GETTING STARTED WITH THE MISMO API TOOLKIT

How can the MISMO API Toolkit help organizations adopt the microservices architecture style?

To accelerate the development of new Mortgage APIs, MISMO has developed the API Toolkit. It supports small, independent services that can be composed into larger transactions. The Toolkit provides a fast, efficient way to develop APIs down to a microservice level, resulting in shorter time-to-market and improved interoperability. The core offering in this toolkit is a set of standalone OpenAPI objects defined as individual YAML files (for example, Address.yaml, Borrower.yaml, and Asset.yaml). These YAML files can be used as building blocks to define an API. No single schema with a pre-defined hierarchy is provided. The toolkit offers the flexibility to design data structures appropriate for a specific business use case.

The Use Case

This article will illustrate how to use the API Toolkit to design the schema for a simple use case. Future articles will build on this and provide examples for a full OpenAPI specification.

To get started, we will focus exclusively on the API request for the following use case.

"As a product owner, I need to enable external clients to submit basic borrower information, such as their legal name and date of birth, so that I can determine if the borrower can be pre-approved for a mortgage."

Designing the schema

This use case requires the borrower first name, middle name, last name, birthdate, and type (Primary/Secondary). The schema design should use an optimal hierarchy depth, eliminating unnecessary hierarchy where possible and retaining hierarchy when required to express the intent of the interface and to align with MISMO guidelines.

Step 1 – Identify the core entities: The YAML files containing the required datapoints can be identified by reviewing the Enhanced Logical Data Dictionary or searching the contents of the provided YAML files. In this case, Name.yaml and Borrower.yaml contain all the necessary datapoints.

Step 2 – Determine the desired level of MISMO compatibility:

According to the current MISMO model, ***Name*** is an entity that can be used across different parties – hence it is modeled separately. We will provide two separate ways to model the borrower to illustrate the levels of MISMO compatibility.

- **Approach 1: Schema Aligned with the MISMO Model (naming and structure)**

The first schema will follow the naming and structure within the MISMO model. Then, the flattening rules outlined in the MISMO API Toolkit Implementation Guide will be applied. Hierarchy will be retained to reflect the cardinality in the MISMO model and eliminate ambiguity for entities that exists under various parents within the model (for example, “Single-Path Rule”).

- **Approach 2: Schema Aligned with the MISMO Terms and Definitions (naming only)**
  The second schema will introduce a custom structure that leverages the MISMO vocabulary. It will merge the ***Name*** and ***Borrower*** entities into a single ***Borrower*** entity to produce a simpler and more intuitive schema. It will use the appropriate MISMO terms and definitions but will not adhere to the structure of the MISMO model.

Both schemas will enforce necessary constraints at the entity and property level, as required by the use case. Either approach is supported by the MISMO API Toolkit. The tradeoffs of each approach are discussed in this article. Teams must decide which approach best fits their use case.

Building the schema

Step 1 - Define the base types:

Borrower.yaml and Name.yaml define the entities in their entirety with all the necessary properties. The properties that are not needed for this use case will be eliminated. For simplicity, the definitions will be combined into a single file, rather than using external references.

Step 2 – Combine the entities and flatten the hierarchy:

SCHEMA ALGINED WITH THE MISMO MODEL

1. First, we will identify the common parents for **Borrower** and **Name** within the MISMO Model. The following paths apply for this use case:
   ```
   party.individual.name
   party.roles[].borrower
   ```
   Both **Name** and **Borrower** appear under **Party**, so **parties** will be the root for this schema.
2. Apply the Single-Path Rule - The MISMO API Toolkit defines two flattening rules to eliminate unnecessary hierarchy. The Single-Path Rule states that implementers may flatten the hierarchy when there is exactly one path to the datapoint in the MISMO model. Name appears underneath multiple containers (for example, Contact, Alias, Individual, etc.), so this schema will retain the hierarchy required to indicate that the name belongs to an individual. A single path exists for borrower, so the hierarchy can be flattened if it does not violate the Cardinality rule below.
3. Apply the Cardinality Rule – All API objects below the root must use the cardinality in the MISMO model. Roles is plural within the model, and it is not the root object. So, the hierarchy for borrower cannot be flattened.

```
Party:
  type: object
  properties:
    roles:
      type: array
      items:
```

Contents of **Party.yaml**. Only individuals are needed for this use case. Legal entities will not be supported by the API.

```
      $ref: "#/definitions/Role"
    individual:
     $ref: "#/definitions/Individual"
```

```
Role:
  type: object
  properties:
    borrower:
     $ref: "#/definitions/Borrower"
```

Contents of **Role.yaml.**

```
Borrower:
  type: object
  properties:
    borrowerBirthDate:
     $ref: "#/definitions/MISMODate"
    borrowerClassificationType:
     - type: string
       enum:
        - Primary
        - Secondary
```

Contents of **Borrower.yaml.**

```
Individual:
  type: object
  properties:
    name:
     $ref: "#/definitions/Name"
```

Contents of **Individual.yaml**.

```
Name:
  type: object
  properties:
    firstName:
     $ref: "#/definitions/MISMOString50"
    lastName:
     $ref: "#/definitions/MISMOString50"
    middleName:
     $ref: "#/definitions/MISMOString50"
```

Contents of **Name.yaml**.

```
MISMOString50:
  type: string
  maxLength: 50
```

Custom type to restrict the length to 50 characters.

Custom data types are defined to restrict the length of the strings (**MISMOString50**) as depicted above. **MISMOString50** restricts the length to 50 characters. Only the length is restricted, so **MISMOString** is used as the base name for the new type.

Notice the JSON request produced by the schema. The structure aligns with the MISMO model. It allows parties to play multiple roles and clearly identifies the name attributes as the individual's legal name. However, it also includes multiple levels in the hierarchy which introduces complexity.

```
{
   "parties": [
   {
      "roles": [
      {
         "borrower":
         {
            "borrowerBirthDate": "1979-01-01",
            "borrowerClassificationType": "Primary"
         }
      }],
      "individual":
      {
         "name":
         {
            "firstName": "John",
            "lastName": "Cox",
            "middleName": "Henry"
         }
      }
   },
   {
      "roles": [
      {
         "borrower":
         {
            "borrowerBirthDate": "1980-11-11",
            "borrowerClassificationType": "Secondary"
         }
      }],
      "individual":
      {
         "name":
         {
            "firstName": "Jane",
            "lastName": "Cox"
         }
      }
   }]
}
```

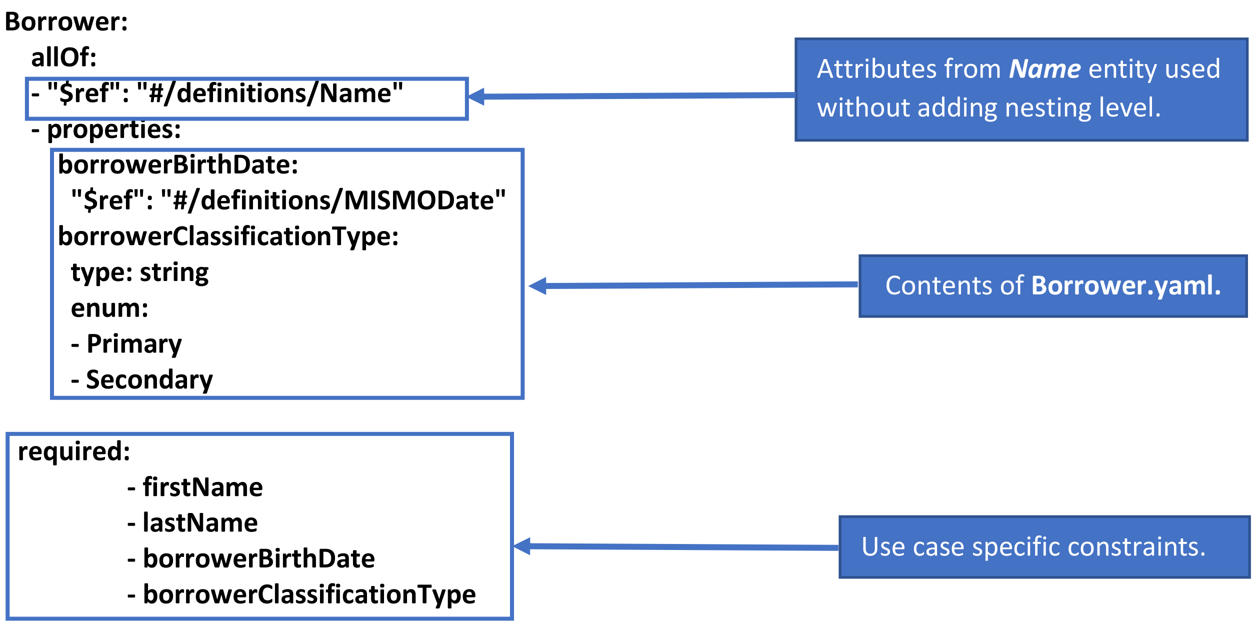
1. Parties is the root of the document.
2. There are multiple paths to *Name* in the MISMO model, so the hierarchy is retained to eliminate ambiguity.
3. *Roles* is plural, so an array is used to retain cardinality.

SCHEMA ALGINED WITH MISMO TERMS AND DEFINITIONS

For the minimum level of MISMO Compatibility, a composite ***Borrower*** object will include properties from Borrower.yaml and Name.yaml. The MISMO datapoint names will be leveraged, however the hierarchy in the MISMO model will be eliminated to produce a flattened structure.

```
Name:
  type: object
  properties:
   firstName:
    "$ref": "#/definitions/MISMOString50"
   lastName:
    "$ref": "#/definitions/MISMOString50"
   middleName:
    "$ref": "#/definitions/MISMOString50"
```

References to custom data types.



```
Borrower:
  allOf:
  - "$ref": "#/definitions/Name"
  - properties:
      borrowerBirthDate:
        "$ref": "#/definitions/MISMODate"
      borrowerClassificationType:
        type: string
        enum:
        - Primary
        - Secondary

required:
          - firstName
          - lastName
          - borrowerBirthDate
          - borrowerClassificationType
```

Attributes from ***Name*** entity used without adding nesting level.
Contents of **Borrower.yaml.**
Use case specific constraints.

Notice the JSON request produced by the schema. The datapoint names are the same as the previous example. However, the nested objects (such as **Roles** and **Individual**) are eliminated producing a structure that is completely flattened. This streamlined API request results in the following consequences:

- A party cannot play multiple roles.
- The name attributes are not explicitly identified as the legal name for an Individual.

Software engineers need to consider the tradeoffs and choose the appropriate schema design for their use case. The nesting depth and context specific constraints are important considerations for transactional schema design.

```
{
   "Borrowers": [
     {
       "firstName": "John",
       "lastName": "Cox",
       "middleName": "Henry",
       "borrowerBirthDate": "1979-01-01",
       "borrowerClassificationType": "Primary",
     },
     {
       "firstName": "Jane",
       "lastName": "Cox",
       "borrowerBirthDate": "1980-01-01",
       "borrowerClassificationType": "Secondary",
     }
   ]
 }
```

1. Borrowers is the root.
2. Hierarchy is completely flattened.
3. Secondary borrower does not have the optional middle name.

The API Toolkit provides a fast, efficient way to develop MISMO aligned APIs down to a microservice level, resulting in shorter time-to-market and improved interoperability. The examples above can be expanded to suit any use case your entity might have. If you have any questions or comments, the API Community of Practice meets the 1st and 3rd Thursday of each month. We would love to hear from you.

Note: MISMO's API Toolkit is available to its members free of charge. Contact us at info@MISMO.org to learn about the benefits of MISMO membership.