JSON SCHEMAの命名規則と設計規則の技術仕様

Mar. 22, 2023

観光技術検討会 リード
板垣和芳

1. JSON Schemaの特徴

- 報告書名称：JSON Schema Naming and Design Rules V1.0 2022-09-13

JSON Schemaの特徴

3.7.2 ドキュメントベースおよびリソースベースの情報をサポートする JSON Schemaでの ASBIE 表現

- CCTS (Core Component Techbnical Specification) は、従来の EDI (Electronic Data Interchange) メッセージを標準化およびモデル化する目的で発明されました。 現在でも、特に B2B および B2A 環境では、ドキュメントベースのデータ交換が依然として主流です。
- JSON SchemaがサポートするREST API はビジネス文書の交換ではなく、リソースの管理に基づいているという事実によって特徴付けられます。 これは、たとえば、取引先情報を請求書や輸送オーダーなどの取引データとは別に管理できることを意味します。 CCTS では、これらはすべて、ABIE が ASBIE の形式で相互に関連付けられる場所です。
- JSON Schema アーティファクトを介して REST API をサポートすることを目的として、まさにこの時点で、ドキュメント中心のデータ交換からリソース中心のデータ交換に切り替えるオプションをサポートする必要があります。
- リソースベースのデータ管理とは、リソースが一意の識別子を持つ必要があることを意味します。 したがって、それらの ABIE のみを一意の識別子を持つリソースに変換できます。 URI として表されるこの一意の識別子を使用すると、注文の購入者に関する情報を取得し、URI に続いて購入者の当事者情報を取得できます。

CCTSからJSON Schemaへの変換における留意事項

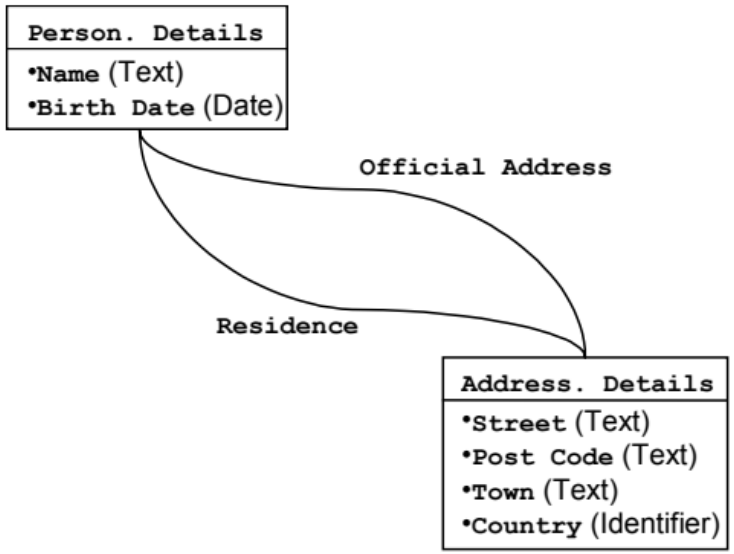
- 3.7.2に記載されているように、このNDRではCCTSからJSON Schemaへの変換方法が主題となっています。
- したがって、以下のような問題が生じます。（3.1.1 RESTful コンテキストでの JSONのシリアル化　参照）
- CCTS は、リソース ベースのモデリング アプローチと同様にドキュメント中心のモデリング アプローチをサポートする階層データ モデル (hierarchical data model) の命名規則と設計規則を定義します。 ドキュメント中心のモデリング アプローチをサポートし、下位互換性を保つために、それは階層構造で設計されています。
- 一方、REST API はリソースベースのみです。 これは、JSON Schemaを使用して CCTS から REST API に移行する場合、両方のオプションを考慮する必要があることを意味します。
- さらに、JSON 構文には、CCTS の命名規則および設計規則とは異なる、独自の命名規則および設計規則があります。 このセクションでは、CCTS から JSON Schemaに移行する方法について詳しく説明します。

RESTful コンテキストでの JSONのシリアル化

- REST API 仕様で JSON Schema アーティファクトを使用するには、階層構造がどのレベルでリソースベースの構造に分割されるかという問題が生じます。
- UN/CEFACT プロジェクト API Town Plan は、すでにこの根本的な問題に取り組んでいます。 それは、決定が事前に中央で下すことはできないと定式化しました。 むしろ、それはそれぞれの具体的なプロジェクトまたは具体的なドメインにおける実装のニーズによって異なります。
- このため、JSON Schema NDR 内では、ドキュメント中心の階層を保持するか、リソースに応じて分離するかという両方のオプションを各決定ポイントに許可するシリアル化の形式が選択されます。
- すべての ASBIE (Associated Business Information Entity; 連結BIE) 接続がこの影響を受けます。
- 対応するデータ型は 3.7 章 (3.7 JSON Schemaでの ABIE および BBIE 表現) でモデル化されています。

CCTS と JSON Schema の構造比較

[Example] *Association Core Component*



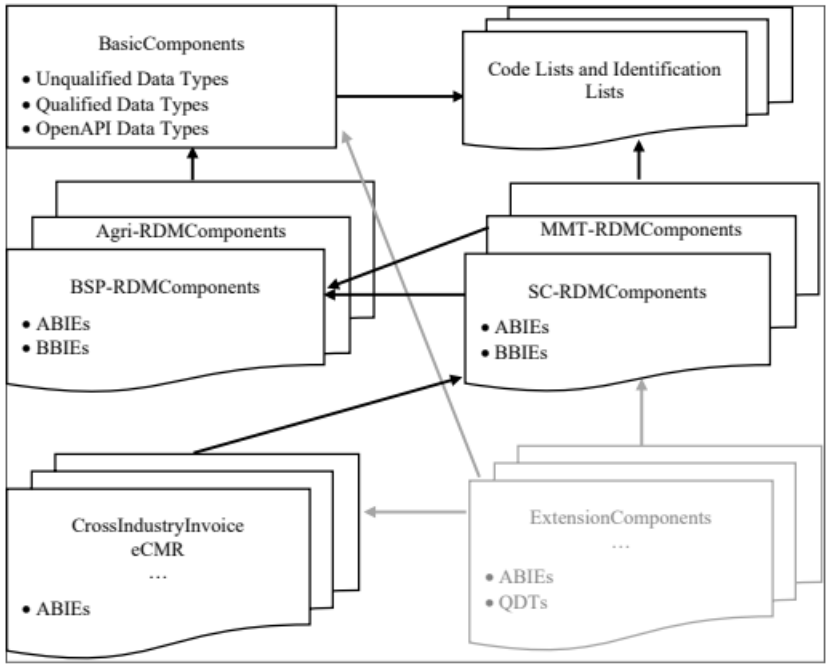


Figure 1 – JSON schema landscape

3.7 JSON Schemaでの ABIE および BBIE 表現

- [R 41|1]
- 各 ABIE は JSON サブSchemaで表現されるものとします (SHALL)。 以前のバージョンで非推奨 (deprecated) としてマークされた ABIE は、JSON サブSchemaで表現してはなりません (SHALL NOT)。
- [注記] たとえば、ABIE はバージョン D20B 以降で非推奨になるように定義されています。 バージョン D21A の JSON Schema アーティファクトがエクスポートされる場合、ABIE はこのエクスポートで表されてはなりません (SHALL NOT)。
- [R 42|1] extensibleType への参照
- JSON サブSchema内のすべての ABIE 表現には、extensibleType への参照が含まれているものとします (SHALL)
- [Example]

```
"abieType": {
    "title": "The Dictionary Entr y Name",
    "description": "The description",
    "type": "object",
    "properties": {
      "p1": { "type": "string" }
    },
    "required": ["p1"],
```

3.7.1 ABIE および BBIE の一般的な取り扱い

```
    "$ref": "UNECE-BasicComponents.json#/$defs/extensibleType",
    "unevaluatedProperties": false
  }
}
[Example of a valid JSON object]
{
    "p1": "value",
    "x-addedStringProperty": "added value",
    "x-addedObjectProperty": { "content": "a123"}
}
[Example of an invalid JSON object]
{
    "p1": "value",
    "addedStringProperty": "added value"
}
```

- [R 43|2]
- 拡張プロパティ名は、この技術仕様で定義されているのと同じ命名規則に従う必要があります (SHOULD)。

3.7.2 ドキュメントベースおよびリソースベースの情報をサポートする JSON Schema での ASBIE 表現

- [R 44|1]
- BasicComponents は、リソースベースのデータ交換のための JSON サブSchemaを次のように定義するものとします。

```
"$defs": {
  "resourceType": {
    "type": "string",
    "format": "uri"
  }
}
```

- [R 45|1]
- ABIE に識別子が含まれるすべての ASBIE は、resourceType と関連する ABIE の間の oneOf の選択肢を使用してモデル化されるものとします (SHALL)。 他のすべての ASBIE は直接参照されるものとします (SHALL)。
- どちらの場合も、定義されたカーディナリティを遵守する必要があります(SHALL)。
- 次の例では、注目を保つため、タイトル、説明などは表示されていません。

3.7.2 ドキュメントベースおよびリソースベースの情報をサポートする JSON Schema での ASBIE 表現

[Example]

```
"$defs": {
  "invoiceType": {
    "type": "object",
    "properties": {
      "buyer": {
        "oneOf": [
          {"$ref": "UNECE-BasicComponents.json#/$defs/resourceType" },
          {"$ref": "#/$defs/partyType" }
        ]
      }
    },
    "required": [ "buyer" ],
    "$ref": "UNECE-BasicComponents.json#/$defs/extensibleType",
    "unevaluatedProperties": false
  },
```

3.7.2 ドキュメントベースおよびリソースベースの情報をサポートするJSON Schema での ASBIE 表現

```
"partyType": {
   "type": "object",
   "properties": {
     "id": {
       "type": "array",
       "items": {
         "$ref":"UNECE-BasicComponents.json#/$defs/udt/identifierType"
       }
     },
     "name": { "type": "string" },
     "postalTradeAddress": { "$ref": "#/$defs/addressType" }
   },
   "$ref": "UNECE-BasicComponents.json#/$defs/extensibleType",
   "unevaluatedProperties": false
 },
```

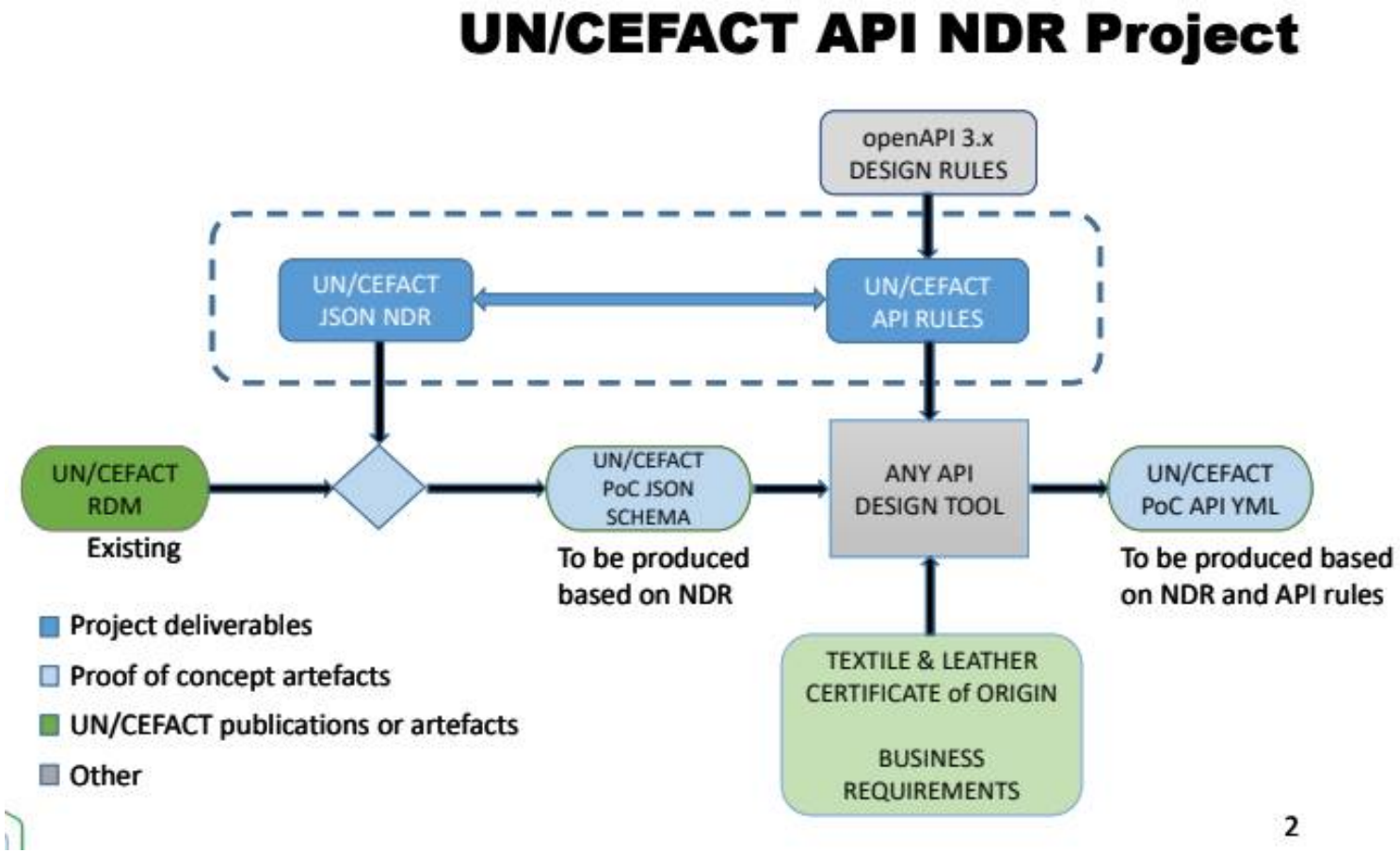
3.7.2 ドキュメントベースおよびリソースベースの情報をサポートするJSON Schema での ASBIE 表現

```
 "addressType": {
    "type": "object",
    "properties": {
      "street": { "type": "string"},
      "city": { "type": "string"},
      "postalCode": { "type": "string"},
      "countryCode": { "$ref": "UNECE-BasicComponents.json#/$defs/qdt/countryIdType"}
    },
    "$ref": "UNECE-BasicComponents.json#/$defs/extensibleType",
    "unevaluatedProperties": false
  }
}
```

摘要と目的

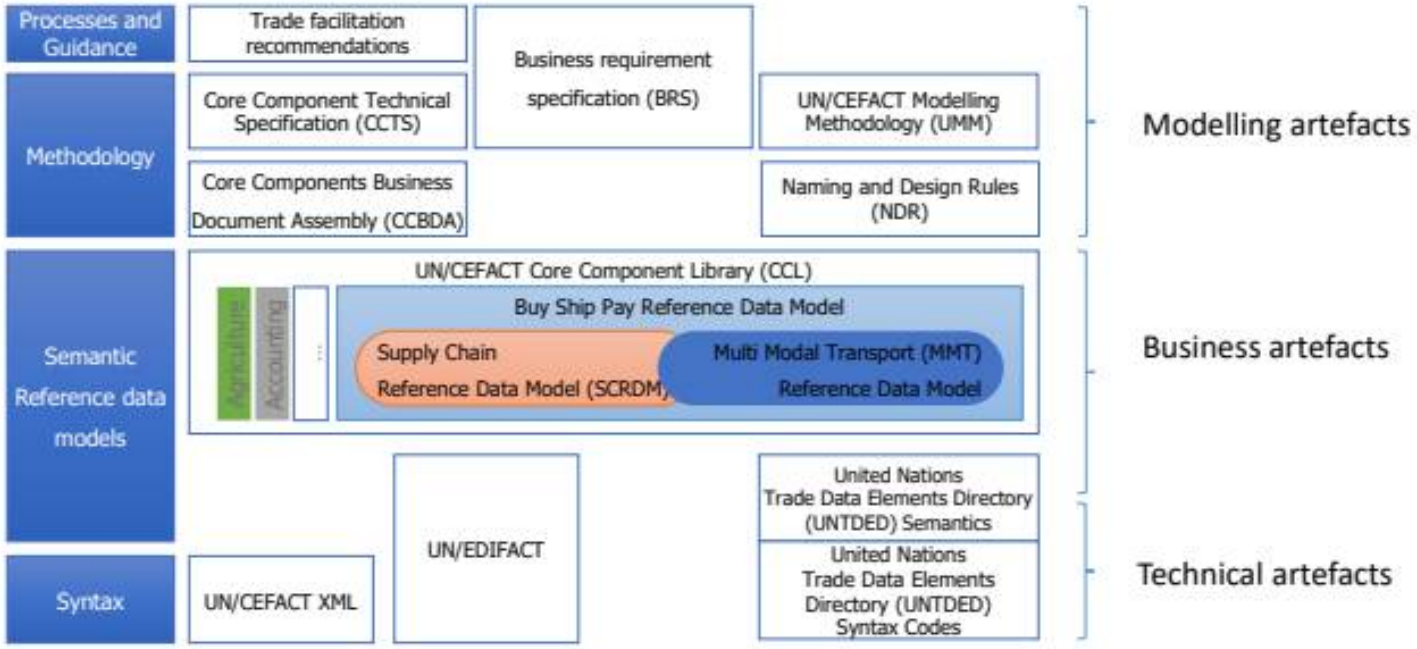
- この JSON Schemaの命名規則と設計規則の技術仕様では、JSON を定義、記述、使用して、つまり REST API を介してビジネス情報交換を一貫して表現するために必要なアーキテクチャと一連の規則を定義します。
- これは、JSON Schema チームの仕様と UN/CEFACT コア コンポーネント技術仕様(CCTS)に基づいています。
- この仕様は、UN/CEFACT によって JSON Schemaおよび JSON Schema文書を定義するために使用され、UN/CEFACT 標準として発行されます。 また、業界間および業界内の相互運用性を最大限に高めることに関心のある他の組織でも使用される予定です。
- この JSON Schema NDR 技術仕様ドキュメントは、最新の Web 開発者が UN/CEFACT セマンティクスを利用できるようにサポートすることを目的とした一連のドキュメントの一部を形成します。
- これを UN/CEFACT 参照データ モデル (UN/CEFACT Reference Data Models) の任意のレイヤーに適用して、UN/CEFACT コア コンポーネント技術仕様 (UN/CEFACT CCTS) バージョン2.01 に従って 適合する JSON 仕様を作成できます。
- これには、Supply-Chain-Reference-Data-Model (SCR-DM), Multi-Modal-Transport-Reference-Data-Model (MMT-RDM)などのコンテキスト化と同様なBuy-Ship-Pay、Accountingなどの包括的な RDM から Road Consignment Note (eCMR) や原産地証明書 (COO; certificate of origin) などの単一メッセージの実装までが含まれます。

JSON Schema NDR and OpenAPI 関連資料 From Lunch & Learn 8 May 2023



JSON Schema NDR and OpenAPI 関連資料 From Lunch & Learn 8 May 2023

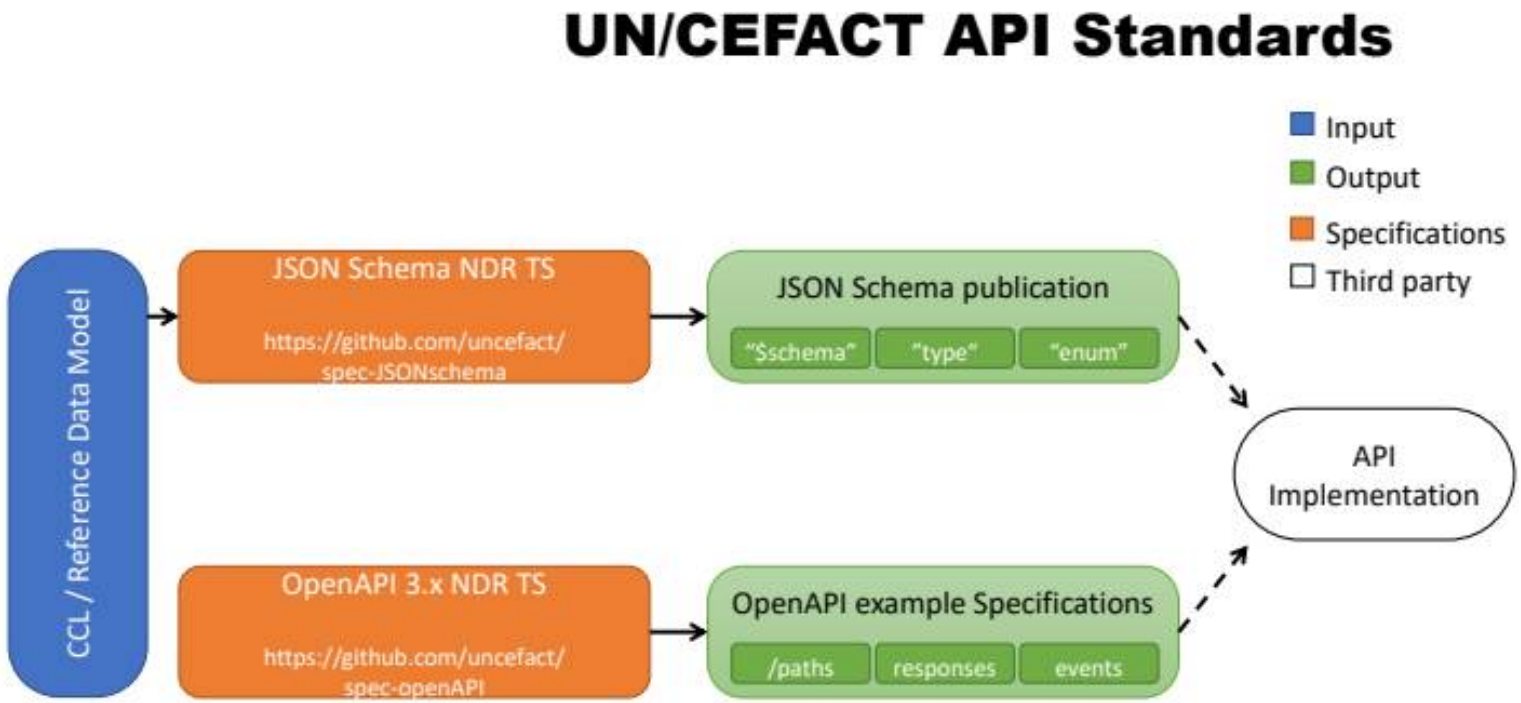
UN/CEFACT Modelling Framework



JSON Schema NDR and OpenAPI 関連資料 From Lunch & Learn 8 May 2023

From generic to concrete: Layers of profiles / subsets

JSON Schema NDR and OpenAPI 関連資料 From Lunch & Learn 8 May 2023



JSON Schema NDR and OpenAPI 関連資料 From Lunch & Learn 8 May 2023

Using a Business Requirement Specification

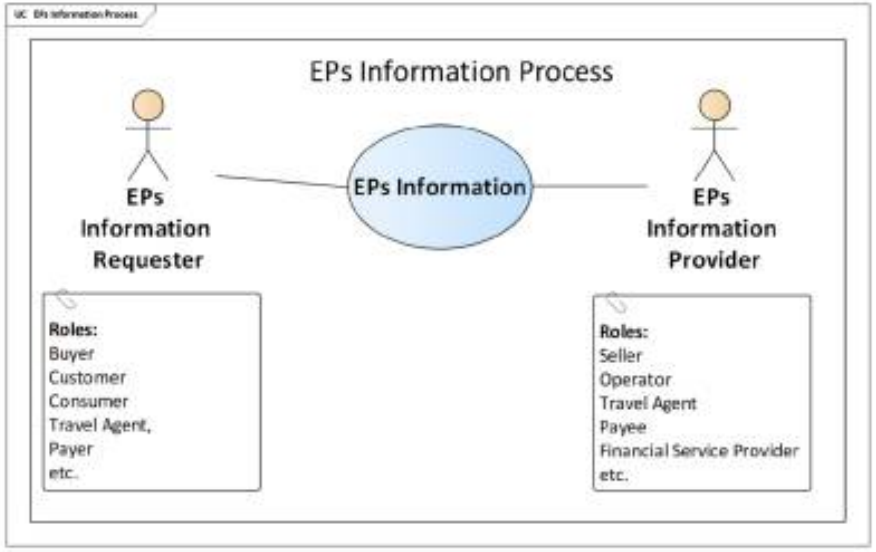


Figure 2-1 Use case diagram - EPs Information Exchange

BRS
- Defines the processes and requirements

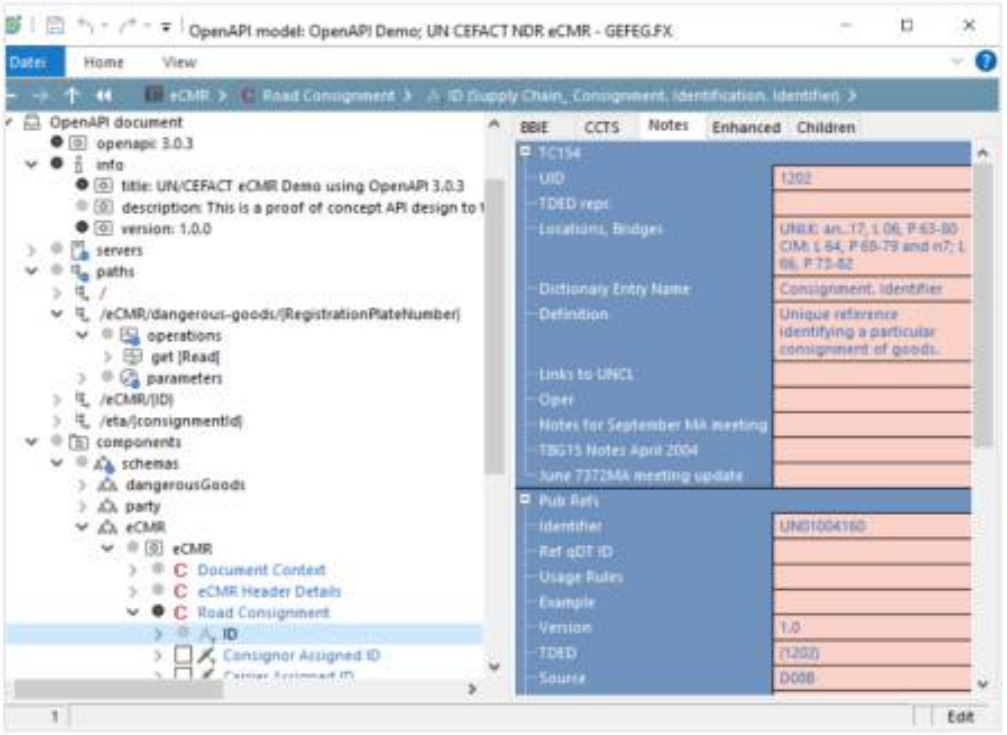
JSON Schema
- Contextualise RDM

OpenAPI
- Use JSON-Schema
- Create services for process implementation

JSON Schema NDR and OpenAPI 関連資料 From Lunch & Learn 8 May 2023

Re-Use the wisdom of decades

- Import or re-use your own or your industries
  - XSD specification
  - JSON schema
  - JSON-LD
- Compliant to the UN/CEFACT standards
- Tick-off what you do not need



JSON Schema NDR and OpenAPI 関連資料 From Lunch & Learn 8 May 2023

From Model to Specification including harmonised definitions an code lists

```yaml
openapi: 3.0.3
info:
  title: UN/CEFACT eCMR Demo using OpenAPI 3.0.3
  description: |
  version: 1.0.0
servers:
paths:
  /:
  /eCMR/dangerous-goods/{RegistrationPlateNumber}:
    get:
      tags:
      description: Gets a list of dangeours goods on a
      transport means.
      operationId: getDangeoursGoodsInfo
      parameters:
      responses:
        200:
          description: OK
          headers:
            Link:
              description: |
                Link header providing information on
                pagination.

                Examples:

                Link:
                <https://api.unece.org/demo?cursor=A7HFJ98M>;
                 rel="current"
```

```yaml
DocumentCode:
  description: Code specifying the name of a
  document such as 352 for Proforma invoice,
  380 for Commercial invoice.
  type: object
  properties:
    content:
      description: |
        Applicable codes:
        * '730' - Road consignment note
      enum:
        - '730'  # Road consignment note
      type: string
    listAgencyID:
      enum:
```

JSON Schema NDR and OpenAPI 関連資料 From Lunch & Learn 8 May 2023

Use JSON Schema to create API Specs with the tool of your choice

3.1.2 全体的な JSON Schema構造

- [R 2|1]
- この仕様の範囲において、JSON Schemaは、https://json-schema.org で定義されている JSON Schema定義に準拠するファイルです。
- $defs section で定義されたサブSchemaが含まれる場合があります。
- JSON Schema フラグメントは、JSON Schema全体とそれに含まれる各サブSchemaの両方を意味します。
- [R 3|1]
- 各 JSONSchemaは、https://json-schema.org/draft/2020-12/schema として定義された適切な $schema 文字列プロパティを持つ「JSON Draft 2020-12 schema3」として宣言されるものとします (SHALL)。
- [R 4|2]
- セクション 3.8.1 では、JSON Schema バージョン 2
- 020-12 をまだサポートしていない多くのツールとの互換性を実現できる一連の規則が定義されています。この規則のセットは出版物に適用できます (MAY)。
- または、結果のSchemaを「非推奨の互換性セット」としてマークされた JSON Schemaの 2 番目のセットとして公開することもできます。

3.1.2 全体的な JSON Schema構造

- [R 5|1]
- 各 JSON Schemaには「タイトル」注釈が含まれているものとします (SHALL)。 それは、全体的な説明のタイトルである必要があります。
- [R 6|1]
- 各 JSON Schemaには「説明 (description)」注釈が含まれているものとします (SHALL)。 これには、そのファイルの全体的な説明と著作権情報が含まれています。
- [R 7|1]
- 宣言された各ドキュメントおよびライブラリの ABIE 定義とその BBIE および ASBIE メンバーには、「タイトル」注釈と「説明」注釈が含まれなければなりません (SHALL)。
- 「タイトル」注釈は、BIEへの CCTS 辞書エントリ名でなければなりません。
- 文脈に応じた(contextualised)ビジネス名が存在する場合は、代わりにそれを使用するものとします(SHALL)。
- 「説明」注釈は CCTS 定義値となります。
- enum のメンバーには、「タイトル」または「説明」の注釈を含めてはなりません (SHALL NOT)。

3.1.2 全体的な JSON Schema構造

- [R 8|1]
- 各 JSON Schema フラグメントの「未評価プロパティ(unevaluated Property)」プロパティは、プリミティブ データ型、修飾されていない(unqualified)データ型、および修飾データ型のサブSchemaを除き、false に設定されるものとします。
- プリミティブ データ型、修飾されていないデータ型、または修飾データ型を指定するサブSchemaの場合、「未評価プロパティ」は、このドキュメントで定義されているように記述されるものとします(SHALL)。

3.2 バージョン管理と「$id」

- [R 9|1]
- JSON Schema ファイル名にはバージョン情報を含めてはなりません (SHALL NOT)。
- バージョンの違いは、$id と JSON Schema アーティファクトが配置されているフォルダー構造によってのみ示されます。
- [R 10|1]
- ユーザーグループまたは標準化組織によって公開されている各 JSON Schemaには、適切な $id URI プロパティにSchemaへの識別子が含まれているものとします(SHALL)。
- 閉じた環境 (テストなど) でのみ使用される JSON Schema エクスポートには、$id プロパティを含める必要はありません。
- URI は次の形式に従うものとします:

3.2 バージョン管理と「$id」

- [R 8|1]

"$id": "<basepath>/<variant>/<domain>/<version>[/<RDM>]/<BIE>"

with <basepath> identifying the originator. (発信者を識別する<basepath>)

UNECE アーティファクトの場合それは、

“https://github.com/uncefact/spec-JSONschema

<variant> は、JSON Schema ドラフト バージョンとエクスポート バリアントを表します。 例えば “JSONschema2020-12/library”

<domain> "BuyShipPay"のような部門。

<version> UNECE 出版形式でのバージョン。例えば “D22A”

<BIE> 以下の1つが付きます

- ファイル拡張子のない各メッセージ アセンブリ ABIE用の個別の名前 (例えばCross Industry Invoice)
- すべての BBIE コンポーネント用の名前: “BasicComponents”
- ライブラリ ABIE コンポーネントのすべての RDM セット用の個別の名前:

例えば “BSP-RDMComponents” または "SC-RDMComponents"

- 各拡張機能コレクション用の個別の名前

<RDM> スナップショット バリアントの場合、追加の構造化が許可されます。

3.2 バージョン管理と「$id」

JSON Schema ファイル名は次の形式で構築されるものとします:

<originator>-<abbreviation>.json

with

- <originator> identifying the originator. (発信者を識別する<originator>)

  UNECE のアーティファクトの場合それは、UNECE であるものとします。

- <abbreviation（略語）> ライブラリ ABIE コンポーネントの RDM セットを識別します。
  メッセージ構造にコンテキスト化されたビジネス名が存在する場合は、代わりにそれを使用するものとします(SHALL)。
  この名前の .json ファイルがすでに存在する場合は、メッセージ モデル名を別のハイフンで区切って追加するものとします (SHALL)。

[Examples]

"$id":"https://github.com/uncefact/spec-JSONschema/JSONschema2020-12/library/BuyShipPay/D22A/BasicComponents"

UNECE-BasicComponents.json

"$id":"https://github.com/uncefact/spec-JSONschema/JSONschema2020-12/library/BuyShipPay/D22A/CrossIndustryInvoice"

UNECE-CrossIndustryInvoice.json

"$id":"https://github.com/uncefact/spec-JSONschema/JSONschema2020-12/library/BuyShipPay/D22A/CrossIndustryInvoice-Variant"

UNECE-CrossIndustryInvoice-Variant.json

3.3 CCTS から JSON への一般的な命名規則

- [R 11|1]
- BasicComponents のJSON Schema ファイルには、プリミティブ データ型、修飾されていないデータ型 (unqualified data types)、および修飾されたデータ型 (qualified data types)のすべてのサブSchemaが含まれているものとします。
- [R 12|1]
- プロパティは、JSON オブジェクト内の名前と値のペアです。
- プロパティ名は、プロパティのキーまたは名前の部分です。 プロパティ値は、プロパティの値の部分です。

[Example]

```
{
    "propertyName": "propertyValue"
}
```

- [R 13|1]
- JSON プロパティ名は、辞書エントリ名 (DEN; Dictionary Entry Names) から派生するものとします (SHALL)。
- たとえば、 BBIE または ASBIE では、DEN に周囲の ABIE の DEN が含まれている場合、それは削除されるものとします (SHALL)。
- BBIE または ASBIE に連続した同一の単語が含まれる場合、重複は削除されるものとします(SHALL)。 NDR 規則を適用することによって DEN 内の単語が重複した場合、その重複は削除されるものとします (SHALL)。

3.3 CCTS から JSON への一般的な命名規則

- [R 14|1]
- 終止符 . 、非改行スペース (ASCII コード 160)、およびアンダースコア _ などの特殊文字は、基になる辞書エントリ名から削除されるものとします。
- 数字 (0-9) が空白の前にあり、別の数字が空白の後にある場合、空白はハイフン - に置き換えられるものとします(SHALL)。

[Example]

"This. is_ a. class. name" is represented as "thisIsAClassName"
"ISO 4217 3 A" is represented as "ISO4217-3A"

- [R 15|1]
- JSON プロパティ名は、小文字のキャメルケース*の ASCII 文字列であり、JSON Schemaに準拠している必要があります。
- 空白の後の文字は大文字でなければなりません。
- DEN の大文字は使用しないでください。

[Example]

"Specified. IBAN. Identifier" is represented as "specifiedIbanId"

"AAA Archive_ Document. Specified. AAA Archive_ Archive Parameter" is represented as "specifiedAaaArchiveParameter"

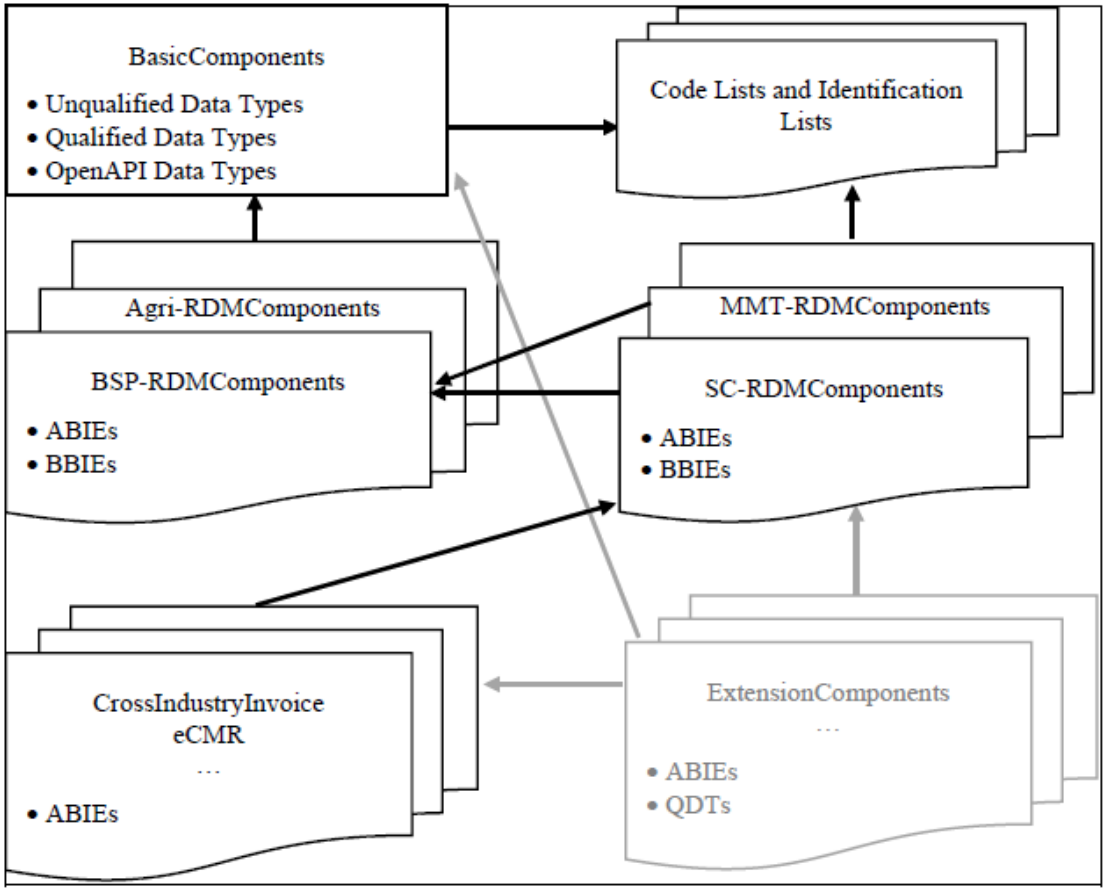
3.3 CCTS から JSON への一般的な命名規則

- [R 16|1]
- 略語と頭字語は表 4 の定義に従って使用するものとします (SHALL)。[R 15|1] を考慮するものとします (SHALL)。

CCTSの外観	JSON表現
"Indicator"	省略されます。"isOrHas"がプレフィックスとして追加されます。
"Identification. Identifier"	"Id"
"Text"	省略されます。
"Specified_"	省略されます。
初めに"AAA "のあるもの。 "TT_" "Transport_" "Supply Chain_" "CI_"	ABIE の結果の名前が一意である場合は省略され (SHALL)、それ以外の場合はそのまま保持されます (SHALL)。
"Formatted_"	省略されます。
最後に“Trade_ Party”のあるもの。	省略されます。

- [R 17|1]
- オブジェクトクラス用語「識別スキーム」は「スキーム」と表すものとします。 [R 15|1]が考慮されるものとします。

3.4 JSON Schemaの展望



3.5.1 プリミティブ データ型

- 10 進データ型は、測定値だけでなく、特に金額 (特定の通貨) を表すために使用されるため、特別な処理が必要です。
- JSON は、このような 10 進データ型をサポートしていません。
- データ型「数値」のみをサポートします。
- これは技術的には浮動小数点数または倍精度データ型として実装されます。
- 多くの議論がありますが、10 進データ型の代わりに浮動小数点データ型を使用することの難しさを示す実践的な経験 (EN16931 の実装による検証規則の適用などに基づく) もあります。
- 要約すると、float データ型を使用すると、必然的に丸めの違いが生じ、送信される値が不正確に表現されると言えます。
- UNECE 参照データ モデルの実装にはビジネス データの交換が含まれるため、値の正確な送信が最優先事項です。
- これを念頭に置いて、10 進データ型は JSON Schemaでは文字列表現として表されます。
- これは、JSON レベルで直接演算を使用できない場合でも、さまざまな実装言語で損失なくクリーンに実装できます。

3.5.1 プリミティブ データ型

- [R 18|1]
- プリミティブ データ型 (PDT) は、表 6 に示すように、JSON Schemaで表現されるものとします (SHALL)。それらは、$defs/pdt/ の下に配置されるものとします (SHALL)。

CCTS プリミティブ データ型	JSON表現

Binary	OpenAPI 3.0.x で使用する場合の表現:
	`"binaryType":`
	`{`
	`"title": "Binary",`
	`"description": "",`
	`"type": "string",`
	`"format": "byte"`
	`}`
	
	OpenAPI 3.1.x で使用する場合の表現 (JSON Schemaの完全なサポート、デフォルトのパブリケーション バリアント):
	`"binaryType":`
	`{`
	`"title": "Binary",`
	`"description": "",`
	`"type": "string",`
	`"contentEncoding": "base64"`
	`}`

3.5.1 プリミティブ データ型

CCTS プリミティブ データ型	JSON表現
Boolean	`"type": "boolean"`
Decimal	`"decimalType": { "title": "Decimal", "description": "", "type": "string", "pattern": "^([+-]?(0?|[1-9][0-9]*)(¥¥.?¥¥d+))$" }`
Integer	`"type": "integer"`
String	`"type": "string"`

3.5.2 承認されたコアコンポーネント型

- 承認されたコア コンポーネント型 は、JSON Schemaで直接表現されません。
- 代わりに、UDT は JSON Schemaに直接マッピングされます。

3.5.3 修飾されていないデータ型

- UDT は、CCTS のその他すべてのデータ構造の基礎を形成します。
- これらは、実際の値（コンテンツ）と、通常はオプションの補足コンポーネントで構成されます。
- 文脈化の際、これらの補足コンポーネントの一部は省略されることがよくあります。
- 実際、これにより実際の実装では UDT の数が倍増し、技術的に複雑になります。
- このため、UDT のコンテキスト化は JSON Schemaにマッピングされません。
- 代わりに、より高いデータ型の完全な UDT が常に使用されます。
- [R 19|1]
- 修飾されていないデータ型はサブSchemaで表現されるものとします (SHALL)。
- 辞書エントリ名の一部としての「Type」は保持されるものとします (SHALL)。

3.5.3 修飾されていないデータ型

- [R 20|1]
- CCTS コンテンツ プロパティは、「content(内容)」という名前のサブSchemaで表されるものとします (SHALL)。
- そのデータ型は基礎となる PDT を使用するものとします (SHALL)。 content-property は必須であるものとします (SHALL)。
- [R 21|1]
- 補足コンポーネントのプロパティ名は、JSON サブSchemaのプロパティ名を繰り返してはなりません。
- [R 22|1]
- 補足コンポーネントは、コードリストおよび/または識別スキームを参照する場合があります。
- この場合、JSON プロパティは、セクション 3.5.5 その他の修飾されたデータ型で定義されている適切なコード リストまたは識別スキームを参照するものとします (SHALL)。
- [R 23|1]
- 修飾されていないデータ型は、表 7 に示すようにサブSchemaで表現されるものとします (SHALL)。タ
- イトルと説明のプロパティは次の表には示されていません。
- 時間の経過とともに変更される可能性があるため、代わりにプレースホルダー <タイトルと説明> で示されます。 これらは規則 [R 5|1]、[R 6|1]、および [R 7|1] に従って公開されるものとします。
- これらは $defs/udt の下に配置されるものとします (SHALL)。

3.5.3 修飾されていないデータ型

CCTS 修飾されていない データ型	JSON表現

• Binary Object. Type
• Binary Object. Content
• Binary Object. Format. Text
• Binary Object. Mime . Code
• Binary Object . Encoding. Code
• Binary Object. Character Set. Code
• Binary Object. Uniform Resource. Identifier
• Binary Object. Filename. Text
	"binaryObjectType": {
	<<title and description>>
	"type": "object",
	"properties": {
	"content": {
	<<title and description>>
	"$ref": "#/$defs/pdt/binaryType"
	},
	"format": {
	<<title and description>>
	"type": "string"
	},
	"mimeCode": {
	<<title and description>>
	"type": "string"
	},
	"encodingCode": {
	<<title and description>>
	"$ref":
	"UNECE_CharacterSetEncoding.json#/$defs/codeList/characterSetEncodingType"
	},

3.5.3 修飾されていないデータ型

CCTS 修飾されていない データ型	JSON表現

	"characterSetCode": {
	<<title and description>>
	"$ref": "UNECE_CharacterSets.json#/$defs/codeList/characterSetsType"
	},
	"uri": {
	<<title and description>>
	"type": "string",
	"format": "uri"
	},
	"filename": {
	<<title and description>>
	"type": "string"
	}
	},
	"required": ["content"] ,
	"unevaluatedProperties": false
	}

3.5.3 修飾されていないデータ型

CCTS 修飾されていない データ型	JSON表現

• Code. Type
• Code. Content
• Code List. Identifier
• Code List. Agency. Identifier
• Code List. Agency Name. Text
• Code List. Name. Text
• Code List. Version. Identifier
• Code. Name. Text
• Language. Identifier
• Code List. Uniform Resource. Identifier
Code List Scheme. Uniform Resource. Identifier
	<pre>"codeType": {
	  <<title and description>>
	  "type": "object",
	  "properties": {
	  "content": {
	  <<title and description>>
	  "type": "string"
	  },
	  "listId": {
	  <<title and description>>
	  "type": "string"
	  },
	  "listAgencyId": {
	  <<title and description>>
	  "$ref": "UNECE_UNTDID-3055.json#/$defs/codeList/untdid3055Type"
	  },
	  "listAgencyName": {
	  <<title and description>>
	  "type": "string"
	  },</pre>

3.5.3 修飾されていないデータ型

CCTS 修飾されていない データ型	JSON表現

	"listName": {
	<<title and description>>
	"type": "string"
	},
	"listVersionId": {
	<<title and description>>
	"type": "string"
	},
	"name": {
	<<title and description>>
	"type": "string"
	},
	"languageId": {
	<<title and description>>
	"$ref": "UNECE_UNTDID-3453.json#/$defs/codeList/untdid3453Type"
	},

3.5.3 修飾されていないデータ型

CCTS 修飾されていない データ型	JSON表現

	"listUri": {
	<<title and description>>
	"type": "string",
	"format": "uri"
	},
	"listSchemaUri": {
	<<title and description>>
	"type": "string",
	"format": "uri"
	}
	},
	"required": ["content"] ,
	"unevaluatedProperties": false
	}

3.5.3 修飾されていないデータ型

CCTS 修飾されていない データ型	JSON表現

• Date Time. Type	"dateTimeType": {
	<<title and description>>
	"type": "string",
	"format": "date-time"
	}

• Date. Type	"graphicType": {
	<<title and description>>
	"$ref": #/$defs/udt/binaryObjectType"
	}

• Graphic. Type	"graphicType": {
	<<title and description>>
	"$ref": #/$defs/udt/binaryObjectType"
	}

3.5.3 修飾されていないデータ型

CCTS 修飾されていない データ型	JSON表現

• Identifier. Type
• Identifier. Content
• Identification Scheme. Identifier
• Identification Scheme. Name. Text
• Identification Scheme Agency. Identifier
• Identification Scheme. Agency Name. Text
• Identification Scheme. Version. Identifier
• Identification Scheme Data. Uniform Resource. Identifier
• Identification Scheme. Uniform Resource. Identifier
	<pre>"identifierType": {
	  <<title and description>>
	  "type": "object",
	  "properties": {
	  "content": {
	  <<title and description>>
	  "type": "string"
	  },
	  "schemeId": {
	  <<title and description>>
	  "type": "string"
	  },
	  "schemeName": {
	  <<title and description>>
	  "type": "string"
	  },
	  "schemeAgencyId": {
	  <<title and description>>
	  "$ref": "UNECE_UNTDID-3055.json#/$defs/codeList/untdid3055Type"
	  },</pre>

3.5.3 修飾されていないデータ型

CCTS 修飾されていない データ型	JSON表現

	<pre>"schemeAgencyName": {
	<<title and description>>
	"type": "string"
	},
	"schemeVersionId": {
	<<title and description>>
	"type": "string"
	},
	"schemeDataUri": {
	<<title and description>>
	"type": "string",
	"format": "uri"
	},
	"schemeUri": {
	<<title and description>>
	"type": "string",
	"format": "uri"
	}
	},
	"required": ["content"],
	"unevaluatedProperties": false
	}</pre>

3.5.3 修飾されていないデータ型

CCTS 修飾されていない データ型	JSON表現

• Indicator. Type	"indicatorType": {
	<<title and description>>
	"type": "boolean"
	}

• Measure. Type
• Measure. Content
• Measure Unit. Code
• Measure Unit. Code List Version. Identifier
	"measureType": {
	<<title and description>>
	"type": "object",
	"properties": {
	"content": {
	<<title and description>>
	"$ref": "#/$defs/pdt/decimalType"
	},
	"unitCode": {
	<<title and description>>
	"$ref": "UNECE_UNTDID-6411.json#/$defs/codeList/untdid6411Type"
	},

3.5.3 修飾されていないデータ型

CCTS 修飾されていない データ型	JSON表現

	"unitCodeListVersionId": {
	<<title and description>>
	"type": "string"
	}
	},
	"required": ["content"],
	"unevaluatedProperties": false
	}

• Name. Type
• Text. Content
• Language. Identifier
• Language. Locale. Identifier
	"nameType": {
	<<title and description>>
	"$ref": "#/$defs/udt/textType"
	}

3.5.3 修飾されていないデータ型

CCTS 修飾されていない データ型	JSON表現

	"unitCodeListVersionId": {
	<<title and description>>
	"type": "string"
	}
	},
	"required": ["content"],
	"unevaluatedProperties": false
	}

• Name. Type
• Text. Content
• Language. Identifier
• Language. Locale. Identifier
	"nameType": {
	<<title and description>>
	"$ref": "#/$defs/udt/textType"
	}

• Numeric. Type
• Numeric. Content
• Numeric. Format. Text
	"numericType": {
	<<title and description>>
	"type": "object",
	"properties": {
	"content": {
	<<title and description>>
	"$ref": "#/$defs/pdt/decimalType"
	},

3.5.3 修飾されていないデータ型

CCTS 修飾されていない データ型	JSON表現

	"format": {
	<<title and description>>
	"type": "string"
	}
	},
	"required": ["content"] ,
	"unevaluatedProperties": false
	}

• Percent. Type	"percentType": {
	<<title and description>>
	"$ref": "#/$defs/udt/numericType"
	}

• Picture. Type	"pictureType": {
	<<title and description>>
	"$ref": #/$defs/udt/binaryObjectType"
	}

3.5.3 修飾されていないデータ型

CCTS 修飾されていない データ型	JSON表現

• Quantity. Type
• Quantity. Content
• Quantity Unit. Code
• Quantity Unit. Code List. Identifier
• Quantity Unit. Code List Agency. Identifier
• Quantity Unit. Code List Agency Name. Text
	<pre>"quantityType": {
	<<title and description>>
	"type": "object",
	"properties": {
	"content": {
	<<title and description>>
	"$ref": "#/$defs/pdt/decimalType"
	},
	"unitCode": {
	<<title and description>>
	"$ref": "UNECE_REC-20+21.json#/$defs/codeList/rec20+21Type"
	},
	"unitCodeListId": {
	<<title and description>>
	"type": "string"
	},

3.5.3 修飾されていないデータ型

CCTS 修飾されていない データ型	JSON表現

• Quantity. Type
• Quantity. Content
• Quantity Unit. Code
• Quantity Unit. Code List. Identifier
• Quantity Unit. Code List Agency. Identifier
• Quantity Unit. Code List Agency Name. Text
	<pre>"unitCodeListAgencyId": {
	<<title and description>>
	"$ref": "UNECE_UNTDID-3055.json#/$defs/codeList/untdid3055Typ"
	},
	"unitCodeListAgencyName": {
	<<title and description>>
	"type": "string"
	}
	},
	"required": ["content"],
	"unevaluatedProperties": false
	}</pre>

	[注記] Rec 20 は、Rec 21 コード値にプレフィックスを追加することで、Rec 21 との組み合わせをサポートします。 この JSON サブSchemaの使用では、必要に応じて結合コード リストを制限できます。

3.5.3 修飾されていないデータ型

CCTS 修飾されていない データ型	JSON表現

• Rate. Type	"rateType": {
	<<title and description>>
	"$ref": "#/$defs/udt/numericType"
	}

• Sound. Type	"soundType": {
	<<title and description>>
	"$ref": "#/$defs/udt/binaryObjectType"
	}

3.5.3 修飾されていないデータ型

CCTS 修飾されていない データ型	JSON表現

• Text. Type
• Text. Content
• Language. Identifier
• Language. Locale. Identifier
	"textType": {
	<<title and description>>
	"type": "object",
	"properties": {
	"content": {
	<<title and description>>
	"type": "string"
	},
	"languageId": {
	<<title and description>>
	"$ref": "ISO_6391-1-2A.json#/$defs/codeList/iso6391-1-2AType"
	},
	"languageLocaleId": {
	<<title and description>>
	"type": "string"
	}
	},
	"required": ["content"],
	"unevaluatedProperties": false
	}

3.5.3 修飾されていないデータ型

CCTS 修飾されていない データ型	JSON表現

• Time. Type	"timeType": {
	<<title and description>>
	"type": "string",
	"format": "time"
	}

• Value. Type	"valueType": {
	<<title and description>>
	"$ref": "#/$defs/udt/numericType"
	}

• Video. Type	"videoType": {
	<<title and description>>
	"$ref": "#/$defs/udt/binaryObjectType"
	}

3.5.4 日付と時刻の修飾データ型

- CCTS は、ISO 8601 のさまざまな日付と時刻の形式の広範なサブセットをサポートします。
- ただし、この柔軟性は実際のアプリケーションでのみ必要とされ、限られた範囲で使用されます。
- 多くの場合、日付、時刻、および結合された情報は、JSON Schemaによって直接サポートされる単純な表現形式に縮小できます。
- いくつかの例外が存在するため、CCTS ではいくつかの特殊な QDT が定義されています。
- これらの QDT のモデリングは初期の EDIFACT 定義に戻っており、JSON Schemaを使用する OpenAPI のアプリケーションにとっては最新ではないようです。
- それにもかかわらず、この表記法は依然として幅広いコミュニティで使用されています。
- このような背景に対して、これらの QDT を次のように簡略化して使用します。
- [R 24|1]
- The ”Date Mandatory_ Date Time.Type”は、formattedDateTimeType に置き換えるものとします (SHALL)。
- [R 25|1]
- The ”Time Only_ Formatted_ Date Time.Type”は、formattedDateTimeType に置き換えるものとします (SHALL)。

3.5.4 日付と時刻の修飾データ型

- [R 26|1]

The “Formatted_DateTime.Type” は次のように表現されるものとします(SHALL)。

```
"formattedDateTimeType":
  <<title and description>>
    "oneOf": [
      { "type": "string", "format": "date-time" },
      { "type": "string", "format": "time" },
      { "type": "string", "format": "date" },
      { "type": "string", "format": "duration" },
      { "type": "object",
        "properties": {
          "content": { "type": "string" }
          "format": { "$ref": "UNECE_UNTDID2379-
JSON.json#/$defs/codeList/untdid2379JsonType" }
        }
        "required": ["content", "format"]
      }
    ]
}
```

3.5.4 日付と時刻の修飾データ型

[Example]

JSON schema definition:

```
{   "properties": {
    "myDateTime": { "$ref": "#/$defs/formattedDateTimeType"}
    }
}
```

JSON instance:

Hint: The presence of "content" indicates that it is a UNECE specific format not directly supported by JSON schema.

```
{
    "myDateTime": {"content": "2022-W02", "format": "CCYY-Www"},
    "myDateTime": {"content": "1T10:00/1T12:00", "format":
"NThh:mm/NThh:mm"},
    "myDateTime": "2022-02-11",
    "myDateTime": "2022-02-11T12:23:58Z",
    "myDateTime": "12:23:58Z",
    "myDateTime": "P10W"
}
```

3.5.4 日付と時刻の修飾データ型

- [R 27|1]
- コードリスト「UNTDID 2379」に基づいて、追加のコードリスト「UNTDID 2379 json」を指定する必要があります。
- 既存の JSON 日付および時刻形式によってその意味がすでに表現されている形式定義はすべて省略されます (SHALL)。
- このコードリストは、UNTDID 2379 に従って維持されるものとします (SHALL)。
- 他のすべての形式は次のように表されるものとします (SHALL)。

```
"untdid2379JsonType": {

  "title": "Date and Time format codes for JSON representation.",

  "description": "This code list is based on UNTDID 2379. It is adjusted to take JSON date and time representation into account.¥n

# The following abbreviations are used¥n

*  'C' - Century¥n

*  'Y' - Year¥n

*  'M' - Month¥n

*  'D' - Day¥n

*  'h' - Hour¥n
```

3.5.4 日付と時刻の修飾データ型

```
*  'm' - Minute¥n
*  's' - Second¥n
*  'w' - Week¥n
*  'T' - Time zone offset separator (+/-/Z) ¥n
¥n
*  'A' - 10 day period within a month of a year¥n
*  'B' - 1: First half month; 2: second half month¥n
*  'E' - Week of a month¥n
*  'G' - Working days¥n
*  'H' - Half month¥n
*  'I' - 1-9: Shift in a day¥n
*  'K' - 1-5: First to fifth week in a month¥n
*  'M' - Trimester: A period of three months¥n
*  'N' - 1-7: Numeric representation of the day (Monday = 1, Sunday = 7)¥n
*  'P' - A period of 4 months¥n
*  'Q' - Quarter¥n
*  'RR' - 00-99: Time period¥n
*  'S' - Semester¥n
*¥n
```

3.5.4 日付と時刻の修飾データ型

```
* Hyphens and additional character in a format string are kept. According
to ISO 8601 a slash is used to separate time spans.¥n
# Codes from UNTDID 2379 and their representation in JSON¥n
* '2' - is represented as 'date' format¥n
* '3' - is represented as 'date' format¥n
* '4' - is represented as 'date' format¥n
* '5' - is represented as 'date-time' format¥n
* '6' - is represented as 'CCYY-MM-B'¥n
* '7' - is represented as 'CCYY-MM-K'¥n
* '8' - is represented as 'CCYY-MM-DD-I'¥n
* '9' - is represented as 'CCYY-MM-DD-RR'¥n
* '10' - is represented as 'date-time' format¥n
* '101' - is represented as 'date' format¥n
* '102' - is represented as 'date' format¥n
* '103' - is represented as 'YY-Www-N'; 01 is first week of January; 1 is always Monday¥n
* '104' - is represented as 'MM-WEE/MM-WEE'¥n
* '105' - is represented as 'YY-DDD'; January the first = Day 001; Always start numbering the
days of the year from January 1st through December
31st ¥n
```

3.5.4 日付と時刻の修飾データ型

* '106' - is represented as '-MM-DD'¥n
* '107' - is represented as 'DDD'¥n
* '108' - is represented as 'WW'¥n
* '109' - is represented as '-MM-'¥n
* '110' - is represented as '--DD'¥n
* '201' - is represented as 'date-time' format¥n
* '202' - is represented as 'date-time' format¥n
* '203' - is represented as 'date-time' format¥n
* '204' - is represented as 'date-time' format¥n
* '205' - is represented as 'date-time' format¥n
* '206' - is represented as 'date-time' format¥n
* '207' - is represented as 'date-time' format¥n
* '208' - is represented as 'date-time' format¥n
* '209' - is represented as 'date-time' format¥n
* '210' - is represented as 'hh:mm:ssZhh:mm/hh:mm:ssZhh:mm'¥n
* '301' - is represented as 'date-time' format¥n
* '302' - is represented as 'date-time' format¥n
* '303' - is represented as 'date-time' format¥n

3.5.4 日付と時刻の修飾データ型

```
* '304' - is represented as 'date-time' format¥n
* '305' - is represented as '-MM-DDThh:mm' format¥n
* '306' - is represented as '--DDThh:mm' format¥n
* '307' - is represented as 'date-time' format¥n
* '308' - is represented as 'CCYY-MM-DDThh:mmZhh:mm/CCYY-MM-DDThh:mmZhh:mm' ¥n
* '401' - is represented as 'time' format¥n
* '402' - is represented as 'time' format¥n
* '404' - is represented as 'time' format¥n
* '405' - is represented as 'duration' format¥n
* '406' - is represented as 'Zhh:mm'¥n
* '501' - is represented as 'hh:mm/hh:mm' ¥n
* '502' - is represented as 'hh:mm:ss/hh:mm:ss' ¥n
* '503' - is represented as 'hh:mm:ssZhh:mm/hh:mm:ssZhh:mm' ¥n
* '600' - is represented as 'CC'¥n
* '601' - is represented as 'YY' ¥n
* '602' - is represented as 'CCYY' ¥n
* '603' - is represented as 'YY-S' ¥n
* '604' - is represented as 'CCYY-S' ¥n
```

3.5.4 日付と時刻の修飾データ型

```
* '608' - is represented as 'CCYY-Q' ¥n
* '609' - is represented as 'YY-MM' ¥n
* '610' - is represented as 'CCYY-MM' ¥n
* '613' - is represented as 'YY-MM-A' ¥n
* '614' - is represented as 'YY-MM-A' ¥n
* '615' - is represented as 'YY-Www ¥n
* '616' - is represented as 'CCYY-Www' ¥n
* '701' - is represented as 'YY/YY' ¥n
* '702' - is represented as 'CCYY/CCYY' ¥n
* '703' - is represented as 'YY-S/YY-S' ¥n
* '704' - is represented as 'CCYY-S/CCYY-S' ¥n
* '705' - is represented as 'YY-P/YY-P' ¥n
* '706' - is represented as 'CCYY-P/CCYY-P' ¥n
* '707' - is represented as 'YY-Q/YY-Q' ¥n
* '708' - is represented as 'CCYY-Q/CCYY-Q' ¥n
* '709' - is represented as 'YY-MM/YY-MM' ¥n
* '710' - is represented as 'CCYY-MM/CCYY-MM' ¥n
* '713' - is represented as 'YY-MM-DDThh:mm/YY-MM-DDThh:mm' ¥n
```

3.5.4 日付と時刻の修飾データ型

```
* '715' - is represented as 'YY-Www/YY-Www' ¥n
* '716' - is represented as 'CCYY-Www/CCYY-Www' ¥n
* '717' - is represented as 'YY-MM-DD/YY-MM-DD' ¥n
* '718' - is represented as 'CCYY-MM-DD/CCYY-MM-DD' ¥n
* '719' - is represented as 'CCYY-MM-DDThh:mm/CCYY-MM-DDThh:mm' ¥n
* '720' - is represented as 'NThh:mm/NThh:mm' ¥n
* '801' - is represented as 'duration' format ¥n
* '802' - is represented as 'duration' format ¥n
* '803' - is represented as 'duration' format ¥n
* '804' - is represented as 'duration' format ¥n
* '805' - is represented as 'duration' format ¥n
* '806' - is represented as 'duration' format ¥n
* '807' - is represented as 'duration' format ¥n
* '808' - is represented as 'S' ¥n
* '809' - is represented as 'P' ¥n
* '810' - is represented as 'M' ¥n
* '811' - is represented as 'H' ¥n
* '812' - is represented as 'A' ¥n
* '813' - is represented as 'N' ¥n
* '814' - is represented as 'G' ¥n
```

3.5.4 日付と時刻の修飾データ型

```
",
"oneOf": [
{ "const": "CCYY-MM-B" },
{ "const": "CCYY-MM-K" },
{ "const": "CCYY-MM-DD-I" },
{ "const": "CCYY-MM-DD-RR" },
{ "const": "YY-Www-N" },
{ "const": "MMWEE/MMWEE" },
{ "const": "YY-DDD" },
{ "const": "-MM-DD" },
{ "const": "DDD" },
{ "const": "-WW" },
{ "const": "-MM-" },
{ "const": "--DD" },
{ "const": "hh:mm:ssZhh:mm/hh:mm:ssZhh:mm" },
{ "const": "-MM-DDThh:mm" },
{ "const": "--DDThh:mm" },
{ "const": "CCYY-MM-DDThh:mmZhh:mm/CCYY-MM-DDThh:mmZhh:mm" },
```

3.5.4 日付と時刻の修飾データ型

```
{ "const": "Zhh:mm" },
{ "const": "hh:mm/hhmm" },
{ "const": "hh:mm:ss/hh:mm:ss" },
{ "const": "hh:mm:ssZhh:mm/hh:mm:ssZhh:mm" }, { "const": "CC" },
{ "const": "YY" },
{ "const": "CCYY" },
{ "const": "CCYY-S" }, { "const": "CCYY-Q" },
{ "const": "YY-MM" },
{ "const": "CCYY-MM" },
{ "const": "YY-MM-A" },
{ "const": "CCYY-MM-A" },
{ "const": "YY-Www" },
{ "const": "CCYY-Www" },
{ "const": "YY/YY" },
{ "const": "CCYY/CCYY" },
{ "const": "YY-S/YY-S" },
{ "const": "CCYY-S/CCYY-S" },
{ "const": "YY-P/YY-P" },
```

3.5.4 日付と時刻の修飾データ型

```
{ "const": "CCYY-P/CCYY-P" },
{ "const": "YY-Q/YY-Q" },
{ "const": "CCYY-Q/CCYY-Q" },
{ "const": "YY-MM/YY-MM" },
{ "const": "CCYY-MM/CCYY-MM" },
{ "const": "YY-MM-DDThh:mm/YY-MM-DDThh:mm" },
{ "const": "YYWww/YYWww" },
{ "const": "CCYYWww/CCYYWww" },
{ "const": "YY-MM-DD/YY-MM-DD" },
{ "const": "CCYY-MM-DD/CCYY-MM-DD" },
{ "const": "CCYY-MM-DDThh:mm/CCYY-MM-DDThh:mm" }, { "const": "NThh:mm/NThh:mm" },
{ "const": "S" },
{ "const": "P" },
{ "const": "M" },
{ "const": "H" },
{ "const": "A" },
{ "const": "N" },
{ "const": "G" }
]
}
```

3.5.5 他の修飾データ型

- CCTS コードおよび識別子リストでは、修飾データ型 (QDT) として指定されます。
- これらは UDT codeType または idType に基づいています。
- UDT codeType と前述の idType は、発行機関や使用されているコード リストのバージョンやSchema バージョンなど、コード リストまたは識別スキーム固有のプロパティを記述する機能を提供します。
- すべてのコード リスト、識別スキーム、または修飾されたデータ型にこれらのプロパティがすべて適用できるわけではありませんが、それが考慮されます。
- [R 28|1]
- セクション 3.5.4 に該当しない各 QDT は、セクション 3.6.1 で説明される方法を適用するその定義に従って制限されるものとします (SHALL)。

3.5.5 他の修飾データ型

[Example]

```
"unitMeasureType": {
  "title": "Unit_ Measure. Type",
  "description": "A numeric value determined by measuring an object along with the specified unit of measure.",
  "$ref" : "#/$defs/udt/measureType",
  "required": ["unitCode"],
  "properties": {
    "unitCodeListVersionId": false
  }
}
```

3.5.5 他の修飾データ型

- [R 29|1]
- 各 QDT はサブSchemaで表されるものとします (SHALL)。
- code または id 値がローカルで指定される場合、それらは const 定義のいずれかの組み合わせであるものとします (SHALL)。
- これらは enumarray として指定してはなりません (SHALL NOT)。
- 各コード値は文字列型として表されるものとします (SHALL)。
- コードと ID の値がコードおよび識別スキームで編成されている場合、対応する JSON Schemaは適切なコード リストまたは識別スキームを参照するものとします (SHALL)。
- [R 30|1]
- 各コード リストと識別スキーム (identification scheme)は、別個の JSON Schema ファイルで指定する必要があります。
- JSON Schema ファイルは、使用されているコード リストと識別スキームごとに作成されるものとします (SHALL)。
- その名前は、コードリストまたは識別スキームの名前を表すものとし、次の形式で一意であるものとします。
- <Code List Agency Name>_<Code List Name or Identifier>.json
- <Identification Scheme Agency Name>_<Identification Scheme Name or Identifier>.json

3.5.5 他の修飾データ型

- ここでは：
- すべての特殊文字は名前から削除されるものとします (SHALL)。
- ピリオド ... バージョン番号の は文字 p に置き換えられます。
- <コード リスト機関名> – コード リストを管理する機関。
- <識別スキーム機関名> – 識別スキームを管理する機関。
- <コード リスト名または識別子> – コード リスト識別子が UNTDID に存在する場合、識別子は UNTDID<識別子> の形式で指定されます。
- それ以外の場合、コード リスト名は発行代理店によって割り当てられたものとして記載されています。
- <識別スキーム名または識別子> – 識別スキーム識別子が UNTDID に存在する場合、識別子は UNTDID<識別子> の形式で指定されます。
- それ以外の場合、識別スキーム名は発行機関によって割り当てられたものとして記載されています。

- ファイルはエクスポート パスのサブフォルダー codelist に配置されるものとします (SHALL)。
- $id プロパティは、このサブフォルダー構造を反映するものとします。

3.5.5 他の修飾データ型

[Example]

UNECE_UNTDID-1001.json

OpenPEPPOL_DocumentTypes.json

- [R 31|2]
- SON Schema アーティファクトとコード リストおよび識別スキームとの互換性を可能な限り維持することは、明確な目標です。
- このため、コード リストのバージョンと識別スキームのバージョンは、.json ファイル名の一部でも型名の一部でもありません。
- ただし、これは $id の一部であるため、必要に応じてバージョンを区別するために JSON Schema ファイルを使用できます。
- 何らかの理由で、特定のシナリオでコード リストまたは識別スキームの複数のバージョンを使用する必要がある場合は、<コード リスト バージョン> または <識別スキーム バージョン> を次の形式でファイル名に追加する必要があります。
- <Code List Agency Name>_<Code List Name or Identifier>_<Code List Version>.json
- <Identification Scheme Agency Name>_<Identification Scheme Name or Identifier>_<Identification Scheme Version>.json

3.5.5 他の修飾データ型

- JSON が発明されて以来、JSON がSchema ファイル内のコメントをサポートすべきかどうかについて議論が繰り返されてきました。
- 基本的な概念に関して、JSON はデータのみであり、コメントをサポートしないことが意図的に決定されました。
- それにもかかわらず、バージョン管理が進むにつれて、description や $comment などの注釈が導入されました。
- 後者はパーサーによって無視されることになっており、Schema ユーザーに情報を提示するために使用されるべきではありません。
- 代わりに $comment は、将来のSchema開発者のための情報を含めることのみを目的としています。
- Schemaのメンテナンス情報を強調表示します。
- 何年もの間よく議論されてきたトピックは、列挙型のコメント化です。
- JSON Schemaは、C言語などのダブルスラッシュや PHP のハッシュタグに相当する .JSON ファイル内のコメントをサポートしません。
- 一部の JSON エディターは、このようなコメントを独自にサポートしています。
- ただし、通常は 2 つのバリアントのうちの 1 つだけであり、多くの場合、独自のプログラミング言語の規則に対応します。
- したがって、世界共通の規約がないため、UNECE JSON Schemaのコードと識別子のリストでは、そのような独自のコメントが不要になります。

3.5.5 他の修飾データ型

- この NDR 技術仕様は、OpenAPI 仕様で使用する JSON Schema アーティファクトの適用性を目的として作成されています。
- これは、そのような仕様の実装者にとって、個々のコードまたは識別子の値の文書化が重要であることを意味します。
- OpenAPI 3.1 以降、コード リストの推奨される表現は const 定義の 1 つの組み合わせ(oneOf combination of const definition) です。
- これにより、コード名と定義を個々のコードの定義に直接追加できます。
- さらに、個々のコード値の有効期間を追加するなどのさらなる修正も可能になります。
- [R 32|1]
- コードまたは識別子リストを指定する JSON Schemaの description プロパティには、CCL で定義されている著作権表示情報をリストする必要があります (SHALL)。
- これには、コードまたは識別子リストの名前、コードまたは識別子リストの発行機関、コードまたは識別子リストのバージョン、および著作権情報が含まれます。
- [R 33|1]
- コードまたは識別子リストの値を保持する const 定義を指定するサブSchemaの title プロパティは、英語のコード名の値である必要があります (SHOULD)。
- コードまたは識別子リストの値を保持する const 定義を指定するサブSchemaの description プロパティは、英語のコード定義値である必要があります (SHOULD)。

3.5.5 他の修飾データ型

- [R 34|1]
- コード リストは、次の命名規則に従って、対応するSchema ファイルのサブSchema内で表現されるものとします (SHALL):
- $defs/codeList/<Code List Name or Identifier>Type
- with <Code List Name or Identifier> – コード リスト識別子が UNTDID に存在する場合、識別子は untdid<識別子> の形式で指定されます。
- それ以外の場合、コード リスト名は、発行機関によって割り当てられたもののまま、特殊文字が削除されて記載されます。
- 次の例は、完全なコード リストの JSON Schema ファイルの内容を示しています。

[Example]

```
{
  "$schema":  "https://json-schema.org/draft/2020-12/schema",
  "$id":  "https://service.unece.org/trade/uncefact/json-schema/D22A/UNECE_UNTDID-3131",
  "title": "Address type code",
  "description": "<<copyright notice information>>",
  "$defs": {
    "codeList":  {
      "untdid3131Type": {
```

3.5.5 他の修飾データ型

```
"title": "Address type code",
"oneOf":  [
  {
    "const": "1",
    "title": "Postal Address"
  },
  {
    "const": "2",
    "title": "Fiscal Address"
  },
  {
    "const": "3",
    "title": "Physical Address"
  },
  {
    "const": "4",
    "title": "Business Address"
  },
```

3.5.5 他の修飾データ型

```
            {
              "const": "5",
              "title": "Delivery To Address"
            },
            {
              "const": "6",
              "title": "Residential Address"
            },
            {
              "const": "7",
              "title": "Mail To Address"
            },
            {
              "const": "8",
              "title": "Postbox Address"
            }
          ]
        }
      }
    }
}
```

3.5.5 他の修飾データ型

- [R 35|1]
- 識別スキームは、次の命名規則に従って、対応するSchema ファイルのサブSchema内で表現されるものとします (SHALL):
- $defs/identificationScheme/<Indentification Scheme Name or Identifier>Type
  with < Identification Scheme Name or Identifier> – 識別スキーム識別子が UNTDID に存在する場合、識別子は untdid<識別子> の形式で指定されます。
- それ以外の場合、コードまたは識別スキーム名は、発行機関によって割り当てられたもののまま、特殊文字が削除されて記載されています。
- 次の例は、完全な識別スキームの JSON Schema ファイルの内容を示しています。

[Example]

```
{
  "$schema":  "https://json-schema.org/draft/2020-12/schema",
  "$id": "https://service.unece.org/trade/uncefact/json-schema/D22A/ISO_639-1-2A",
  "title": "Language identifier", "description": "<<copyright notice information>>",
  "$defs":  {
    "identificationScheme": {
      "iso639-1-2AType": {
      "title": "Language identifier",
```

3.5.5 他の修飾データ型

```
"oneOf": [
  {
    "const": "AR",
    "title": "ARABIC"
  },
  {
    "const": "AS",
    "title": "ASSAMESE"
  },
  {
```

3.6.1 制限事項

- CCTS は、作成する制限方法を定義します、例えばCCL の業界固有のプロファイル。
- このプロセスの出力の 1 つは、サプライチェーン参照データ モデル (SCRDM) やマルチモーダル トランスポート参照データ モデル (MMT-RDM) のように公開されている参照データ モデル (RDM) です。
- メッセージを介したデータ送信の場合、この制限方法は、コードまたは識別子リストのカーディナリティ（濃度）と値を制限するためにも使用されます (追記: コードまたは識別子リストの値を制限する場合は、修飾されたデータ型が作成されます)。
- UN/CEFACT の標準化活動の重要な部分は、まさにこの問題に長年取り組んできました。
- ルール [R 10|1] で定義されているように、データ モデルの個々のレイヤーごとに個別の JSON Schema ファイルが公開されます。

3.6.1 制限事項

- [R 36|1]
- CCTS オブジェクトに対する制限は、サブSchemaで次のように表現されるものとします (SHALL)。
- Cardinalities（基数、濃度）
- From 0..1 to 1..1

[Example]

```
"toBeRestrictedType":  {
  "type": "object",
  "properties": {
    "id": { "type": "string" }
  }
},
"restrictingType":  {
  "$ref": "#/$defs/toBeRestrictedType",
  "required": ["id"]
}
```

3.6.1 制限事項

- From 0..1 to 0..0 (forbidden; 禁じられた)

[Example]

```
"toBeRestrictedType": {
  "type": "object",
  "properties": {
    "id": { "type": "string" },
    "name": { "type": "string" }
  }
},
"restrictingType": {
  "$ref": "#/$defs/toBeRestrictedType",
  "properties": {
    "id": false
  }
}
```

3.6.1 制限事項

- From 0..unbounded to 0..n with n < unbounded (制限されない)

[Example]

```
"toBeRestrictedType": {
  "type": "object",
  "properties": {
    "id": {
      "type": "array",
      "items": { "type": "string }
    }
  }
},
"restrictingType": {
  "$ref": "#/$defs/toBeRestrictedType",
  "properties": {
    "id": { "maxItems": 2 }
  }
}
```

3.6.1 制限事項

- From 0..unbounded to n.. unbounded

[Example]

```
"toBeRestrictedType": {
  "type": "object",
  "properties": {
    "id": {
      "type": "array",
      "items": { "type": "string }
    }
  }
},
"restrictingType": {
  "$ref": "#/$defs/toBeRestrictedType",
  "properties": {
    "id": { "minItems": 2 }
  }
}
```

3.6.1 制限事項

- Restriction of value ranges（値の範囲の制限）

[Example restricting content to values with exact 2 fraction digits]

```
"restrictingType": {
  "allOf": [
    { "$ref": "UNECE-BasicComponents.json#/$defs/udt/amountType" },
    { "properties": {
        "content": { "pattern": "^.*¥..{2}$" }
      }
    }
  ]
}
```

3.6.1 制限事項

- Restriction of value ranges（値の範囲の制限）

[Example restricting content to values with exact 2 fraction digits]

```
"restrictingType": {
  "allOf": [
    { "$ref": "UNECE-BasicComponents.json#/$defs/udt/amountType" },
    { "properties": {
        "content": { "pattern": "^.*¥..{2}$" }
      }
    }
  ]
}
```

3.6.1 制限事項

- Restriction of const（constの制限）

[Example restricting content to a code list subset]

```
"addressType": {
  "type": "object",
  "properties": {
    "countryId": { "$ref": "UNECE-BasicComponents.json#/$defs/qdt/countryIdType"}
  }
},
"restrictingType": {
  "allOf": [
    { "$ref": " #/$defs/addressType" },
    { "properties": {
        "countryId": { "const": ["CH", "DE", "FR"] }
      }
    }
  ]
}
```

3.6.1 制限事項

- 制限が下位レベルで定義されている場合、同じタイプの制限を適用できます。

[Example]

```
{
  "$defs": {
  "restriction": {
    "allOf": [
      {
        "$ref": "#/$defs/levelOne"
      },
      {
        "properties": {
          "oneFirst": {
            "properties": {
              "twoFirst": false
            }
          }
        }
      }
    ]
  },
```

3.6.1 制限事項

```
  "levelOne": {
    "type": "object",
    "properties": {
      "oneFirst": {
        "$ref": "#/$defs/levelTwo"
      },
      "oneSecond": {
        "type": "string"
      }
    }
  },
  "levelTwo": {
    "type": "object",
    "properties": {
      "twoFirst": {
        "type": "string"
      },
      "twoSecond": {
        "type": "string"
      }
```

3.6.1 制限事項

```
  "levelOne": {
    "type": "object",
    "properties": {
      "oneFirst": {
        "$ref": "#/$defs/levelTwo"
      },
      "oneSecond": {
        "type": "string"
      }
    }
  },
  "levelTwo": {
    "type": "object",
    "properties": {
      "twoFirst": {
        "type": "string"
      },
      "twoSecond": {
        "type": "string"
      }
```

3.6.1 制限事項

```
        }
      }
    }
}
```

3.6.2 拡張

- CCTS は拡張機能をサポートしていません。
- したがって、「制限」の章に類似した、カーディナリティ、値の範囲、列挙型 (enum) を拡張する NDR ルールを CCTS に対して設定することはできません。
- それでも実装にそのような拡張が必要な場合、その結果はこの技術仕様に従ったアーティファクトに準拠しなくなります。
- 技術的には、これはSchemaと anyOf を組み合わせることによって実現できます。
- ただし、特に OpenAPI 仕様を実装する場合は、プロパティの拡張が必要です。 たとえば、API エンドポイントにメタデータを追加します。

3.6.2 拡張

- [R 37|1]
- BasicComponents は、次のように拡張用の JSON サブSchemaを定義するものとします。

```
"$defs": {
  "extensibleType": {
  "patternProperties": {
    "^x-": true}
  }
}
```

- extensibleType を使用すると、ユーザーは独自の JSON プロパティを JSON サブSchemaに追加できます。
- 彼らが従わなければならない唯一のルールは、x- で始めなければならないということです。
- これにより、OpenAPI 仕様で定義された拡張メソッドに準拠します。
- 例は、ルール [R 42|1] の次のセクションにあります。

3.6.3 コンテキスト化の公開と再利用

- CCL は継続的に開発されています。
- このようにして、新しいバージョンでは使用されなくなった定義が含まれます。
- もう使用されなくなった公開データ タイプとの混乱を避けるため、RDM レベルは、UN/CEFACT 公開の最低のエクスポート レベルです。
- [R 38|1]
- すべての JSON Schema エクスポートのベースは RDM レベルである必要があります。
- これは、基礎となる各 CCL 基本データ型が、RDM 定義に従ってプロファイリングされ、コンテキスト化されるものとする (SHALL) ことを意味します。
- RDM で使用されるデータ型のみがエクスポートされるものとします (SHALL)。

- このセクションで定義されたルールが CCL 全体に適用される場合、結果として得られる JSON アーティファクトは複雑で非常に大きくなる可能性があります。
- このアプローチにより、制限の高レベルのトレーサビリティが実現され、個々のデータ型の一貫した (再) 利用が保証されます。
- ただし、API の実際のアプリケーションでは、これらのライブラリが不必要に大きくなる可能性があります。
- 特に、CCL のサブセットのみが使用されている場合はそうです。

3.6.3 コンテキスト化の公開と再利用

- したがって、必要な (サブ) 構造の「スナップショット」を JSON アーティファクトとしてエクスポートすると便利です。
- ここでの手順は、XML 設計原則「ベネチアン ブラインド」に対応しています。作成される JSON Schema ファイルは 1 つだけで、ユースケースに必要なデータ型がすべて含まれています。
- 必須ではないプロパティはすべてエクスポートされません。 制限は最小限に抑えられます。
- CCL への準拠は必須です。

- [R 39|2]
- ユーザー コミュニティは、CCL の特定のサブセットに対して「スナップショット」JSON Schema アーティファクトを作成することを決定する場合があります。
- 「スナップショット」JSON Schema アーティファクトには、サブセットの定義に必要なすべての関連データ型が含まれているものとします (SHALL)。
- 「スナップショット」JSON Schema アーティファクトには、追加の制限と拡張が含まれる場合があります。

- 「スナップショット」エクスポートと合わせて、JSON Schema アーティファクトを作成するには 3 つの方法が考えられます。

SON Schema アーティファクトを作成するには 3 つの方法

Export variant	説　明
ライブラリのエクスポート Library export	ライブラリのエクスポートでは、UN/CEFACT 標準で定義されているコンテキスト化のレベルごとに 1 つの JSON Schema ファイルが作成されます。 これにより、基盤として 1 つの大きな CCL JSON Schema表現が作成されます。 その上に、定義された RDM およびドキュメント中心の構造に CCL をコンテキスト化および制限する 1 つの JSON Schema ファイルが作成されます。 各レベルでは、そのレベルで正確に制限されている制限されたデータ型がすでに使用されている場合があります。このタイプのエクスポートを作成する場合は、これを考慮する必要があります。 Pro (賛) 完全な CCL、すべての RDM、およびすべての (ドキュメント中心の) メッセージ構造定義が、UN/CEFACT 標準の定義に従ってエクスポートされます。 再利用可能なデータ構造と定義が最大限に作成されます。 設計により、あらゆる実装に一貫性があり、あらゆるプロセス修正に対応できることが保証されます。 Contra (否) どの実装でも、UN/CEFACT 標準で定義されているマルチレイヤー制限だけでなく、ベース インポートとして巨大な CCL ライブラリを処理する必要があります。 たとえば、eCMR メッセージは、UN/CEFACT によって定義されたすべてのドキュメント中心のメッセージのマスター メッセージ構造のコンテキスト化として定義されます。 含まれるデータ構造は、マルチモーダル トランスポート参照データ モデルのプロセス固有のコンテキスト化です。さらに、これもまた、基礎となる CCL の文脈化です。 したがって、実装はかなり複雑になる可能性がありますが、同時に最大のコンプライアンス レベルを達成することもできます。

SON Schema アーティファクトを作成するには 3 つの方法

Export variant	説　明
サブセットのエクスポート Subset export	サブセットのエクスポートは、ライブラリのエクスポートと同じ原則に従いますが、大きな違いが 1 つあります。選択したサブセットの必要なデータ構造のみがエクスポートされます。 他のすべてのデータ構造は省略されます。 このようにして、ファイル サイズとコンテンツが最小限の情報セットに削減され、同時にすべての関係が利用可能な状態に保たれます。 あらゆるレベルの制限を考慮することが重要です。 すべてのレベルの制限を階層順に適用した結果のみが、結果として得られる技術的成果物に表されます。 Pro (賛) ライブラリ エクスポートで定義された引数に加えて、サブセット エクスポートは、ファイル サイズとデータ オブジェクトの量の点で処理が容易です。 Contra (否) コンテキスト化のレイヤーの複雑さは、ライブラリのエクスポートの場合と同じです。 サブセットを修正すると、基礎となるオブジェクトが変更されます。 特定のサブセットに必要なデータ オブジェクトのみがエクスポートされます。 将来のバージョンでサブセットの範囲が拡大されると、基礎となるデータ構造に追加のオブジェクトが必要になる可能性があります。 これは、サブセットの実装をすべてのプレーヤーで同時に更新する必要があることを意味します。

SON Schema アーティファクトを作成するには 3 つの方法

Export variant	説 明
スナップショットのエクスポート Snapshot export	内容的には、スナップショットのエクスポートはサブセットのエクスポートと同じです。 主な違いは、複数の JSON Schema ファイルのセットに対する多層コンテキスト化が削除されていることです。スナップショット オブジェクトの必要なデータ構造がすべて含まれる単一の JSON Schema ファイルが 1 つだけ作成されます。 これは、XML の「Ventian Blind」アプローチに相当します。 基礎となるデータ オブジェクトは依然として定義されています (パーティ データ タイプなど)。 ただし、スナップショット選択で使用されているSchema オブジェクトのみが含まれます。 Pro (賛) 特定のスナップショットの複雑さは最小限に抑えられます。 自己完結型の JSON Schema ファイルが 1 つだけ作成されます。 JSON Schema ファイルは、すべての一般的な JSON ツールおよび OpenAPI 設計ツールで簡単に使用できます。 エクスポートされたデータ構造は UN/CEFACT 標準に準拠しており、すべての制限とコンテキスト化の「編集」を反映しています。 Contra (否) 個々のスナップショットごとに 1 つの自己完結型 JSON Schema ファイルが作成されます。 このアプローチを事前定義された環境で使用すると、非常にうまく機能します。 したがって、スナップショットの内容を事前に明確に定義することが重要です。1 つの実装で複数の自己完結型 JSON Schema ファイルが使用される場合、事態は複雑になり始めます。 たとえば、ドキュメント中心のメッセージごとに 1 つの自己完結型 JSON Schema ファイルが作成されると仮定します (XML Schema ファイルの場合と同様)。 これらの各 JSON Schema ファイルは、基礎となるデータ型 (パーティなど) を定義しました。

SON Schema アーティファクトを作成するには 3 つの方法

Export variant	説　明
スナップショットのエクスポート Snapshot export	OpenAPI 仕様では、基礎となるデータ型間で競合が発生する可能性があるため、これらの複数の Schema ファイルを 1 つの OpenAPI ファイルに結合することはそれほど簡単ではありません。 その理由は、同じ名前の同じデータ型でも、異なる JSON Schema ファイル間で異なるコンテキストが存在する可能性があるためです。

- [R 40|1]
- UNECE 出版物は、公開されたアーティファクトにアクセスするグローバル コミュニティに必要な要件を処理できるサーバー上でライブラリ エクスポートを提供するものとします (SHALL)。
- さらに、UNECE は、コンテキスト化されたドキュメント ABIE ごとに追加のスナップショット エクスポートを提供する必要があります。

- [注記]
- JSON Schemaの $id プロパティは有効な URL を表す必要があるため、提供されるサービスのスケーラビリティの側面を考慮する必要があります。
- 1 つのオプションは、GIT 準拠のリポジトリでパブリケーションを提供することです。