Canonical Account Quote Model

Version 1.0.0 (meta rev 5)

Abstract

This document defines canonical JSON entity models for Account, Census, and Quote, along with references between models using JSON Schema 2020-12. It also provides a FHIR-aligned representation for Quote using standard resources for insurance quoting workflows.

Model-level key concepts (Vision2030 Canonical Models)
1) Canonical namespace & document identity

Canonical URI (stable, no version): the name of the model family.
https://schemas.example.com/models/Account

Versioned $id (immutable): the specific release of that model document.
https://schemas.example.com/models/Account/1.0.0

$anchor (export): a stable label for the root (or any sub-schema) others can import.
"$anchor": "Account"

Why: gives every model a permanent home, a frozen release URL, and a stable anchor to point at.

2) Dialect & vocabularies

$schema = JSON Schema Draft 2020-12 (core + validation vocabularies).

Use standard keywords only for validation/structure; add your own annotations separately (see “Model metadata”).

Why: ensures validators agree on behavior and formats (date-time, uuid, etc.).

3) Model metadata (annotations, not validation)

Keep a non-standard annotation block (ignored by validators) to carry governance info.

```
"model": {
  "canonical": "https://schemas.example.com/models/Account",
  "version": "1.0.0",         // semantic version (SemVer) of meaning/shape
  "metaVersion": 2,           // document revision counter (reprints)
  "status": "active",         // active | deprecated | withdrawn
  "dependsOn": {              // model-level dependencies (version ranges)
    "https://schemas.example.com/models/Census": "^1.0.0"
  },
  "digest": "sha256:…",       // optional content hash for provenance
  "changelog": [ { "rev": 2, "note": "Fixed typos in descriptions." } ]
}
```

Why: separates meaning (version) from reprint (metaVersion) and enables dependency checks and registry automation.

4) Semantic version vs. metaVersion

version (SemVer) = shape/meaning of the model.

MAJOR: breaking changes (rename/remove/requiredness changes).

MINOR: additive, backward-compatible (new optional fields, new enum values).

PATCH: annotation fixes only (no shape changes).

metaVersion = document reprint counter for the same semantic version (typos, links, formatting).

Rule of thumb: change version when a validator’s verdict could change; else bump metaVersion.

5) Model-level referencing ($ref to other models)

Models may import the structure of other models via absolute $ref to a versioned $id + #anchor.

```
"account": { "$ref": "https://schemas.example.com/models/Account/1.0.0#Account" },
"census":  { "$ref": "https://schemas.example.com/models/Census/1.0.0#Census" }
```

Policy:

In source control, pin $refs to a specific $id.

In build/CI, resolve model.dependsOn ranges (e.g., ^1.0.0) to pinned $ids and fail on incompatible MAJORs.

6) Composition & reuse

Use $defs for internal reusable shapes; export selected ones with $anchor.

Compose with allOf / oneOf / anyOf when building layered models.

Add unevaluatedProperties: false at the final level when composing to keep a closed shape.

7) Closed content (catch typos)

On every object, set additionalProperties: false.

When using composition, close with unevaluatedProperties: false to block stray fields introduced via allOf.

Why: prevents accidental keys from silently passing validation.

8) Property design (schema-level)

Naming: lowerCamelCase; clear, unabbreviated names.

Descriptions: every property has a description (annotation).

Constraints: prefer explicit minLength, maxLength, pattern, minimum, maximum, enum, format.

Nullability: model explicitly (e.g., type: ["string","null"]) if allowed.

Note: title, description, examples, $comment, deprecated are annotations—they don’t affect validity.

9) Compatibility contract

What’s allowed without a MAJOR bump:

Add an optional property.

Add enum members (if consumers are written to ignore unknown values).

Loosen constraints (e.g., allow a longer maxLength).

Improve annotations (description, examples, $comment).

What requires a MAJOR bump:

Remove/rename a property, change required, tighten constraints, change types/formats, repurpose semantics.

10) Deprecation signals

Mark fields as "deprecated": true (annotation).

Keep them unchanged until the next MAJOR; document migration in model.changelog.

JSON schema Group {

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "https://schemas.example.com/models/Group-SF/1.0.0",
  "$anchor": "GroupSF",
  "title": "Group (Salesforce-aligned, Stepwise-ready)",
  "type": "object",
  "additionalProperties": false,
  "unevaluatedProperties": false,
  "properties": {
    "sfAccountId": { "type": "string", "pattern": "^[A-Za-z0-9]{15,18}$", "description": "Salesforce Account Id." },
    "Name": { "type": "string", "minLength": 1, "maxLength": 255, "description": "Account.Name" },
    "Group_Number__c": { "type": "string", "pattern": "^[A-Za-z0-9\\-]{1,30}$" },
    "ShippingStreet": { "type": "string", "maxLength": 255 },
    "ShippingStreetLines": {
      "type": "array",
      "items": { "type": "string", "minLength": 1, "maxLength": 200 },
      "minItems": 1,
      "maxItems": 3,
      "description": "Normalized split of ShippingStreet into lines."
    },
    "ShippingCity": { "type": "string", "maxLength": 40 },
    "ShippingState": { "type": "string", "maxLength": 80 },
    "Shipping_County__c": { "type": "string", "maxLength": 80 },
    "ShippingPostalCode": { "type": "string", "maxLength": 20 },
    "Sic": { "type": "string", "pattern": "^[0-9]{4}$" },
    "Value_Based_Programs__c": { "type": "boolean" },
    "Identity_Protection_ASO__c": { "type": "boolean" },
    "OON_Saving_Fee__c": { "type": "number", "minimum": 0 },
    "OON_Percentage__c": { "type": "number", "minimum": 0, "maximum": 1 },
    "OON_Per_Claim_Cap__c": { "type": "number", "minimum": 0 },
    "Payment_Integrity_Participation__c": { "type": "boolean" },
    "Payment_Integrity_Savings_Retained__c": { "type": "number", "minimum": 0, "maximum": 1 },
    "Repooled_to_Experience__c": { "type": "boolean" },
    "Class_of_Business__c": { "type": "string", "maxLength": 50, "pattern": "^[A-Z0-9_.\\- ]{1,50}$" },
    "Rating_Type__c": { "type": "string", "maxLength": 50, "pattern": "^[A-Z0-9_.\\- ]{1,50}$" },
    "Payment_Integrity_Effective_Date__c": { "type": "string", "format": "date" },
    "MSK_Participation_Code__c": { "type": "string", "pattern": "^[A-Z0-9\\-]{1,30}$" },
    "MSK_Participation_Date__c": { "type": "string", "format": "date" },
    "Code_Advisor_Participation_Code__c": { "type": "string", "pattern": "^[A-Z0-9\\-]{1,30}$" },
    "Code_Advisor_Participation_Percent__c": { "type": "number", "minimum": 0, "maximum": 1 },
    "Code_Advisor_Effective_Date__c": { "type": "string", "format": "date" },
    "Gaps_in_Care_Participation_Code__c": { "type": "string", "pattern": "^[A-Z0-9\\-]{1,30}$" },
    "Gaps_in_Care_Participation_Date__c": { "type": "string", "format": "date" },
    "Notes__c": { "type": ["string","null"], "maxLength": 4000 }
  },
  "required": ["sfAccountId", "Name"],
  "allOf": [
    {
      "if": { "properties": { "Repooled_to_Experience__c": { "const": true } }, "required": ["Repooled_to_Experience__c"] },
      "then": { "required": ["Class_of_Business__c", "Rating_Type__c"] }
    }
  ]
}
```

Address note: Salesforce has a single multi-line ShippingStreet. If Stepwise needs separate lines, we expose both: raw ShippingStreet (string) and normalized ShippingStreetLines (array, 1..3). Use whichever your pipeline expects.

Scenario overlays: different constrainss{

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "https://schemas.example.com/models/Group-SF-QuoteProfile/1.0.0",
  "title": "Group (SF Quote Profile)",
  "allOf": [
    { "$ref": "https://schemas.example.com/models/Group-SF/1.0.0#GroupSF" },
    {
      "type": "object",
      "required": ["sfAccountId","Name"],
      "anyOf": [
        { "required": ["ShippingStreet","ShippingCity","ShippingState","ShippingPostalCode"] },
        { "required": ["ShippingStreetLines","ShippingCity","ShippingState","ShippingPostalCode"] }
      ],
      "properties": {
        "OON_Percentage__c": { "type": "number", "minimum": 0, "maximum": 1 },
        "Payment_Integrity_Participation__c": { "type": "boolean" }
      }
    }
  ]
}
```

Quote cardinality (delta):

Address present (1..1 via either ShippingStreet* or ShippingStreetLines path).

Program fields optional, but if provided must meet constraints (percentages 0..1).

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "https://schemas.example.com/models/Group-SF-RenewalProfile/1.0.0",
  "title": "Group (SF Renewal Profile)",
  "allOf": [
    { "$ref": "https://schemas.example.com/models/Group-SF/1.0.0#GroupSF" },
    {
      "type": "object",
      "required": [
        "sfAccountId",
        "Name",
        "Group_Number__c",
        "ShippingStreet",
        "ShippingCity",
        "ShippingState",
        "ShippingPostalCode"
      ],
      "properties": {
        "Payment_Integrity_Participation__c": { "type": "boolean" },
        "Payment_Integrity_Effective_Date__c": { "type": "string", "format": "date" }
      }
    }
  ]
}
```

Renewal cardinality (delta):

Group_Number__c: 1..1 required.

Shipping address: 1..1 required (raw fields; if you prefer the split lines, swap to ShippingStreetLines in the required list).

Payment Integrity effective date required only if participation is true (add an if/then if you want that enforced at validation time).

Example conditional add-on:

```
{
  "if": { "properties": { "Payment_Integrity_Participation__c": { "const": true } }, "required":
["Payment_Integrity_Participation__c"] },
  "then": { "required": ["Payment_Integrity_Effective_Date__c"] }
}
```

4) Quick sanity checks (so Stepwise won't choke)

Percentages: store as 0..1 (divide by 100 on ingest).

Codes: 1..30 chars, uppercase letters/digits/dash (^[A-Z0-9\-]{1,30}$).

Caps/fees: non-negative numbers; decide whether you want integers (minor units) or decimals as strings; above uses JSON numbers.

Address lines: if Stepwise truly needs indexed lines [0..2], keep ShippingStreetLines with minItems:1,maxItems:3.

Closed content: additionalProperties:false catches typos.

Label (from sheet)	Proposed API name	Type / constraints	Cardinality	Notes
Account Name	`Name`	string, 1..255	**1..1**	SF Account.Name (group display name)
Group Number	`Group_Number__c`	string `^[A-Za-z0-9\-]{1,30}$`	0..1	Required at **Renewal** (see profile)
ShippingStreet\[0]	`ShippingStreetLines[0]`	string, 1..200	**1..1** if lines used	First line; see note on SF `ShippingStreet`
ShippingStreet\[1]	`ShippingStreetLines[1]`	string, 1..200	0..1	Optional line 2
ShippingStreet\[2]	`ShippingStreetLines[2]`	string, 1..200	0..1	Optional line 3
ShippingCity	`ShippingCity`	string, ≤40	0..1	
ShippingState	`ShippingState`	string, ≤80	0..1	
Shipping County	`Shipping_County__c`	string, ≤80	0..1	custom
ShippingPostalCode	`ShippingPostalCode`	string, ≤20	0..1	
SIC Code	`Sic`	string `^[0-9]{4}$`	0..1	4-digit SIC
Value Based Programs	`Value_Based_Programs__c`	boolean	0..1	yes/no
Identity Protection (ASO)	`Identity_Protection_ASO__c`	boolean	0..1	yes/no
OON Saving Fee	`OON_Saving_Fee__c`	number ≥0	0..1	currency or fee amount
OON % (÷100)	`OON_Percentage__c`	number 0..1	0..1	store as **0..1** (e.g., 0.25 = 25%)
OON \$ Per Claim Cap	`OON_Per_Claim_Cap__c`	number ≥0	0..1	currency cap
Payment Integrity – Participation	`Payment_Integrity_Participation__c`	boolean	0..1	yes/no
Payment Integrity – % Savings Retained (÷100)	`Payment_Integrity_Savings_Retained__c`	number 0..1	0..1	store 0..1
Repooled to Experience?	`Repooled_to_Experience__c`	boolean	0..1	if **true**, require Class & Rating Type
Class of Business	`Class_of_Business__c`	string `^[A-Z0-9_\- ]{1,50}$`	0..1	required if repoled=true
Rating Type	`Rating_Type__c`	string `^[A-Z0-9_\- ]{1,50}$`	0..1	required if repoled=true
Payment Integrity – Effective Date	`Payment_Integrity_Effective_Date__c`	date (YYYY-MM-DD)	0..1	
MSK Participation Code	`MSK_Participation_Code__c`	string `^[A-Z0-9\-]{1,30}$`	0..1	“W-007555”, etc.
MSK Participation Date	`MSK_Participation_Date__c`	date	0..1	
Code Advisor Participation Code	`Code_Advisor_Participation_Code__c`	string `^[A-Z0-9\-]{1,30}$`	0..1	“W-008025–W-0080272”
Code Advisor % (÷100)	`Code_Advisor_Participation_Percent__c`	number 0..1	0..1	0..1
Code Advisor Effective Date	`Code_Advisor_Effective_Date__c`	date	0..1	
Gaps in Care Participation Code	`Gaps_in_Care_Participation_Code__c`	string `^[A-Z0-9\-]{1,30}$`	0..1	
Gaps in Care Participation Date	`Gaps_in_Care_Participation_Date__c`	date	0..1	

1.1 Introduction

The canonical model organizes business data into three entity models: Account (the customer organization), Census (population snapshot used for underwriting), and Quote (pricing proposal derived from the Account and Census).

Each model is published as an independent JSON Schema with its own semantic version and metaVersion. Quote references Account and Census by model-level $ref, enabling consistent validation of embedded sub-objects across services and databases.

Figure 1 shows the high-level UML structure and relationships.

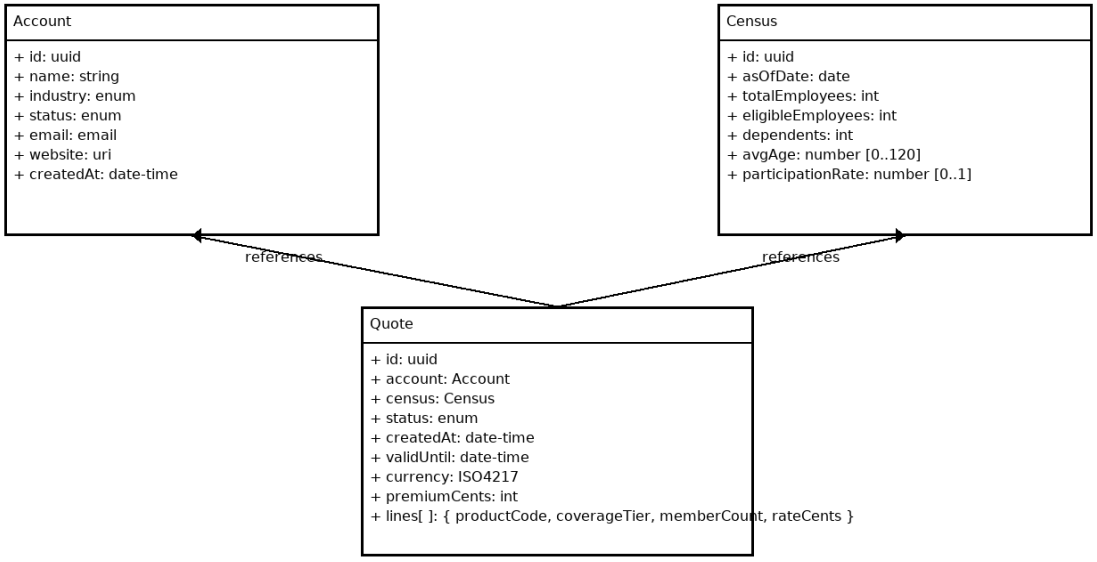

Figure 1 — UML-style structure: Quote references Account and Census.

1.2 Key Concepts

1. **Model-level references:** Use $ref to import the structure of another entity model at design time (not instance IDs).
2. **Semantic version:** SemVer string that signals meaning/shape changes (e.g., 1.2.0).
3. **Meta version:** Document revision counter for republishing without meaning change.
4. **Closed content:** Use additionalProperties:false (and when composed, unevaluatedProperties:false) to prevent typos.
5. **Reference options at instance level:** foreign keys, link objects, or embedded snapshots. (Out of scope here; this doc focuses on model-level typing.)
6. **Validation:** All schemas target JSON Schema Draft 2020-12 and use standard format, enum, and numeric constraints.

1.3 JSON Entity Modeling Best Practices

Naming: lowerCamelCase for properties; avoid abbreviations; positive booleans (isActive).

Identity: Use opaque IDs (UUID/ULID). Don’t encode semantics in identifiers.

Dates & money: ISO-8601 UTC date-time; currency in minor units (integer) or decimal string; avoid binary float for currency.

Nullability: Document semantics: missing = unknown; null = intentionally empty; model nullable with type: [T, "null"].

Versioning: Pin $id with semantic version; bump *metaVersion* on non-functional republishes; keep old IDs resolvable.

Documentation: Use description annotations; ship minimal and maximal examples; maintain a changelog.

1.4 Quote Model Structure — JSON

The Quote schema references Account and Census models by their versioned $id and $anchor. Business rules and field constraints are captured with standard JSON Schema keywords.

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "https://schemas.example.com/models/Quote/1.0.0",
  "$anchor": "Quote",
  "title": "Quote",
  "type": "object",
  "additionalProperties": false,
  "model": {
    "canonical": "https://schemas.example.com/models/Quote",
    "version": "1.0.0",
    "metaVersion": 5,
    "status": "active",
    "dependsOn": {
      "https://schemas.example.com/models/Account": "^1.0.0",
      "https://schemas.example.com/models/Census": "^1.0.0"
    }
  },
  "properties": {
    "id": {
      "type": "string",
      "format": "uuid",
      "description": "Quote identifier (opaque UUID)."
    },
    "account": {
      "$ref": "https://schemas.example.com/models/Account/1.0.0#Account",
      "description": "The Account this quote is for (typed by the Account model)."
    },
    "census": {
      "$ref": "https://schemas.example.com/models/Census/1.0.0#Census",
      "description": "Census data used to produce this quote (typed by the Census model)."
    },
    "status": {
      "type": "string",
      "enum": [
        "draft",
        "proposed",
        "accepted",
        "rejected",
        "expired"
      ],
      "description": "Quote lifecycle status."
    },
    "createdAt": {
      "type": "string",
      "format": "date-time",
      "description": "UTC creation timestamp."
    },
    "validUntil": {
      "type": "string",
      "format": "date-time",
      "description": "UTC expiration timestamp for this quote."
    },
    "currency": {
      "type": "string",
      "pattern": "^[A-Z]{3}$",
      "description": "ISO 4217 currency (e.g., 'USD')."
    },
    "premiumCents": {
      "type": "integer",
      "minimum": 0,
      "description": "Total premium in minor units (e.g., cents)."
    },
    "lines": {
      "type": "array",
      "minItems": 1,
      "description": "Line items comprising the quote.",
      "items": {
        "type": "object",
        "additionalProperties": false,
        "properties": {
```

```
            "productCode": {
              "type": "string",
              "maxLength": 50,
              "pattern": "^[A-Z0-9_.-]{1,50}$",
              "description": "Product/plan identifier (system-specific)."
            },
            "coverageTier": {
              "type": "string",
              "enum": [
                "EE",
                "ES",
                "EC",
                "EF"
              ],
              "description": "Coverage tier: Employee, Employee+Spouse, Employee+Children, Employee+Family."
            },
            "memberCount": {
              "type": "integer",
              "minimum": 1,
              "description": "Members covered by this line."
            },
            "rateCents": {
              "type": "integer",
              "minimum": 0,
              "description": "Rate per member or unit, in minor units."
            }
          },
          "required": [
            "productCode",
            "coverageTier",
            "memberCount",
            "rateCents"
          ]
        }
      },
      "notes": {
        "type": [
          "string",
          "null"
        ],
        "maxLength": 2000,
        "description": "Underwriting comments or caveats."
      }
    },
    "required": [
      "id",
      "account",
      "census",
      "status",
      "createdAt",
      "currency",
      "premiumCents",
      "lines"
    ]
  }
```

Business Rules (informative):

- validUntil SHALL be >= createdAt (enforced in service logic).
- status transitions: draft → proposed → accepted I rejected; expired is terminal.
- currency SHALL be a valid ISO 4217 code (pattern enforces 3 letters; service validates membership).
- premiumCents equals the sum of lines.memberCount * lines.rateCents (computed rule).

1.5 Quote Model Structure — FHIR-aligned

A price quote can be represented in FHIR using the Claim resource with use='predetermination'. The adjudicated totals may be returned as a ClaimResponse. The customer account maps to Organization, and the census (population snapshot) maps to Group (type=person, actual=true).

Alternative/adjacent flow: CoverageEligibilityRequest/Response can be used to check eligibility and plan-specific benefits; however, for full pricing with line items, Claim/ClaimResponse is the canonical fit.

FHIR Bundle (Quote as Predetermination)

```
Bundle.type = 'collection'
  - Organization (Account)
  - Group (Census)
  - Claim (use='predetermination')
      .provider -> Organization
      .item[n].productOrService -> CodeableConcept (plan/benefit code)
      .item[n].quantity.value   -> memberCount
      .item[n].unitPrice        -> rate (Money)
  - ClaimResponse (optional result, totals)
```

Notes:

- Group captures the employee population; characteristics or contained members may be used depending on privacy and size.
- Organization identifies the quoting customer (your Account).
- Claim.item lines map naturally from Quote.lines; totals belong in ClaimResponse.
- Use standard terminologies where available; internal product codes may be profiled as CodeSystems.

1.5 Glossary

Semantic version	SemVer string indicating meaning/shape changes of the model (e.g., 1.2.0).
Meta version	Document revision counter for republishing without meaning change.
$id	Canonical, version-pinned URI of a schema, used as the base for $ref resolution.
$anchor	Local anchor name to reference a particular schema location (e.g., the root).
$ref	Reference to another schema location (can be absolute URL + #anchor).
additionalProperties:false	Disallow unspecified properties to catch typos and ensure closed content.
FHIR Claim (predetermination)	A 'what-if' claim used for quoting or cost estimation.
FHIR Group	Collection of persons used to represent a census/population.
FHIR Organization	Entity representing an account/customer organization.

1.6 References

- JSON Schema Draft 2020-12 — Validation: https://json-schema.org/draft/2020-12/json-schema-validation
- JSON (RFC 8259): https://www.rfc-editor.org/info/rfc8259
- JSON Pointer (RFC 6901): https://www.rfc-editor.org/info/rfc6901
- FHIR Claim (use: predetermination): https://build.fhir.org/claim.html
- FHIR ValueSet: claim-use: https://build.fhir.org/valueset-claim-use.html
- FHIR Group: https://build.fhir.org/group.html
- FHIR Organization: https://build.fhir.org/organization.html
- CoverageEligibilityRequest: https://build.fhir.org/coverageeligibilityrequest.html