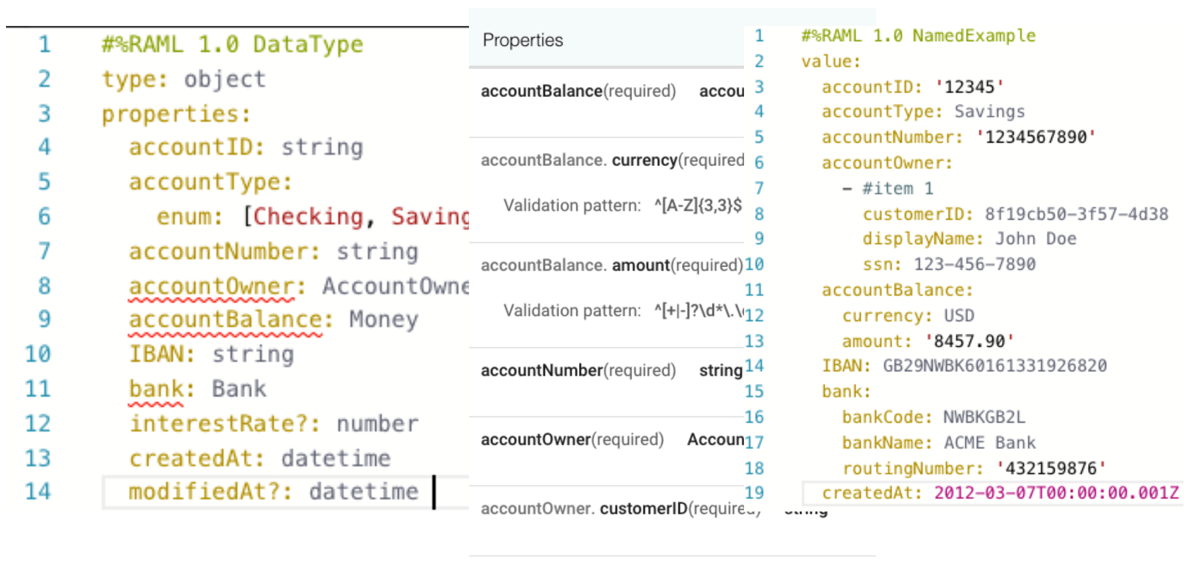
Module 6: Modeling Data

Goal



```
#%RAML 1.0 DataType
type: object
properties:
  accountID: string
  accountType:
    enum: [Checking, Saving
  accountNumber: string
  accountOwner: AccountOwne
  accountBalance: Money
  IBAN: string
  bank: Bank
  interestRate?: number
  createdAt: datetime
  modifiedAt?: datetime
```

Properties

accountBalance(required) accou

accountBalance. currency(required

Validation pattern: ^[A-Z]{3,3}$

accountBalance. amount(required)

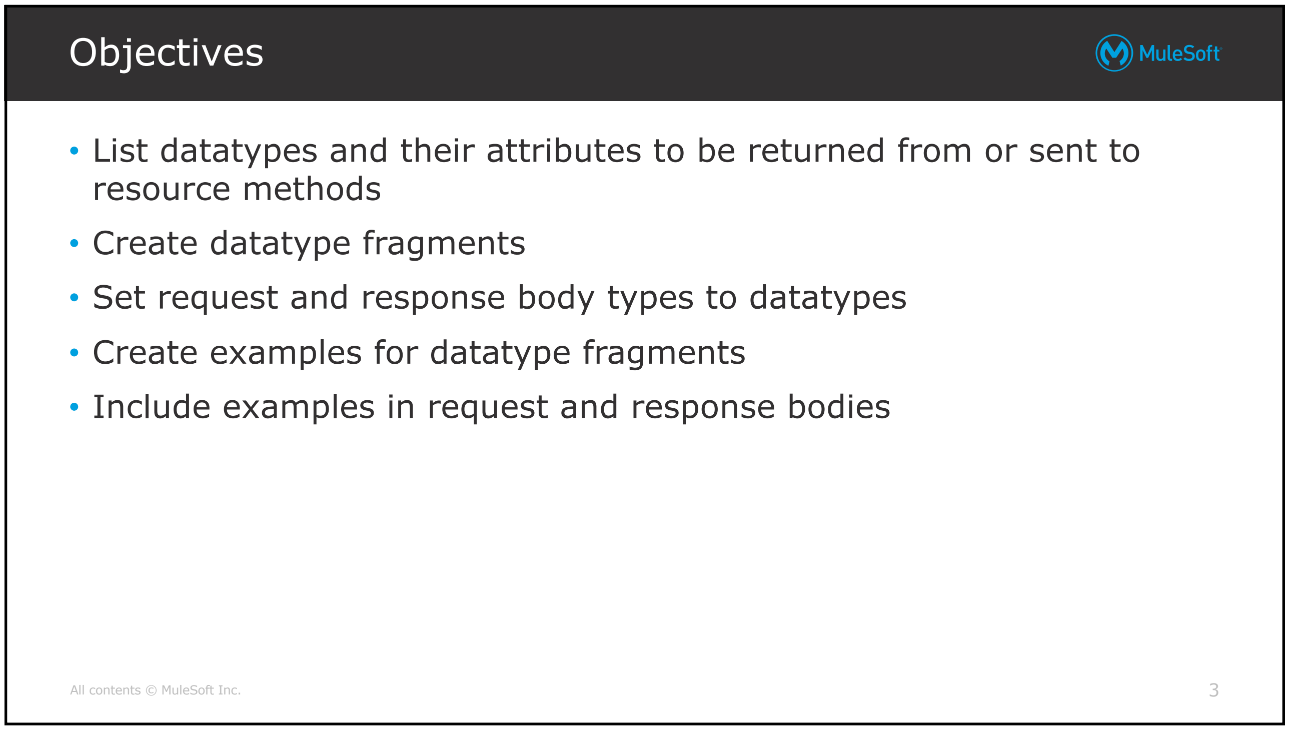
Validation pattern: ^[+|-]?\d*\.\

accountNumber(required) string

accountOwner(required) Accoun

accountOwner. customerID(require

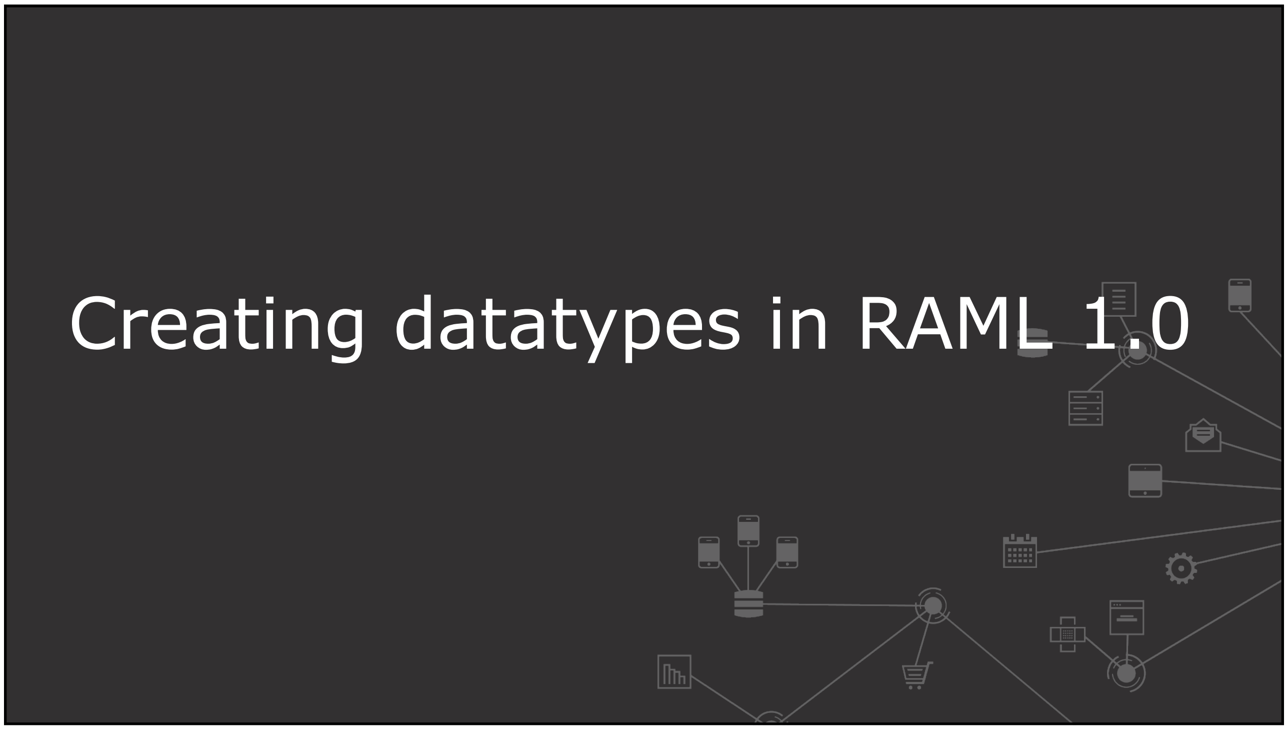
```
#%RAML 1.0 NamedExample
value:
  accountID: '12345'
  accountType: Savings
  accountNumber: '1234567890'
  accountOwner:
    - #item 1
      customerID: 8f19cb50-3f57-4d38
      displayName: John Doe
      ssn: 123-456-7890
  accountBalance:
    currency: USD
    amount: '8457.90'
  IBAN: GB29NWBK60161331926820
  bank:
    bankCode: NWBKGB2L
    bankName: ACME Bank
    routingNumber: '432159876'
  createdAt: 2012-03-07T00:00:00.001Z
```



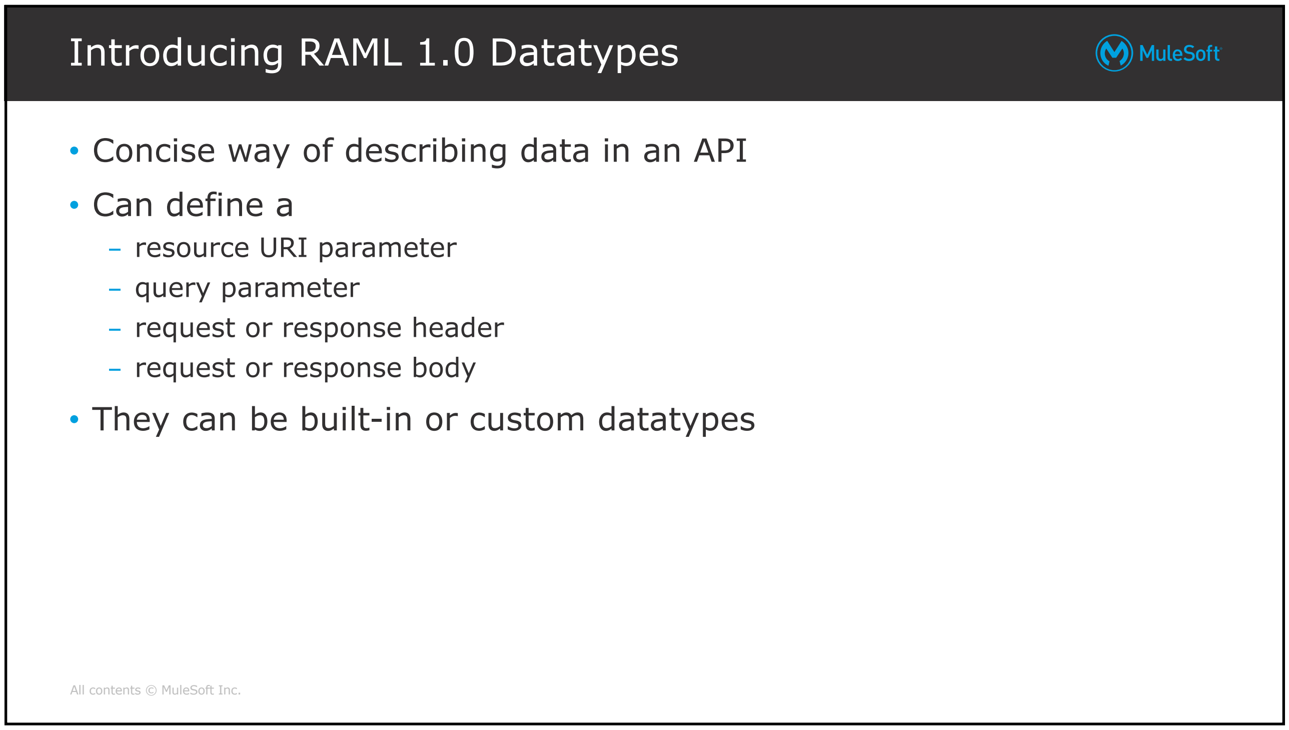
Objectives

MuleSoft

- List datatypes and their attributes to be returned from or sent to resource methods
- Create datatype fragments
- Set request and response body types to datatypes
- Create examples for datatype fragments
- Include examples in request and response bodies

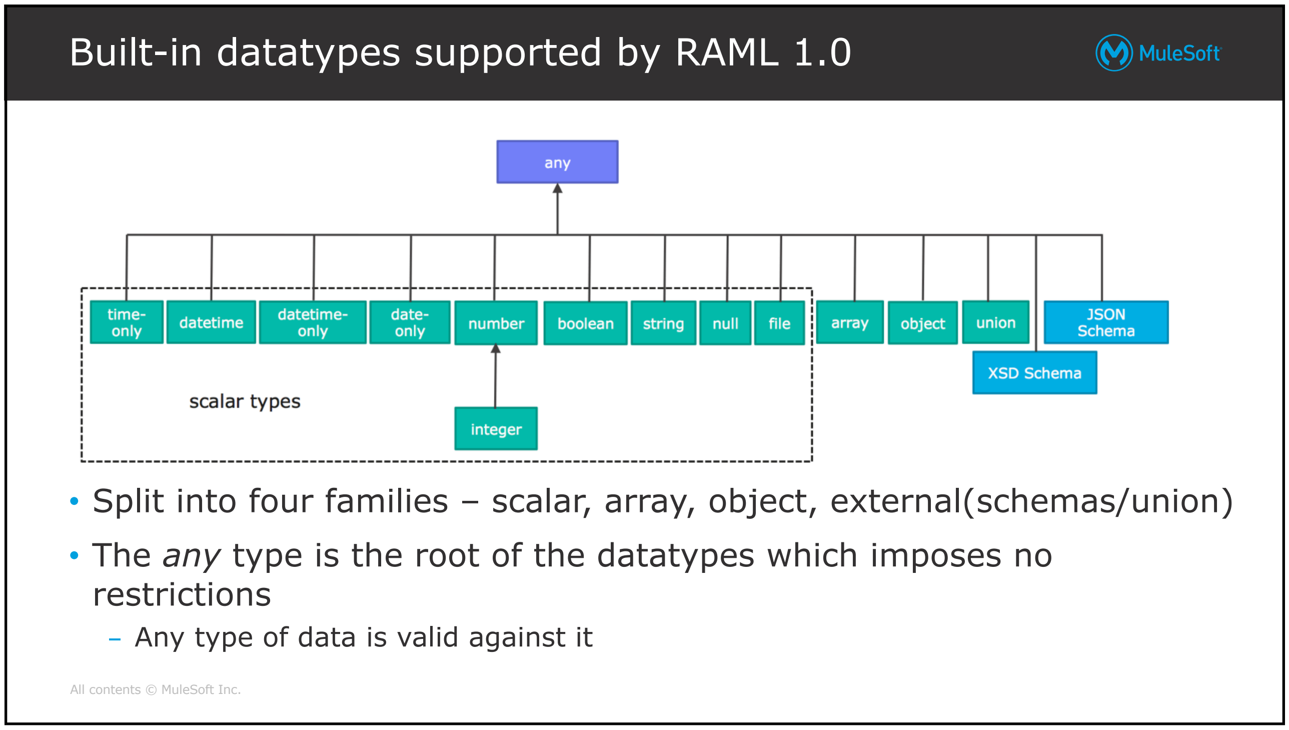


Creating datatypes in RAML 1.0

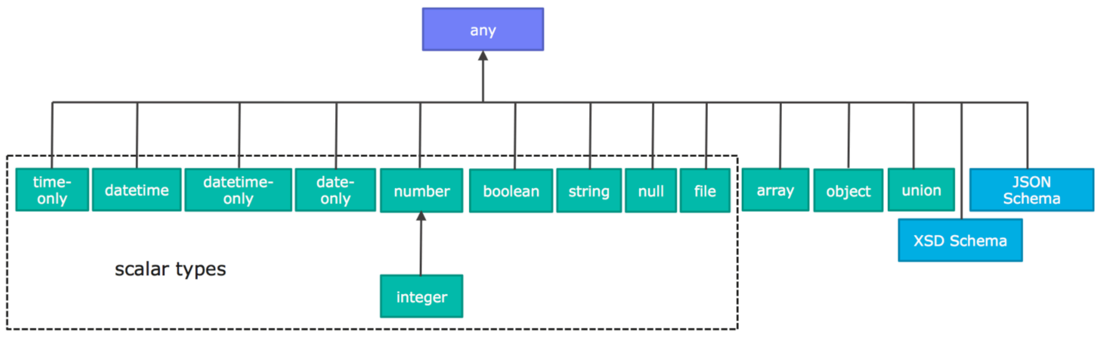


Introducing RAML 1.0 Datatypes

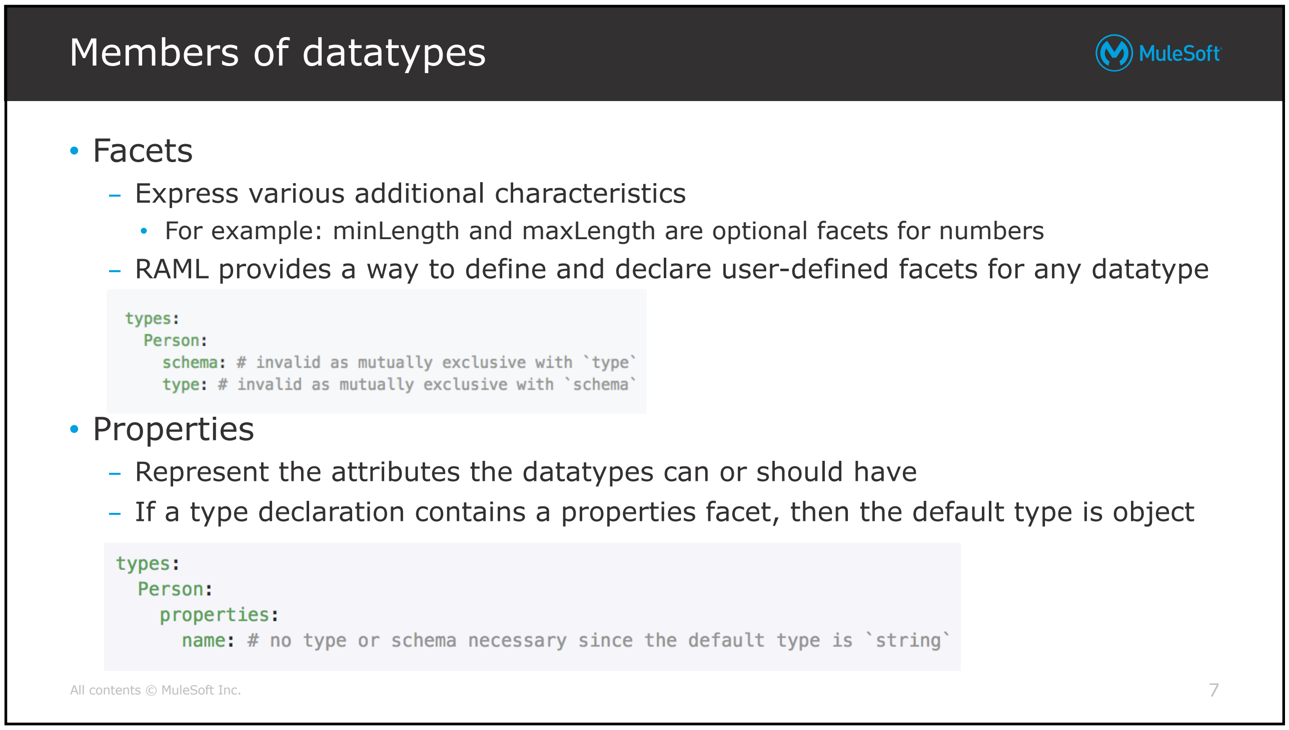
- Concise way of describing data in an API
- Can define a
  - resource URI parameter
  - query parameter
  - request or response header
  - request or response body
- They can be built-in or custom datatypes



Built-in datatypes supported by RAML 1.0



- Split into four families – scalar, array, object, external(schemas/union)
- The *any* type is the root of the datatypes which imposes no restrictions
  - Any type of data is valid against it



Members of datatypes

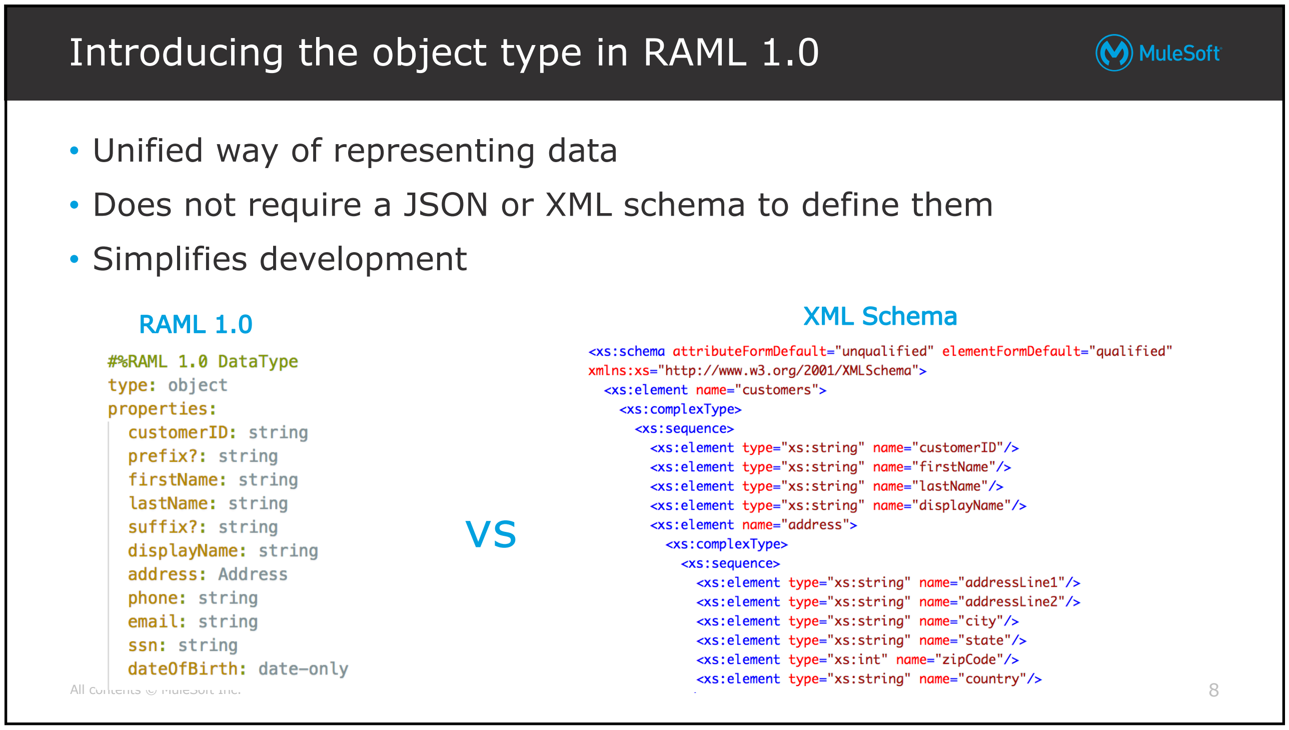
- Facets
  - Express various additional characteristics
    - For example: minLength and maxLength are optional facets for numbers
  - RAML provides a way to define and declare user-defined facets for any datatype

```
types:
  Person:
    schema: # invalid as mutually exclusive with `type`
    type: # invalid as mutually exclusive with `schema`
```

- Properties
  - Represent the attributes the datatypes can or should have
  - If a type declaration contains a properties facet, then the default type is object

```
types:
  Person:
    properties:
      name: # no type or schema necessary since the default type is `string`
```

7



Introducing the object type in RAML 1.0

- Unified way of representing data
- Does not require a JSON or XML schema to define them
- Simplifies development

RAML 1.0

```
#%RAML 1.0 DataType
type: object
properties:
  customerID: string
  prefix?: string
  firstName: string
  lastName: string
  suffix?: string
  displayName: string
  address: Address
  phone: string
  email: string
  ssn: string
  dateOfBirth: date-only
```

VS

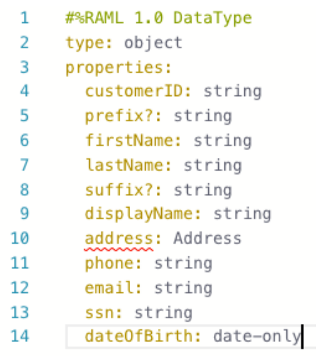
XML Schema

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="customers">
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:string" name="customerID"/>
        <xs:element type="xs:string" name="firstName"/>
        <xs:element type="xs:string" name="lastName"/>
        <xs:element type="xs:string" name="displayName"/>
        <xs:element name="address">
          <xs:complexType>
            <xs:sequence>
              <xs:element type="xs:string" name="addressLine1"/>
              <xs:element type="xs:string" name="addressLine2"/>
              <xs:element type="xs:string" name="city"/>
              <xs:element type="xs:string" name="state"/>
              <xs:element type="xs:int" name="zipCode"/>
              <xs:element type="xs:string" name="country"/>
```

8

Defining datatypes in RAML 1.0

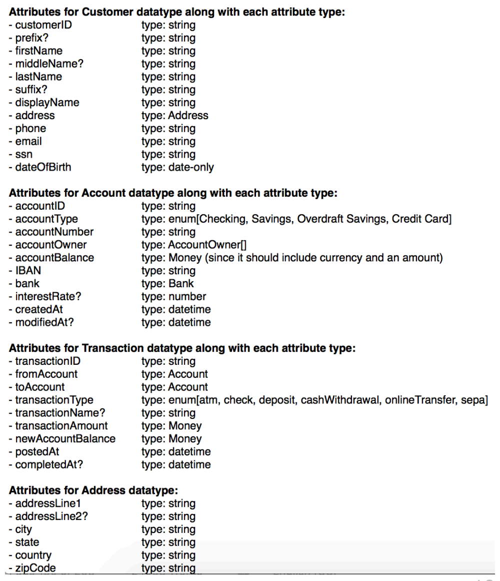
- Datatypes can be defined inline inside RAML 1.0 API Definition
- They can also be defined as a DataType fragment and included in the main API definition where they will be used
  - A fragment is a RAML document that lives outside the root RAML API definition
  - Helps break up the code into smaller reusable and readable components



```
#%RAML 1.0 DataType
type: object
properties:
  customerID: string
  prefix?: string
  firstName: string
  lastName: string
  suffix?: string
  displayName: string
  address: Address
  phone: string
  email: string
  ssn: string
  dateOfBirth: date-only
```

Walkthrough 6-1: List datatypes and their attributes for an API

- List the datatypes required for the resource methods
- Identify the attributes for each datatype
- Create necessary additional datatypes to simplify the identified datatypes
- Identify optional attributes in datatypes



Attributes for Customer datatype along with each attribute type:

- customerID	type: string
- prefix?	type: string
- firstName	type: string
- middleName?	type: string
- lastName	type: string
- suffix?	type: string
- displayName	type: string
- address	type: Address
- phone	type: string
- email	type: string
- ssn	type: string
- dateOfBirth	type: date-only

Attributes for Account datatype along with each attribute type:

- accountID	type: string
- accountType	type: enum[Checking, Savings, Overdraft Savings, Credit Card]
- accountNumber	type: string
- accountOwner	type: AccountOwner[]
- accountBalance	type: Money (since it should include currency and an amount)
- IBAN	type: string
- bank	type: Bank
- interestRate?	type: number
- createdAt	type: datetime
- modifiedAt?	type: datetime

Attributes for Transaction datatype along with each attribute type:

- transactionID	type: string
- fromAccount	type: Account
- toAccount	type: Account
- transactionType	type: enum[atm, check, deposit, cashWithdrawal, onlineTransfer, sepa]
- transactionName?	type: string
- transactionAmount	type: Money
- newAccountBalance	type: Money
- postedAt	type: datetime
- completedAt?	type: datetime

Attributes for Address datatype:

- addressLine1	type: string
- addressLine2?	type: string
- city	type: string
- state	type: string
- country	type: string
- zipCode	type: string

Walkthrough 6-2: Create datatype fragments

- Create individual datatype fragment files for the identified datatypes
- Define the datatypes with required and optional attributes
- Include datatype fragments in the main RAML API definition

```
#%RAML 1.0 DataType
type: object
properties:
  customerID: string
  prefix?: string
  firstName: string
  lastName: string
  suffix?: string
  displayName: string
  address: Address
  phone: string
  email: string
  ssn: string
  dateOfBirth: date-onl
```

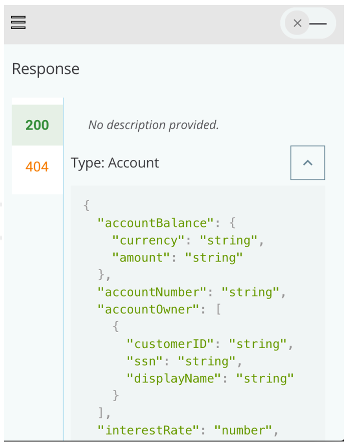
```
#%RAML 1.0
title: ACME Banking API
mediaType: application/json

types:
  Customer: !include datatypes/Customer.raml
  Address: !include datatypes/Address.raml
  Account: !include datatypes/Account.raml
  AccountOwner: !include datatypes/AccountOwner.raml
  Bank: !include datatypes/Bank.raml
  Money: !include datatypes/Money.raml
  Transaction: !include datatypes/Transaction.raml
  CustomErrorMessage: !include datatypes/CustomErrorMessage.raml
```

Walkthrough 6-3: Specify datatypes in resource methods

- Add the type parameter with a reference to the datatype for all resource method response bodies

```
    put:
      body:
      responses:
        200:
          body:
            type: Account
```


Response

200 404

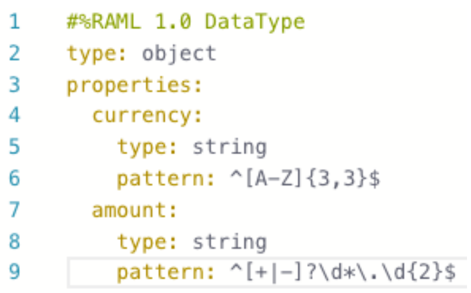
No description provided.

Type: Account

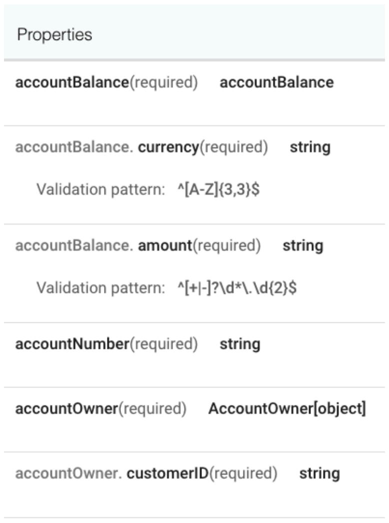
```
{
  "accountBalance": {
    "currency": "string",
    "amount": "string"
  },
  "accountNumber": "string",
  "accountOwner": [
    {
      "customerID": "string",
      "ssn": "string",
      "displayName": "string"
    }
  ],
  "interestRate": "number",
```

Walkthrough 6-4: Validate datatype attribute values using patterns

- Specify attribute patterns and format values for certain datatype attributes



```
#%RAML 1.0 DataType
type: object
properties:
  currency:
    type: string
    pattern: ^[A-Z]{3,3}$
  amount:
    type: string
    pattern: ^[+|-]?\d*\.\d{2}$
```



Properties

accountBalance(required)	accountBalance
accountBalance. currency(required)	string
Validation pattern:	^[A-Z]{3,3}$
accountBalance. amount(required)	string
Validation pattern:	^[+|-]?\d*\.\d{2}$
accountNumber(required)	string
accountOwner(required)	AccountOwner[object]
accountOwner. customerID(required)	string

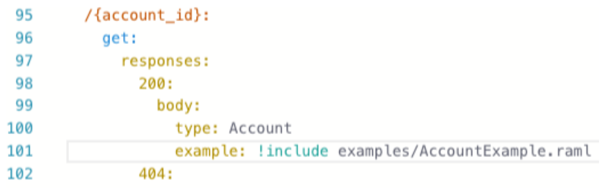
13

Specifying examples in RAML 1.0

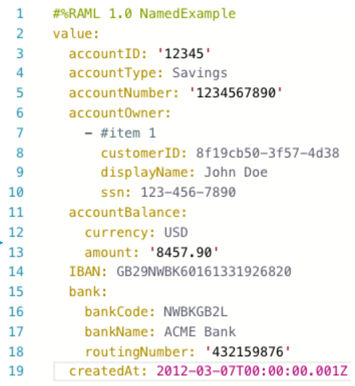
Introducing examples

MuleSoft

- Examples allow for complete documentation of the API
  - Leads to faster development time for API consumers
  - Promotes readability
  - Increases longevity and long-term usability



```
/{account_id}:
  get:
    responses:
      200:
        body:
          type: Account
          example: !include examples/AccountExample.raml
      404:
```

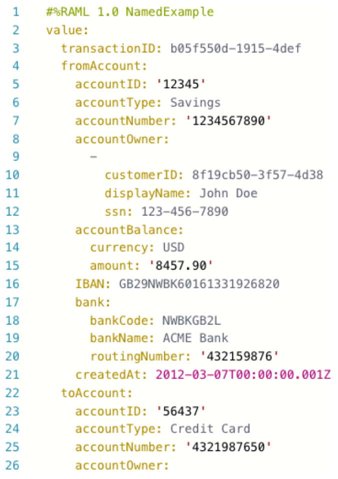


```
#%RAML 1.0 NamedExample
value:
  accountID: '12345'
  accountType: Savings
  accountNumber: '1234567890'
  accountOwner:
    - #item 1
      customerID: 8f19cb50-3f57-4d38
      displayName: John Doe
      ssn: 123-456-7890
  accountBalance:
    currency: USD
    amount: '8457.90'
  IBAN: GB29NWBK60161331926820
  bank:
    bankCode: NWBKGB2L
    bankName: ACME Bank
    routingNumber: '432159876'
  createdAt: 2012-03-07T00:00:00.001Z
```

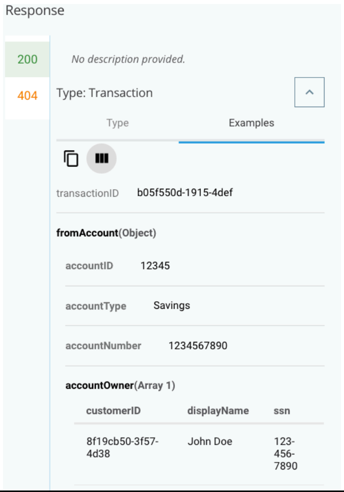
Walkthrough 6-5: Define example fragments for datatypes

MuleSoft

- Create example fragments for the datatypes
- Include example fragments in response bodies



```
#%RAML 1.0 NamedExample
value:
  transactionID: b05f550d-1915-4def
  fromAccount:
    accountID: '12345'
    accountType: Savings
    accountNumber: '1234567890'
    accountOwner:
      -
        customerID: 8f19cb50-3f57-4d38
        displayName: John Doe
        ssn: 123-456-7890
    accountBalance:
      currency: USD
      amount: '8457.90'
    IBAN: GB29NWBK60161331926820
    bank:
      bankCode: NWBKGB2L
      bankName: ACME Bank
      routingNumber: '432159876'
    createdAt: 2012-03-07T00:00:00.001Z
  toAccount:
    accountID: '56437'
    accountType: Credit Card
    accountNumber: '4321987650'
    accountOwner:
```



Response

200 *No description provided.*

404 Type: Transaction

Type | Examples

transactionID b05f550d-1915-4def

fromAccount(Object)

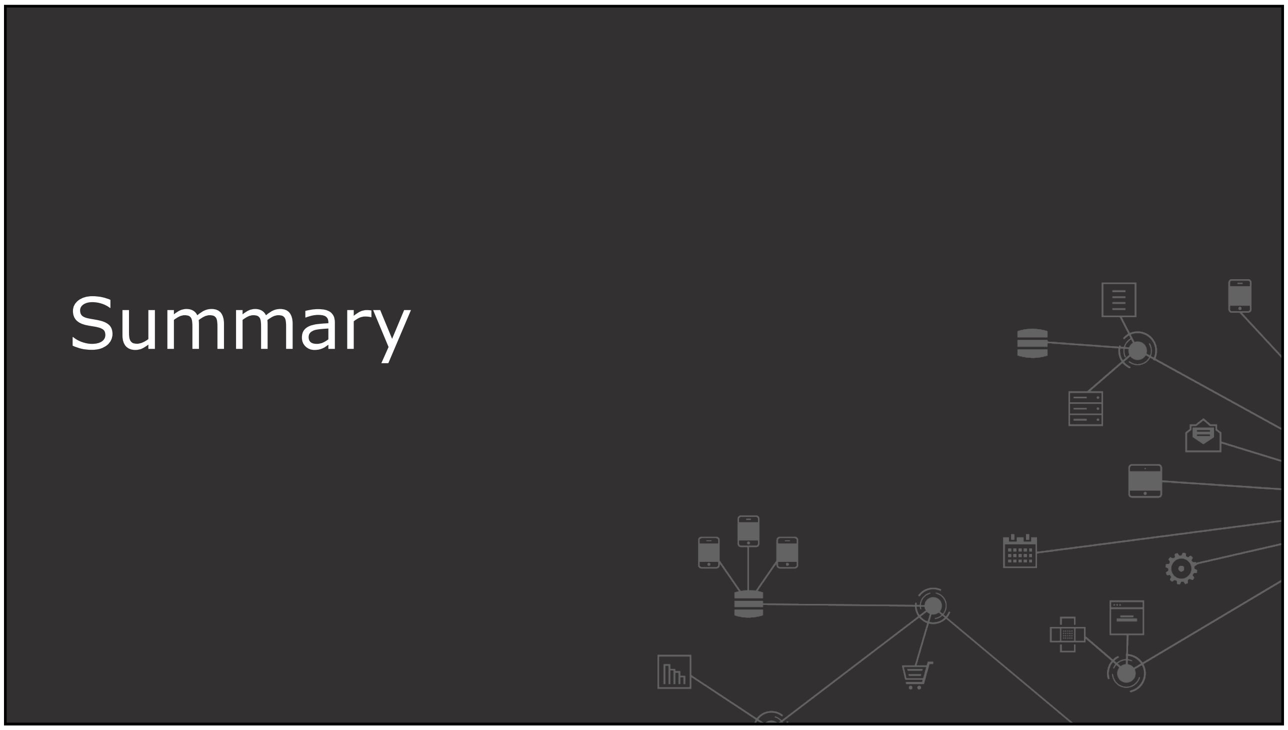
accountID 12345

accountType Savings

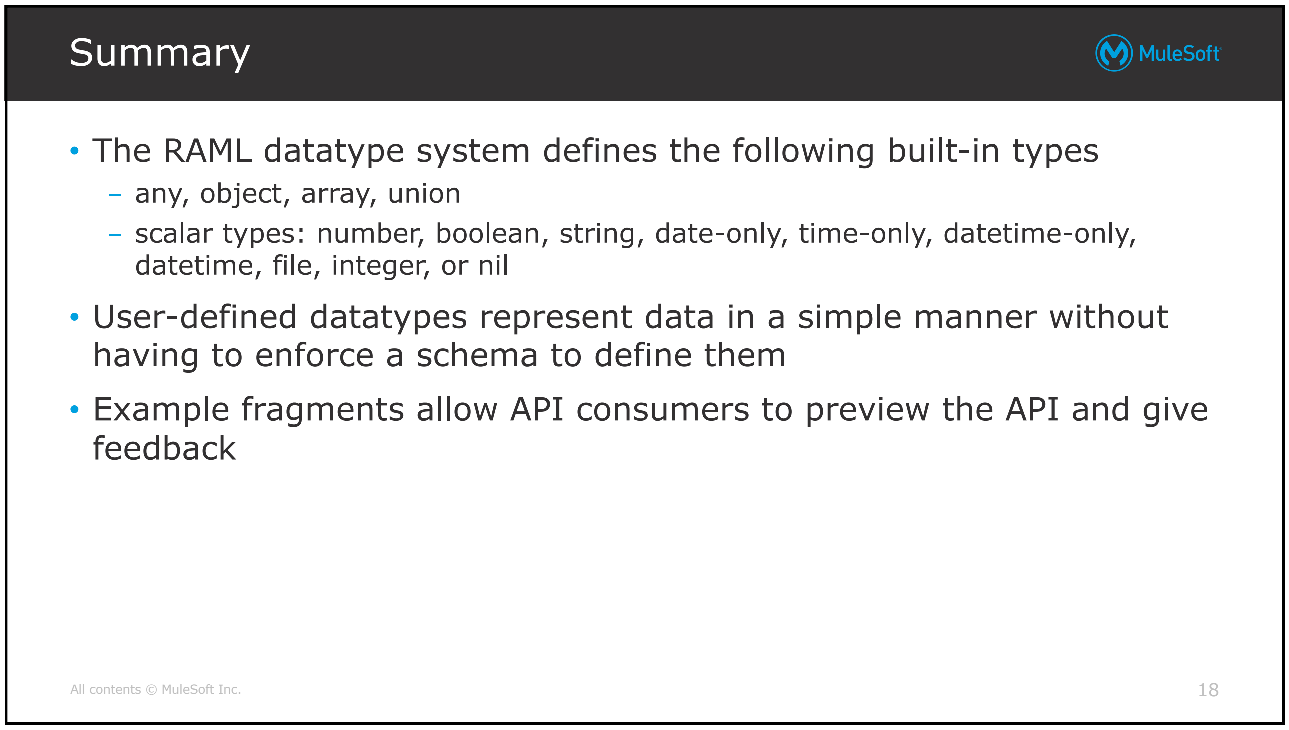
accountNumber 1234567890

accountOwner(Array 1)

customerID	displayName	ssn
8f19cb50-3f57-4d38	John Doe	123-456-7890



Summary



Summary

- The RAML datatype system defines the following built-in types
  - any, object, array, union
  - scalar types: number, boolean, string, date-only, time-only, datetime-only, datetime, file, integer, or nil
- User-defined datatypes represent data in a simple manner without having to enforce a schema to define them
- Example fragments allow API consumers to preview the API and give feedback