Описание формата данных JSON 1С.

Содержание.

0. Общие понятия.

Согласно классификации объектов предметной области прикладного решения, которая была введена в широкую практику Мартином Фаулером и Эриком Эвансом, прикладные объекты 1С могут быть ссылочными (entity или reference object), значимыми (value object) и агрегатами (aggregate). Значимые типы данных далее будут называться табличными.

https://martinfowler.com/bliki/EvansClassification.html
https://martinfowler.com/bliki/ValueObject.html
https://martinfowler.com/bliki/DDD_Aggregate.html

К ссылочным объектам в 1С относятся справочники и документы. Справочники и документы, имеющие табличные части, образуют агрегаты. Ссылочные объекты в 1С идентифицируются при помощи UUID (RFC 4122).

Справочники хранят данные о редко изменяемых сущностях прикладного решения. Например, это могут быть сведения о номенклатуре, контрагентах, местах хранения товара и т.п.

Документы фиксируют события, которые происходят со справочными сущностями, изменяя таким образом их учётное состояние в количественном или качественном выражении. Например, это может быть изменение остатка товара на складе или изменение его статуса.

Только ссылочные объекты могут иметь табличные части. В табличных частях хранится, связанная с данным объектом неразрывным образом информация, образуя в таком случае агрегат. Например, это может быть список товаров документа "Приходная накладная" или "Заказ покупателя".

Хорошей практикой в 1С считается хранение состояния сущностей отдельно от них самих. Для этого используются табличные типы данных (value object).

К табличным типам данных в 1С относятся регистры сведений и регистры накопления. Управление данными этих объектов осуществляется при помощи наборов записей, которые, как правило, имеют составной ключ.

Регистры сведений используются для хранения любой информации, которая связана с теми или иными ссылочными объектами прикладного решения. Регистры сведений бывают периодическими, подчинёнными регистратору и независимыми. Первый тип регистров сведений хранит информацию в разрезе дат и времени. Второй — в разрезе документов. Третий — настраивается произвольным образом согласно решаемым прикладным решением задачам.

Регистры накопления являются учётными регистрами и хранят информацию о состоянии сущностей в количественном выражении. Регистры накопления бывают двух типов: остатков и оборотов. Например, регистр остатков "Остатки товара на складах" будет хранить записи о приходах и расходах товаров, а регистр оборотов "Продажи товара" будет хранить движения выбытия товаров в результате их продажи.

Данные в регистрах накопления всегда связанны с тем или иным документом. Такие документы по отношению к регистрам накопления называются регистраторами.

1. Формат JSON ссылочных типов данных.

Пример сериализованного объекта типа справочник "Валюты":

```
{
   "#type": "jcfg:CatalogObject.Валюты",
   "#value":
   {
      "Ref": "9c556d4d-720f-11df-b436-0015e92f2802",
      "DeletionMark": false,
      "Code": "840",
      "Description": "USD",
      "НаименованиеПолное": "Доллар США"
   }
}
```

Пример документа с табличной частью (массив "Товары"):

```
{
  "#type": "jcfg:DocumentObject.ЗаказКлиента",
  "#value": {
    "Ref": "0227135d-296e-11e5-92f1-0050568b35ac",
    "DeletionMark": false,
    "Date": "2015-07-13T18:46:57",
    "Number": "ТД00-000028",
    "Posted": true,
    "Контрагент": "2f5f7e5d-f873-11df-aecd-0015e9b8c48d",
    "Валюта": "26093579-c180-11e4-a7a9-000d884fd00d",
    "СуммаДокумента": 117625,
    "Автор": {
      "#type": "jcfg:CatalogRef.Пользователи",
      "#value": "a4212b3d-730a-11df-b338-0011955cba6b"
    },
```

```
    "Товары": [
      {
        "Номенклатура": "bd72d927-55bc-11d9-848a-00112f43529a",
        "Количество": 3,
        "Цена": 6750,
        "Сумма": 20250
      }
    ]
  }
}
```

Для обмена данными об удаляемых ссылочных объектах используется следующая сериализация JSON:

```
{
  "#type": "jent:ObjectDeletion",
  "#value": {
    "Ref": {
      "#type": "jcfg:CatalogRef.Валюты",
      "#value": "9c556d4d-720f-11df-b436-0015e92f2802"
    }
  }
}
```

Системные свойства справочников:

Имя	Синоним	Тип данных	Описание
Ref	Ссылка (обязательный)	UUID	Ссылка на данный объект в базе данных
DeletionMark	ПометкаУдаления (обязательный)	boolean	Признак удаления объекта
Owner	Владелец (необязательный)	Ссылка	Внешний ключ на объект справочника, который "владеет" данным объектом
Code	Код (необязательный)	number string	Код объекта
Description	Наименование (необязательный)	string	Наименование объекта
Parent	Родитель (необязательный)	Ссылка	Родительский объект для данного объекта. Таким образом реализуется иерархия объектов одного и того же типа.
IsFolder	ЭтоГруппа (необязательный)	boolean	Вид родительского объекта: группа или объект.

Системные свойства документов:

Имя	Синоним	Тип данных	Описание

Ref	Ссылка (обязательный)	UUID	Ссылка на данный объект в базе данных
DeletionMark	ПометкаУдаления (обязательный)	boolean	Признак удаления объекта
Date	Дата (обязательный)	string (date-time) ISO 8601	Дата и время документа
Number	Номер (необязательный)	number string	Номер документа
Posted	Проведен (обязательный)	boolean	Признак принятия документа к учёту

2. Концепция ссылки на объект.

1С широко оперирует понятием "Ссылка". Ссылка - это указатель на ссылочный объект определённого типа данных. Идентификатором ссылки является UUID. Например, в документации по 1С можно часто увидеть описание ссылки таким образом: СправочникСсылка.Номенклатура. В выше приведённых примерах JSON ссылками являются, например, свойства "Валюта" и "Автор" документа "ЗаказКлиента".

В формате JSON 1С часто можно увидеть следующие имена для ссылочных типов данных:
jcfg:EnumRef.СпособОплаты — перечисление "Способ оплаты"
jcfg:CatalogRef.Валюты - справочник "Валюты"
jcfg:DocumentRef.ЗаказКлиента — документ "Заказ клиента"

Для объектов переноса данных (data transfer object) используются аналогичное именование:
jcfg:CatalogObject.Валюты - DTO справочника "Валюты"
jcfg:DocumentObject.ЗаказКлиента — DTO документа "Заказ клиента"
jent:ObjectDeletion - DTO удаления объекта (см. JSON выше)

Ссылки на объекты бывают "пустыми". Дело в том, что в 1С значением ссылки не может быть значение null. Вместо этого используется понятие "пустая ссылка". В формате JSON это выглядит так:

```
{
  "Валюта": "00000000-0000-0000-0000-000000000000"
}
```

или так:

```
{
  "Автор": {
    "#type": "jcfg:CatalogRef.Пользователи",
    "#value": "00000000-0000-0000-0000-000000000000"
  }
}
```

Такая ссылка имеет тип данных и значение нулевого UUID.

3. Концепция составного типа данных.

1С поддерживает концепцию множественных типов данных (multiple type). Для этого платформа вводит такое понятие как "составной тип данных". Значение такого типа может в каждый момент времени иметь разный тип данных. Например, оно может быть строкой, числом или ссылкой на объект.

Для сравнения в языке программирования C++ такой тип данных является экземпляром класса std::variant или объединением (union).

Рассмотрим реализацию этой концепции подробнее. Как следует из примера выше, значения ссылок могут кодироваться в JSON как простой UUID:

```
{
  "Валюта": "26093579-c180-11e4-a7a9-000d884fd00d"
}
```

или как объект со свойствами #type и #value:

```
{
  "Автор": {
    "#type": "jcfg:CatalogRef.Пользователи",
    "#value": "a4212b3d-730a-11df-b338-0011955cba6b"
  }
}
```

Во втором случае очень просто понять какого типа ссылка закодирована в качестве значения свойства "Автор". В первом случае о типе значения остаётся только догадываться.

Это происходит потому, что при сериализации объекта в JSON 1С учитывает имеет ли данное свойство множественный тип данных или одиночный (single type value). Во втором случае свойство "Автор" имеет составной тип данных 1С, а в первом — одиночный. В тех случаях, когда тип значения ссылки известен платформе 1С (для этого есть метаданные) и может быть только одного типа, платформа сериализует значение ссылки как простой UUID.

При десериализации одиночного значения ссылки узнать тип этой ссылки можно только из метаданных 1С. Для внешней системы эти метаданные можно выгрузить из 1С в виде XSD схемы. При помощи сторонних утилит можно по метаданным 1С генерировать JSON схему.

Таким образом, сериализация любого значения множественного типа данных (примитивного или ссылочного) выглядит следующим образом:

```
{
  "#type": "jcfg:CatalogRef.Пользователи",
  "#value": "a4212b3d-730a-11df-b338-0011955cba6b"
```

```
}
```

Пример для примитивного типа данных:

```
{
  "#type": "jxs:string",
  "#value": "это строковое значение"
}
```

Возможные значения свойств #type и #value:

#type	#value
jxs:string	"строковое значение"
jxs:decimal	123 или 123.45
jxs:boolean	true или false
jxs:dateTime	"2021-01-01T00:00:00"
jcfg:EnumRef.ИмяПеречисления	"НаличнаяОплата" (*)
jcfg:CatalogRef.ИмяСправочника	"a4212b3d-730a-11df-b338-0011955cba6b"
jcfg:DocumentRef.ИмяДокумента	"a4212b3d-730a-11df-b338-0011955cba6b"

(*) Не смотря на то, что перечисления в 1С это ссылочные типы данных, значением перечисления в JSON 1С всегда является строка - имя для значения UUID в базе данных.

Составной тип данных является обнуляемым типом данных, то есть его значением может быть значение null. Например:

```
{
  "Автор": null
}
```

4. Формат JSON значимых типов данных.

Пример для регистра сведений "КурсыВалют":

```
{
  "#type": "jcfg:InformationRegisterRecordSet.КурсыВалют",
  "#value": {
    "Filter": [
      {
        "Name": {
          "#type": "jxs:string",
          "#value": "Валюта"
        },
        "Value": {
          "#type": "jcfg:CatalogRef.Валюты",
          "#value": "9c556d4d-720f-11df-b436-0015e92f2802"
        }
      }
    ],
    "Record": [
      {
```

```
        "Period": "2021-07-13T00:00:00",
        "Валюта": "9c556d4d-720f-11df-b436-0015e92f2802",
        "Курс": 82.55
      },
      {
        "Period": "2021-07-14T00:00:00",
        "Валюта": "9c556d4d-720f-11df-b436-0015e92f2802",
        "Курс": 81.3
      },
      {
        "Period": "2021-07-15T00:00:00",
        "Валюта": "9c556d4d-720f-11df-b436-0015e92f2802",
        "Курс": 80.45
      }
    ]
  }
}
```

Пример для регистра накопления "Расчёты с клиентами":

```
{
  "#type": "jcfg:AccumulationRegisterRecordSet.РасчетыСКлиентами",
  "#value": {
    "Filter": [
      {
        "Name": {
          "#type": "jxs:string",
          "#value": "Recorder"
        },
        "Value": {
          "#type": "jcfg:DocumentRef.ЗаказКлиента",
          "#value": "0227135d-296e-11e5-92f1-0050568b35ac"
        }
      }
    ],
    "Record": [
      {
        "Recorder": {
          "#type": "jcfg:DocumentRef.ЗаказКлиента",
          "#value": "0227135d-296e-11e5-92f1-0050568b35ac"
        },
        "Period": "2015-07-18T23:59:59",
        "RecordType": "Receipt",
        "Active": true,
        "ЗаказКлиента": {
          "#type": "jcfg:DocumentRef.ЗаказКлиента",
          "#value": "0227135d-296e-11e5-92f1-0050568b35ac"
        },
        "Валюта": "26093579-c180-11e4-a7a9-000d884fd00d",
        "Сумма": 0,
        "ФормаОплаты": "ПлатежнаяКарта"
      }
    ]
```

```
  }
}
```

Системные свойства регистра сведений:

Имя	Синоним	Тип данных	Описание
Period	Период (необязательный)	string (date-time) ISO 8601	Дата и время записи. Используется только для периодических регистров сведений.
Recorder	Регистратор (необязательный)	Ссылка	Ссылка на документ, "владеющий" данной записью. Используется только для регистров сведений подчинённых регистратору.
Active	Активность (необязательный)	boolean	Признак использования записи для учёта. Может использоваться, например, для разделения фискального и управленческого учёта.

Системные свойства регистров накопления остатков и оборотов:

Имя	Синоним	Тип данных	Описание
Period	Период (обязательный)	string (date-time) ISO 8601	Дата и время записи. Как правило совпадает с датой документа.
Recorder	Регистратор (обязательный)	Ссылка	Ссылка на документ, который выполнил регистрацию движений — создал данную запись.
Active	Активность (обязательный)	boolean	Признак использования записи для учёта. Может использоваться, например, для разделения фискального и управленческого учёта.
RecordType	ВидДвижения (необязательный)	string (enum)	"Receipt" — приход "Expense" - расход Не используется для регистров накопления оборотов.

Как следует из примеров структура набора записей в формате JSON 1С выглядит следующим образом:

```
{
  "#type": "jcfg:AccumulationRegisterRecordSet.ИмяРегистра",
  "#value": {
```

```
    "Filter": [ { /* ... */ } ],
    "Record": [ { /* ... */ } ]
  }
}
```

Свойство #type нам уже знакомо. Его использование в данном случае ровно такое же, как в случае ссылочных типов данных, то есть определение типа объекта переноса данных (data transfer object).

Это свойство может иметь следующий вид:
jcfg:InformationRegisterRecordSet.ИмяРегистраСведений
jcfg:AccumulationRegisterRecordSet.ИмяРегистраНакопления

Свойство #value создержит значение типа "набор записей".

Свойство Record это массив записей. Каждая запись это объект и её кодирование интуитивно понятно. Оно аналогично кодированию объектов ссылочного типа.

Свойство Filter это массив объектов типа "Элемент отбора". Каждый элемент отбора имеет свойства Name и Value. Свойство Name указывает на свойство записи, по которому выполняется отбор данных для набора записей. В свою очередь свойство Value содержит значение для соответствующего элемента отбора. По этому значению всегда выполняется фильтрация данных на равенство. Все элементы отбора объединяются между собой при помощи логического оператора AND.

При этом следует иметь ввиду, что свойство Filter может отсутствовать или иметь в качестве своего значения пустой массив. Это будет означать, что отбор не используется. То же самое можно сказать про свойство Record.

5. Концепция набора записей.

1С выполняет модифицирование табличных типов данных при помощи наборов записей. Набор записей включает в себя отбор и сами записи. Алгоритм обработки набора записей платформой 1С выглядит следующим образом:
1. Выполняется удаление записей по отбору.
2. Выполняется вставка записей.

Соответственно, если отбор не используется, то удаляются все записи как если бы в выражении SQL DELETE не было бы указано предложение WHERE.

Каждый табличный тип данных имеет уникальный составной ключ, идентифицирующий запись в таблице СУБД. Таким образом отбор может быть полным или неполным.

Полным отбором является такой отбор, который включает в себя значения для всех свойств ключа, то есть идентифицирует одну запись.

Неполным отбором является такой отбор, который устанавливает значения только для части свойств ключа, то есть потенциально идентифицирует несколько записей.

Если отбор используется в наборе записей, то его значения для соответствующих свойств записей набора должны быть идентичны.

Таким образом, в случае с табличными типами значений, в отличие от ссылочных, объект переноса данных типа "Удаление объекта" не используется (см. раздел № 1). Вместо этого используется отбор набора записей.

6. Метаданные, схема данных прикладного решения 1С.

Для описания структур данных объектов прикладного решения 1С используются метаданные. Метаданные являются схемой данных прикладного решения. Частота изменения схемы данных зависит от интенсивности разработки прикладного решения.

1С имеет штатные средства для выгрузки схемы данных в формате XSD. Для выгрузки схемы данных в формате, например, JSON Schema требуется использование сторонних утилит.

Пример схемы XSD для документа "Заказ клиента":

```
<objectType name="DocumentObject.ЗаказКлиента">
     <property name="Ref" type="DocumentRef.ЗаказКлиента"/>
     <property name="DeletionMark" type="xs:boolean"/>
     <property name="Date" type="xs:dateTime"/>
     <property name="Number" type="xs:string"/>
     <property name="Posted" type="xs:boolean"/>
     <property name="Контрагент" type="CatalogRef.Контрагенты"/>
     <property name="Валюта" type="CatalogRef.Валюты"/>
     <property name="СуммаДокумента" type="xs:decimal"/>
     <property name="Автор" nillable="true"/>
     <property name="Товары"
          type="DocumentTabularSectionRow.ЗаказКлиента.Товары"
          lowerBound="0" upperBound="99999"/>
</objectType>

<objectType name="DocumentTabularSectionRow.ЗаказКлиента.Товары">
     <property name="Номенклатура" type="CatalogRef.Номенклатура"/>
     <property name="Количество" type="xs:decimal"/>
     <property name="Цена" type="xs:decimal"/>
     <property name="Сумма" type="xs:decimal"/>
</objectType>
```

Пример JSON Schema для документа "Заказ клиента":

```
{
  "$id": "Документ.ЗаказКлиента",
  "type": "object",
  "additionalProperties": false,
  "required": [ "#type", "#value" ],
  "properties": {
    "#type": { "const": "jcfg:DocumentObject.ЗаказКлиента" },
    "#value": {
      "type": "object",
      "properties": {
        "Ref": { "type": "string", "format": "uuid" },
        "DeletionMark": { "type": "boolean" },
        "Date": { "type": "string", "format": "date-time" },
        "Number": { "type": "string", "maxLength": 9 },
        "Posted": { "type": "boolean" },
        "Контрагент": { "$ref": "Справочник.Контрагенты" },
        "Валюта": { "$ref": "Справочник.Валюты" },
        "СуммаДокумента": { "type": "number" },
        "Автор": {
          "oneOf": [
            { "type": "null" },
            { "type": "object", "required": [ "#type", "#value" ],
              "properties": {
                "#type": { "type": "string" },
                "#value": {"type": ["string","number","boolean"] }
              }
            }
          ]
        },
        "Товары": { "type": "array", "items": { "type": "object",
            "properties": {
              "Номенклатура": {"$ref": "Справочник.Номенклатура"},
              "Количество": { "type": "number" },
              "Цена": { "type": "number" },
              "Сумма": { "type": "number" }
            }
          }
        }
      }
    }
  }
}
```