines	Computing environments ranging from hosts with abstracted operating systems to bare-metal servers.
Compute	Other (Compute)	Compute services that do not fall into one of the defined subcategories.
Databases	Caching	Low-latency and high-throughput access to frequently accessed data.
Databases	Data Warehouses	Big data storage and querying capabilities.
Databases	Ledger Databases	Immutable and transparent databases to record tamper-proof and cryptographically secure transactions.
Databases	NoSQL Databases	Unstructured or semi-structured data storage and querying capabilities.
Databases	Relational Databases	Structured data storage and querying capabilities.
Databases	Time Series Databases	Time-stamped data storage and querying capabilities.
Databases	Other (Databases)	Database services that do not fall into one of the defined subcategories.
Developer Tools	Developer Platforms	Unified solution that combines technologies across multiple areas of the software development lifecycle.
Developer Tools	Continuous Integration and Deployment	CI/CD tools and services that support building and deploying code for software and systems.
Developer Tools	Development Environments	Tools and services that support authoring code for software and systems.
Developer Tools	Source Code Management	Tools and services that support version control of code for software and systems.
Developer Tools	Quality Assurance	Tools and services that support testing code for software and systems.
Developer Tools	Other (Developer Tools)	Developer Tools services that do not fall into one of the defined subcategories.
Identity	Identity and Access Management	Technologies that ensure users have appropriate access to resources.
Identity	Other (Identity)	Identity services that do not fall into one of the defined subcategories.
Integration	API Management	Creation, publishing, and management of application programming interfaces.
Integration	Messaging	Asynchronous communication between distributed applications.
Integration	Workflow Orchestration	Design, execution, and management of business processes and workflows.
Integration	Other (Integration)	Integration services that do not fall into one of the defined subcategories.

Service Category	Service Subcategory	Service Subcategory Description
Internet of Things	IoT Analytics	Examination of data collected from IoT devices.
Internet of Things	IoT Platforms	Unified solution that combines IoT data collection, processing, visualization, and device management.
Internet of Things	Other (Internet of Things)	Internet of Things (IoT) services that do not fall into one of the defined subcategories.
Management and Governance	Architecture	Planning, design, and construction of software systems.
Management and Governance	Compliance	Adherence to regulatory standards and industry best practices.
Management and Governance	Cost Management	Monitoring and controlling expenses of systems and services.
Management and Governance	Data Governance	Management of the availability, usability, integrity, and security of data.
Management and Governance	Disaster Recovery	Plans and procedures that ensure systems and services can recover from disruptions.
Management and Governance	Endpoint Management	Tools that configure and secure access to devices.
Management and Governance	Observability	Monitoring, logging, and tracing of data to track the performance and health of systems.
Management and Governance	Support	Assistance and expertise supplied by service providers.
Management and Governance	Other (Management and Governance)	Management and governance services that do not fall into one of the defined subcategories.
Media	Content Creation	Production of media content.
Media	Gaming	Development and delivery of gaming services.
Media	Media Streaming	Multimedia delivered and rendered in real-time on devices.
Media	Mixed Reality	Technologies that blend real-world and computer-generated environments.
Media	Other (Media)	Media services that do not fall into one of the defined subcategories.
Migration	Data Migration	Movement of stored data from one location to another.
Migration	Resource Migration	Movement of resources from one location to another.
Migration	Other (Migration)	Migration services that do not fall into one of the defined subcategories.
Mobile	Other (Mobile)	All Mobile services.
Multicloud	Multicloud Integration	Environments that facilitate consumption of services from multiple cloud service providers.
Multicloud	Other (Multicloud)	Multicloud services that do not fall into one of the defined subcategories.

Service Category	Service Subcategory	Service Subcategory Description
Networking	Application Networking	Distribution of incoming network traffic across application-based workloads.
Networking	Content Delivery	Distribution of digital content using a network of servers (CDNs).
Networking	Network Connectivity	Facilitates communication between networks or network segments.
Networking	Network Infrastructure	Configuration, monitoring, and troubleshooting of network devices.
Networking	Network Routing	Services that select paths for traffic within or across networks.
Networking	Network Security	Protection from unauthorized network access and cyber threats using firewalls and anti-malware tools.
Networking	Other (Networking)	Networking services that do not fall into one of the defined subcategories.
Security	Secret Management	Information used to authenticate users and systems, including secrets, certificates, tokens, and other keys.
Security	Security Posture Management	Tools that help organizations configure, monitor, and improve system security.
Security	Threat Detection and Response	Collect and analyze security data to identify and respond to potential security threats and vulnerabilities.
Security	Other (Security)	Security services that do not fall into one of the defined subcategories.
Storage	Backup Storage	Secondary storage to protect against data loss.
Storage	Block Storage	High performance, low latency storage that provides random access.
Storage	File Storage	Scalable, sharable storage for file-based data.
Storage	Object Storage	Highly available, durable storage for unstructured data.
Storage	Storage Platforms	Unified solution that supports multiple storage types.
Storage	Other (Storage)	Storage services that do not fall into one of the defined subcategories.
Web	Application Platforms	Integrated environments that run web applications.
Web	Other (Web)	Web services that do not fall into one of the defined subcategories.
Other	Other (Other)	Services that do not fall into one of the defined categories.

3.1.57.6. Introduced (version)

1.1

3.1.58. SKU ID

A SKU ID is a service-provider-specified unique identifier that represents a specific [SKU](#). *SKUs* are quantifiable goods or service offerings in a Cost and Usage [FOCUS dataset](#) that represent specific functionality and technical specifications. Examples of *SKUs* include but are not limited to:

- A product license that is purchased or subscribed to.
- Usage of a deployed resource from direct user interaction (e.g., request count).

- Usage by a deployed resource based on the resource's configuration (e.g., running hours, storage space).

Each SKU ID represents a unique set of features that can be sold at different price points or [SKU Prices](#). SKU ID is consistent across all pricing variations, which may differ based on multiple factors beyond the common functionality and technical specifications. Examples include but are not limited to:

- Date the [charge](#) was incurred.
- Pricing tiers (e.g., free tier or volume-based tiers).
- Commitment discount pricing [period](#) (e.g., 1 year, 3 years).
- Negotiated discounts or other contractual terms or conditions.

SKU ID should be consistent across pricing variations of a good or service to facilitate price comparisons for the same functionality, like where the functionality is provided or how it's paid for. SKU ID can be referenced on a catalog or [price list](#) published by a service provider to look up detailed information about the *SKU*. The composition of the properties associated with the SKU ID may differ across service providers. SKU ID is commonly used for analyzing and comparing costs for the same SKU across different price details (e.g., *period*, tier, location).

3.1.58.1. Requirements

Skuld adheres to the following requirements:

- Skuld MUST be present in a Cost and Usage [FOCUS dataset](#) when the service provider supports unit pricing concepts and publishes price lists, publicly or as part of contracting.
- Skuld MUST be of type String.
- Skuld MUST conform to [StringHandling](#) requirements.
- Skuld nullability is defined as follows:
 - Skuld MUST be null when [ChargeCategory](#) is "Tax".
 - Skuld MUST NOT be null when ChargeCategory is "Usage" or "Purchase" and [ChargeClass](#) is not "Correction".
 - Skuld MAY be null in all other cases.
- Skuld for a given *SKU* adheres to the following additional requirements:
 - Skuld MUST remain consistent across [billing accounts](#) or contracts.
 - Skuld MUST remain consistent across [PricingCategory](#) values.
 - Skuld MUST remain consistent regardless of any other factors that might impact the price but do not affect the functionality of the *SKU*.
- Skuld MUST be associated with a given [resource](#) or [service](#) when ChargeCategory is "Usage" or "Purchase".
- Skuld MAY equal [SkuPriceld](#).

3.1.58.2. Column ID

Skuld

3.1.58.3. Display Name

SKU ID

3.1.58.4. Description

Service-provider-specified unique identifier that represents a specific *SKU* (e.g., a quantifiable good or service offering).

3.1.58.5. Content constraints

Constraint	Value
Column type	Dimension
Feature level	Conditional
Allows nulls	True
Data type	String
Value format	<not specified>

3.1.58.6. Introduced (version)

1.0-preview

3.1.59. SKU Meter

SKU Meter describes the functionality being metered or measured by a particular SKU in a [charge](#).

Service providers often have billing models in which multiple SKUs exist for a given service to describe and bill for different functionalities for that service. For example, an object storage service may have separate SKUs for functionalities such as object storage, API requests, data transfer, encryption, and object management. This field helps practitioners understand which functionalities are being metered by the different SKUs that appear in a Cost and Usage [FOCUS dataset](#).

3.1.59.1. Requirements

SkuMeter adheres to the following requirements:

- SkuMeter MUST be present in a Cost and Usage *FOCUS dataset* when the service provider supports unit pricing concepts and publishes [price lists](#), publicly or as part of contracting.
- SkuMeter MUST be of type String.
- SkuMeter MUST conform to [StringHandling](#) requirements.
- SkuMeter nullability is defined as follows:
 - SkuMeter MUST be null when [Skuld](#) is null.
 - SkuMeter SHOULD NOT be null when Skuld is not null.
- SkuMeter SHOULD remain consistent over time for a given Skuld.

3.1.59.2. Examples

Compute Usage, Block Volume Usage, Data Transfer, API Requests

3.1.59.3. Column ID

SkuMeter

3.1.59.4. Display Name

SKU Meter

3.1.59.5. Description

Describes the functionality being metered or measured by a particular SKU in a *charge*.

3.1.59.6. Content Constraints

Constraint	Value
Column type	Dimension
Feature level	Conditional
Allows nulls	True
Data type	String
Value format	<not specified>

3.1.59.7. Introduced (version)

1.1

3.1.60. SKU Price Details

SKU Price Details represent a list of [SKU Price](#) properties (key-value pairs) associated with a specific [SKU Price ID](#). These properties include qualitative and quantitative properties of a [SKUs](#) (e.g., functionality and technical specifications), along with core stable pricing properties (e.g., pricing [periods](#), tiers, etc.), excluding dynamic or negotiable pricing elements such as unit price amounts, currency (and related exchange rates), temporal validity (e.g., effective dates), and contract- or negotiation-specific factors (e.g., contract or account identifiers, and negotiable discounts).

The composition of properties associated with a specific *SKU Price* may differ across service providers and across *SKUs* within the same service provider. However, the exclusion of dynamic or negotiable pricing properties should ensure that all [charges](#) with the same SKU Price ID share the same SKU Price Details, i.e., that SKU Price Details remains consistent across different [billing periods](#) and [billing accounts](#) within a service provider.

SKU Price Details helps practitioners understand and distinguish *SKU Prices*, each identified by a SKU Price ID and associated with a used or purchased [resource](#) or [service](#). It can also help determine the quantity of units for a property when it holds a numeric value (e.g., CoreCount), even when its unit differs from the one in which the *SKU* is priced and charged, thus supporting FinOps capabilities such as unit economics. Additionally, the SKU Price Details may be used to analyze costs based on pricing properties such as *periods* and tiers.

3.1.60.1. Requirements

SkuPriceDetails adheres to the following requirements:

- SkuPriceDetails MUST be present in a Cost and Usage [FOCUS dataset](#) when the service provider supports unit pricing concepts and publishes [price lists](#), publicly or as part of contracting.
- SkuPriceDetails MUST conform to [KeyValueFormat](#) requirements.
- SkuPriceDetails property keys SHOULD conform to [PascalCase](#) format.
- SkuPriceDetails nullability is defined as follows:
 - SkuPriceDetails MUST be null when SkuPriceld is null.
 - SkuPriceDetails MAY be null when SkuPriceld is not null.
- When SkuPriceDetails is not null, SkuPriceDetails adheres to the following additional requirements:
 - SkuPriceDetails MUST be associated with a given SkuPriceld.
 - SkuPriceDetails MUST include the FOCUS-defined SKU Price property when an equivalent property is included as a custom property.
 - SkuPriceDetails MUST NOT include properties that are not applicable to the corresponding SkuPriceld.
 - SkuPriceDetails SHOULD include all FOCUS-defined SKU Price properties listed below that are applicable to the corresponding SkuPriceld.
 - SkuPriceDetails SHOULD include all custom SKU Price properties that are applicable to the corresponding SkuPriceld when there is no equivalent FOCUS-defined property.
 - SkuPriceDetails MAY include properties that are already captured in other dedicated columns.
 - SkuPriceDetails properties for a given SkuPriceld adhere to the following additional requirements:
 - Existing SkuPriceDetails properties SHOULD remain consistent over time.
 - Existing SkuPriceDetails properties SHOULD NOT be removed.
 - Additional SkuPriceDetails properties MAY be added over time.
 - Property key SHOULD remain consistent across comparable *SKUs* having that property, and the values for this key SHOULD remain in a consistent format.
 - Property key MUST begin with the string "x_" unless it is a FOCUS-defined property.
 - Property value MUST represent the value for a single [PricingUnit](#) when the property holds a numeric value.
- FOCUS-defined SKU Price properties adhere to the following additional requirements:
 - Property key MUST match the spelling and casing specified for the FOCUS-defined property.
 - Property value MUST be of the type specified for that property.
 - Property value MUST represent the value for a single PricingUnit, denominated in the unit of measure specified for that property when the property holds a numeric value.

3.1.60.2. Examples

```
{
  "StorageClass": "Archive",
  "CoreCount": 4,
  "x_PremiumProcessing": true
}
```

3.1.60.3. Column ID

SkuPriceDetails

3.1.60.4. Display Name

SKU Price Details

3.1.60.5. Description

A set of properties of a SKU Price ID which are meaningful and common to all instances of that SKU Price ID.

3.1.60.6. Content Constraints

Constraint	Value
Column type	Dimension
Feature level	Conditional
Allows nulls	True
Data type	JSON
Value format	Key-Value Format

3.1.60.6.1. FOCUS-Defined Properties

The following keys should be used when applicable to facilitate cross-SKU and cross-service-provider queries for the same conceptual property. FOCUS-defined keys will appear in the list below and custom (e.g., service-provider-defined) keys will be prefixed with "x_" to make them easy to identify as well as prevent collisions.

Key	Description	Data Type	Unit of Measure (numeric) or example values (string)
CoreCount	Number of physical or virtual CPUs available ¹	Numeric	Measure: Quantity of Cores
DiskMaxIops	Storage maximum sustained input/output operations per second ¹	Numeric	Measure: Input/Output Operations per Second (IOPS)
DiskSpace	Storage capacity available	Numeric	Measure: Gibibytes (GiB)
DiskType	Kind of disk used	String	Examples: "SSD", "HDD", "NVMe"
GpuCount	Number of GPUs available	Numeric	Measure: Quantity of GPUs
InstanceType	Common name of the instance including size, shape, series, etc.	String	Examples: "m5d.2xlarge", "NC24rs_v3", "P50"
InstanceSeries	Common name for the series and/or generation of the instance	String	Examples: "M5", "Dadv5", "N2D"
MemorySize	RAM allocated for processing	Numeric	Measure: Gibibytes (GiB ²)
NetworkMaxIops	Network maximum sustained input/output operations per second ¹	Numeric	Measure: Input/Output Operations per Second (IOPS)

Key	Description	Data Type	Unit of Measure (numeric) or example values (string)
NetworkMaxThroughput	Network maximum sustained throughput for data transfer ¹	Numeric	Measure: Megabits per second (Mbps)
OperatingSystem	Operating system family ³	String	Examples: "Linux", "MacOS", "Windows"
Redundancy	Level of redundancy offered by the SKU	String	Examples: "Local", "Zonal", "Global"
StorageClass	Class or tier of storage provided	String	Examples: "Hot", "Archive", "Nearline"

Notes

¹ In the case of "burstable" SKUs offering variable levels of performance, the baseline or guaranteed value should be used.

² Memory manufacturers still commonly uses "GB" to refer to 2³⁰ bytes, which is known as GiB in other contexts.

³ This is the operating system family of the SKU, if it's included with the SKU or the SKU only supports one type of operating system.

3.1.60.7. Introduced (version)

1.1

3.1.61. SKU Price ID

SKU Price ID is a service-provider-specified unique identifier that represents a specific [SKU Price](#) associated with a [resource](#) or [service](#) used or purchased. It serves as a key reference for a *SKU Price* in a [price list](#) published by a service provider, allowing practitioners to look up detailed information about the *SKU Price*.

The composition of properties associated with the SKU Price ID may differ across service providers and across *SKUs* within the same service provider. However, the exclusion of dynamic or negotiable pricing properties, such as unit price amount, currency (and related exchange rates), temporal validity (e.g., effective dates), and contract- or negotiation-specific elements (e.g., contract or account identifiers, and negotiable discounts), ensures that the SKU Price ID remains consistent across different billing periods and billing accounts within a service provider. This consistency enables efficient filtering of [charges](#) to track price fluctuations (e.g., changes in unit price amounts) over time and across billing accounts, for both list and contracted unit prices. Additionally, the SKU Price ID is commonly used to analyze costs based on pricing properties such as [periods](#) and tiers.

3.1.61.1. Requirements

SkuPriceId adheres to the following requirements:

- SkuPriceId MUST be present in a Cost and Usage [FOCUS dataset](#) when the service provider supports unit pricing concepts and publishes *price lists*, publicly or as part of contracting.
- SkuPriceId MUST be of type String.
- SkuPriceId MUST conform to [String Handling](#) requirements.
- SkuPriceId nullability is defined as follows:
 - SkuPriceId MUST be null when [ChargeCategory](#) is "Tax".
 - SkuPriceId MUST NOT be null when ChargeCategory is "Usage" or "Purchase" and

[ChargeClass](#) is not "Correction".

- SkuPricelId MAY be null in all other cases.
- When SkuPricelId is not null, SkuPricelId adheres to the following additional requirements:
 - SkuPricelId MUST have one and only one parent [Skuld](#).
 - SkuPricelId MUST remain consistent over time.
 - SkuPricelId MUST remain consistent across [billing accounts](#) or contracts.
 - SkuPricelId MAY equal Skuld.
 - SkuPricelId MUST be associated with a given [resource](#) or [service](#) when ChargeCategory is "Usage" or "Purchase".
 - SkuPricelId MUST reference a *SKU Price* in a service-provider-supplied *price list*, enabling the lookup of detailed information about the *SKU Price*.
 - SkuPricelId MUST support the lookup of the [ListUnitPrice](#) when the service provider publishes unit prices exclusive of discounts.
 - SkuPricelId MUST support the verification of the given [ContractedUnitPrice](#) when the service provider supports negotiated pricing concepts.

See [Examples: Commitment Discount Flexibility](#) for more details around *commitment discount flexibility*.

3.1.61.2. Column ID

SkuPricelId

3.1.61.3. Display Name

SKU Price ID

3.1.61.4. Description

A service-provider-specified unique identifier that represents a specific *SKU Price* associated with a *resource* or *service* used or purchased.

3.1.61.5. Content constraints

Constraint	Value
Column type	Dimension
Feature level	Conditional
Allows nulls	True
Data type	String
Value format	<not specified>

3.1.61.6. Introduced (version)

1.0-preview

3.1.62. Sub Account ID

A Sub Account ID is a service-provider-assigned identifier assigned to a [sub account](#). Sub Account ID is commonly used for scenarios like grouping based on organizational constructs, access management needs, and cost allocation strategies.

3.1.62.1. Requirements

SubAccountId adheres to the following requirements:

- SubAccountId MUST be present in a Cost and Usage [FOCUS dataset](#) when the service provider supports a *sub account* construct.
- SubAccountId MUST be of type String.
- SubAccountId MUST conform to [StringHandling](#) requirements.
- SubAccountId nullability is defined as follows:
 - SubAccountId MUST be null when a [charge](#) is not related to a *sub account*.
 - SubAccountId MUST NOT be null when a *charge* is related to a *sub account*.

See [Appendix: Grouping constructs for resources or services](#) for details and examples of the different grouping constructs supported by FOCUS.

3.1.62.2. Column ID

SubAccountId

3.1.62.3. Display Name

Sub Account ID

3.1.62.4. Description

An ID assigned to a grouping of [resources](#) or [services](#), often used to manage access and/or cost.

3.1.62.5. Content constraints

Constraint	Value
Column type	Dimension
Feature level	Conditional
Allows nulls	True
Data type	String
Value format	<not specified>

3.1.62.6. Introduced (version)

0.5

3.1.63. Sub Account Name

A Sub Account Name is a display name assigned to a [sub account](#). Sub account Name is commonly used for scenarios like grouping based on organizational constructs, access management needs, and cost allocation strategies.

3.1.63.1. Requirements

SubAccountName adheres to the following requirements:

- SubAccountName MUST be present in a Cost and Usage [FOCUS dataset](#) when the service provider supports a *sub account* construct.
- SubAccountName MUST be of type String.
- SubAccountName MUST conform to [StringHandling](#) requirements.
- SubAccountName nullability is defined as follows:
 - SubAccountName MUST be null when [SubAccountid](#) is null.
 - SubAccountName MUST NOT be null when SubAccountid is not null.

See [Appendix: Grouping constructs for resources or services](#) for details and examples of the different grouping constructs supported by FOCUS.

3.1.63.2. Column ID

SubAccountName

3.1.63.3. Display Name

Sub Account Name

3.1.63.4. Description

A name assigned to a grouping of [resources](#) or [services](#), often used to manage access and/or cost.

3.1.63.5. Content constraints

Constraint	Value
Column type	Dimension
Feature level	Conditional
Allows nulls	True
Data type	String
Value format	<not specified>

3.1.63.6. Introduced (version)

0.5

3.1.64. Sub Account Type

Sub Account Type is a service-provider-assigned name to identify the type of *sub account*. Sub Account Type is a readable display name and not a code. Sub Account Type is commonly used for scenarios like mapping FOCUS and service provider constructs, summarizing costs across service providers, or invoicing and chargeback.

3.1.64.1. Requirements

SubAccountType adheres to the following requirements:

- SubAccountType MUST be present in a Cost and Usage *FOCUS dataset* when the service provider supports more than one possible SubAccountType value.
- SubAccountType MUST be of type String.
- SubAccountType MUST conform to *StringHandling* requirements.
- SubAccountType nullability is defined as follows:
 - SubAccountType MUST be null when *SubAccountId* is null.
 - SubAccountType MUST NOT be null when SubAccountId is not null.
- SubAccountType MUST be a consistent, readable display value.

3.1.64.2. Column ID

SubAccountType

3.1.64.3. Display Name

Sub Account Type

3.1.64.4. Description

A service-provider-assigned name to identify the type of *sub account*.

3.1.64.5. Content Constraints

Constraint	Value
Column type	Dimension
Feature level	Conditional
Allows nulls	True
Data type	String
Value format	<not specified>

3.1.64.6. Introduced (version)

1.2

3.1.65. Tags

The Tags column represents the set of [tags](#) assigned to [tag sources](#) that also account for potential provider-defined or user-defined tag evaluations. Tags are commonly used for scenarios like adding business context to cost and usage data to identify and accurately allocate [charges](#). Tags may also be referred to by data generators using other terms such as labels.

A tag becomes [finalized](#) when a single value is selected from a set of possible tag values assigned to the tag key. When supported by a data generator, this can occur when a tag value is set by provider-defined or user-defined rules.

3.1.65.1. Requirements

Tags adheres to the following requirements:

- Tags MUST be present in a Cost and Usage [FOCUS dataset](#) when the data generator supports setting user or provider-defined tags.
- Tags MUST conform to [KeyValueFormat](#) requirements.
- Tags MAY be null.
- When Tags is not null, Tags adheres to the following additional requirements:
 - Tags MUST include all user-defined and provider-defined tags.
 - Tags MUST only include finalized tags.
 - Tags SHOULD include tag keys with corresponding non-null values for a given [resource](#).
 - Tags MAY include tag keys with a null value for a given *resource* depending on the data generator's tag finalization process.
 - Tag keys that do not support corresponding values, MUST have a corresponding true (boolean) value set.
 - Data generator SHOULD publish tag finalization methods and semantics within their respective documentation.
 - Data generator MUST NOT alter tag values unless applying true (boolean) to valueless tags.
- Provider-defined tags adhere to the following additional requirements:
 - Provider-defined tag keys MUST be prefixed with a predetermined, provider-specified tag key prefix that is unique to each corresponding provider-specified tag scheme.
 - Data generator SHOULD publish all provider-specified tag key prefixes within their respective documentation.
- User-defined tags adhere to the following additional requirements:
 - Data generator MUST prefix all but one user-defined tag scheme with a predetermined, provider-specified tag key prefix that is unique to each corresponding user-defined tag scheme when the data generator has more than one user-defined tag scheme.
 - Data generator MUST NOT prefix tag keys when the data generator has only one user-defined tag scheme.
 - Data generator MUST NOT allow reserved tag key prefixes to be used as prefixes for any user-defined tag keys within a prefixless user-defined tag scheme.

3.1.65.2. Provider-Defined vs. User-Defined Tags

This example illustrates various tags produced from multiple user-defined and provider-defined tag schemes. The first three tags illustrate examples from three different, user-defined tag schemes. The data generator predetermined that 1 user-defined tag scheme (i.e., "foo": "bar") does not have a prepended prefix, but the remaining two user-defined tag schemes (i.e., "userDefinedTagScheme2/foo": "bar", "userDefinedTagScheme3/foo": true) do have provider-defined and reserved prefixes. Additionally, the third tag is produced from a valueless, user-defined tag scheme, so the data generator also applies true as its default value.

The last two tags illustrate examples from two different, provider-defined tag schemes. Since all provider-defined tag schemes require a prefix, the data generator has prepended predefined and reserved prefixes (providerDefinedTagScheme1/, providerDefinedTagScheme2/) to each tag.

```

{
  "foo": "bar",
  "userDefinedTagScheme2/foo": "bar",
  "userDefinedTagScheme3/foo": true,
  "providerDefinedTagScheme1/foo": "bar",
  "providerDefinedTagScheme2/foo": "bar"
}

```

3.1.65.3. Finalized Tags

Within a data generator, tag keys may be associated with multiple values, and potentially defined at different levels within the data generator, such as accounts, folders, [resource](#) and other *resource* grouping constructs. When finalizing, *data generator* must reduce these multiple levels of definition to a single value where each key is associated with exactly one value. The method by which this is done and the semantics are up to each data generator but must be documented within their respective documentation.

As an example, let's assume 1 [sub account](#) exists with 1 virtual machine with the following details, and tag inheritance favors Resources over *Sub Accounts*.

- Sub Account
 - id: *my-sub-account*
 - user-defined tags: *team:ops, env:prod*
- Virtual Machine
 - id: *my-vm*
 - user-defined tags: *team:web*

The table below represents a finalized dataset with these *resources*. It also shows the finalized state after all resource-oriented, tag inheritance rules are processed.

ResourceType	ResourceId	Tags
Sub Account	my-sub-account	{ "team": "ops", "env": "prod" }
Virtual Machine	my-vm	{ "team": "web", "env": "prod" }

Because the Virtual Machine Resource did not have an env tag, it inherited tag, env:prod (italicized), from its parent *sub account*. Conversely, because the Virtual Machine Resource already has a team tag (team:web), it did not inherit team:ops from its parent *sub account*.

3.1.65.4. Column ID

Tags

3.1.65.5. Display Name

Tags

3.1.65.6. Description

The set of tags assigned to *tag sources* that account for potential provider-defined or user-defined tag evaluations.

3.1.65.7. Content Constraints

Constraint	Value
Column type	Dimension
Feature level	Conditional
Allows nulls	True
Data type	JSON
Value format	Key-Value Format

3.1.65.8. Introduced (version)

1.0-preview

3.2. Contract Commitment

The Contract Commitment dataset is a supporting dataset that describes the terms of contracts agreed between a service provider and a customer.

Columns

Column	Column Type	Feature Level	Allows Nulls	Data Type
Billing Currency	Dimension	Mandatory	True	String
Contract Commitment Category	Dimension	Mandatory	False	String
Contract Commitment Cost	Metric	Mandatory	True	Numeric
Contract Commitment Description	Dimension	Mandatory	True	String
Contract Commitment ID	Dimension	Mandatory	False	String
Contract Commitment Period End	Dimension	Mandatory	False	Date/Time
Contract Commitment Period Start	Dimension	Mandatory	False	Date/Time
Contract Commitment Quantity	Metric	Mandatory	True	Numeric
Contract Commitment Type	Dimension	Mandatory	False	String
Contract Commitment Unit	Dimension	Mandatory	True	String
Contract ID	Dimension	Mandatory	False	String
Contract Period End	Dimension	Mandatory	False	Date/Time
Contract Period Start	Dimension	Mandatory	False	Date/Time

Relationships

The Contract Commitment dataset can be joined to the Cost and Usage dataset through the use of Contract Commitment ID.

- In the Contract Commitment dataset, Contract Commitment ID is a column.
- In the Cost and Usage dataset, Contract Commitment ID is a property within a JSON object array provided in Contract Applied column.

Dataset A	Dataset A Column	Dataset B	Dataset B Column
-----------	------------------	-----------	------------------

Dataset A	Dataset A Column	Dataset B	Dataset B Column
Contract Commitment	Contract Commitment ID	Cost and Usage	Contract Applied

Requirements

ContractCommitment adheres to the following requirements:

- ContractCommitment MUST be present when the service provider supports *contract commitments*.
- ContractCommitment MUST conform to [ColumnHandling](#) requirements.
- ContractCommitment MUST conform to [NullHandling](#) requirements.

Dataset ID

ContractCommitment

Display Name

Contract Commitment

Description

Describes the terms of contracts agreed between a service provider and a customer.

Introduced (version)

1.3

3.2.1. Billing Currency

[Billing currency](#) is an identifier that represents the currency of a [contract commitment](#).

3.2.1.1. Requirements

BillingCurrency adheres to the following requirements:

- BillingCurrency MUST be present in a Contract Commitment [FOCUS dataset](#).
- BillingCurrency MUST be of type String.
- BillingCurrency MUST conform to [StringHandling](#) requirements.
- BillingCurrency MUST conform to [CurrencyFormat](#) requirements.
- BillingCurrency MUST NOT be null when [ContractCommitmentCategory](#) is "Spend".
- BillingCurrency MUST match the currency used in the invoice generated by the invoice issuer.
- BillingCurrency MUST be expressed in [national currency](#) (e.g., USD, EUR).

3.2.1.2. Column ID

BillingCurrency

3.2.1.3. Display Name

Billing Currency

3.2.1.4. Description

Represents the currency of a *contract commitment*.

3.2.1.5. Content Constraints

Constraint	Value
Column type	Dimension
Feature level	Mandatory
Allows nulls	True
Data type	String
Value format	Currency Format

3.2.1.6. Introduced (version)

1.3

3.2.2. Contract Commitment Cost

Contract Commitment Cost represents the monetary value of the *contract commitment*. Contract Commitment Cost is commonly used for monitoring the progress towards fulfilling contractual commitments that may facilitate discounts for *resources* or *services* as agreed between a service provider and a customer.

3.2.2.1. Requirements

ContractCommitmentCost adheres to the following requirements:

- ContractCommitmentCost MUST be present in a Contract Commitment [FOCUS dataset](#).
- ContractCommitmentCost MUST be of type Decimal.
- ContractCommitmentCost MUST conform to [NumericFormat](#) requirements.
- ContractCommitmentCost nullability is defined as follows:
 - ContractCommitmentCost MUST NOT be null when [ContractCommitmentCategory](#) is "Spend".
 - ContractCommitmentCost MAY be null when ContractCommitmentCategory is "Usage".
- ContractCommitmentCost MUST be a valid decimal value.
- ContractCommitmentCost MUST be denominated in the [BillingCurrency](#).

3.2.2.2. Column ID

ContractCommitmentCost

3.2.2.3. Display Name

3.2.2.4. Description

The monetary value of the *contract commitment*.

3.2.2.5. Content Constraints

Constraint	Value
Column type	Metric
Feature level	Mandatory
Allows nulls	True
Data type	Decimal
Value format	Numeric Format
Number range	Any valid decimal value

3.2.2.6. Introduced (version)

1.3

3.2.3. Contract Commitment ID

Contract Commitment ID is a service-provider-assigned identifier describing a single contract term agreed between a provider and a customer. Contracts can include commitments to a certain amount of spend or usage over an agreed period of time.

3.2.3.1. Requirements

ContractCommitmentId adheres to the following requirements:

- ContractCommitmentId MUST be present in a Contract Commitment [FOCUS dataset](#).
- ContractCommitmentId MUST be of type String.
- ContractCommitmentId MUST conform to [StringHandling](#) requirements.
- ContractCommitmentId MUST NOT be null.
- When ContractCommitmentId is not null, ContractCommitmentId adheres to the following additional requirements:
 - ContractCommitmentId MUST be a unique identifier within the service provider.
 - ContractCommitmentId SHOULD be a fully-qualified identifier.
- ContractCommitmentId MUST have one and only one parent [ContractId](#).
- ContractCommitmentId MAY be equal to ContractId.
- ContractCommitmentId MUST be unique across the Contract Commitment dataset.

3.2.3.2. Column ID

ContractCommitmentId

3.2.3.3. Display Name

Contract Commitment ID

3.2.3.4. Description

A service-provider-assigned identifier describing a single contract term agreed between a service provider and a customer.

3.2.3.5. Content Constraints

Constraint	Value
Column type	Dimension
Feature level	Mandatory
Allows nulls	True
Data type	String
Value format	<not specified>

3.2.3.6. Introduced (version)

1.3

3.2.4. Contract Commitment Category

Contract Commitment Category represents the highest-level classification of a [contract commitment](#) based on the nature of how it is applied to a charge. Contract Commitment Category is commonly used to identify and distinguish between categories of contract commitments that may require different handling.

3.2.4.1. Requirements

ContractCommitmentCategory adheres to the following requirements:

- ContractCommitmentCategory MUST be present in a Contract Commitment [FOCUS dataset](#).
- ContractCommitmentCategory MUST be of type String.
- ContractCommitmentCategory MUST NOT be null.
- ContractCommitmentCategory MUST be one of the allowed values.

3.2.4.2. Column ID

ContractCommitmentCategory

3.2.4.3. Display Name

3.2.4.4. Description

Represents the highest-level classification of a *contract commitment* based on the nature of how it is applied to a charge.

3.2.4.5. Content Constraints

Constraint	Value
Column type	Dimension
Feature level	Mandatory
Allows nulls	False
Data type	String
Value format	Allowed values

Allowed values:

Value	Description
Spend	Contract commitments that require a predetermined amount of spend.
Usage	Contract commitments that require a predetermined amount of usage.

3.2.4.6. Introduced (version)

1.3

3.2.5. Contract Commitment Description

Contract Commitment Description provides a high-level context of a [contract commitment](#) without requiring additional discovery. Contract Commitment Description is a self-contained summary of the contract commitment's terms, which may not be sufficiently described by the other columns of the Contract Commitment dataset.

3.2.5.1. Requirements

ContractCommitmentDescription adheres to the following requirements:

- ContractCommitmentDescription MUST be present in a Contract Commitment [FOCUS dataset](#).
- ContractCommitmentDescription MUST be of type String.
- ContractCommitmentDescription MUST conform to [StringHandling](#) requirements.
- ContractCommitmentDescription SHOULD NOT be null.
- ContractCommitmentDescription maximum length SHOULD be provided in the corresponding FOCUS Metadata Schema.

3.2.5.2. Column ID

3.2.5.3. Display Name

Contract Commitment Description

3.2.5.4. Description

The self-contained summary of the *contract commitment's* terms.

3.2.5.5. Content Constraints

Constraint	Value
Column type	Dimension
Feature level	Mandatory
Allows nulls	True
Data type	String
Value format	<not specified>

3.2.5.6. Introduced (version)

1.3

3.2.6. Contract Commitment Period End

Contract Commitment Period End represents the *exclusive end bound* of a *contract commitment period*. For example, a time period where *Contract Commitment Period Start* is '2024-01-01T00:00:00Z' and *Contract Commitment Period End* is '2024-01-02T00:00:00Z' includes January 1 2024 since *Contract Commitment Period Start* represents the *inclusive start bound*, but does not include January 1 2025 since *Contract Commitment Period End* represents the *exclusive end bound*.

3.2.6.1. Requirements

ContractCommitmentPeriodEnd adheres to the following requirements:

- ContractCommitmentPeriodEnd MUST be present in a Contract Commitment [FOCUS dataset](#).
- ContractCommitmentPeriodEnd MUST be of type Date/Time.
- ContractCommitmentPeriodEnd MUST conform to [DateTimeFormat](#) requirements.
- ContractCommitmentPeriodEnd MUST NOT be null.
- ContractCommitmentPeriodEnd MUST be the *exclusive end bound* of the effective period of the *contract commitment*.

3.2.6.2. Column ID

ContractCommitmentPeriodEnd

3.2.6.3. Display Name

Contract Commitment Period End

3.2.6.4. Description

The *exclusive end bound* of a *contract commitment period*.

3.2.6.5. Content constraints

Constraint	Value
Column type	Dimension
Feature level	Mandatory
Allows nulls	False
Data type	Date/Time
Value format	Date/Time Format

3.2.6.6. Introduced (version)

1.3

3.2.7. Contract Commitment Period Start

Contract Commitment Period Start represents the *inclusive start bound* of a *contract commitment period*. For example, a time period where Contract Commitment Period Start is '2024-01-01T00:00:00Z' and [Contract Commitment End](#) is '2025-01-01T00:00:00Z' includes January 1 2024 since Contract Commitment Period Start represents the *inclusive start bound*, but does not include *charges* for January 2 2025 since Contract Commitment Period End represents the *exclusive end bound*.

3.2.7.1. Requirements

ContractCommitmentPeriodStart adheres to the following requirements:

- ContractCommitmentPeriodStart MUST be present in a Contract Commitment [FOCUS dataset](#).
- ContractCommitmentPeriodStart MUST be of type Date/Time.
- ContractCommitmentPeriodStart MUST conform to [DateTimeFormat](#) requirements.
- ContractCommitmentPeriodStart MUST NOT be null.
- ContractCommitmentPeriodStart MUST be the *inclusive start bound* of the effective period of the *contract commitment*.

3.2.7.2. Column ID

ContractCommitmentPeriodStart

3.2.7.3. Display Name

Contract Commitment Period Start

3.2.7.4. Description

The *inclusive start bound* of a *contract commitment period*.

3.2.7.5. Content constraints

Constraint	Value
Column type	Dimension
Feature level	Mandatory
Allows nulls	False
Data type	Date/Time
Value format	Date/Time Format

3.2.7.6. Introduced (version)

1.3

3.2.8. Contract Commitment Quantity

Contract Commitment Quantity represents the amount associated with the [contract commitment](#), denominated in a service-provider-defined [Contract Commitment Unit](#). Contract Commitment Quantity is commonly used for monitoring the progress towards fulfilling contractual commitments that may facilitate discounts for [resources](#) or [services](#) as agreed between a provider and a customer.

3.2.8.1. Requirements

ContractCommitmentQuantity adheres to the following requirements:

- ContractCommitmentQuantity MUST be present in a Contract Commitment [FOCUS dataset](#).
- ContractCommitmentQuantity MUST be of type Decimal.
- ContractCommitmentQuantity MUST conform to [NumericFormat](#) requirements.
- ContractCommitmentQuantity nullability is defined as follows:
 - ContractCommitmentQuantity MUST NOT be null when [ContractCommitmentCategory](#) is "Usage".
 - ContractCommitmentQuantity MAY be null when ContractCommitmentCategory is "Spend".
- ContractCommitmentQuantity MUST be a valid decimal value.

3.2.8.2. Column ID

ContractCommitmentQuantity

3.2.8.3. Display Name

Contract Commitment Quantity

3.2.8.4. Description

The amount associated with the *contract commitment*.

3.2.8.5. Content Constraints

Constraint	Value
Column type	Metric
Feature level	Mandatory
Allows nulls	True
Data type	Decimal
Value format	Numeric Format
Number range	Any valid decimal value

3.2.8.6. Introduced (version)

1.3

3.2.9. Contract Commitment Type

Contract Commitment Type is a service-provider-assigned name to identify the type of [contract commitment](#). Contract Commitment Type is a readable display name and not a code. Contract Commitment Type is commonly used for displaying and aggregating the types of commitments the practitioner has made, stated in service-provider-specific terms.

3.2.9.1. Requirements

ContractCommitmentType adheres to the following requirements:

- ContractCommitmentType MUST be present in a Contract Commitment [FOCUS dataset](#).
- ContractCommitmentType MUST be of type String.
- ContractCommitmentType MUST conform to [StringHandling](#) requirements.
- ContractCommitmentType MUST NOT be null.
- ContractCommitmentType MUST be a consistent, readable display value.

3.2.9.2. Column ID

ContractCommitmentType

3.2.9.3. Display Name

Contract Commitment Type

3.2.9.4. Description

A service-provider-assigned name to identify the type of *contract commitment*.

3.2.9.5. Content Constraints

Constraint	Value
Column type	Dimension
Feature level	Mandatory
Allows nulls	False
Data type	String
Value format	<not specified>

3.2.9.6. Introduced (version)

1.3

3.2.10. Contract Commitment Unit

The Contract Commitment Unit represents a service-provider-specified measurement unit for the amount declared in Contract Commitment Quantity. Contract Commitment Unit complements the Contract Commitment Quantity metric.

3.2.10.1. Requirements

ContractCommitmentUnit adheres to the following requirements:

- ContractCommitmentUnit MUST be present in a Contract Commitment [FOCUS dataset](#).
- ContractCommitmentUnit MUST be of type String.
- ContractCommitmentUnit MUST conform to [StringHandling](#) requirements.
- ContractCommitmentUnit SHOULD conform to [UnitFormat](#) requirements.
- ContractCommitmentUnit nullability is defined as follows:
 - ContractCommitmentUnit MUST be null when ContractCommitmentQuantity is null.
 - ContractCommitmentUnit MUST NOT be null when ContractCommitmentQuantity is not null.

3.2.10.2. Column ID

ContractCommitmentUnit

3.2.10.3. Display Name

Contract Commitment Unit

3.2.10.4. Description

A service-provider-specified measurement unit for the amount declared in Contract Commitment Quantity.

3.2.10.5. Content Constraints

Constraint	Value
Column type	Dimension
Feature level	Mandatory
Allows nulls	True
Data type	String
Value format	Unit Format recommended

3.2.10.6. Introduced (version)

1.3

3.2.11. Contract ID

Contract ID is a service-provider-assigned identifier for a contract describing the agreed terms between a service provider and a customer. Contracts can include commitment to a certain amount of spend or usage over an agreed period of time.

3.2.11.1. Requirements

ContractId adheres to the following requirements:

- ContractId MUST be present in a Contract Commitment [FOCUS dataset](#).
- ContractId MUST be of type String.
- ContractId MUST conform to [StringHandling](#) requirements.
- ContractId MUST NOT be null.
- When ContractId is not null, ContractId adheres to the following additional requirements:
 - ContractId MUST be a unique identifier within the service provider.
 - ContractId SHOULD be a fully-qualified identifier.

3.2.11.2. Column ID

ContractId

3.2.11.3. Display Name

Contract ID

3.2.11.4. Description

A service-provider-assigned identifier for a contract describing the agreed terms between a service provider and a customer.

3.2.11.5. Content constraints

Constraint	Value
Column type	Dimension
Feature level	Mandatory
Allows nulls	True
Data type	String
Value format	<not specified>

3.2.11.6. Introduced (version)

1.3

3.2.12. Contract Period End

Contract Period End represents the *exclusive end bound* of a *contract period*. For example, a time period where *Contract Period Start* is '2024-01-01T00:00:00Z' and Contract Period End is '2024-01-02T00:00:00Z' includes January 1 2024 since Contract Period Start represents the *inclusive start bound*, but does not include January 1 2025 since Contract Period End represents the *exclusive end bound*.

3.2.12.1. Requirements

ContractPeriodEnd adheres to the following requirements:

- ContractPeriodEnd MUST be present in a Contract Commitment *FOCUS dataset*.
- ContractPeriodEnd MUST be of type Date/Time.
- ContractPeriodEnd MUST conform to *DateTimeFormat* requirements.
- ContractPeriodEnd MUST NOT be null.
- ContractPeriodEnd MUST be the *exclusive end bound* of the effective period of the *contract*.

3.2.12.2. Column ID

ContractPeriodEnd

3.2.12.3. Display Name

Contract Period End

3.2.12.4. Description

The *exclusive end bound* of a *contract period*.

3.2.12.5. Content constraints

Constraint	Value
Column type	Dimension
Feature level	Mandatory
Allows nulls	False
Data type	Date/Time
Value format	Date/Time Format

3.2.12.6. Introduced (version)

1.3

3.2.13. Contract Period Start

Contract Period Start represents the *inclusive start bound* of a *contract period*. For example, a time period where Contract Period Start is '2024-01-01T00:00:00Z' and [Contract Period End](#) is '2025-01-01T00:00:00Z' includes January 1 2024 since Contract Period Start represents the *inclusive start bound*, but does not include January 2 2025 since Contract Period End represents the *exclusive end bound*.

3.2.13.1. Requirements

ContractPeriodStart adheres to the following requirements:

- ContractPeriodStart MUST be present in a Contract Commitment [FOCUS dataset](#).
- ContractPeriodStart MUST be of type Date/Time.
- ContractPeriodStart MUST conform to [DateTimeFormat](#) requirements.
- ContractPeriodStart MUST NOT be null.
- ContractPeriodStart MUST be the *inclusive start bound* of the effective period of the *contract*.

3.2.13.2. Column ID

ContractPeriodStart

3.2.13.3. Display Name

Contract Period Start

3.2.13.4. Description

The *inclusive start bound* of a *contract period*.

3.2.13.5. Content constraints

Constraint	Value
Column type	Dimension
Feature level	Mandatory
Allows nulls	False
Data type	Date/Time
Value format	Date/Time Format

3.2.13.6. Introduced (version)

1.3

4. Attributes

Attributes are requirements that apply either across a [FOCUS dataset](#) or an individual column level. An attribute is associated with a dataset or column in their respective Requirements sections. Requirements on data content can include naming conventions, data types, formatting standardizations, etc. Attributes may introduce high-level requirements for data granularity, recency, frequency, etc. Requirements defined in attributes are necessary for servicing [FinOps capabilities](#) accurately using a standard set of instructions regardless of the origin of the data.

4.1. Column Handling

A [FOCUS dataset](#) consists of a set of columns that convey information about the charges incurred with a service provider. Each column describes an aspect of the charge, including but not limited to:

- Who is responsible for incurring or delivering the service.
- What the charge is for.
- When the charge was incurred.
- Where the service was delivered.
- Why the charge was incurred for a specific price.
- How much the charge is and how that cost is calculated.

While FOCUS establishes the core structure and standardizes columns for consistent reporting of cost and usage data, the diverse and evolving landscape of service providers and service offerings may require service providers and data generators to include supplemental columns in the FOCUS dataset. These additional columns may enable deeper analysis and provide more detailed descriptions of usage that may not be fully captured by standard FOCUS dataset columns.

In such cases, service providers and data generators are responsible for ensuring that their usage and cost data is accurately and comprehensively represented by including necessary supplemental columns without duplicating data in FOCUS columns. Rows in a FOCUS dataset may be aggregated or split differently than non-FOCUS datasets to align with FOCUS requirements (e.g., Discount Handling), while enriching the dataset, providers and data generators must maintain the integrity of FOCUS-defined dimensions and metrics. When performing these transformations, providers and data generators must ensure the accuracy of all dimensions and metrics, particularly summable values such as costs and quantities.

Columns within FOCUS include an ID and a display name. Column IDs are used in files and database tables and display names can be used in report output and other descriptive content, like documentation. Column IDs provided in a *FOCUS dataset* follow consistent naming and ordering conventions for FinOps practitioners who consume the data for analysis, reporting, and other use cases.

All columns defined in the FOCUS specification MUST follow the naming and ordering requirements listed below.

4.1.1. Attribute ID

ColumnHandling

4.1.2. Attribute Name

Column Handling

4.1.3. Description

Naming and ordering convention for columns appearing in a *FOCUS dataset*.

4.1.4. Requirements

4.1.4.1. Column Names

- All columns defined by FOCUS MUST follow the following rules:
 - Column IDs MUST use [Pascal case](#).
 - Column IDs MUST NOT use abbreviations.
 - Column IDs MUST be alphanumeric with no special characters.
 - Column IDs SHOULD NOT use acronyms.
 - Column IDs SHOULD NOT exceed 50 characters to accommodate column length restrictions of various data repositories.
 - Columns that have an ID and a Name MUST have the ID or Name suffix in the Column ID.
 - Column display names MUST be consistent with their Column IDs, with spaces inserted between words (e.g., Column ID "BillingAccountName" and display name "Billing Account Name").
 - Columns with the Category suffix MUST be normalized.
- Custom (e.g., service-provider-defined) columns that are not defined by FOCUS but included in a *FOCUS dataset* MUST follow the following rules:
 - Custom columns MUST be prefixed with a consistent x_ prefix to identify them as

external, custom columns and distinguish them from FOCUS columns to avoid conflicts in future releases.

- Custom columns SHOULD follow the same rules listed above for FOCUS columns.

4.1.4.2. Column Order

- All FOCUS columns SHOULD be first in the provided dataset.
- Custom columns SHOULD be listed after all FOCUS columns and SHOULD NOT be intermixed.
- Columns MAY be sorted alphabetically, but custom columns SHOULD be after all FOCUS columns.

4.1.5. Exceptions

- Identifiers will use the "Id" abbreviation since this is a standard pattern across the industry.
- Product offerings that incur charges will use the "Sku" abbreviation because it is a well-understood term both within and outside the industry.

4.1.6. Introduced (version)

0.5

4.2. Currency Format

Columns that contain currency information in cost data following a consistent format reduce friction for FinOps practitioners who consume the data for analysis, reporting, and other use cases.

A currency may be one of the following currency types:

- National currency (e.g., USD, EUR).
- Virtual currency (e.g., tokens, credits).

All columns capturing a currency value, defined in the FOCUS specification, MUST follow the requirements listed below. Custom currency-related columns SHOULD also follow the same formatting requirements.

4.2.1. Attribute ID

CurrencyFormat

4.2.2. Attribute Name

Currency Format

4.2.3. Description

Formatting for currency columns appearing in a [FOCUS dataset](#).

4.2.4. Requirements

- Currency-related columns MUST be represented as a three-letter alphabetic code as dictated in the governing document [ISO 4217:2015](#) when the value is presented in national currency (e.g., USD, EUR).
- Currency-related columns MUST conform to [StringHandling](#) requirements when the value is presented in virtual currency (e.g., credits, tokens).

4.2.5. Exceptions

None

4.2.6. Introduced (version)

0.5

4.3. Data Generator-Calculated Split Cost Allocation Handling

The data generator-calculated split cost allocation for data generator-defined services is a capability that can be offered by data generators which allocates (or in some cases provides more granular detail about) a charge to a more granular level. This is accomplished by taking a charge record present in a FOCUS dataset (origin charge) and splitting it into multiple charge records (allocated charges) to reflect the more granular detail, while ensuring the origin charge can be derived from the combination of allocated charges. This feature is used by practitioners to conduct chargebacks and better understand the usage of resources.

4.3.1. Attribute ID

DataGeneratorCalculatedSplitCostAllocationHandling

4.3.2. Attribute Name

Data Generator-Calculated Split Cost Allocation Handling

4.3.3. Description

An attribute that allows data generators to offer more detailed cost and usage information based on a method defined and documented by the data generator, including support for allocating costs in cases where the usage of a resource might not match the units the resource is measured in.

4.3.4. Requirements

- A FOCUS dataset MUST include the following columns when the data generator supports data generator-calculated split cost allocation:
 - [AllocatedMethodId](#)

- [AllocatedResourceId](#)
- [AllocatedResourceName](#)
- [AllocatedResourceTags](#)
- A FOCUS dataset SHOULD include the following column when the data generator supports data generator-calculated split cost allocation:
 - [AllocatedMethodDetails](#)
- Allocated charge records in a FOCUS dataset MUST sum up to the origin charge record for all aggregatable metric columns.
- For each allocated charge records in a FOCUS dataset, all dimension columns and non-aggregatable metric columns MUST match the values of the origin charge record.
- Allocated charge records MUST include the same keys and values present in the [Tags](#) column for the origin charge.
- Allocated charge records MUST satisfy normative requirements for all columns.
- The method used for allocating origin charges to create allocated charges MUST be documented by the data generator and accessible to practitioners.
- A FOCUS dataset MAY contain records for concepts not related to resource usage, if documented in the split cost allocation method.
- A FOCUS dataset MAY contain records for the unused or unallocated usage from the origin charge as separate allocated charges, if it aligns to the data generator's documented allocation method.
- Allocated charge records MAY contain apportioned costs for the unused or unallocated usage from the origin charge, if it aligns to the data generator's documented allocation method.
- Split cost allocation is RECOMMENDED to be applied to charges on an opt-in basis.

4.3.5. Exceptions

None

4.3.6. Introduced (version)

1.3

4.4. Date/Time Format

Columns that provide date and time information conforming to specified rules and formatting requirements ensure clarity, accuracy, and ease of interpretation for both humans and systems.

All columns capturing a date/time value, defined in the FOCUS specification, MUST follow the formatting requirements listed below. Custom date/time-related columns SHOULD also follow the same formatting requirements.

4.4.1. Attribute ID

DateTimeFormat

4.4.2. Attribute Name

Date/Time Format

4.4.3. Description

Rules and formatting requirements for date/time-related columns appearing in a [FOCUS dataset](#).

4.4.4. Requirements

- Date/time values MUST be in UTC (Coordinated Universal Time) to avoid ambiguity and ensure consistency across different time zones.
- Date/time values format MUST be aligned with ISO 8601 standard, which provides a globally recognized format for representing dates and times (see [ISO 8601-1:2019](#) governing document for details).
- Values providing information about a specific moment in time MUST be represented in the extended ISO 8601 format with UTC offset ('YYYY-MM-DDTHH:mm:ssZ') and conform to the following guidelines:
 - Include the date and time components, separated with the letter 'T'
 - Use two-digit hours (HH), minutes (mm), and seconds (ss).
 - End with the 'Z' indicator to denote UTC (Coordinated Universal Time)

4.4.5. Exceptions

None

4.4.6. Introduced (version)

0.5

4.5. Discount Handling

A discount is a pricing construct where service providers offer a reduced price for [services](#). Service providers may have many types of discounts, including but not limited to commercially [negotiated discounts](#), [commitment discounts](#) when you agree to a certain amount of usage or spend, and bundled discounts where you receive free or discounted usage of one product or *service* based on the usage of another. Discount Handling is commonly used in scenarios like verifying discounts were applied and calculating cost savings.

Some discount offers can be purchased from a service provider to get reduced prices. The most common example is a *commitment discount*, where you "purchase" a commitment to use or spend a specific amount within a period. When a commitment isn't fully utilized, the unused amount reduces the potential savings from the discount and can even result in paying higher costs than without the discount. Due to this risk, unused commitment amounts need to be clearly identifiable at a granular level. To facilitate this, unused commitments are recorded with a separate row for each charge period where the commitment was not fully utilized. To show the impact of purchased discounts on each discounted row, discount purchases need the purchase amount to be amortized over the [period](#) the discount is applied to (e.g., 1 year) with each [charge period](#) split and applied to each row that received the discount.

Amortization is a process used to break down and spread purchase costs over a *period* of time. When a purchase is applicable to resources, like *commitment discounts*, the amortized cost of a resource takes the initial payment and *period* into account and distributes it out based on the resource's usage, attributing the prorated cost for each unit of billing. Amortization enables users of billing data to distribute purchase charges to the appropriate audience in support of cost allocation efforts. Discount Handling for purchased commitments is commonly used for scenarios like calculating utilization and implementing chargeback for the purchase amount.

While service providers may use different terms to describe discounts, FOCUS identifies a discount as being a reduced price applied directly to a row. Any price or cost reductions that are awarded after the fact are identified as a "Credit" Charge Category. One example might be when a service provider offers a reduced rate after passing a certain threshold of usage or spend.

All rows defined in FOCUS MUST follow the discount handling requirements listed below.

4.5.1. Attribute ID

DiscountHandling

4.5.2. Attribute Name

Discount Handling

4.5.3. Description

Indicates how to include and apply discounts to usage charges or rows in a FOCUS dataset.

4.5.4. Requirements

- All applicable discounts SHOULD be applied to each row they pertain to and SHOULD NOT be negated in a separate row.
- All discounts applied to a row MUST apply to the entire charge.
 - Multiple discounts MAY apply to a row, but they MUST apply to the entire charge covered by that row.
 - If a discount only applies to a portion of a charge, then the discounted portion of the charge MUST be split into a separate row.
 - Each discount MUST be identifiable using existing FOCUS columns.
 - Rows with a *commitment discount* applied to them MUST include a CommitmentDiscountId.
 - If a service provider applies a discount that cannot be represented by a FOCUS column, they SHOULD include additional columns to identify the source of the discount.
- Purchased discounts (e.g., *commitment discounts*) MUST be amortized.
 - The BilledCost MUST be 0 for any row where the commitment covers the entire cost for the charge period.
 - The EffectiveCost MUST include the portion of the amortized purchase cost that applies to this row.
 - The sum of the EffectiveCost for all rows where CommitmentDiscountStatus is "Used" or "Unused" for each CommitmentDiscountId over the entire commitment *period* MUST be the same as the total BilledCost of the *commitment discount*.
 - The CommitmentDiscountId and ResourceId MUST be set to the ID assigned to the *commitment discount*. ChargeCategory MUST be set to "Purchase" on rows that represent a purchase of a *commitment discount*.
 - CommitmentDiscountStatus MUST be "Used" for ChargeCategory "Usage" rows that received a reduced price from a commitment. CommitmentDiscountId MUST be set to the ID assigned to the discount. ResourceId MUST be set to the ID of the resource that received the discount.
 - If a commitment is not fully utilized, the service provider MUST include a row that represents the unused portion of the commitment for that *charge period*. These rows

MUST be represented with CommitmentDiscountStatus set to "Unused" and ChargeCategory set to "Usage". Such rows MUST have their CommitmentDiscountId and ResourceId set to the ID assigned to the *commitment discount*.

- Credits that are applied after the fact MUST use a ChargeCategory of "Credit".

4.5.5. Exceptions

None

4.5.6. Introduced (version)

1.0-preview

4.6. JSON Object Format

JSON Objects extend the [Key-Value Format](#) to add support for complex data types like arrays and nested key-value pairs. This format is used when the Key-Value Format is insufficient to represent the complexity, such as when multiple sets of key-value pairs apply to the same charge record. JSON Objects are also referred to as maps, trees, or hashtables.

All complex JSON Object columns defined in the FOCUS specification MUST follow the object formatting requirements listed below.

4.6.1. Attribute ID

JsonObjectFormat

4.6.2. Attribute Name

JSON Object Format

4.6.3. Description

Rules and formatting requirements for columns appearing in a [FOCUS dataset](#) that convey data as complex, hierarchical objects.

4.6.4. Requirements

- JsonObjectFormat columns MUST contain a serialized JSON string, consistent with the [ECMA 404](#) definition of an object.
- Objects used within JsonObjectFormat adhere to the following additional requirements:
 - Object keys MUST be unique within an object.
 - Object values MUST be one of the following types: number, string, true, false, array, object, or null.
- Arrays used within JsonObjectFormat adhere to the following additional requirements:
 - Array elements MUST all use the same, consistent type.
 - Array elements MUST NOT be repeated.

- Array elements MUST NOT be null.
- JsonObjectFormat columns MUST conform to all requirements of the corresponding column definition, which may specify or restrict the shape or contents of the Object.
- Data Generator-defined [custom columns](#) whose contents contain a JSON object MUST have their object schema documented by the data generator.
- JsonObjectFormat objects SHOULD NOT exceed 3 levels of nesting.

4.6.5. Exceptions

None

4.6.6. Introduced (version)

1.3

4.7. Invoice Handling

FinOps practitioners must be able to reconcile FOCUS datasets with the corresponding invoices and usage statements they receive from [Invoice Issuers](#). In practice, this means ensuring that all monetary [charges](#) that appear on an invoice or usage statement — including those not tied to metered usage — are represented in the [FOCUS dataset](#). Without this alignment, it becomes difficult to perform accurate invoice reconciliation, financial reporting, and chargeback.

This attribute introduces requirements for how charges such as usage, taxes, credits, refunds, etc, inclusive of support, training, and marketplace transactions, and any other type of charge should be captured and categorized. It also defines expectations around the completeness and consistency of invoice-level totals within the dataset, enabling FOCUS datasets to be used in a system of record for all invoiced costs.

4.7.1. Attribute ID

InvoiceHandling

4.7.2. Attribute Name

Invoice Handling

4.7.3. Description

Indicates how invoice-level *charges*, including those not directly tied to usage, should be represented in a FOCUS Cost and Usage dataset.

4.7.4. Requirements

- All costs that appear on any invoice issued to a [BillingAccountId](#) MUST be included in the *FOCUS dataset*.

- If an invoice-level *charge* appears on a customer invoice but cannot be expressed using existing FOCUS columns, data generators MUST include provider-defined columns (e.g., *x_ChargeSubType*) to capture the non-FOCUS-defined details needed to support invoice *charges* reconciliation using the *FOCUS dataset*.

4.7.5. Exceptions

- Informational line items that have zero monetary impact and are included solely for transparency MAY be excluded. Examples include:
 - Tax exemption notifications
 - SLA credit details when the credit is already applied to the charged amount
- If such informational items are excluded, data generators MUST document this in their FOCUS implementation guide and ensure the sum of included charges still equals the invoice total.

4.7.6. Introduced (version)

1.3

4.8. Key-Value Format

Columns that provide Key-Value information are often used in place of separate columns for enumerating data which would be inherently sparse and/or without predetermined keys. This consolidates related information and provides more consistency in the schema. Key-value pairs are also referred to as name-value pairs, attribute-value pairs, or field-value pairs.

All key-value related columns defined in the FOCUS specification MUST follow the key-value formatting requirements listed below.

4.8.1. Attribute ID

KeyValueFormat

4.8.2. Attribute Name

Key-Value Format

4.8.3. Description

Rules and formatting requirements for columns appearing in a [FOCUS dataset](#) that convey data as key-value pairs.

4.8.4. Requirements

- Key-Value Format columns MUST contain a serialized JSON string, consistent with the [ECMA 404](#) definition of an object.
- Keys in a key-value pair MUST be unique within an object.

- Values in a key-value pair MUST be one of the following types: number, string, true, false, or null.
- Values in a key-value pair MUST NOT be an object or an array.

4.8.5. Exceptions

None

4.8.6. Introduced (version)

1.0-preview

4.9. Null Handling

Cost data [rows](#) that don't have a value that can be presented for a column must be handled in a consistent way to reduce friction for FinOps practitioners who consume the data for analysis, reporting, and other use cases.

All columns defined in the FOCUS specification MUST follow the null handling requirements listed below. Custom columns SHOULD also follow the same formatting requirements.

4.9.1. Attribute ID

NullHandling

4.9.2. Attribute Name

Null Handling

4.9.3. Description

Indicates how to handle columns that don't have a value.

4.9.4. Requirements

- Columns MUST use NULL when there isn't a value that can be specified for a nullable column.
- Columns MUST NOT use empty strings or placeholder values such as 0 for numeric columns or "Not Applicable" for string columns to represent a null or not having a value, regardless of whether the column allows nulls or not.

4.9.5. Exceptions

None

4.9.6. Introduced (version)

0.5

4.10. Numeric Format

Columns that provide numeric values conforming to specified rules and formatting requirements ensure clarity, accuracy, and ease of interpretation for humans and systems. The FOCUS specification does not require a specific level of precision for numeric values. The level of precision required for a given column is determined by the provider and should be part of a data definition published by the provider.

All columns capturing a numeric value, defined in the FOCUS specification, MUST follow the formatting requirements listed below. Custom numeric value capturing columns SHOULD adopt the same format requirements over time.

4.10.1. Attribute ID

NumericFormat

4.10.2. Attribute Name

Numeric Format

4.10.3. Description

Rules and formatting requirements for numeric columns appearing in a [FOCUS dataset](#).

4.10.4. Requirements

- Columns with a Numeric value format MUST contain a single numeric value.
- Numeric values MUST be expressed as an integer value, a decimal value, or a value expressed in scientific notation. Fractional notation MUST NOT be used.
- Numeric values expressed using scientific notation MUST be expressed using E notation "mEn" with a real number m and an integer n indicating a value of " $m \times 10^n$ ". The sign of the exponent MUST only be expressed as part of the exponent value if n is negative.
- Numeric values MUST NOT be expressed with mathematical symbols, functions, or operators.
- Numeric values MUST NOT contain qualifiers or additional characters (e.g., currency symbols, units of measure, etc.).
- Numeric values MUST NOT contain commas or punctuation marks except for a single decimal point (".") if required to express a decimal value.
- Numeric values MUST NOT include a character to represent a sign for a positive value. A negative sign (-) MUST indicate a negative value.
- Columns with a Numeric value format MUST present one of the following values as the "Data type" in the column definition.
 - Allowed values:

Data Type	Type Description
-----------	------------------

Data Type	Type Description
Integer	Specifies a numeric value represented by a whole number or by zero. Integer number formats correspond to standard data types defined by ISO/IEC 9899:2018
Decimal	Specifies a numeric value represented by a decimal number. Decimal formats correspond to ISO/IEC/IEEE 60559:2011 and IEEE 754-2008 definitions.

- Providers SHOULD define precision and scale for Numeric Format columns using one of the following precision values in a data definition document that providers publish.

- Allowed values:

Data Type	Precision	Definition	Range / Significant Digits
Integer	Short	16-bit signed short int ISO/IEC 9899:2018	-32,767 to +32,767
Integer	Long	32-bit signed long int ISO/IEC 9899:2018	-2,147,483,647 to +2,147,483,647
Integer	Extended	64-bit signed two's complement integer <i>or higher</i>	$-(2^{63} - 1)$ to $(2^{63} - 1)$
Decimal	Single	32-bit binary format IEEE 754-2008 floating-point (decimal32)	9
Decimal	Double	64-bit binary format IEEE 754-2008 floating-point (decimal64)	16
Decimal	Extended	128-bit binary format IEEE 754-2008 floating-point (decimal128) or higher	36+

4.10.4.1. Examples

This format requires that single numeric values be represented using an integer or decimal format without additional characters or qualifiers. The following lists provide examples of values that meet the requirements and those that do not.

- Values Meeting Numeric Requirements:
 - 100.2
 - 3
 - 4
 - 35.2E-7
 - 1.234
- Values NOT Meeting Numeric Requirements
 - 1 1/2 - contains fractional notation
 - 35.2E+7 - contains a positive exponent with a sign
 - 35.24 x 10⁷ - contains an invalid format for scientific notation
 - [3,5,8] - contains an array
 - [4:5] - contains a range
 - 5i + 4 - contains a complex number
 - sqrt(2) - contains a mathematical symbol or operation
 - 2.3³ - contains an exponent
 - 32 GiB - contains a unit of measure

- \$32 - contains a currency symbol
- 3,432,342 - contains a comma
- +333 - contains a positive sign

4.10.5. Exceptions

None

4.10.6. Introduced (version)

1.0-preview

4.11. String Handling

Columns that capture string values conforming to specified requirements foster data integrity, interoperability, and consistency, improve data analysis and reporting, and support reliable data-driven decision-making.

All columns capturing a string value, defined in the FOCUS specification, MUST follow the requirements listed below. Custom string value capturing columns SHOULD adopt the same requirements over time.

4.11.1. Attribute ID

StringHandling

4.11.2. Attribute Name

String Handling

4.11.3. Description

Requirements for string-capturing columns appearing in a [FOCUS dataset](#).

4.11.4. Requirements

- String values MUST maintain the original casing, spacing, and other relevant consistency factors as specified by data generators and end-users.
- [Charges](#) to mutable entities (e.g., resource names) MUST be accurately reflected in corresponding *charges* incurred after the change and MUST NOT alter *charges* incurred before the change, preserving data integrity and auditability for all *charge* records.
- Immutable string values that refer to the same entity (e.g., resource identifiers, region identifiers, etc.) MUST remain consistent and unchanged across all [billing periods](#).
- Empty strings and strings consisting solely of spaces SHOULD NOT be used in not-nullable string columns.

4.11.5. Exceptions

- When a record is provided after a change to a mutable string value and the [ChargeClass](#) is "Correction", the record MAY contain the altered value.

4.11.6. Introduced (version)

1.0

4.12. Unit Format

Billing data frequently captures data measured in units related to data size, count, time, and other [dimensions](#). The Unit Format attribute provides a standard for expressing units of measure in columns appearing in a [FOCUS dataset](#).

All columns defined in FOCUS specifying Unit Format as a value format MUST follow the requirements listed below.

4.12.1. Attribute ID

UnitFormat

4.12.2. Attribute Name

Unit Format

4.12.3. Description

Indicates standards for expressing measurement units in columns appearing in a *FOCUS dataset*.

4.12.4. Requirements

- Units SHOULD be expressed as a single unit of measure adhering to one of the following three formats.
 - `<plural-units>` - "GB", "Seconds"
 - `<singular-unit>-<plural-time-units>` - "GB-Hours", "MB-Days"
 - `<plural-units>/<singular-time-unit>` - "GB/Hour", "PB/Day"
- Units MAY be expressed with a unit quantity or time interval. If a unit quantity or time interval is used, the unit quantity or time interval MUST be expressed as a whole number. The following formats are valid:
 - `<quantity> <plural-units>` - "1000 Tokens", "1000 Characters"
 - `<plural-units>/<interval> <plural-time-units>` - "Units/3 Months"
- Unit values and components of columns using the Unit Format MUST use a capitalization scheme that is consistent with the capitalization scheme used in this attribute if that term is listed in this section. For example, a value of "gigabyte-seconds" would not be compliant with this specification as the terms "gigabyte" and "second" are listed in this section with the appropriate capitalization. If the unit is not listed in the table, it is to be used over a functional equivalent with a similar meaning with the same capitalization scheme.
- Units SHOULD be composed of the list of recommended units listed in this section unless the

unit value covers a *dimension* not listed in the recommended unit set, or if the unit covers a count-based unit distinct from recommended values in the count *dimension* listed in this section.

4.12.4.1. Data Size Unit Names

Data size unit names MUST be abbreviated using one of the abbreviations in the following table. For example, a unit name of "TB" is a valid unit name, and a unit name of "terabyte" is an invalid unit name. Data size abbreviations can be considered both the singular and plural form of the unit. For example, "GB" is both the singular and plural form of the unit "gigabyte", and "GBs" would be an invalid unit name. Values that exceed 10^{18} MUST use the abbreviation for exabit, exabyte, exhibit, and exbibyte, and values smaller than a byte MUST use the abbreviation for bit or byte. For example, the abbreviation "YB" for "yottabyte" is not a valid data size unit name as it represents a value larger than what is listed in the following table.

The following table lists the valid abbreviations for data size units from a single bit or byte to 10^{18} bits or bytes.

Data size in bits	Data size in bytes
b (bit) = 10^1	B (byte = 10^1)
Kb (kilobit = 10^3)	KB (kilobyte = 10^3)
Mb (megabit = 10^6)	MB (megabyte = 10^6)
Gb (gigabit = 10^9)	GB (gigabyte = 10^9)
Tb (terabit = 10^{12})	TB (terabyte = 10^{12})
Pb (petabit = 10^{15})	PB (petabyte = 10^{15})
Eb (exabit = 10^{18})	EB (exabyte = 10^{18})
Kib (kibibit = 2^{10})	KiB (kibibyte = 2^{10})
Mib (mebibit = 2^{20})	MiB (mebibyte = 2^{20})
Gib (gibibit = 2^{30})	GiB (gibibyte = 2^{30})
Tib (tebibit = 2^{40})	TiB (tebibyte = 2^{40})
Pib (pebibit = 2^{50})	PiB (pebibyte = 2^{50})
Eib (exbibit = 2^{60})	EiB (exbibyte = 2^{60})

4.12.4.2. Count-based Unit Names

A count-based unit is a noun that represents a discrete number of items, events, or actions. For example, a count-based unit can be used to represent the number of requests, instances, tokens, or connections.

If the following list of recommended values does not cover a count-based unit, a service provider/data generator MAY introduce a new noun representing a count-based unit. All nouns appearing in units that are not listed in the recommended values table will be considered count-based units. A new count-based unit value MUST be capitalized.

Count
Count
Unit
Request
Token
Connection
Certificate

Count
Domain
Core

4.12.4.3. Time-based Unit Names

A time-based unit is a noun that represents a time interval. Time-based units can be used to measure consumption over a time interval or in combination with another unit to capture a rate of consumption. Time-based units MUST match one of the values listed in the following table.

Time
Year
Month
Day
Hour
Minute
Second

4.12.4.4. Composite Units

If the unit value is a composite value made from combinations of one or more units, each component MUST also align with the set of recommended values.

Instead of "per" or "-" to denote a Composite Unit, slash ("/") and space(" ") MUST be used as a common convention. Count-based units like requests, instances, and tokens SHOULD be expressed using a value listed in the count *dimension*. For example, if a usage unit is measured as a rate of requests or instances over a period of time, the unit SHOULD be listed as "Requests/Day" to signify the number of requests per day.

4.12.5. Exceptions

None

4.12.6. Introduced (version)

1.0-preview

5. Metadata

The FOCUS specification defines a metadata structure to be supplied by data providers to facilitate practitioners' use of FOCUS data. This metadata includes general information about the [dataset artifact](#).

The metadata includes the following sections:

Metadata Section	Description
Data Generator	Describes the entity delivering the dataset artifact.
Dataset Instance	Describes the nature of the dataset artifact.
Recency	Describes the recency and completeness of data within the artifact.
Schema	Describes the schema of data within the artifact.

Requirements

Metadata adheres to the following requirements:

- Data generators SHOULD provide FOCUS metadata in a format that is accessible programmatically, such as a file, website, API, or table.
- Data generators SHOULD provide documentation on their implementation of the FOCUS metadata.

Metadata ID

Metadata

Metadata Name

Metadata

Introduced (version)

1.0

5.1. Data Generator

The FOCUS metadata about the generator of the FOCUS data.

5.1.1. Requirements

DataGenerator adheres to the following requirements:

- DataGenerator MUST be present in the [Metadata](#).
- DataGenerator MUST be of type Object.
- DataGenerator MUST NOT be null.

5.1.2. Metadata ID

DataGenerator

5.1.3. Metadata Name

Data Generator

5.1.4. Examples

For an example of the FOCUS Data Generator metadata please refer to: [Data Generator Example](#).

5.1.5. Introduced (version)

1.0

5.1.6. Data Generator

Human-readable name of the entity that generated the dataset instance. The Data Generator ensures the technical accuracy and delivery of the data.

DataGenerator property adheres to the following requirements:

- DataGenerator MUST be present in the [Data Generator](#) object.
- DataGenerator MUST be of type String.
- DataGenerator MUST conform to [StringHandling](#) requirements.
- DataGenerator MUST NOT be null.
- DataGenerator SHOULD reflect the entity that generated the *FOCUS dataset*.

5.1.6.1. Metadata ID

DataGenerator

5.1.6.2. Metadata Name

Data Generator

5.1.6.3. Content constraints

Constraint	Value
Feature level	Mandatory
Allows nulls	False
Data type	String
Value format	<not specified>

5.1.6.4. Introduced (version)

5.2. Dataset Instance

The Dataset Instance metadata object provides information about the [dataset instance](#) and its content. Dataset Instance is a data generator-delivered instance of a FOCUS Dataset. For example, a Data Generator may provide multiple datasets utilizing the FOCUS specification, including multiple instances of the FOCUS Cost and Usage dataset, each representing a different time granularity.

5.2.1. Requirements

DatasetInstance adheres to the following requirements:

- DatasetInstance MUST be present in the [Metadata](#).
- DatasetInstance MUST be structured as a collection of objects.
- DatasetInstance MUST NOT be null.
- DatasetInstance collection MUST contain at least one object for every [FOCUS dataset](#) supported by the data generator.
- DatasetInstance collection object MUST NOT be null.
- DatasetInstance collection object MUST be associated with one and only one *FOCUS dataset*.

5.2.2. Metadata ID

DatasetInstance

5.2.3. Metadata Name

Dataset Instance

5.2.4. Examples

For an example of the FOCUS dataset instance metadata, please refer to: [Dataset Instance Metadata Example](#).

5.2.5. Introduced (version)

1.3

5.2.6. Dataset Instance ID

The Dataset Instance ID is a data generator-specified unique identifier that represents a specific FOCUS dataset instance provided by the data generator.

DatasetInstanceid adheres to the following requirements:

- DatasetInstanceid MUST be present in an object within the [DatasetInstance](#) collection.
- DatasetInstanceid MUST be of type String.
- DatasetInstanceid MUST NOT contain null values.
- DatasetInstanceid MUST be a unique identifier within a data generator.

- DatasetInstanceId SHOULD be a Globally Unique Identifier (GUID).

5.2.6.1. Metadata ID

DatasetInstanceId

5.2.6.2. Metadata Name

Dataset Instance ID

5.2.6.3. Content constraints

Constraint	Value
Feature level	Mandatory
Allows nulls	False
Data type	String
Value format	GUID (recommended)

5.2.6.4. Introduced (version)

1.3

5.2.7. Dataset Instance Name

The human-readable name of the dataset instance as provided by the data generator.

DatasetInstanceName adheres to the following requirements:

- DatasetInstanceName MUST be present in an object within the [DatasetInstance](#) collection.
- DatasetInstanceName MUST be of type String.
- DatasetInstanceName MUST NOT be null.

5.2.7.1. Metadata ID

DatasetInstanceName

5.2.7.2. Metadata Name

Dataset Instance Name

5.2.7.3. Content constraints

Constraint	Value
------------	-------

Constraint	Value
Feature level	Mandatory
Allows nulls	False
Data type	String
Value format	<not specified>

5.2.7.4. Introduced (version)

1.3

5.2.8. FOCUS Dataset ID

The identifier of the FOCUS dataset for which the schema and its data conform to. This indicates which FOCUS dataset the data artifact aligns with, such as "FOCUS Cost and Usage" or "FOCUS Contract."

The FocusDatasetId property adheres to the following requirements:

- FocusDatasetId MUST be present in an object within the [DatasetInstance](#) collection.
- FocusDatasetId MUST be of type String.
- FocusDatasetId MUST NOT be null.
- FocusDatasetId MUST match the Dataset ID of one of the [FOCUS datasets](#) defined in the FOCUS specification.

5.2.8.1. Metadata ID

FocusDatasetId

5.2.8.2. Metadata Name

FOCUS Dataset ID

5.2.8.3. Content constraints

Constraint	Value
Feature level	Mandatory
Allows nulls	False
Data type	String
Value format	<not specified>

5.2.8.4. Introduced (version)

1.3

5.3. Recency

The recency metadata object and its contents provide information about how up-to-date FOCUS datasets are and when they have been updated.

5.3.1. Requirements

Recency adheres to the following requirements:

- Recency MAY be present in the [Metadata](#).
- Recency MUST be structured as a collection of objects.
- Recency MUST NOT be null.
- Recency collection MUST NOT contain null objects.
- Recency collection MAY contain one and only one object for every [DatasetInstance](#) object.
- Recency collection object MUST be associated with one and only one DatasetInstance object.
- Recency collection object MUST be retrievable without inspection of the contents of [dataset instance artifacts](#).
- Recency collection object SHOULD be updated when the data generator updates the corresponding *dataset instance artifact*.

5.3.2. Metadata ID

Recency

5.3.3. Metadata Name

Recency

5.3.4. Examples

Example scenarios include but are not limited to:

- [Updating Recency metadata for a time series dataset](#)
- [Updating Recency metadata for a non time series dataset](#)

For an example of the FOCUS recency metadata, please refer to: [Recency Metadata Example](#).

5.3.5. Introduced (version)

1.3

5.3.6. Dataset Instance ID

The Dataset Instance ID provides the reference item to associate which Dataset Instance the recency metadata is for.

DatasetInstanceid adheres to the following requirements:

- DatasetInstanceid MUST be present in an object within the [Recency](#) collection.
- DatasetInstanceid MUST be of type String.

- DatasetInstanceId MUST NOT be null.
- DatasetInstanceId SHOULD be a Globally Unique Identifier (GUID).

5.3.6.1. Metadata ID

DatasetInstanceId

5.3.6.2. Metadata Name

Dataset Instance ID

5.3.6.3. Content constraints

Constraint	Value
Feature level	Mandatory
Allows nulls	False
Data type	String
Value format	GUID (recommended)

5.3.6.4. Introduced (version)

1.3

5.3.7. Dataset Instance Complete

Dataset Instance Complete provides a boolean value to indicate that the dataset instance is considered complete by the Data Generator. The definition of complete is determined by the Data Generator and should be provided in documentation provided by the Data Generator. For Datasets that are time series, the Complete value indicates that the time sector is complete and therefore is located in as key value in a time sector. For datasets that are not time series, a value of "true" indicates that the dataset is complete and therefore is a property of the recency object.

DatasetInstanceComplete adheres to the following requirements:

- DatasetInstanceComplete MUST be present in an object within the [Recency](#) collection when the dataset instance is not a time-series dataset.
- DatasetInstanceComplete MUST be of type Boolean.
- DatasetInstanceComplete MUST NOT be null.

5.3.7.1. Metadata ID

DatasetInstanceComplete

5.3.7.2. Metadata Name

5.3.7.3. Content constraints

Constraint	Value
Feature level	Conditional
Allows nulls	False
Data type	Boolean
Value format	<not specified>

5.3.7.4. Introduced (version)

1.3

5.3.8. Dataset Instance Last Updated

Datetime when the data present in the Dataset Instance was updated.

DatasetInstanceLastUpdated adheres to the following requirements:

- DatasetInstanceLastUpdated MUST be present in an object within the [Recency](#) collection.
- DatasetInstanceLastUpdated MUST be of type Date/Time.
- DatasetInstanceLastUpdated MUST conform to [DateTimeFormat](#) requirements.
- DatasetInstanceLastUpdated MUST NOT be null.

5.3.8.1. Metadata ID

DatasetInstanceLastUpdated

5.3.8.2. Metadata Name

Dataset Instance Last Updated

5.3.8.3. Content constraints

Constraint	Value
Feature level	Mandatory
Allows nulls	False
Data type	Date/Time
Value format	Date/Time Format

5.3.8.4. Introduced (version)

1.3

5.3.9. Recency Last Updated

Datetime the recency metadata object was updated.

RecencyLastUpdated adheres to the following requirements:

- RecencyLastUpdated MUST be present in the metadata.
- RecencyLastUpdated MUST be of type Date/Time.
- RecencyLastUpdated MUST NOT be null.
- RecencyLastUpdated MUST conform to [DateTimeFormat](#).

5.3.9.1. Metadata ID

RecencyLastUpdated

5.3.9.2. Metadata Name

Recency Last Updated

5.3.9.3. Content constraints

Constraint	Value
Feature level	Mandatory
Allows nulls	False
Data type	Date/Time
Value format	Date/Time Format

5.3.9.4. Introduced (version)

1.3

5.3.10. Time Sectors

The FOCUS recency metadata's Time Sectors provide a list of time periods and metadata about them. Time Sectors are used when the associated [FOCUS dataset](#) is defined as a time series dataset (i.e., its dataset artifacts represent data distributed over time). Each time sector represents a single time period and the completeness of that time period as it pertains to the dataset artifact. Time sectors do not represent start and end dates of the dataset artifact but rather periods of time relative to the datasets Charge Period Start and Charge Period End. Length of time sectors can be determined by the Data Generator, though it is suggested to align time sector periods to the reports time granularity (Hourly cost reports = hourly time sectors).

5.3.10.1. Requirements

TimeSectors adheres to the following requirements:

- TimeSectors MUST be present in an object within the [Recency](#) collection when the associated *FOCUS dataset* is defined as a time series dataset.
- TimeSectors MUST be structured as a collection of objects.
- TimeSectors MUST NOT be null.
- TimeSectors collection MUST contain at least one object.
- TimeSectors collection MUST NOT contain null objects.
- TimeSectors collection object MUST be updated, if already present, or added to the collection whenever the data generator updates or provides new dataset artifacts.

5.3.10.2. Metadata ID

TimeSectors

5.3.10.3. Metadata Name

Time Sectors

5.3.10.4. Introduced (version)

1.3

5.3.10.5. Time Sector Complete

Time Sector Complete provides a boolean value to indicate that the time sector is considered complete by the DataGenerator. The definition of complete is determined by the DataGenerator and should be provided in documentation provided by the DataGenerator.

TimeSectorComplete adheres to the following requirements:

- TimeSectorComplete MUST be present in the [TimeSectors](#) subsection of the [Recency](#) metadata section.
- TimeSectorComplete MUST be of type Boolean.
- TimeSectorComplete MUST not be null.

5.3.10.5.1. Metadata ID

TimeSectorComplete

5.3.10.5.2. Metadata Name

Time Sector Complete

5.3.10.5.3. Content constraints

Constraint	Value
------------	-------

Constraint	Value
Feature level	Mandatory
Allows nulls	False
Data type	Boolean
Value format	<not specified>

5.3.10.5.4. Introduced (version)

1.3

5.3.10.6. Time Sector Last Updated

Datetime the data in the time sector was last updated.

TimeSectorLastUpdated adheres to the following requirements:

- TimeSectorLastUpdated MUST be present in [TimeSectors](#) subsection of the [Recency](#) metadata section.
- TimeSectorLastUpdated MUST be of type Date/Time.
- TimeSectorLastUpdated MUST NOT be null.
- TimeSectorLastUpdated MUST conform to [DateTimeFormat](#).

5.3.10.6.1. Metadata ID

TimeSectorLastUpdated

5.3.10.6.2. Metadata Name

Time Sector Last Updated

5.3.10.6.3. Content constraints

Constraint	Value
Feature level	Mandatory
Allows nulls	False
Data type	Date/Time
Value format	Date/Time Format

5.3.10.6.4. Introduced (version)

1.3

5.3.10.7. Time Sector Start

The Time Sector Start is the datetime of the start of the time sector.

TimeSectorStart adheres to the following requirements:

- TimeSectorStart MUST be present in the [TimeSectors](#) subsection of the [Recency](#) metadata section.
- TimeSectorStart MUST be of type Date/Time.
- TimeSectorStart MUST NOT be null.
- TimeSectorStart MUST conform to [DateTimeFormat](#).

5.3.10.7.1. Metadata ID

TimeSectorStart

5.3.10.7.2. Metadata Name

Time Sector Start

5.3.10.7.3. Content constraints

Constraint	Value
Feature level	Mandatory
Allows nulls	False
Data type	Date/Time
Value format	Date/Time Format

5.3.10.7.4. Introduced (version)

1.3

5.3.10.8. Time Sector End

The Time Sector End is the datetime of the end of the time sector. The EndTime MUST be later than the StartTime.

TimeSectorEnd adheres to the following requirements:

- TimeSectorEnd MUST be present in the [TimeSectors](#) subsection of the [Recency](#) metadata section.
- TimeSectorEnd MUST be of type Date/Time.
- TimeSectorEnd MUST NOT be null.
- TimeSectorEnd MUST conform to [DateTimeFormat](#).
- TimeSectorEnd must be exclusive of the end time of the subsequent time sector.
- TimeSectorEnd MUST be later than TimeSectorStart.

5.3.10.8.1. Metadata ID

TimeSectorEnd

5.3.10.8.2. Metadata Name

Time Sector End

5.3.10.8.3. Content constraints

Constraint	Value
Feature level	Mandatory
Allows nulls	False
Data type	Date/Time
Value format	Date/Time Format

5.3.10.8.4. Introduced (version)

1.3

5.4. Schema

The schema metadata object and its content provide information about the structure of the data provided.

5.4.1. Requirements

Schema adheres to the following requirements:

- Schema MUST be present in the [Metadata](#).
- Schema MUST be structured as a collection of objects.
- Schema MUST NOT be null.
- Schema collection MUST contain at least one object for every [DatasetInstance](#) object.
- Schema collection MUST NOT contain null objects.
- Schema collection object MUST be associated with one and only one DatasetInstance object.
- Schema collection object MUST be added to the collection whenever the structure of the [dataset instance artifacts](#) changes (including, but not limited to, additions or removals of columns, modifications to any ColumnDefinition, or updates to the FOCUSVersion or DataGeneratorVersion).
- Schema collection object MUST be referenced by *dataset instance artifacts* that conform to the structure defined by that Schema collection object.
- Schema collection object MUST define the exact structure of the *dataset instance artifacts* that reference it.
- Schema collection object MUST be retrievable independently from the *dataset instance artifacts* that conform to the structure defined by that Schema collection object.
- Schema collection object SHOULD be provided separately from the *dataset instance artifacts* that conform to the structure defined by that Schema collection object.
- Schema collection object MAY be provided through the structure and/or schema of the delivery mechanism (e.g., database tables).

5.4.2. Examples

There are many scenarios that would result in an update to the Schema metadata. These scenarios include but are not limited to:

- [Adding a new column](#)
- [Removing a column](#)
- [Changing column metadata](#)
- [FOCUS Version has changed](#)
- [Data Generator Version has changed](#)
- [Correcting schema metadata errors](#)

For an example of the FOCUS schema metadata, please refer to: [Schema Metadata Example](#).

5.4.3. Metadata ID

Schema

5.4.4. Metadata Name

Schema

5.4.5. Introduced (version)

1.0

5.4.6. Schema ID

The Schema ID provides the reference item to associate which Schema was used for the generation of a [FOCUS dataset](#).

Schemald adheres to the following requirements:

- Schemald MUST be present in an object within the [Schema](#) collection.
- Schemald MUST be of type String.
- SchemaID MUST NOT be null.
- Schemald SHOULD be a Globally Unique Identifier (GUID).

5.4.6.1. Metadata ID

Schemald

5.4.6.2. Metadata Name

Schema ID

5.4.6.3. Content constraints

Constraint	Value
Feature level	Mandatory
Allows nulls	False
Data type	String
Value format	GUID (recommended)

5.4.6.4. Introduced (version)

1.0

5.4.7. Creation Date

Date the schema was created.

CreationDate adheres to the following requirements:

- CreationDate MUST be present in an object within the [Schema](#) collection.
- CreationDate MUST be of type Date/Time.
- CreationDate MUST conform to [DateTimeFormat](#).
- CreationDate MUST NOT be null.

5.4.7.1. Metadata ID

CreationDate

5.4.7.2. Metadata Name

Creation Date

5.4.7.3. Content constraints

Constraint	Value
Feature level	Mandatory
Allows nulls	False
Data type	Date/Time
Value format	Date/Time Format

5.4.7.4. Introduced (version)

1.0

5.4.8. FOCUS Version

The version of FOCUS utilized for building the dataset.

FocusVersion adheres to the following requirements:

- FocusVersion MUST be present in an object within the [Schema](#) collection.
- FocusVersion MUST be of type String.
- FocusVersion MUST NOT be null.
- FocusVersion MUST match one of the published versions of the FOCUS specification.
- FocusVersion MUST match the version of the FOCUS specification that the [dataset instance artifact](#) conforms to.

5.4.8.1. Metadata ID

FocusVersion

5.4.8.2. Metadata Name

FOCUS Version

5.4.8.3. Content constraints

Constraint	Value
Feature level	Mandatory
Allows nulls	False
Data type	String
Value format	<not specified>

5.4.8.4. Introduced (version)

1.0

5.4.9. Data Generator Version

The DataGeneratorVersion may be supplied to declare the version of logic by which the [dataset instance artifact](#) was generated and is separate from FOCUS Version. DataGeneratorVersion allows for the provider to specify changes that may not result in a structural change in the data. It is suggested that the DataGeneratorVersion use a versioning approach such as [SemVer](#) version.

DataGeneratorVersion adheres to the following requirements:

- DataGeneratorVersion MAY be present in an object within the [Schema](#) collection.
- DataGeneratorVersion MUST be of type String.
- DataGeneratorVersion MUST conform to [StringHandling](#) requirements.
- DataGeneratorVersion MUST NOT be null.
- DataGeneratorVersion MUST be changed when [FocusVersion](#) is changed.
- Data generators MUST document what changes are present in the DataGeneratorVersion.

5.4.9.1. Metadata ID

5.4.9.2. Metadata Name

Data Generator Version

5.4.9.3. Content constraints

Constraint	Value
Feature level	Optional
Allows nulls	False
Data type	String
Value format	<not specified>

5.4.9.4. Introduced (version)

1.1

5.4.10. Dataset Instance ID

The Dataset Instance ID is a unique identifier for the specific dataset instance provided by the data generator. It identifies the dataset instance that this schema and the corresponding dataset artifacts are aligned with.

DatasetInstanceID adheres to the following requirements:

- DatasetInstanceID MUST be present in an object within the [Schema](#) collection.
- DatasetInstanceID MUST be of type String.
- DatasetInstanceID MUST NOT be null.
- DatasetInstanceID MUST be a unique identifier within a data generator.

5.4.10.1. Metadata ID

DatasetInstanceID

5.4.10.2. Metadata Name

Dataset Instance ID

5.4.10.3. Content constraints

Constraint	Value
Feature level	Mandatory
Allows nulls	False

Constraint	Value
Data type	String
Value format	GUID (recommended)

5.4.10.4. Introduced (version)

1.3

5.4.11. Column Definition

The FOCUS metadata schema column definition provides a list of the columns present in the [dataset instance artifact](#) along with metadata about the columns.

5.4.11.1. Requirements

ColumnDefinition adheres to the following requirements:

- ColumnDefinition MUST be present in an object within the [Schema](#) collection.
- ColumnDefinition MUST be structured as a collection of objects.
- ColumnDefinition MUST NOT be null.
- ColumnDefinition collection MUST contain one and only one object for every column provided in *dataset instance artifacts* that reference the parent Schema object.
- ColumnDefinition collection MUST NOT contain null objects.

5.4.11.2. Metadata ID

ColumnDefinition

5.4.11.3. Metadata Name

Column Definition

5.4.11.4. Introduced (version)

1.0

5.4.11.5. Column Name

The name of the column provided in the [FOCUS dataset](#).

ColumnName adheres to the following requirements:

- ColumnName MUST be present in an object within the [ColumnDefinition](#) collection.
- ColumnName MUST be of type String.
- ColumnName MUST NOT be null.

5.4.11.5.1. Metadata ID

ColumnName

5.4.11.5.2. Metadata Name

Column Name

5.4.11.5.3. Content constraints

Constraint	Value
Feature level	Mandatory
Allows nulls	False
Data type	String
Value format	<not specified>

5.4.11.5.4. Introduced (version)

1.0

5.4.11.6. Data Type

The data type of the column provided in the [FOCUS dataset](#).

DataType adheres to the following requirements:

- DataType MUST be present in an object within the [ColumnDefinition](#) collection.
- DataType MUST be of type String.
- DataType MUST NOT contain null values.

5.4.11.6.1. Metadata ID

DataType

5.4.11.6.2. Metadata Name

Data Type

5.4.11.6.3. Content constraints

Constraint	Value
Feature level	Mandatory
Allows nulls	False
Data type	String

Constraint	Value
------------	-------

Value format	<not specified>
--------------	-----------------

5.4.11.6.4. Introduced (version)

1.0

5.4.11.7. Deprecated

The deprecation status of a column in a [Dataset Instance](#).

Deprecated adheres to the following requirements:

- Deprecated MUST be present in an object within the [ColumnDefinition](#) collection when the column is planned for removal.
- Deprecated MUST be of type Boolean.
- Deprecated MUST NOT contain null values.
- Deprecated SHOULD only be "true" if the column is deprecated.
- Deprecated MUST be "true" when the data generator removes a column at a future date, or the column has been identified for deprecation for the FOCUS version identified in the schema definition.

5.4.11.7.1. Metadata ID

Deprecated

5.4.11.7.2. Metadata Name

Deprecated

5.4.11.7.3. Content constraints

Constraint	Value
Feature level	Conditional
Allows nulls	False
Data type	Boolean
Value format	<not specified>

5.4.11.7.4. Introduced (version)

1.2

5.4.11.8. Numeric Precision

Numeric Precision is the maximum number of digits for the values in the column.

NumericPrecision adheres to the following requirements:

- NumberPrecision SHOULD be present in an object within the [ColumnDefinition](#) collection when the column is of Decimal data type.
- NumericPrecision MUST be of type Integer.
- NumericPrecision MUST NOT contain null values.

5.4.11.8.1. Metadata ID

NumericPrecision

5.4.11.8.2. Metadata Name

Numeric Precision

5.4.11.8.3. Content constraints

Constraint	Value
Feature level	Recommended
Allows nulls	False
Data type	Integer
Value format	Numeric Format

5.4.11.8.4. Introduced (version)

1.0

5.4.11.9. Number Scale

The number scale of the data provides the maximum number of digits after the decimal point in decimal numbers.

NumberScale adheres to the following requirements:

- NumberScale SHOULD be present in an object within the [ColumnDefinition](#) collection when the column is of Decimal data type.
- NumberScale MUST be of type Integer.
- NumberScale MUST conform to [NumericFormat](#) requirements.
- NumberScale MUST NOT be null.

5.4.11.9.1. Metadata ID

NumberScale

5.4.11.9.2. Metadata Name

Number Scale

5.4.11.9.3. Content constraints

Constraint	Value
Feature level	Recommended
Allows nulls	False
Data type	Integer
Value format	Numeric Format

5.4.11.9.4. Introduced (version)

1.0

5.4.11.10. PreviousColumnName

The PreviousColumnName field indicates that on that schema the column where the key is included was renamed.

PreviousColumnName adheres to the following requirements:

- PreviousColumnName MUST be present in an object within the [ColumnDefinition](#) collection when the column was renamed.
- When PreviousColumnName is present, PreviousColumnName adheres to the following normative requirements:
 - PreviousColumnName MUST be of type String.
 - PreviousColumnName MUST not be null.
 - PreviousColumnName MUST be the name used in previous versions of the schema.
 - PreviousColumnName MUST NOT be present in schema versions created after the rename.

5.4.11.10.1. Metadata ID

PreviousColumnName

5.4.11.10.2. Metadata Name

Previous Column Name

5.4.11.10.3. Content constraints

Constraint	Value
Feature level	Conditional

Constraint	Value
Allows nulls	False
Data type	String
Value format	<not specified>

5.4.11.10.4. Introduced (version)

1.2

5.4.11.11. Provider Tag Prefixes

The Provider Tag Prefixes define the list of prefixes used in the tag name of provider-defined [tags](#). This metadata is useful for the consumer to identify which tags are provider-defined vs user-defined.

ProviderTagPrefixes adheres to the following requirements:

- ProviderTagPrefixes MUST be present in an object within the [ColumnDefinition](#) collection when [ColumnName](#) is "Tags".
- ProviderTagPrefixes MUST be of type Collection of Strings.
- ProviderTagPrefixes SHOULD be easily associated with the data generator who generated the [dataset instance](#) and the corresponding [dataset instance artifacts](#).

5.4.11.11.1. Metadata ID

ProviderTagPrefixes

5.4.11.11.2. Metadata Name

Provider Tag Prefixes

5.4.11.11.3. Content constraints

Constraint	Value
Feature level	Conditional
Allows nulls	False
Data type	Collection of Strings
Value format	<not specified>

5.4.11.11.4. Introduced (version)

1.0

5.4.11.12. String Encoding

The string encoding scheme of the column provided in the [FOCUS dataset](#).

StringEncoding adheres to the following requirements:

- StringEncoding MUST be present in an object within the [ColumnDefinition](#) collection when this information is required in order to successfully read the data.
- StringEncoding MUST be of type String.
- StringEncoding MUST NOT be null.

5.4.11.12.1. Metadata ID

StringEncoding

5.4.11.12.2. Metadata Name

StringEncoding

5.4.11.12.3. Content constraints

Constraint	Value
Feature level	Conditional
Allows nulls	False
Data type	String
Value format	<not specified>

5.4.11.12.4. Introduced (version)

1.0

5.4.11.13. String Max Length

The string max length of the data that can be stored in the column.

StringMaxLength adheres to the following requirements:

- StringMaxLength SHOULD be present in an object within the [ColumnDefinition](#) collection when the column is of String data type.
- StringMaxLength MUST be of type Integer.
- StringMaxLength MUST NOT be null.

5.4.11.13.1. Metadata ID

StringMaxLength

5.4.11.13.2. Metadata Name

String Max Length

5.4.11.13.3. Content constraints

Constraint	Value
Feature level	Recommended
Allows nulls	False
Data type	Integer
Value format	Numeric Format

5.4.11.13.4. Introduced (version)

1.0

6. Use Case Library

This specification is based on a set of common FinOps use cases, which are publicly available at <https://focus.finops.org/use-cases/>. Developed by FinOps practitioners, these use cases are organized by persona and capability, making it easy to find relevant scenarios. Each use case includes sample SQL queries to help you get started with implementation.

7. Glossary

Adjustment

A charge representing a modification to billing data to account for certain events or circumstances not previously captured, or captured incorrectly. Examples include billing errors, service disruptions, or pricing changes.

Allocated Charge

The [charge](#) that was created as the result of an allocation operation. This is used in the context of [Data Generator-Calculated Split Cost Allocation](#) to identify the charges that were created from the [origin charge](#) resulting from the application of Data Generator-Calculated Split Cost Allocation.

Allocated Method

The process or formula by which cost is being allocated from an [origin charge](#) to produce [allocated charges](#). This is used in the context of [Data Generator-Calculated Split Cost Allocation](#) which requires documentation of the method to be provided for any and all allocated methods used. May also be colloquially referred to as allocation method.

Amortization

The distribution of upfront costs over time to accurately reflect the consumption or benefit derived from the associated resources or services. Amortization is valuable when the commitment [period](#) extends beyond the granularity of the source report.

Availability Zone

A collection of geographically separated locations containing a data center or cluster of data centers. Each availability zone (AZ) should have its own power, cooling, and networking, to provide redundancy and fault tolerance.

Billed Cost

A charge that serves as the basis for invoicing. It includes the total amount of fees and discounts, signifying a monetary obligation. Valuable when reconciling cash outlay with incurred expenses is required, such as cost allocation, budgeting, and invoice reconciliation.

Billing Account

A container for resources and/or services that are billed together in an invoice. A billing account may have sub accounts, all of whose costs are consolidated and invoiced to the billing account.

Billing Currency

An identifier that represents the currency that a charge for resources and/or services was billed in.

Billing Period

The time window that an organization receives an invoice for, inclusive of the start date and exclusive of the end date. It is independent of the time of usage and consumption of resources and services.

Block Pricing

A pricing approach where the cost of a particular resource or service is determined based on predefined quantities or tiers of usage. In these scenarios, the Pricing Unit and the corresponding Pricing Quantity can be different from the Consumed Unit and Consumed Quantity.

Capacity Reservation

A capacity reservation is an agreement that secures a dedicated amount of resources or services for a specified period. This ensures the reserved capacity is always available and accessible, even if it's not fully utilized. Customers are typically charged for the reserved capacity, regardless of actual

consumption.

Charge

A row in a FOCUS-compatible cost and usage dataset.

Charge Period

The time window for which a charge is effective, inclusive of the start date and exclusive of the end date. The charge period for continuous usage should match the time granularity of the dataset (e.g., 1 hour for hourly, 1 day for daily). The charge period for a non-usage charge with time boundaries should match the period of eligibility.

Cloud Service Provider (CSP)

A company or organization that provides remote access to computing resources, infrastructure, or applications for a fee.

Commitment

A customer's agreement to either spend a defined monetary amount or consume a specific quantity of resources or services over a specified [period](#).

Commitment Discount

A billing discount model that offers reduced rates on preselected SKUs in exchange for an obligated usage or spend amount over a specified [period](#). Commitment discount purchases, made upfront and/or with recurring monthly payments are amortized evenly across predefined charge periods (i.e., hourly), and unused amounts cannot be carried over to subsequent charge periods. Commitment discounts are publicly available to customers without special contract arrangements.

Commitment Discount Flexibility

A feature of [commitment discounts](#) that may further transform the predetermined amount of usage purchased or consumed based on additional, service-provider-specific requirements.

Contract

A collection of agreed terms between a service provider and a customer.

Contract Commitment

A specific term within a [contract](#) that defines a measurable obligation agreed upon by a provider and a customer, such as a minimum spend or usage over an agreed period of time.

Contracted Unit Price

The agreed-upon unit price for a single [Pricing Unit](#) of the associated SKU, inclusive of negotiated discounts, if present, and exclusive of any other discounts. This price is denominated in the [Billing Currency](#).

Correction

A charge to correct cost or usage data in a previously invoiced [billing period](#).

Credit

A financial incentive or allowance granted by a service provider unrelated to other past/current/future charges.

Dataset Artifact

An abbreviated term for [dataset instance artifact](#).

Dataset Instance

A specific implementation of a [FOCUS dataset](#) provided by a [data generator](#). A Data Generator may provide multiple dataset instances of the same *FOCUS dataset*, each with different properties such as time granularity or differing custom column inclusions. For example, the same 'FOCUS Cost and Usage' *FOCUS dataset* may be provided at an hourly or daily time granularity by a Data Generator.

Each would be a distinct Dataset Instance.

Dataset Instance Artifact

A physical representation of a specific [dataset instance](#) delivered by a [data generator](#).

Dimension

A specification-defined categorical attribute that provides context or categorization to billing data.

Effective Cost

The amortized cost of the charge after applying all reduced rates, discounts, and the applicable portion of relevant, prepaid purchases (one-time or recurring) that covered this charge.

Exclusive End Bound

A Date/Time Format value that is not contained within the ending bound of a time period.

Finalized Tag

A tag with one tag value chosen from a set of possible tag values after being processed by a set of service-provider-defined or user-defined rules.

FinOps Cost and Usage Specification (FOCUS)

An open-source specification that defines requirements for billing data.

FOCUS Dataset

A structured collection of columns that conforms to the BCP14 criteria established by FOCUS. All columns included must be defined in the FOCUS Columns section of the specification.

In addition to these standardized columns, [data generators](#) may include custom columns (prefixed with x_) where additional context is needed beyond what is captured in the defined FOCUS columns. If custom columns introduce record-splitting (i.e., a single original charge results in multiple rows), the data generator is responsible for ensuring that all cost and quantity metrics still meet the aggregation and consistency rules required by the specification.

The collection of datasets are designed to provide billing insight, additional context, metadata, mapping, or enrichment information that enhances the interpretability or completeness.

Inclusive Start Bound

A Date/Time Format value that is contained within the beginning bound of a time period.

Interruptible

A category of compute resources that can be paused or terminated by the CSP within certain criteria, often advertised at reduced unit pricing when compared to the equivalent non-interruptible resource.

JSON

A common acronym for JavaScript Object Notation, a data format codified in [ECMA-404](#) as a standard for human-readable, serializable data objects. This data format is used in FOCUS to communicate multiple pieces of information about a charge (tags, properties, etc.) in a single column.

List Unit Price

The suggested service-provider-published unit price for a single [Pricing Unit](#) of the associated [SKU](#), exclusive of any discounts. This price is denominated in the [Billing Currency](#).

Managed Service Provider (MSP)

A company or organization that provides outsourced management and support of a range of IT services, such as network infrastructure, cybersecurity, cloud computing, and more.

Metric

A FOCUS-defined column that provides numeric values, allowing for aggregation operations such as arithmetic operations (sum, multiplication, averaging etc.) and statistical operations.

National Currency

A government-issued currency (e.g., US dollars, Euros).

Negotiated Discount

A contractual agreement where a customer commits to specific spend or usage goals over a specified [period](#) in exchange for discounted rates across varying SKUs. Unlike [commitment discounts](#), negotiated discounts are typically more customized to customer's accounts, can be utilized at varying frequencies, and may overlap with *commitment discounts*.

On-Demand

A service that is available and provided immediately or as needed, without requiring a pre-scheduled appointment or prior arrangement. In cloud computing, virtual machines can be created and terminated as needed, i.e., on demand.

Origin Charge

The [charge](#) that existed prior to an operation. This is used in the context of [Data Generator-Calculated Split Cost Allocation](#) to identify the charge that existed prior to the application of Data Generator-Calculated Split Cost Allocation to produce [allocated charges](#).

Pascal Case

Pascal Case (PascalCase, also known as UpperCamelCase) is a format for identifiers which contain one or more words meaning the words are concatenated together with no delimiter and the first letter of each word is capitalized.

Period

A time window, with a specifically defined start and end date/time.

Potato

A long and often painful conversation had by the FOCUS contributors. Sometimes the name of a thing that we could not yet name. No starchy root vegetables were harmed during the production of this specification. We thank potato for its contribution in the creation of this specification.

Practitioner

An individual who performs FinOps within an organization to maximize the business value of using cloud and cloud-like services.

Price List

A comprehensive list of prices offered by a service provider.

Service Provider

An entity that provides the [resources](#) or [services](#) available for usage or purchase.

Refund

A return of funds that have previously been charged.

Resource

A unique component that incurs a charge.

Row

A row in a FOCUS-compatible cost and usage dataset.

Service

An offering that can be purchased from a service provider, and can include many types of usage or other charges; eg., a cloud database service may include compute, storage, and networking charges.

SKU

A construct composed of the common properties of a product offering associated with one or many SKU Prices.

SKU Price

A pricing construct that encompasses SKU properties (e.g., functionality and technical specifications), along with core stable pricing details for a particular SKU, while excluding dynamic or negotiable pricing elements such as unit price amounts, currency (and related exchange rates), temporal validity (e.g., effective dates), and contract- or negotiation-specific factors (e.g., contract or account identifiers, and negotiable discounts).

Sub Account

A sub account is an optional service-provider-supported construct for organizing resources and/or services connected to a billing account. Sub accounts must be associated with a billing account as they do not receive invoices.

Tag

A metadata label assigned to a resource to provide information about it or to categorize it for organizational and management purposes.

Tag Source

A Resource or Service-Provider-defined construct for grouping resources and/or other Service-Provider-defined construct that a Tag can be assigned to.

Term

An agreement specified on a [contract](#).

Virtual Currency

A proprietary currency (e.g., credits, tokens) issued by service providers and independent of government regulation.

8. Appendix

This section is non-normative.

8.1. Commitment Discounts

8.1.1. Examples: Commitment Discount Scenarios

A [commitment discount](#) is a billing discount model that offers reduced rates on preselected [SKUs](#) in exchange for an obligated usage or spend amount over a specified [period](#). *Commitment discounts* typically consist of purchase and usage records within cost and usage datasets.

Usage-based *commitment discounts* obligate a customer to a predetermined amount of usage over a specified [period](#). In some cases, usage-based *commitment discounts* also feature [commitment discount flexibility](#) which may expand the types of [resources](#) that a *commitment discount* can cover. It is important to note when mixing *commitment discounts* with and without *commitment discount flexibility*, the [CommitmentDiscountUnit](#) should reflect this difference.

Spend-based commitment discounts obligate a customer to a predetermined amount of spend over a specified [period](#). In the usage examples below, each [row](#) measures the monetary amount of the hourly commit consumed by the *commitment discount*, so the [CommitmentDiscountUnit](#) chosen is "USD", or the [billing currency](#).

8.1.2. Purchasing

While customers are bound to the [period](#) of a *commitment discount*, service providers offer some or all of the following payment options before and/or during the [period](#):

- *All Upfront* - The *commitment discount* is paid in full before the [period](#) begins.
- *No Upfront* - The *commitment discount* is paid on a repeated basis, typically over each [billing period](#) of the [period](#).
- *Partial Upfront* - Some of the *commitment discount* is paid before the [period](#) begins, and the rest is paid repeatedly over the [period](#).

For example, if a customer buys a 1-year, spend-based *commitment discount* with a \$1.00 hourly commit and pays with the partial option, the *commitment discount's* payment consists of a one-time purchase in the beginning of the [period](#) and monthly recurring purchases with the following totals:

1. *One-Time* - \$4,380 (24 hours * 365 days * $1.00 * 0.5)
2. *Recurring* - \$182.50 (24 hours * 365 days * $1.00 / 12 months)

8.1.3. Usage

Commitment discounts follow a "use-it-or-lose-it" model where the [amortization](#) of a *commitment discount's* purchase applies evenly to eligible [resources](#) over each [charge period](#) of the [period](#).

For example, if a customer buys a spend-based *commitment discount* with a \$1.00 hourly commit in January (31 days), only \$1.00 is eligible for consumption for each hourly [charge period](#). If a customer has eligible [resources](#) running during this [charge period](#), an amount of up to \$1.00 will be allocated to these [resources](#). Conversely, if a customer does not have eligible [resources](#) running that fully take advantage of this \$1.00 during this [charge period](#), then some or all of this amount will go to waste.

8.1.4. Commitment Discounts in FOCUS

Within the FOCUS specification, the following examples demonstrate how a *commitment discount* appears across various payment and usage scenarios.

8.1.4.1. Purchase Rows

All *commitment discount* purchases appear with a positive [BilledCost](#), [PricingCategory](#) as "Standard", and with the *commitment discount's* id populating both the [ResourceId](#) and [CommitmentDiscountId](#) value. One-time purchases appear as a single record with [ChargeCategory](#) as "Purchase", [ChargeFrequency](#) as "One-Time", and the total quantity and units for *commitment discount's period* reflected as [CommitmentDiscountQuantity](#) and [CommitmentDiscountUnit](#), respectively.

Recurring purchases are allocated across all corresponding *charge periods* of the *period* when [ChargeCategory](#) is "Purchase", [ChargeFrequency](#) is "Recurring", and [CommitmentDiscountQuantity](#) and [CommitmentDiscountUnit](#) are reflected only for that *charge period*.

Using the same *commitment discount* example as above with a one-year, spend-based *commitment discount* with a \$1.00 hourly commit purchased on Jan 1, 2023, various purchase options are available:

8.1.4.1.1. Scenario #1: All Upfront

The entire *commitment discount* is billed *once* during the first *charge period* of the *period* for \$8,670 (derived as 24 hours * 365 days * $1.00).

[CSV Example](#)

8.1.4.1.2. Scenario #2: No Upfront

The *commitment discount* is billed across all 8,760 (24 hours * 365 days) *charge periods* of the *period* with \$1.00 allocated to each *charge period* over the *period*.

[CSV Example](#)

This example shows the first three hourly rows of 8,760 total rows that are all the same except for the incrementing monthly and hourly timeframes denoted in the Billing Period and Charge Period columns, respectively.

8.1.4.1.3. Scenario #3: Partial Upfront

With a 50/50 split, half of the commitment is billed *once* during the first *charge period* of the *period* for \$4,380 (derived as 24 hours * 182.5 days * $1.00), and the other half is billed across each *charge period* over the commitment *period*, derived as ($1.00 * 8,760 hours * 0.5). Amortized costs incur half of the amount (i.e., \$0.50) from the one-time purchase and the other half from the recurring purchase.

[CSV Example](#)

This example shows the first three hourly rows of 8,760 total rows that are all the same except for the incrementing monthly and hourly timeframes denoted in the Billing Period and Charge Period

columns, respectively.

8.1.4.2. Usage Rows

Amortization of commitment discounts occur similarly regardless of how *commitment discount* purchases are made. The same usage-based or spend-based amount is applied evenly across all *charge periods* and potentially allocated to eligible *resources*. Continuing with the same *commitment discount* example, a one-year, spend-based *commitment discount* with a \$1.00 hourly commit and 1 *resource* (for simplicity) yields 4 types of scenarios that can occur during a *charge period*:

- Scenario #1: An eligible *resource* fully consumes the allocated amount (100% utilization)
- Scenario #2: No eligible *resource* consumes the allocated amount (0% utilization)
- Scenario #3: An eligible *resource* partially consumes the allocated amount (75% utilization)
- Scenario #4: An eligible *resource* fully consumes the \$1.00 hourly commit with an overage (100% utilization + overage)

8.1.4.2.1. Scenario #1: An eligible *resource* fully consumes the allocated amount (100% utilization)

In this scenario, one eligible *resource* runs for the full hour and consumes \$1.00, so one *row* allocated to the *resource* is produced.

[CSV Example](#)

8.1.4.2.2. Scenario #2: No eligible *resource* consumes the allocated amount (0% utilization)

In this situation, the full eligible, \$1.00 amount remained unutilized and results in 1 unused *row*. In this scenario, it is important to note that while *CommitmentDiscountQuantity* is not because \$1 was still drawn down by the *commitment discount* even though, no *resource* was allocated, so [ConsumedQuantity](#) and [ConsumedUnit](#) are null.

[CSV Example](#)

8.1.4.2.3. Scenario #3: An eligible *resource* partially consumes the allocated amount (75% utilization)

In this scenario, one eligible *resource* runs for the full hour and consumes \$0.75 of the \$1.00 allocation. One *row* shows \$0.75 to a *resource*, and the other *row* shows that \$0.25 was unused.

[CSV Example](#)

8.1.4.2.4. Scenario #4: An eligible *resource* fully consumes the \$1.00 hourly commit with an overage (100% utilization + overage)

In this scenario, one eligible *resource* runs for the full hour and is charged \$1.50. One *row* shows that \$1.00 was *amortized* from the *commitment discount*, and the other shows that \$0.50 was charged as standard, on-demand spend.

[CSV Example](#)

8.2. Examples: Commitment Discount Flexibility

A usage-based [commitment discount](#) obligates a customer to a usage amount for one or more related SKUs in return for reduced rates. For example, when a usage-based *commitment discount* is purchased to cover a specific database SKU, this commitment will cover every hour over the period where at least one instance of this SKU is running. The usage-based commitment can cover 1 resource over the hour, or in the case of [commitment discount flexibility](#), it can cover a portion of 1 resource or multiple resources at a time.

When mixing usage-based commitment discounts with and without *commitment discount flexibility* and [CommitmentDiscountQuantity](#) measured by time, it is important to differentiate the [CommitmentDiscountUnit](#) for each type of *commitment discount*. In each scenario below, *commitment discounts without commitment discount flexibility* applied use "Hour" as the *CommitmentDiscountUnit*, and conversely commitment discounts *with commitment discount flexibility* applied use "Normalized Hour" as the *CommitmentDiscountUnit*.

For more details on exactly how *commitment discounts* purchase and usage rows appear with and without *commitment discount flexibility*, see the following scenarios:

8.2.1. 100% utilization without commitment discount flexibility

8.2.1.1. Context

For this example, fictitious service provider, *TinyCloud*, offers the following SKU catalog which is used in the scenario below.

8.2.1.2. SKU Catalog

Service	SKU Id	SKU Price Id	SKU Price Unit Price	Normalization Factor
Compute	VM_SMALL	VM_SMALL_COMMITTED_PURCHASE_NO_UPFRONT	\$0.50	1
Compute	VM_MEDIUM	VM_MEDIUM_COMMITTED_PURCHASE_NO_UPFRONT	\$1.00	2
Compute	VM_LARGE	VM_LARGE_COMMITTED_PURCHASE_NO_UPFRONT	\$1.50	3
Compute	VM_XLARGE	VM_XLARGE_COMMITTED_PURCHASE_NO_UPFRONT	\$2.00	4
Compute	VM_SMALL	VM_SMALL_COMMITTED_HOUR	\$0.50	1
Compute	VM_MEDIUM	VM_MEDIUM_COMMITTED_HOUR	\$1.00	2
Compute	VM_LARGE	VM_LARGE_COMMITTED_HOUR	\$1.50	3
Compute	VM_XLARGE	VM_XLARGE_COMMITTED_HOUR	\$2.00	4
Compute	VM_SMALL	VM_SMALL_ON_DEMAND_HOUR	\$1.00	1
Compute	VM_MEDIUM	VM_MEDIUM_ON_DEMAND_HOUR	\$2.00	2
Compute	VM_LARGE	VM_LARGE_ON_DEMAND_HOUR	\$3.00	3
Compute	VM_XLARGE	VM_XLARGE_ON_DEMAND_HOUR	\$4.00	4

The above SKU Catalog shows that this service provider only has 1 service that offers 4 virtual machine SKUs at various list rates, [commitment discount](#) rates, and normalization factors. Each SKU's normalization factor classifies its relative size to its *commitment discount* rate. Usage-based *commitment discounts* with [commitment discount flexibility](#) can fully cover any combination of 1 or more SKUs where the sum of their normalization factor is less than or equal to the normalization factor of the *commitment discount*.

8.2.1.3. Scenario

- 1 no upfront *commitment discount* is purchased for 1 year (2023) for 1 VM_LARGE.
- 1 VM_LARGE resource runs for 1 hour from 2023-01-01T00:00:00 to 2023-01-01T01:00:00.

8.2.1.4. Outcome

- 1 recurring, purchase record exists for 1 eligible "Hour" of the no upfront, *commitment discount* and incurs a \$1.50 *BilledCost*.
- The *commitment discount* covers the first *charge period* for 1 VM_LARGE resource incurring a \$1.50 *EffectiveCost*.

[CSV Example](#)

8.2.2. 0% utilization without commitment discount flexibility

8.2.2.1. Context

For this example, fictitious service provider, *TinyCloud*, offers the following SKU catalog which is used in the scenario below.

8.2.2.2. SKU Catalog

Service	SKU Id	SKU Price Id	SKU Price Unit Price	Normalization Factor
Compute	VM_SMALL	VM_SMALL_COMMITTED_PURCHASE_NO_UPFRONT	\$0.50	1
Compute	VM_MEDIUM	VM_MEDIUM_COMMITTED_PURCHASE_NO_UPFRONT	\$1.00	2
Compute	VM_LARGE	VM_LARGE_COMMITTED_PURCHASE_NO_UPFRONT	\$1.50	3
Compute	VM_XLARGE	VM_XLARGE_COMMITTED_PURCHASE_NO_UPFRONT	\$2.00	4
Compute	VM_SMALL	VM_SMALL_COMMITTED_HOUR	\$0.50	1
Compute	VM_MEDIUM	VM_MEDIUM_COMMITTED_HOUR	\$1.00	2
Compute	VM_LARGE	VM_LARGE_COMMITTED_HOUR	\$1.50	3
Compute	VM_XLARGE	VM_XLARGE_COMMITTED_HOUR	\$2.00	4
Compute	VM_SMALL	VM_SMALL_ON_DEMAND_HOUR	\$1.00	1
Compute	VM_MEDIUM	VM_MEDIUM_ON_DEMAND_HOUR	\$2.00	2
Compute	VM_LARGE	VM_LARGE_ON_DEMAND_HOUR	\$3.00	3
Compute	VM_XLARGE	VM_XLARGE_ON_DEMAND_HOUR	\$4.00	4

The above SKU Catalog shows that this service provider only has 1 service that offers 4 virtual machine SKUs at various list unit prices, *commitment discount* unit prices, and normalization factors. Each SKU's normalization factor classifies its relative size to its *commitment discount* unit price. Usage-based *commitment discounts* with *commitment discount flexibility* can fully cover any combination of 1 or more SKUs where the sum of their normalization factor is less than or equal to the normalization factor of the *commitment discount*.

8.2.2.3. Scenario

- 1 no upfront *commitment discount* is purchased for 1 year (2023) for 1 VM_LARGE.
- 1 VM_MEDIUM resource runs for 1 hour from 2023-01-01T00:00:00 to 2023-01-01T01:00:00.

8.2.2.4. Outcome

- 1 recurring, purchase record exists for 1 eligible "Hour" of the no upfront, *commitment discount* and incurs a \$1.50 *BilledCost*.
- The VM_LARGE *commitment discount* is unused for the corresponding *charge period* because no VM_LARGE resources are running and incurs a \$1.50 *EffectiveCost*.
- 1 hour of on-demand usage is incurred by the VM_MEDIUM resource and incurs a \$2.00 *BilledCost* and *EffectiveCost*.

[CSV Example](#)

8.2.3. 100% utilization with commitment discount flexibility with 1 resource

8.2.3.1. Context

For this example, fictitious service provider, *TinyCloud*, offers the following SKU catalog which is used in the scenario below.

8.2.3.2. SKU Catalog

Service	Sku Id	Sku Price Id	Sku Price Unit Price	Normalization Factor
Compute	VM_SMALL	VM_SMALL_COMMITTED_PURCHASE_NO_UPFRONT	\$0.50	1
Compute	VM_MEDIUM	VM_MEDIUM_COMMITTED_PURCHASE_NO_UPFRONT	\$1.00	2
Compute	VM_LARGE	VM_LARGE_COMMITTED_PURCHASE_NO_UPFRONT	\$1.50	3
Compute	VM_XLARGE	VM_XLARGE_COMMITTED_PURCHASE_NO_UPFRONT	\$2.00	4
Compute	VM_SMALL	VM_SMALL_COMMITTED_HOUR	\$0.50	1
Compute	VM_MEDIUM	VM_MEDIUM_COMMITTED_HOUR	\$1.00	2
Compute	VM_LARGE	VM_LARGE_COMMITTED_HOUR	\$1.50	3
Compute	VM_XLARGE	VM_XLARGE_COMMITTED_HOUR	\$2.00	4
Compute	VM_SMALL	VM_SMALL_ON_DEMAND_HOUR	\$1.00	1
Compute	VM_MEDIUM	VM_MEDIUM_ON_DEMAND_HOUR	\$2.00	2
Compute	VM_LARGE	VM_LARGE_ON_DEMAND_HOUR	\$3.00	3
Compute	VM_XLARGE	VM_XLARGE_ON_DEMAND_HOUR	\$4.00	4

The above SKU catalog shows that this service provider only has 1 service that offers 4 virtual machine SKUs at various list rates, *commitment discount* rates, and normalization factors. Each SKU's normalization factor classifies its relative size to its *commitment discount* rate. Usage-based *commitment discounts* with *commitment discount flexibility* can fully cover any combination of 1 or

more SKUs where the sum of their normalization factor is less than or equal to the normalization factor of the *commitment discount*.

8.2.3.3. Scenario

- 1 no upfront *commitment discount* is purchased for 1 year (2023) for 1 VM_SMALL which has a normalization factor of 1.
- 1 VM_LARGE resource runs for 1 hour from 2023-01-01T00:00:00 to 2023-01-01T01:00:00 with a normalization factor of 4.

8.2.3.4. Outcome

- 1 recurring, purchase record exists for 1 eligible "Normalized Hour" of the no upfront, *commitment discount* and incurs a \$0.50 *BilledCost*.
- The VM_SMALL *commitment discount* is fully utilized within the corresponding *charge period*, covers 25% of the VM_LARGE resource, and incurs a \$0.50 *EffectiveCost*.
- The VM_LARGE resource incurs an additional, on-demand \$2.25 *BilledCost* and *EffectiveCost*.

[CSV Example](#)

8.2.4. 100% utilization with commitment discount flexibility with 2 resources

8.2.4.1. Context

For this example, fictitious service provider, *TinyCloud*, offers the following SKU catalog which is used in the scenario below.

8.2.4.2. SKU Catalog

Service	SKU Id	SKU Price Id	SKU Price Unit Price	Normalization Factor
Compute	VM_SMALL	VM_SMALL_COMMITTED_PURCHASE_NO_UPFRONT	\$0.50	1
Compute	VM_MEDIUM	VM_MEDIUM_COMMITTED_PURCHASE_NO_UPFRONT	\$1.00	2
Compute	VM_LARGE	VM_LARGE_COMMITTED_PURCHASE_NO_UPFRONT	\$1.50	3
Compute	VM_XLARGE	VM_XLARGE_COMMITTED_PURCHASE_NO_UPFRONT	\$2.00	4
Compute	VM_SMALL	VM_SMALL_COMMITTED_HOUR	\$0.50	1
Compute	VM_MEDIUM	VM_MEDIUM_COMMITTED_HOUR	\$1.00	2
Compute	VM_LARGE	VM_LARGE_COMMITTED_HOUR	\$1.50	3
Compute	VM_XLARGE	VM_XLARGE_COMMITTED_HOUR	\$2.00	4
Compute	VM_SMALL	VM_SMALL_ON_DEMAND_HOUR	\$1.00	1
Compute	VM_MEDIUM	VM_MEDIUM_ON_DEMAND_HOUR	\$2.00	2
Compute	VM_LARGE	VM_LARGE_ON_DEMAND_HOUR	\$3.00	3
Compute	VM_XLARGE	VM_XLARGE_ON_DEMAND_HOUR	\$4.00	4

The above SKU Catalog shows that this service provider only has 1 service that offers 4 virtual machine SKUs at various list rates, *commitment discount* rates, and normalization factors. Each SKU's normalization factor classifies its relative size to its *commitment discount* rate. Usage-based [commitment discounts](#) with [commitment discount flexibility](#) can fully cover any combination of 1 or more SKUs where the sum of their normalization factor is less than or equal to the normalization factor of the *commitment discount*.

8.2.4.3. Scenario

- 1 no upfront *commitment discount* is purchased for 1 year (2023) for 1 VM_XLARGE which has a normalization factor of 8.
- 2 VM_MEDIUM resources run for 1 hour from 2023-01-01T00:00:00 to 2023-01-01T01:00:00 with a normalization factor of 4 for each.

8.2.4.4. Outcome

- 1 recurring, purchase record exists for 1 eligible "Normalized Hour" for a no upfront, *commitment discount* and incurs a \$2.00 [BilledCost](#).
- With *commitment discount flexibility*, 1 *commitment discount* for a VM_XLARGE covers 2 VM_MEDIUM resources within the corresponding [charge period](#) and incurs a \$2.00 total [EffectiveCost](#).
 - 1 *commitment discount* with a normalization factor of 8 covers 2 resources with normalization factors of 4 (i.e $4 + 4 = 8$).

[CSV Example](#)

8.3. Examples: Metadata

The following sections contain examples of metadata provided by a hypothetical FOCUS data generator called ACME to supply the required reference between the [FOCUS dataset artifacts](#) and the [Metadata](#). Data Generator implementations will vary on how the metadata is disseminated; however, the data generator's chosen metadata delivery approach should be able to support the structure represented in this example.

In this example, the data generator supports delivery of FOCUS data via file export to a data storage system. It uses JSON as the format for providing the metadata. The data generator delivers data every 12 hours into a path structure described below:

Type of data	Path
Export location	/FOCUS
Metadata location	/FOCUS/metadata
Cost data location	/FOCUS/data

Here are some metadata examples for various scenarios:

8.3.1. Data Generator Metadata

8.3.1.1. Scenario

Acme provides metadata about the data generator as a part of their FOCUS data export. They provide the relevant data via the [Data Generator](#) schema object.

8.3.1.2. Supplied Metadata

Metadata can be provided at a location such as /FOCUS/metadata/data_generator.json.

The updated Data Generator-related metadata could look like this:

```
{  
  "DataGenerator": "Acme"  
}
```

8.3.2. Dataset Metadata Example

8.3.2.1. Scenario

ACME provides two FOCUS datasets: Cost and Usage and Contract. Each [Schema](#) metadata object includes the [Dataset](#) metadata to indicate which FOCUS Dataset the Schema conforms to.

8.3.2.2. Supplied Metadata

Metadata can be provided at a location such as /FOCUS/metadata/schemas/schema-1234-abcde-12345-abcde-12345.json.

The schema for the data artifact conforming to the dataset FOCUS Cost and Usage.

```

{
  "SchemaId": "1234-abcde-12345-abcde-12345",
  "FocusVersion": "1.0",
  "CreationDate": "2024-01-01T12:01:03.083z",
  "DatasetInstancelId": "178151-dbad145e-178151-dbad145e-178151",
  "ColumnDefinition": [
    {
      "ColumnName": "BillingAccountId",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8"
    },
    {
      "ColumnName": "BillingAccountName",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8"
    },
    {
      "ColumnName": "ChargePeriodStart",
      "DataType": "DATETIME"
    },
    {
      "ColumnName": "ChargePeriodEnd",
      "DataType": "DATETIME"
    },
    {
      "ColumnName": "BilledCost",
      "DataType": "DECIMAL",
      "NumericPrecision": 20,
      "NumberScale": 10
    },
    {
      "ColumnName": "EffectiveCost",
      "DataType": "DECIMAL",
      "NumericPrecision": 20,
      "NumberScale": 10
    },
    {
      "ColumnName": "Tags",
      "DataType": "JSON",
      "ProviderTagPrefixes": ["acme", "ac"]
    }
  ]
}

```

The schema for the data artifact conforming to the dataset FOCUS Contracts.

```

{
  "SchemaId": "1234-abcde-12345-abcde-12345",
  "FocusVersion": "1.0",
  "CreationDate": "2024-01-01T12:01:03.083z",
  "DatasetInstanceid": "178151-dbad145e-178151-dbad145e-246811",
  "ColumnDefinition": [
    {
      "ColumnName": "ContractId",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8"
    },
    {
      "ColumnName": "OverColumnName",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8"
    }
  ]
}

```

8.3.3. Deprecating Columns

8.3.3.1. Scenario

ACME has decided to deprecate columns prior to removal from their FOCUS data export. The column for deprecation is `x_awesome_column3`. The data generator creates a new [Schema](#) object to represent the new schema, with a unique [SchemaId](#).

8.3.3.2. Supplied Metadata

Metadata can be provided at a location such as `/FOCUS/metadata/schemas/schema-34567-abcde-34567-abcde-34567.json`.

The updated schema-related metadata could look like this:

```

{
  "SchemaId": "34567-abcde-34567-abcde-34567",
  "FocusVersion": "1.0",
  "CreationDate": "2024-03-02T12:01:03.083z",
  "DatasetInstanceid": "178151-dbad145e-178151-dbad145e-178151",
  "ColumnDefinition": [
    {
      "ColumnName": "BillingAccountid",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8"
    },
    {
      "ColumnName": "BillingAccountName",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8"
    }
  ]
}

```

```

    },
    {
      "ColumnName": "ChargePeriodStart",
      "DataType": "DATETIME"
    },
    {
      "ColumnName": "ChargePeriodEnd",
      "DataType": "DATETIME"
    },
    {
      "ColumnName": "BilledCost",
      "DataType": "DECIMAL",
      "NumericPrecision": 20,
      "NumberScale": 10
    },
    {
      "ColumnName": "EffectiveCost",
      "DataType": "DECIMAL",
      "NumericPrecision": 20,
      "NumberScale": 10
    },
    {
      "ColumnName": "Tags",
      "DataType": "JSON",
      "ProviderTagPrefixes": ["acme", "ac"]
    },
    {
      "ColumnName": "x_awesome_column1",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8"
    },
    {
      "ColumnName": "x_awesome_column2",
      "DataType": "DATETIME"
    },
    {
      "ColumnName": "x_awesome_column3",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8",
      "Deprecated": true
    }
  ]
}

```

For an example of how ACME ensures the schema metadata reference requirement is met see: [Schema Metadata to FOCUS Data Reference](#)

8.3.4. Renaming Columns

8.3.4.1. Scenario

ACME has decided to rename a column in their FOCUS data export. The column for rename is `x_awesome_column1` and will be renamed to `x_awesome_column_one`. The data generator creates a new [Schema](#) object to represent the new schema, with a unique [SchemaId](#). After this schema definition is created if the data generator creates another schema, the `PreviousColumnName` is

removed.

8.3.4.2. Supplied Metadata

Metadata can be provided at a location such as /FOCUS/metadata/schemas/schema-34567-abcde-34567-abcde-34567.json.

The updated schema related metadata for the schema where the rename took place could look like this:

```
{
  "SchemaId": "34567-abcde-34567-abcde-34567",
  "FocusVersion": "1.0",
  "CreationDate": "2024-03-02T12:01:03.083z",
  "DatasetInstanceId": "178151-dbad145e-178151-dbad145e-178151",
  "ColumnDefinition": [
    {
      "ColumnName": "BillingAccountId",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8"
    },
    {
      "ColumnName": "BillingAccountName",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8"
    },
    {
      "ColumnName": "ChargePeriodStart",
      "DataType": "DATETIME"
    },
    {
      "ColumnName": "ChargePeriodEnd",
      "DataType": "DATETIME"
    },
    {
      "ColumnName": "BilledCost",
      "DataType": "DECIMAL",
      "NumericPrecision": 20,
      "NumberScale": 10
    },
    {
      "ColumnName": "EffectiveCost",
      "DataType": "DECIMAL",
      "NumericPrecision": 20,
      "NumberScale": 10
    },
    {
      "ColumnName": "Tags",
      "DataType": "JSON",
      "ProviderTagPrefixes": ["acme", "ac"]
    },
    {
      "ColumnName": "x_awesome_column_one",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8",
      "ProviderTagPrefixes": ["acme", "ac"]
    }
  ]
}
```

```

        "PreviousColumnName": "x_awesome_column1"
    },
    {
        "ColumnName": "x_awesome_column2",
        "DataType": "DATETIME"
    },
    {
        "ColumnName": "x_awesome_column3",
        "DataType": "STRING",
        "StringMaxLength": 64,
        "StringEncoding": "UTF-8",
        "Deprecated": true
    }
]
}

```

The subsequent new schema metadata after the rename could look like this:

```

{
    "SchemaId": "34567-abcde-34567-abcde-34567",
    "FocusVersion": "1.0",
    "CreationDate": "2024-03-02T12:01:03.083z",
    "ColumnDefinition": [
        {
            "ColumnName": "BillingAccountId",
            "DataType": "STRING",
            "StringMaxLength": 64,
            "StringEncoding": "UTF-8"
        },
        {
            "ColumnName": "BillingAccountName",
            "DataType": "STRING",
            "StringMaxLength": 64,
            "StringEncoding": "UTF-8"
        },
        {
            "ColumnName": "ChargePeriodStart",
            "DataType": "DATETIME"
        },
        {
            "ColumnName": "ChargePeriodEnd",
            "DataType": "DATETIME"
        },
        {
            "ColumnName": "BilledCost",
            "DataType": "DECIMAL",
            "NumericPrecision": 20,
            "NumberScale": 10
        },
        {
            "ColumnName": "EffectiveCost",
            "DataType": "DECIMAL",
            "NumericPrecision": 20,
            "NumberScale": 10
        },
        {
            "ColumnName": "Tags",
            "DataType": "JSON",
            "ProviderTagPrefixes": ["acme", "ac"]
        }
    ]
}

```

```

    {
      "ColumnName": "x_awesome_column_one",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8"
    },
    {
      "ColumnName": "x_awesome_column2",
      "DataType": "DATETIME"
    },
    {
      "ColumnName": "x_awesome_column3",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8",
      "Deprecated": true
    }
  ]
}

```

For an example of how ACME ensures the schema metadata reference requirement is met see: [Schema Metadata to FOCUS Data Reference](#)

8.3.5. Schema Metadata

8.3.5.1. Scenario

ACME has only provided one [Schema](#) for their FOCUS data export. ACME provides a directory of schemas and each schema is a single file. Acme provides a file representing the schema for the data they provide.

8.3.5.2. Supplied Metadata

Metadata can be provided at a location such as /FOCUS/metadata/schemas/schema-1234-abcde-12345-abcde-12345.json.

The updated schema-related metadata could look like this:

```

{
  "SchemaId": "1234-abcde-12345-abcde-12345",
  "FocusVersion": "1.0",
  "CreationDate": "2024-01-01T12:01:03.083z",
  "DatasetInstanceId": "178151-dbad145e-178151-dbad145e-178151",
  "ColumnDefinition": [
    {
      "ColumnName": "BillingAccountId",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8"
    },
    {
      "ColumnName": "BillingAccountName",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8"
    },
    {
      "ColumnName": "ChargePeriodStart",
      "DataType": "DATETIME"
    },
    {
      "ColumnName": "ChargePeriodEnd",
      "DataType": "DATETIME"
    },
    {
      "ColumnName": "BilledCost",
      "DataType": "DECIMAL",
      "NumericPrecision": 20,
      "NumberScale": 10
    },
    {
      "ColumnName": "EffectiveCost",
      "DataType": "DECIMAL",
      "NumericPrecision": 20,
      "NumberScale": 10
    },
    {
      "ColumnName": "Tags",
      "DataType": "JSON",
      "ProviderTagPrefixes": ["acme", "ac"]
    }
  ]
}

```

8.3.6. Schema Metadata to FOCUS Data Reference

8.3.6.1. Scenario

ACME makes a change to the [Schema](#) of their data exports. For each FOCUS data export, ACME includes a metadata reference to the schema object. Because multiple files are provided in each export, Acme has elected to include a metadata file in each export folder that includes the FOCUS schema reference that applies to the data export files within that folder. When the schema changes, they include the new [Schema ID](#) in their export metadata file of the new folder.

8.3.6.2. Supplied Metadata

Metadata can be provided at a location such as /FOCUS/data/export1-metadata.json

The export metadata could look like this:

```
{
  "SchemaId": "1234-abcde-12345-abcde-12345",
  "data_location":
  [
    {
      "filepath": "/FOCUS/data/export1/export1-part1.csv",
      "total_bytes": 9010387,
      "total_rows": 4450
    },
    {
      "filepath": "/FOCUS/data/export1/export1-part2.csv",
      "total_bytes": 9010387,
      "total_rows": 4450
    },
    {
      "filepath": "/FOCUS/data/export1/export1-part3.csv",
      "total_bytes": 9010387,
      "total_rows": 4450
    },
    {
      "filepath": "/FOCUS/data/export1/export1-part4.csv",
      "total_bytes": 9010387,
      "total_rows": 4450
    }
  ]
}
```

New metadata can be provided at a location such as /FOCUS/data/export2-metadata.json.

The new export metadata could look like this:

```

{
  "SchemaId": "23456-abcde-23456-abcde-23456",
  "data_location":
  [
    {
      "filepath": "/FOCUS/data/export2/export2-part1.csv",
      "total_bytes": 9010387,
      "total_rows": 4450
    },
    {
      "filepath": "/FOCUS/data/export2/export2-part2.csv",
      "total_bytes": 9010387,
      "total_rows": 4450
    },
    {
      "filepath": "/FOCUS/data/export2/export2-part3.csv",
      "total_bytes": 9010387,
      "total_rows": 4450
    },
    {
      "filepath": "/FOCUS/data/export2/export2-part4.csv",
      "total_bytes": 9010387,
      "total_rows": 4450
    }
  ]
}

```

8.3.7. Data Changed by Data Generator Using Data Generator Version

8.3.7.1. Scenario

ACME specifies the optional metadata property [Data Generator Version](#) in their [Schema](#) object. They made a change to the Cost and Usage [FOCUS dataset](#) they produce that does not adopt a new FOCUS Version, nor does it make a change to the included columns, but does impact values in the data. This example illustrates that Data Generator Version changes are independent of column changes, however data generator version changes may include column changes.

The data generator creates a new schema object to represent the new schema. The data generator includes both the FOCUS Version and Data Generator Version in the schema object.

8.3.7.2. Supplied Metadata

Metadata can be provided at a location such as /FOCUS/metadata/schemas/schema-56789-abcde-56789-abcde-56789.json.

The updated schema-related metadata could look like this:

```

{
  "SchemaId": "56789-abcde-56789-abcde-56789",
  "FocusVersion": "1.1",
  "DataGeneratorVersion": "2.4",
  "CreationDate": "2024-05-02T12:01:03.083z",
  "DatasetInstancelId": "178151-dbad145e-178151-dbad145e-178151",
  "ColumnDefinition": [
    {
      "ColumnName": "BillingAccountId",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8"
    },
    {
      "ColumnName": "BillingAccountName",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8"
    },
    {
      "ColumnName": "ChargePeriodStart",
      "DataType": "DATETIME"
    },
    {
      "ColumnName": "ChargePeriodEnd",
      "DataType": "DATETIME"
    },
    {
      "ColumnName": "BilledCost",
      "DataType": "DECIMAL",
      "NumericPrecision": 20,
      "NumberScale": 10
    },
    {
      "ColumnName": "EffectiveCost",
      "DataType": "DECIMAL",
      "NumericPrecision": 20,
      "NumberScale": 10
    },
    {
      "ColumnName": "Tags",
      "DataType": "JSON",
      "DataGeneratorTagPrefixes": ["acme", "ac"]
    },
    {
      "ColumnName": "x_awesome_column1",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8"
    },
    {
      "ColumnName": "x_awesome_column2",
      "DataType": "DATETIME"
    }
  ]
}

```

For an example of how ACME ensures the schema metadata reference requirement is met see:

[Schema Metadata to FOCUS Data Reference](#)

8.3.8. Adding New Columns

8.3.8.1. Scenario

ACME has decided to add additional columns to their FOCUS data export. The new columns are `x_awesome_column1`, `x_awesome_column2`, and `x_awesome_column3`. The data generator creates a new [Schema](#) object to represent the new schema, this schema object has a unique [Schemald](#). The subsequent data exports that use the new schema include the new schema's id as a reference to their corresponding schema object.

8.3.8.2. Supplied Metadata

Metadata can be provided at a location such as `/FOCUS/metadata/schemas/schema-23456-abcde-23456-abcde-23456.json`.

The updated schema-related metadata could look like this:

```
{
  "Schemald": "23456-abcde-23456-abcde-23456",
  "FocusVersion": "1.0",
  "CreationDate": "2024-02-02T12:01:03.083z",
  "DatasetInstanceid": "178151-dbad145e-178151-dbad145e-178151",
  "ColumnDefinition": [
    {
      "ColumnName": "BillingAccountId",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8"
    },
    {
      "ColumnName": "BillingAccountName",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8"
    },
    {
      "ColumnName": "ChargePeriodStart",
      "DataType": "DATETIME"
    },
    {
      "ColumnName": "ChargePeriodEnd",
      "DataType": "DATETIME"
    },
    {
      "ColumnName": "BilledCost",
      "DataType": "DECIMAL",
      "NumericPrecision": 20,
      "NumberScale": 10
    },
    {
      "ColumnName": "EffectiveCost",
      "DataType": "DECIMAL",
      "NumericPrecision": 20,
      "NumberScale": 10
    }
  ]
}
```

```

    "ColumnName": "Tags",
    "DataType": "JSON",
    "ProviderTagPrefixes": ["awecorp", "ac"]
  },
  {
    "ColumnName": "x_awesome_column1",
    "DataType": "STRING",
    "StringMaxLength": 64,
    "StringEncoding": "UTF-8"
  },
  {
    "ColumnName": "x_awesome_column2",
    "DataType": "DATETIME"
  },
  {
    "ColumnName": "x_awesome_column3",
    "DataType": "STRING",
    "StringMaxLength": 64,
    "StringEncoding": "UTF-8"
  }
]
}

```

For an example of how ACME ensures the schema metadata reference requirement is met see: [Schema Metadata to FOCUS Data Reference](#)

8.3.9. Removing Columns

8.3.9.1. Scenario

ACME has decided to remove columns from their FOCUS data export. The column removed is `x_awesome_column3`. The data generator creates a new [Schema](#) object to represent the new schema, with a unique [Schemald](#).

8.3.9.2. Supplied Metadata

Metadata can be provided at a location such as `/FOCUS/metadata/schemas/schema-34567-abcde-34567-abcde-34567.json`.

The updated schema related metadata could look like this:

```

{
  "SchemaId": "34567-abcde-34567-abcde-34567",
  "FocusVersion": "1.0",
  "CreationDate": "2024-03-02T12:01:03.083z",
  "DatasetInstanceId": "178151-dbad145e-178151-dbad145e-178151",
  "ColumnDefinition": [
    {
      "ColumnName": "BillingAccountId",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8"
    },
    {
      "ColumnName": "BillingAccountName",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8"
    },
    {
      "ColumnName": "ChargePeriodStart",
      "DataType": "DATETIME"
    },
    {
      "ColumnName": "ChargePeriodEnd",
      "DataType": "DATETIME"
    },
    {
      "ColumnName": "BilledCost",
      "DataType": "DECIMAL",
      "NumericPrecision": 20,
      "NumberScale": 10
    },
    {
      "ColumnName": "EffectiveCost",
      "DataType": "DECIMAL",
      "NumericPrecision": 20,
      "NumberScale": 10
    },
    {
      "ColumnName": "Tags",
      "DataType": "JSON",
      "ProviderTagPrefixes": ["acme", "ac"]
    },
    {
      "ColumnName": "x_awesome_column1",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8"
    },
    {
      "ColumnName": "x_awesome_column2",
      "DataType": "DATETIME"
    }
  ]
}

```

For an example of how ACME ensures the schema metadata reference requirement is met see: [Schema Metadata to FOCUS Data Reference](#)

8.3.10. Changing Column Metadata

8.3.10.1. Scenario

ACME has decided to change the datatype of column `x_awesome_column1` from a string to a number. ACME creates a new [Schema](#) object with the modification to `x_awesome_column2`.

8.3.10.2. Supplied Metadata

Metadata can be provided at a location such as `/FOCUS/metadata/schemas/schema-67891-abcde-67891-abcde-67891.json`.

The updated schema-related metadata could look like this:

```

{
  "SchemaId": "67891-abcde-67891-abcde-67891",
  "FocusVersion": "1.0",
  "CreationDate": "2024-06-02T12:01:03.083z",
  "DatasetInstanceId": "178151-dbad145e-178151-dbad145e-178151",
  "ColumnDefinition": [
    {
      "ColumnName": "BillingAccountId",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8"
    },
    {
      "ColumnName": "BillingAccountName",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8"
    },
    {
      "ColumnName": "ChargePeriodStart",
      "DataType": "DATETIME"
    },
    {
      "ColumnName": "ChargePeriodEnd",
      "DataType": "DATETIME"
    },
    {
      "ColumnName": "BilledCost",
      "DataType": "DECIMAL",
      "NumericPrecision": 20,
      "NumberScale": 10
    },
    {
      "ColumnName": "EffectiveCost",
      "DataType": "DECIMAL",
      "NumericPrecision": 20,
      "NumberScale": 10
    },
    {
      "ColumnName": "Tags",
      "DataType": "JSON",
      "ProviderTagPrefixes": ["acme", "ac"]
    },
    {
      "ColumnName": "x_awesome_column1",
      "DataType": "DECIMAL",
      "NumericPrecision": 20,
      "NumberScale": 10
    },
    {
      "ColumnName": "x_awesome_column2",
      "DataType": "DATETIME"
    }
  ]
}

```

For an example of how ACME ensures the schema metadata reference requirement is met see: [Schema Metadata to FOCUS Data Reference](#)

8.3.11. Data Generator Metadata Error Correction

8.3.11.1. Scenario

ACME has discovered that while their export includes the column `x_awesome_column3`, the [Schema](#) metadata does not include this column. In this case, the data generator fixes the metadata in the existing schema object and does not need to create a new schema object. Reference metadata remains the same.

8.3.11.2. Supplied Metadata

Metadata can be provided at a location such as `/FOCUS/metadata/schemas/schema-34567-abcde-34567-abcde-34567.json`.

The updated schema-related metadata could look like this:

```

{
  "SchemaId": "34567-abcde-34567-abcde-34567",
  "FocusVersion": "1.0",
  "CreationDate": "2024-03-02T12:01:03.083z",
  "DatasetInstanceId": "178151-dbad145e-178151-dbad145e-178151",
  "ColumnDefinition": [
    {
      "ColumnName": "BillingAccountId",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8"
    },
    {
      "ColumnName": "BillingAccountName",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8"
    },
    {
      "ColumnName": "ChargePeriodStart",
      "DataType": "DATETIME"
    },
    {
      "ColumnName": "ChargePeriodEnd",
      "DataType": "DATETIME"
    },
    {
      "ColumnName": "BilledCost",
      "DataType": "DECIMAL",
      "NumericPrecision": 20,
      "NumberScale": 10
    },
    {
      "ColumnName": "EffectiveCost",
      "DataType": "DECIMAL",
      "NumericPrecision": 20,
      "NumberScale": 10
    },
    {
      "ColumnName": "Tags",
      "DataType": "JSON",
      "ProviderTagPrefixes": ["acme", "ac"]
    },
    {
      "ColumnName": "x_awesome_column1",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8"
    },
    {
      "ColumnName": "x_awesome_column2",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8"
    }
  ]
}

```

8.3.12. FOCUS Version Changed

8.3.12.1. Scenario

ACME's previous exports used FOCUS version 1.0. They are now going to adopt FOCUS version 1.1. It is required that they create a new schema metadata object which specifies the new FOCUS version via the [FOCUS Version](#) property—regardless of schema changes. In this example, the new FOCUS version adoption doesn't include columns changes. This is to illustrate that FOCUS version changes are independent of column changes, however, this scenario is unlikely.

8.3.12.2. Supplied Metadata

Metadata can be provided at a location such as `/FOCUS/metadata/schemas/schema-45678-abcde-45678-abcde-45678.json`.

The updated schema-related metadata could look like this:

```

{
  "SchemaId": "45678-abcde-45678-abcde-45678",
  "FocusVersion": "1.1",
  "CreationDate": "2024-04-02T12:01:03.083z",
  "DatasetInstanceId": "178151-dbad145e-178151-dbad145e-178151",
  "ColumnDefinition": [
    {
      "ColumnName": "BillingAccountId",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8"
    },
    {
      "ColumnName": "BillingAccountName",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8"
    },
    {
      "ColumnName": "ChargePeriodStart",
      "DataType": "DATETIME"
    },
    {
      "ColumnName": "ChargePeriodEnd",
      "DataType": "DATETIME"
    },
    {
      "ColumnName": "BilledCost",
      "DataType": "DECIMAL",
      "NumericPrecision": 20,
      "NumberScale": 10
    },
    {
      "ColumnName": "EffectiveCost",
      "DataType": "DECIMAL",
      "NumericPrecision": 20,
      "NumberScale": 10
    },
    {
      "ColumnName": "Tags",
      "DataType": "JSON",
      "ProviderTagPrefixes": ["acme", "ac"]
    },
    {
      "ColumnName": "x_awesome_column1",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8"
    },
    {
      "ColumnName": "x_awesome_column2",
      "DataType": "DATETIME"
    }
  ]
}

```

For an example of how ACME ensures the schema metadata reference requirement is met see: [Schema Metadata to FOCUS Data Reference](#)

8.3.13. FOCUS Version Changed by Data Generator Using Data Generator Version

8.3.13.1. Scenario

ACME specifies the optional metadata property [Data Generator Version](#) in their [Schema](#) object. Their data generator version 2.2 supported FOCUS version 1.0. They are now going to adopt FOCUS Version 1.1 which requires that they update their Data Generator Version when updating the FOCUS Version. They create a new schema object designating that both properties have changed. In this example, the adoption of the new FOCUS version doesn't include additional columns. This is to illustrate that Data Generator Version can change independent of column changes; however, this scenario is unlikely.

The data generator creates a new schema object to represent the new schema. The data generator includes both the new FOCUS Version and Data Generator Version in the schema object.

8.3.13.2. Supplied Metadata

Metadata can be provided at a location such as `/FOCUS/metadata/schemas/schema-45678-abcde-45678-abcde-45678.json`.

The updated schema-related metadata could look like this:

```

{
  "SchemaId": "45678-abcde-45678-abcde-45678",
  "FocusVersion": "1.1",
  "DataGeneratorVersion": "2.3",
  "name": "New Columns",
  "CreationDate": "2024-04-02T12:01:03.083z",
  "DatasetInstanceId": "178151-dbad145e-178151-dbad145e-178151",
  "ColumnDefinition": [
    {
      "ColumnName": "BillingAccountId",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8"
    },
    {
      "ColumnName": "BillingAccountName",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8"
    },
    {
      "ColumnName": "ChargePeriodStart",
      "DataType": "DATETIME"
    },
    {
      "ColumnName": "ChargePeriodEnd",
      "DataType": "DATETIME"
    },
    {
      "ColumnName": "BilledCost",
      "DataType": "DECIMAL",
      "NumericPrecision": 20,
      "NumberScale": 10
    },
    {
      "ColumnName": "EffectiveCost",
      "DataType": "DECIMAL",
      "NumericPrecision": 20,
      "NumberScale": 10
    },
    {
      "ColumnName": "Tags",
      "DataType": "JSON",
      "ProviderTagPrefixes": ["acme", "ac"]
    },
    {
      "ColumnName": "x_awesome_column1",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8"
    },
    {
      "ColumnName": "x_awesome_column2",
      "DataType": "DATETIME"
    }
  ]
}

```

For reference, the prior schema object looked like this:

```

{
  "SchemaId": "34567-abcde-34567-abcde-34567",
  "FocusVersion": "1.0",
  "DataGeneratorVersion": "2.2",
  "CreationDate": "2024-04-02T12:01:03.083z",
  "Dataset": "FOCUS Cost and Usage",
  "ColumnDefinition": [
    {
      "ColumnName": "BillingAccountId",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8"
    },
    {
      "ColumnName": "BillingAccountName",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8"
    },
    {
      "ColumnName": "ChargePeriodStart",
      "DataType": "DATETIME"
    },
    {
      "ColumnName": "ChargePeriodEnd",
      "DataType": "DATETIME"
    },
    {
      "ColumnName": "BilledCost",
      "DataType": "DECIMAL",
      "NumericPrecision": 20,
      "NumberScale": 10
    },
    {
      "ColumnName": "EffectiveCost",
      "DataType": "DECIMAL",
      "NumericPrecision": 20,
      "NumberScale": 10
    },
    {
      "ColumnName": "Tags",
      "DataType": "JSON",
      "ProviderTagPrefixes": ["acme", "ac"]
    },
    {
      "ColumnName": "x_awesome_column1",
      "DataType": "STRING",
      "StringMaxLength": 64,
      "StringEncoding": "UTF-8"
    },
    {
      "ColumnName": "x_awesome_column2",
      "DataType": "DATETIME"
    }
  ]
}

```

For an example of how ACME ensures the schema metadata reference requirement is met see:

[Schema Metadata to FOCUS Data Reference](#)

8.3.14. Dataset Instance Metadata

8.3.14.1. Scenario

ACME provides three dataset instances: "Cost and Usage Daily," "Cost and Usage Hourly," and "Contract Commitments," corresponding to the FOCUS datasets Cost and Usage and Contract Commitment. ACME also provides a metadata directory containing a single file with metadata for each dataset instance.

8.3.14.2. Supplied Metadata

Metadata can be provided at a location such as /FOCUS/metadata/dataset_instances.json.

The updated schema-related metadata could look like this:

```
[
  {
    "DatasetInstanceId": "178151-dbad145e-178151-dbad145e-246811",
    "DatasetInstanceName": "Contract Commitments Report",
    "FocusDatasetId": "ContractCommitment"
  },
  {
    "DatasetInstanceId": "178151-dbad145e-178151-dbad145e-178151",
    "DatasetInstanceName": "Cost and Usage Daily",
    "FocusDatasetId": "CostAndUsage"
  },
  {
    "DatasetInstanceId": "178151-ja23h1287-387151-dbad145e-134657",
    "DatasetInstanceName": "Cost and Usage Hourly",
    "FocusDatasetId": "CostAndUsage"
  }
]
```

8.3.15. Recency Metadata

8.3.15.1. Scenario

ACME has elected to add recency metadata to its FOCUS data export. ACME provides a directory of recency metadata for each dataset they provide and each recency object is a single file.

8.3.15.2. Supplied Metadata

Metadata can be provided at a location such as /FOCUS/metadata/recency/recency-1234-abcde-12345-abcde-12345.json.

The provided recency metadata for a time series dataset could look like this:

```

{
  "DatasetInstanceid": "1234-abcde-12345-abcde-12345",
  "RecencyLastUpdateDate": "2025-01-29T12:01:03.083z",
  "TimeSectors": [
    {
      "TimeSectorStart": "2025-01-27T0:00:00z",
      "TimeSectorEnd": "2025-01-27T1:00:00z",
      "TimeSectorComplete": true,
      "TimeSectorLastUpdated": "2025-01-29T04:00:00z"
    },
    {
      "TimeSectorStart": "2025-01-27T1:00:00z",
      "TimeSectorEnd": "2025-01-27T2:00:00z",
      "TimeSectorComplete": true,
      "TimeSectorLastUpdated": "2025-01-29T04:00:00z"
    },
    {
      "TimeSectorStart": "2025-01-27T2:00:00z",
      "TimeSectorEnd": "2025-01-27T3:00:00z",
      "TimeSectorComplete": true,
      "TimeSectorLastUpdated": "2025-01-29T04:00:00z"
    },
    {
      "TimeSectorStart": "2025-01-27T3:00:00z",
      "TimeSectorEnd": "2025-01-27T4:00:00z",
      "TimeSectorComplete": true,
      "TimeSectorLastUpdated": "2025-01-29T04:00:00z"
    },
    {
      "TimeSectorStart": "2025-01-27T4:00:00z",
      "TimeSectorEnd": "2025-01-27T5:00:00z",
      "TimeSectorComplete": true,
      "TimeSectorLastUpdated": "2025-01-29T04:00:00z"
    },
    {
      "TimeSectorStart": "2025-01-27T5:00:00z",
      "TimeSectorEnd": "2025-01-27T6:00:00z",
      "TimeSectorComplete": false,
      "TimeSectorLastUpdated": "2025-01-29T04:00:00z"
    },
    {
      "TimeSectorStart": "2025-01-27T6:00:00z",
      "TimeSectorEnd": "2025-01-27T7:00:00z",
      "TimeSectorComplete": false,
      "TimeSectorLastUpdated": "2025-01-29T04:00:00z"
    }
  ]
}

```

The provided recency metadata for non-time series dataset could look like this:

```

{
  "DatasetInstanceid": "54321-abcde-12345-abcde-12345",
  "RecencyLastUpdateDate": "2025-01-29T12:01:03.083z",
  "DatasetInstanceLastUpdated": "2025-01-29T04:00:00z",
  "DatasetInstanceComplete": true
}

```

8.3.16. Recency Metadata Update (Non Time Series)

8.3.16.1. Scenario

ACME provides recency metadata to accompany their FOCUS data export. ACME updates their FOCUS Contracts dataset, a non time-series dataset, every day. In this case, the most recent update to the recency data indicates the dataset and associated data artifact has been updated and is considered complete.

8.3.16.2. Supplied Metadata

Metadata can be provided at a location such as /FOCUS/metadata/recency/recency-54321-abcde-12345-abcde-12345.json.

The provided recency metadata for non-time series dataset could look like this:

```
{
  "DatasetInstanceid": "54321-abcde-12345-abcde-12345",
  "RecencyLastUpdate": "2025-01-29T15:01:03.083z",
  "DatasetInstanceLastUpdated" : "2025-01-29T010:00:00z",
  "DatasetInstanceComplete" : true
}
```

8.3.17. Recency Metadata Update (Time Series)

8.3.17.1. Scenario

ACME provides recency metadata to accompany its FOCUS data export. ACME updates its FOCUS Cost and Usage dataset (time series) every hour; however, the data lags by two days. Here, the most recent update to the recency data indicates the previous time sectors are now TimeSectorComplete. It also indicates that previous time sectors have been updated in the dataset. New time sectors have also been added.

8.3.17.2. Supplied Metadata

Metadata can be provided at a location such as /FOCUS/metadata/recency/recency-1234-abcde-12345-abcde-12345.json.

The provided recency metadata for time series dataset could look like this:

```
{
  "DatasetInstanceid": "1234-abcde-12345-abcde-12345",
  "RecencyLastUpdateDate": "2025-01-29T12:01:03.083z",
  "TimeSectors": [
    {
      "TimeSectorStart": "2025-01-27T0:00:00z",
      "TimeSectorEnd" : "2025-01-27T1:00:00z",
      "TimeSectorComplete" : true,
      "TimeSectorLastUpdated" : "2025-01-29T04:00:00z"
    },
  ],
}
```

```
{
  "TimeSectorStart": "2025-01-27T1:00:00z",
  "TimeSectorEnd" : "2025-01-27T2:00:00z",
  "TimeSectorComplete" : true,
  "TimeSectorLastUpdated" : "2025-01-29T04:00:00z"
},
{
  "TimeSectorStart": "2025-01-27T2:00:00z",
  "TimeSectorEnd" : "2025-01-27T3:00:00z",
  "TimeSectorComplete" : true,
  "TimeSectorLastUpdated" : "2025-01-29T04:00:00z"
},
{
  "TimeSectorStart": "2025-01-27T3:00:00z",
  "TimeSectorEnd" : "2025-01-27T4:00:00z",
  "TimeSectorComplete" : true,
  "TimeSectorLastUpdated" : "2025-01-29T04:00:00z"
},
{
  "TimeSectorStart": "2025-01-27T4:00:00z",
  "TimeSectorEnd" : "2025-01-27T5:00:00z",
  "TimeSectorComplete" : true,
  "TimeSectorLastUpdated" : "2025-01-29T04:00:00z"
},
{
  "TimeSectorStart": "2025-01-27T5:00:00z",
  "TimeSectorEnd" : "2025-01-27T6:00:00z",
  "TimeSectorComplete" : true,
  "TimeSectorLastUpdated" : "2025-01-29T04:00:00z"
},
{
  "TimeSectorStart": "2025-01-27T6:00:00z",
  "TimeSectorEnd" : "2025-01-27T7:00:00z",
  "TimeSectorComplete" : true,
  "TimeSectorLastUpdated" : "2025-01-29T11:15:24z"
},
{
  "TimeSectorStart": "2025-01-27T7:00:00z",
  "TimeSectorEnd" : "2025-01-27T8:00:00z",
  "TimeSectorComplete" : true,
  "TimeSectorLastUpdated" : "2025-01-29T11:15:24z"
},
{
  "TimeSectorStart": "2025-01-27T8:00:00z",
  "TimeSectorEnd" : "2025-01-27T9:00:00z",
  "TimeSectorComplete" : false,
  "TimeSectorLastUpdated" : "2025-01-29T04:00:00z"
},
{
  "TimeSectorStart": "2025-01-27T9:00:00z",
  "TimeSectorEnd" : "2025-01-27T10:00:00z",
  "TimeSectorComplete" : false,
  "TimeSectorLastUpdated" : "2025-01-29T10:23:10z"
},
{
  "TimeSectorStart": "2025-01-27T10:00:00z",
  "TimeSectorEnd" : "2025-01-27T11:00:00z",
  "TimeSectorComplete" : false,
  "TimeSectorLastUpdated" : "2025-01-29T11:15:24z"
},
,
```

```

    }
  ]
}

```

The provided recency metadata for non-time series dataset could look like this:

```

{
  "Dataset": "1234-abcde-12345-abcde-12345",
  "RecencyLastUpdateDate": "2025-01-29T12:01:03.083z",
  "DatasetInstanceLastUpdated": "2025-01-29T04:00:00z",
  "DatasetInstanceComplete": true
}

```

8.4. Examples: Participating Entity Identification

Understanding cost and usage data in billing datasets requires identifying the roles of several participating entities involved in resource or service provisioning, invoicing, and data generation. The FOCUS Specification includes multiple columns to identify key participating entities, these include:

- [Service Provider Name](#)
- [Invoice Issuer Name](#)
- [Host Provider Name](#)
- [Data Generator](#)

The value for each of these may vary depending on how *resources* or *services* are obtained — whether directly from a Cloud Service Provider (CSP) or a SaaS provider, via a Managed Service Provider (MSP), through a cloud marketplace, or from internal service offerings. The table below provides examples that illustrate how the value for each dimension may shift depending on the method of acquisition and other contributing factors.

#	Scenario	Service Provider	Invoice Issuer	Host Provider	Data Generator
1.1	Purchasing cloud resources or services directly from a CSP	CSP	CSP	CSP	CSP
1.2	Purchasing cloud resources or services from a CSP, where the underlying resources are operated by a 3rd party.	CSP	CSP	Entity operating the region for the CSP	CSP
2.1	Purchasing cloud resources or services via an MSP, with visibility into the underlying hosting provider.	MSP	MSP	CSP	MSP
2.2	Purchasing cloud resources or services via an MSP, without visibility into the underlying hosting provider.	MSP	MSP	MSP	MSP
2.3	Purchasing cloud-agnostic resources or services from an MSP	MSP	MSP	MSP	MSP
2.4	Purchasing labor services from an MSP	MSP	MSP		MSP

#	Scenario	Service Provider	Invoice Issuer	Host Provider	Data Generator
3.1.1	Purchase records for cloud marketplace offerings running on your CSP infrastructure and billed by the CSP.	Marketplace Seller	CSP	CSP	CSP
3.1.2	CSP Infrastructure usage records for cloud marketplace offerings running on your CSP infrastructure.	CSP	CSP	CSP	CSP
3.1.3	Usage records for cloud marketplace offerings running on your CSP infrastructure and billed by the CSP.	Marketplace Seller	CSP	CSP	Marketplace Seller
3.2.1	Purchase records for cloud marketplace offerings not running on your cloud infrastructure, with visibility into the underlying hosting provider.	Marketplace Seller	CSP	CSP	CSP
3.2.2	Usage records for cloud marketplace offerings not running on your cloud infrastructure, with visibility into the underlying hosting provider.	Marketplace Seller	CSP	CSP	Marketplace Seller
3.3.1	Purchase records for cloud marketplace offerings not running on your cloud infrastructure, without visibility into the underlying hosting provider.	Marketplace Seller	CSP	Marketplace Seller	CSP
3.3.2	Usage records for cloud marketplace offerings not running on your cloud infrastructure, without visibility into the underlying hosting provider.	Marketplace Seller	CSP	Marketplace Seller	Marketplace Seller
3.4.1	Purchase records for SaaS products not running on your cloud infrastructure, purchased via a reseller. Reseller is issuing payable invoices.	SaaS Provider	Reseller	SaaS Provider	SaaS Provider

#	Scenario	Service Provider	Invoice Issuer	Host Provider	Data Generator
3.4.2	Usage records for SaaS products not running on your cloud infrastructure, purchased via a reseller. Reseller is issuing payable invoices.	SaaS Provider	Reseller	SaaS Provider	SaaS Provider
3.5.1	Purchase records for SaaS products not running on your cloud infrastructure, purchased via a reseller. Reseller does not issue payable invoices.	SaaS Provider	SaaS Provider	SaaS Provider	SaaS Provider
3.5.2	Usage records for SaaS products not running on your cloud infrastructure, purchased via a reseller. Reseller does not issue payable invoices.	SaaS Provider	SaaS Provider	SaaS Provider	SaaS Provider
3.6.1	Purchase records for SaaS products that have been white-labeled and sold by a reseller, not running on your cloud infrastructure. Reseller issues payable invoices.	Reseller	Reseller	Reseller	Reseller
3.6.2	Usage records for SaaS products that have been white-labeled and sold by a reseller, not running on your cloud infrastructure. Reseller issues payable invoices.	Reseller	Reseller	Reseller	Reseller
4.1	Purchasing SaaS products directly from a SaaS provider, with visibility into the underlying hosting provider.	SaaS Provider	SaaS Provider	CSP	SaaS Provider
4.2	Purchasing SaaS products directly from a SaaS provider, without visibility into the underlying hosting provider.	SaaS Provider	SaaS Provider	SaaS Provider	SaaS Provider
4.3.1	Purchasing SaaS products running on your cloud infrastructure, purchased directly from a SaaS provider (see 4.3.2 for charges related to the underlying cloud infrastructure).	SaaS Provider	SaaS Provider		SaaS Provider

#	Scenario	Service Provider	Invoice Issuer	Host Provider	Data Generator
4.3.2	Purchasing resources and services from a CSP used to host SaaS products separately acquired from a SaaS provider (see 4.3.1 for charges related to the SaaS products).	CSP	CSP	CSP	CSP
5.1	Purchasing internal resources or services hosted in Data Center	Internal Name	Internal Name	Internal Name	Internal Name
6.1	Software license costs, reported separately from the costs of the resources or services they apply to.	Licensable Software Provider	License Seller		License Seller

8.5. Examples: SaaS

The following section contains examples with how SaaS data generators may implement the FOCUS specification. SaaS data generator implementations will vary on the level of the detail available in their data, contract terms, purchasing options, and other factors.

8.5.1. Simple SaaS Agreements

Many SaaS providers provide simple contract terms, therefore don't need to support complex scenarios like spend commitments or pricing strategies in their billing data.

The scenarios described below illustrate how a Cost and Usage [FOCUS dataset](#) should look for simple SaaS agreement scenarios (these scenarios may not be specific to SaaS agreements only).

8.5.1.1. Scenario A1: Invoice Up-front for a Purchase of a Service

ACME Corp allows its customers to purchase their service for a term (in this case, a year) for a \$10,000. ACME provides AwesomeCorp with a single invoice for their usage. ACME does not provide detailed cost and usage reports to AwesomeCorp throughout the Charge Period after the initial purchase.

Given that ACME does not charge based on or track usage, its usage details are irrelevant to this scenario.

[CSV Example](#)

Note the following details in the example dataset:

- The Charge Period is April 1st 2025 - April 1st 2026. The Billing Period is the month of April 2025 (when the licenses were ordered) and therefore will appear in the April invoice.
- A single charge representing the total payment for the 12-month agreement (\$10,000) is charged in the first invoice. BilledCost and EffectiveCost are realized in the same record since detailed usage records will not be provided during the 12-month period to realize amortized portions of this up-front payment.
- The single charge record does not include a List Unit Price, Pricing Quantity, or SKU-related information. Alternatively, the Pricing Quantity could have been set to 1, and the List Unit Price

could be the same as the total charge.

8.5.1.2. Scenario A2: Invoice Up-front for a Quantity of a Service

ACME Corp offers its customer the ability to purchase a fixed quantity of licenses for their service. ACME provides AwesomeCorp with a single invoice for their usage. ACME does not provide detailed cost and usage reports to AwesomeCorp throughout the Charge Period after the initial purchase.

On April 1st, 2025, ACME executes a contract and invoices AwesomeCorp \$50,000 (Billed Cost) for a Charge Period of April 1st 2025 to April 1st 2026. As there is no negotiated discount, List Cost of the purchase is also \$50,000.

[CSV Example](#)

Note the following details in the example dataset:

- The Charge Period is April 1st 2025 to April 1st 2026. The Billing Period is the month of April 2025 (when the licenses were ordered) and therefore will appear in the April invoice.
- A single charge representing the total payment for the 12-month agreement is charged in the first invoice. Billed Cost and Effective Cost are both realized in the same record since detailed usage records will not be provided during the 12-month period to realize amortized portions of this up-front payment.
- The single charge provided includes a ListUnitPrice for the licenses and a Pricing Quantity.

8.5.1.3. Scenario A3: Additional Purchase Records Provided in the SaaS Data Generator's FOCUS Dataset

On June 1st 2025 ACME provides the following records due to AwesomeCorp's \$1,000 mid-contract purchase of an additional 10 licenses for the same Charge Period (April 1st 2025 to April 1st 2026).

[CSV Example](#)

Note the following additional details in the example dataset:

- The Charge Period is still April 1st 2025 to April 1st 2026. The Billing Period is now the month of June 2025 (when the additional licenses were ordered) and therefore will appear in the June 2025 invoice.

8.5.1.4. Scenario B: Billed in Arrears for a Quantity of a Service

Similar to Scenario A above, ACME Corp offers its customer the ability to purchase their service with a fixed quantity of licenses. However, in Scenario B, ACME issues the invoice at the end of the usage period.

On April 1st, 2026, ACME invoices AwesomeCorp \$50,000 (Billed Cost) for the Charge Period of April 1st 2025 to April 1st 2026. As there is no negotiated discount, List Cost of the purchase is also \$50,000.

[CSV Example](#)

Note the following additional details in the example dataset:

- The Charge Period is April 1st 2025 to April 1st 2026. The Billing Period is now the month of March 2026 (since this charge is invoiced as of the last month of the Charge Period).

8.5.1.5. Scenario C: Simple SaaS Agreement with Monthly Billing

Like Scenario A2 above, ACME Corp offers its customers the ability to purchase their service with a fixed quantity of licenses. However, in Scenario C, ACME issues invoices at the end of each month (usage period). For this scenario, contract terms additionally include the following terms:

- ACME charges users monthly for the licenses that were consumed in that Billing Period
- The licenses are charged at \$20 per license per month

AwesomeCorp's consumption looks like this:

- In April 2025, AwesomeCorp uses 505 licenses
- In May 2025, AwesomeCorp uses 650 licenses
- In June 2025, AwesomeCorp uses 635 licenses

CSV Example

Note the following additional details in the example dataset:

- The Charge Period and Billing Period are April 1st, 2025, to May 1st, 2025, for the first month. Subsequent months increment the Charge Period and Billing Period by one month to match the month the charges are incurred.
- Billed Cost and Effective Cost are the same value since there is no up-front payment to amortize

8.5.2. SaaS Spend Agreements

Many SaaS service providers support billing models that allow (or in some cases require) consumers to agree to an amount to spend over a period. In some cases, customers receive a negotiated discount for usage during that period in exchange for the spend agreement. Spend agreements can have different payment models like billing in arrears or pre-paid contracts and may impose minimum spend requirements for parts of the agreement.

The scenarios described below illustrate how a Cost and Usage [FOCUS dataset](#) should look for various spend agreement scenarios.

8.5.2.1. Baseline Scenario

The following baseline conditions apply to the scenarios described below:

- AwesomeCorp has signed an agreement with SaaS service provider Acme Co to use their database services
- On April 1 2025, AwesomeCorp agrees to spend \$1200 (post-discounts) in the upcoming 12-months
- AwesomeCorp receives a 20% negotiated discount in return for the commitment
- Acme Co calculates the spend counted against the agreements after discounts (like the negotiated discounts). Other service providers may use the cost after discounts i.e., using List Cost for calculating the spend commitment.

8.5.2.2. Scenario A: Billed in arrears

For this scenario A, contract includes the following terms in addition to the baseline scenario mentioned above:

- All charges will be billed in arrears at a monthly frequency

8.5.2.2.1. Scenario A1: Billed in arrears with no minimum spend requirement per

month

For this scenario, contract additionally includes the following terms:

- Committed spend can be used anytime within the 1-year commitment period.

AwesomeCorp's consumption looks like this:

- In the first month, AwesomeCorp uses \$48 of services (4 server hours). This usage has a List Cost of \$60 (before discounts)
- In the following 2 months, AwesomeCorp has some more usage
- For the final 9 months, AwesomeCorp does not use Acme services

[CSV Example](#)

Note the following details in the example dataset:

- A single charge representing the total unused amount from the 12-month agreement is charged during the final month of the 12-month commitment period

8.5.2.2.2. Scenario A2: Billed in arrears with a minimum spend requirement per month

The spend agreement with Acme requires the customer to spend a minimum amount in each Billing Period (monthly). Unused fees are charged per Billing Period when the consumption is below this level (use-it or lose-it). For this scenario, contract additionally includes the following terms:

- A minimum of \$60 needs to be spent each month

AwesomeCorp's consumption looks like this:

- In the first month, the AwesomeCorp uses \$48 of services (4 server hours). This usage has a List Cost of \$60 (before discounts). For this month, Acme charges \$12 (ListCost of \$15) for not meeting the monthly minimum
- In the following 2 months, AwesomeCorp has usage at or above the minimum requirement
- For the final 9 months, AwesomeCorp does not use Acme services

[CSV Example](#)

Note the following details in the example dataset:

- A monthly charge representing the unused minimum monthly amount is charged during months 4 through 11 of the 12-month commitment period
- The final month has a charge that captures the overall unmet spend requirement for the 12-month contract. Alternatively, this could be provided as two charges, one for the unused portion of the final month, and one to capture the overall unmet spend requirement.

8.5.2.3. Scenario B: Prepaid contract

For this scenario B, contract includes the following terms in addition to the baseline scenario mentioned above:

- The charges will be billed in arrears using monthly invoices

8.5.2.3.1. Scenario B1: Prepaid with no minimum spend requirement per month

Scenario B1 is similar to scenario A1 with the difference being that it's a pre-paid contract.

[CSV Example](#)

Note the following details in the example dataset:

- A purchase record for the initial \$1200 payment is present representing List, Billed, and Contracted cost of the purchase
- The charge for the unused amount has a \$0 BilledCost (since the total amount was billed with the prepayment). However, the charge captures the unused portion as an EffectiveCost.
- The unused charge rows apply to the entire Charge Period the contract was signed for.
- This scenario shows List Cost and Contracted Cost column double counting dynamic (described here in [ListCost](#) and [ContractedCost](#)) where either the [ChargeCategory](#) Purchase or Usage rows need to be excluded depending on the reporting scenario.

8.5.2.3.2. Scenario B2: Prepaid with a minimum spend requirement per month

Scenario B2 is similar to A2 with the difference being that it's a pre-paid contract.

[CSV Example](#)

Note the following details in the example dataset:

- A purchase record for the initial \$1200 payment is present representing List, Billed, and Contracted cost of the purchase
- The monthly charge for the unused amount has a \$0 BilledCost (since the total amount was billed with the prepayment). However, the charge captures the unused portion as an EffectiveCost.
- The final month has a charge that captures the overall unmet spend requirement for the 12-month contract. Alternatively, this could be provided as two charges, one for the unused portion of the final month, and one to capture the overall unmet spend requirement.
- This scenario shows List Cost and Contracted Cost column double counting dynamic (described here in [ListCost](#) and [ContractedCost](#)) where either the Purchase or Usage rows need to be excluded depending on the reporting scenario.

8.5.3. Virtual Currency Pricing Model

Many SaaS service providers support pricing models that utilize virtual currencies such as credits, tokens, or points. Charges may be provided using a virtual currency, which can subsequently be converted to a national currency such as USD or EUR at an advertised or agreed-upon conversion rate.

The scenarios described below illustrate how a Cost and Usage [FOCUS dataset](#) should look for various scenarios where a provider utilizes this pricing model.

8.5.3.1. Baseline Scenario

The following baseline conditions apply to the scenarios described below:

- AwesomeCorp has signed an agreement with SaaS service provider Acme Co to use their services.
- Acme Co offers a virtual currency pricing model for their services and requires a purchase of virtual currency in advance of usage. Their denomination of virtual currency is called "tokens".
- Acme Co requires purchase of additional tokens in the event of usage exceeding purchased tokens.
- Acme Co publicly lists the cost of their tokens at \$2 per token.
- Acme Co treats token purchases as resources; therefore, charges for token purchases include values for ResourceId, ResourceName, and ResourceType.
- Acme Co publicly lists their usage to token rates. These rates are as follows:

- 1 Q Widget Execution = 1 token
- 1 Z Widget Execution = 2 tokens
- 1 Workflow Operation = 3 tokens

8.5.3.2. Scenario A: Virtual Currency Not Offered at a Discount

For this scenario, contract terms include the following terms in addition to the baseline scenario mentioned above:

- Acme Corp offers no discount for purchased tokens.

8.5.3.3. Scenario A1: Purchase of Virtual Currency Without a Discount

For this scenario, the initial purchase of virtual currency is executed as follows:

- On April 1, 2025, AwesomeCorp agrees to purchase 100,000 tokens at \$2 per token for a total spend \$200,000. These tokens are only valid for 12 months.

[CSV Example](#)

Note the following details in the example dataset:

- The Charge Period is April 1st 2025 - April 1st 2026. The Billing Period is the month of April 2025 (when the tokens were purchased) and therefore will appear in the April invoice.
- Because Acme Co uses a virtual currency pricing model for usage and publishes their token price in terms of dollars and their usage cost in terms of tokens, their Cost and Usage [FOCUS dataset](#) includes the columns PricingCurrency, PricingCurrencyContractedUnitPrice, PricingCurrencyEffectiveCost, and PricingCurrencyListUnitPrice.
- A single charge representing the total payment for the initial token purchase agreement (\$200,000) is charged in the first invoice.
 - ListCost, BilledCost, and ContractedCost of the purchase are all represented in this charge, however EffectiveCost is zero since the tokens are not yet consumed.
- PricingQuantity is set to the total tokens purchased.
- Because Awesome Corp is paying the list price, ListUnitPrice and ContractedUnitPrice are all set to the same value of \$2.

8.5.3.4. Scenario A2: Usage of Virtual Currency Purchased Without a Discount

Awesome Corp uses Acme's services consuming tokens as follows in the first day:

- 245 executions of Q Widget
- 5 executions of Z Widget
- 120 operations of Workflow

[CSV Example](#)

Note the following details in the example dataset:

- The Charge Period is April 1st 2025 - April 2nd 2025. The Billing Period is the month of April 2025.
- PricingCurrency for these usage charges reflects the per usage token price of the particular usage.
- PricingQuantity reflects the amount of usage of the PricingUnit for each charge and is equivalent to ConsumedQuantity. While relevant to this example, there are scenarios including tiered pricing where ConsumedQuantity and PricingQuantity may not be the same.
- Because Awesome Corp's usage includes no discount on usage to token rates,

PricingCurrencyContractedUnitPrice and PricingCurrencyListUnitPrice are equivalent.

8.5.3.5. Scenario B: Virtual Currency Offered at a Discount

For this scenario, contract terms include the following terms in addition to the baseline scenario mentioned above:

- Acme Corp offers a discount for purchased tokens.

8.5.3.6. Scenario B1: Purchase of Virtual Currency at a Discount

For this scenario, the initial purchase of virtual currency is executed as follows:

- On April 1, 2025, AwesomeCorp agrees to purchase 100,000 tokens at discounted cost of \$1 per token for a total spend \$100,000. These tokens are only valid for 12 months.

[CSV Example](#)

Note the following details in the example dataset:

- The Charge Period is April 1st 2025 - April 1st 2026. The Billing Period is the month of April 2025 (when the tokens were purchased) and therefore will appear in the April invoice.
- Because Acme Co uses a virtual currency pricing model for usage and publishes their token price in terms of dollars and their usage cost in terms of tokens, their *FOCUS dataset* includes the columns PricingCurrency, PricingCurrencyContractedUnitPrice, PricingCurrencyEffectiveCost, and PricingCurrencyListUnitPrice.
- A single charge representing the total payment for the initial token purchase agreement (\$100,000) is charged in the first invoice.
 - ListCost, BilledCost, and ContractedCost of the purchase are all represented in this charge, however EffectiveCost is zero, as required for prepaid purchases.
- PricingQuantity is set to the total tokens purchased.
- Because Awesome Corp is receiving a discount on the token price, the ListUnitPrice is set to \$2 and the ContractedUnitPrice is set to \$1. A ListCost of (\$200,000) and ContractedCost (\$100,000) reflect the cost of the tokens at the list price and contracted price respectively. The BilledCost is set to \$100,000 since this is the amount that Awesome Corp will be charged for the purchase of tokens.

8.5.3.7. Scenario B2: Usage of Virtual Currency Purchased at a Discount

Awesome Corp uses Acme's services, consuming tokens as follows in the first day:

- 245 executions of Q Widget
- 5 executions of Z Widget
- 120 operations of Workflow

[CSV Example](#)

Note the following details in the example dataset:

- PricingQuantity reflects the amount of usage of the PricingUnit for each charge and is equivalent to ConsumedQuantity. While relevant to this example, there are scenarios including tiered pricing where ConsumedQuantity and PricingQuantity may not be the same.
- Because Awesome Corp's usage includes no discount on usage to token rates, PricingCurrencyContractedUnitPrice and PricingCurrencyListUnitPrice are equivalent.

8.5.3.8. Scenario B3: Usage of Virtual Currency at a Modified Rate

Awesome Corp uses Acme's services consuming tokens as follows in the first day:

- 245 executions of Q Widget
- 5 executions of Z Widget
- 120 operations of Workflow

Additionally, Acme Co offers a modified usage to token ratio for one of their services as follows:

- 1 Workflow Operation = 2 tokens

[CSV Example](#)

Note the following details in the example dataset:

- Because of the modified rate for Workflow Operations, the PricingCurrencyContractedUnitPrice and PricingCurrencyListUnitPrice are different for this charge. The ContractedUnitPrice is set to \$1 and the ListUnitPrice is set to \$2.
- The PricingCurrencyEffectiveCost is 240 tokens for this charge, which is less than example B2 above due to the modified rate.
- ListCost reflects the cost of the charge at both the list cost of the tokens and the list rate for which the usage consumes tokens.

8.5.3.9. Scenario C: Handling Virtual Currency Usage Overages

For this scenario, Awesome Corp has exceeded their purchased tokens on October 1st 2025 by 1,500 tokens and Acme Co has charged them for the overage. The following conditions apply:

- Acme Co has charged Awesome Corp for the cost of tokens at the list price of \$2 per token, and this purchase is effective from April 1st 2025 to the date of the purchase, October 1st 2025.
- Awesome Corp purchases an additional 25,000 tokens to facilitate usage to the end of their contract. These tokens are valid from October 1st 2025 to April 1st 2026.

[CSV Example](#)

Note the following details in the example dataset:

- This example focuses on the purchases records only for the overage and additional purchases. Neither usage charges nor earlier purchases are not included in this example.
- The Charge Period for the Overage Purchase is April 1st 2025 - October 1st 2025. This is because the overage charge is to cover the period of time the overage token purchase is applicable to.
- The Charge Period for the Additional Purchase is October 1st 2025 - April 1st 2026. This is because the additional purchase is to cover the period of time to which the additional token purchase is applicable. Because end dates are exclusive, ChargePeriodEnd is April 1st 2026.

8.6. Grouping Constructs for Resources or Services

Service Providers natively support various constructs for grouping [resources](#) or [services](#). These grouping constructs are often used to mimic organizational structures, technical architectures, cost attribution/allocation and access management boundaries, or other customer-specific structures based on requirements.

Service Providers may support multiple levels of resource or service grouping mechanisms. FOCUS supports two distinct levels of groupings that are commonly needed for FinOps capabilities like chargeback, invoice reconciliation and cost allocation.

- [Billing account](#): A mandatory container for [resources](#) or [services](#) that are billed together in an invoice. *Billing accounts* are commonly used for scenarios like grouping based on

organizational constructs, invoice reconciliation and cost allocation strategies.

- **Sub account:** An optional service-provider-supported construct for organizing *resources* and *services* connected to a *billing account*. *Sub accounts* are commonly used for scenarios like grouping based on organizational constructs, access management needs and cost allocation strategies. *Sub accounts* must be associated with a *billing account* as they do not receive invoices.

The table below highlights key properties of the two grouping constructs supported by FOCUS.

Property	Billing account	Sub account
Requirement level	Mandatory	Optional
Receives an invoice?	Yes	No
Invoiced at	Self	Associated billing account
Examples	AWS: Management Account* GCP: Billing Account Azure MCA: Billing Profile Snowflake: Organizational Account	AWS: Member Account GCP: Project Azure MCA: Subscription Snowflake: Account

* For organizations that have multiple AWS Member Accounts within an AWS Organization, consolidated billing is enabled by default and invoices are received at Management Account level. A Member Account can be removed from AWS consolidated billing whereby the removed account receives independent invoices and is responsible for payments.