

Message Transformation

In API Gateway, a mapping template is used to transform data from one format to another. JSON path expressions can be used to map and transform the integration payload to any desired format. In addition, a model (schema) can be created to define the structure of a message payload. Having a model also enables you to generate an SDK that can be used by the client application to send properly formatted messages.

In this section, you will first create a model to represent the schema of the incoming request and response messages. Then you will validate and transform the incoming request message to match the downstream service (lambda) specification. The returned response message will also be formatted to properly capture the unit metrics.

Building a Model

1. First, let's add a new **Model** called **costCalculatorRequest** to the definition file `openapi.yaml`.

Review the code and then **copy/paste** it into the `openapi.yaml` file

```
1  openapi: "3.0.1"
2  info:
3    title: "module-2-my-first-api"
4    description: "First API with IaC example"
5    version: "1.0"
6
7  paths:
8
9    /pricepermeter:
10     post:
11       consumes:
12         - "application/json"
13       produces:
14         - "application/json"
15       responses:
16         "200":
17           description: "200 response"
18       x-amazon-apigateway-integration:
19         httpMethod: "POST"
20         credentials:
21           Fn::GetAtt: [LambdaExecutionRole, Arn]
22         url:
23           Fn::Sub: "arn:aws:apigateway:${AWS::Region}:lambda:path/2015-03-31/functions/${CostCalculator.Arn}/invocations"
24       responses:
25         default:
26           statusCode: "200"
27       passthroughBehavior: "when_no_match"
28       type: "aws"
29
30 components:
31   schemas:
32     costCalculatorRequest:
33       type: object
34       properties:
35         price:
36           type: number
37         size:
38           type: number
39         unit:
40           type: string
41         downPayment:
42           type: number
```

On line 30 we have **components Object**, which holds a set of reusable objects for different aspects of the OAS.

All objects defined within the components object will have no effect on the API unless they are explicitly referenced from properties outside the components object.

On line 31 we have **schemas Object**, which allows the definition of input and output data types. These types can be objects, but also primitives and arrays.

On lines 32 to 42 we have the **costCalculatorRequest** schema, responsible for defining the model.

2. Deploy the project again using SAM to verify that the model was added in your first API.

```
1  sam build && sam deploy
```

Notice that now you don't need to use the `argument` – guided to the `sam deploy` command again, because the deploy settings is already saved locally into the `sam` configuration file `samconfig.toml`

3. After the new changeset is created, review the resources and type `y` to confirm the new deployment.

Operation	LogicalResourceId	ResourceType	Replacement
+ Add	FirstAPIIDeployment12c42beaef	AWS::ApiGateway::Deployment	N/A
* Modify	FirstAPIDevStage	AWS::ApiGateway::Stage	N/A
+ Delete	FirstAPI	AWS::ApiGateway::RestApi	False
	FirstAPIDeployment1982676189	AWS::ApiGateway::Deployment	N/A

Changeset created successfully, arn:aws:cloudformation:us-east-1:907236258605:changeSet/samcli-deploy1608665929/as2594df-bb06-4eeef-8bc1-29fe245ed9c

Previewing CloudFormation changeset before deployment

Deploy this changeset? [y/N]: y

4. Check if the stack was successfully updated and open the **Amazon API Gateway console**.

5. Choose your REST API named, `module-2-my-first-api`.

6. Click in **Models** and then **costCalculatorRequest** to see the model that you just created.

APIs

Custom domain names

VPC links

Resources

Stages

Authorizers

Resource policy

Documentation

Dashboard

API settings

Usage plans

API keys

Client certificates

Settings

Models (1)

Use models to define the format for the body of different requests and responses used by your API.

Name	Content type	Description
costCalculatorRequest	application/json	

Model details

costCalculatorRequest

```
1 {
2   "type": "object",
3   "properties": {
4     "unit": {
5       "type": "string"
6     },
7     "size": {
8       "type": "number"
9     },
10  },
11  },
```

Now that the models are created, the next step is to transform the incoming request and response messages.

Transform Request Payload

1. Go back to the AWS Cloud9 console.
2. Add a new **requestTemplates object** to the integration in your file `openapi.yaml`.

Review the code and then **copy/paste** it into the `openapi.yaml` file

```
1  openapi: "3.0.1"
2  info:
3    title: "module-2-my-first-api"
4    description: "First API with IaC example"
5    version: "1.0"
6
7  paths:
8
9    /pricepermeter:
10     post:
11       consumes:
12         - "application/json"
13       produces:
14         - "application/json"
15       responses:
16         "200":
17           description: "200 response"
18       x-amazon-apigateway-integration:
19         httpMethod: "POST"
20         credentials:
21           Fn::GetAtt: [LambdaExecutionRole, Arn]
22         url:
23           Fn::Sub: "arn:aws:apigateway:${AWS::Region}:lambda:path/2015-03-31/functions/${CostCalculator.Arn}/invocations"
24       responses:
25         default:
26           statusCode: "200"
27       requestTemplates:
28         application/json:
29           #set($inputRoot = $input.path('$'))
30           {
31             "price": "$inputRoot.price",
32             "size": "$inputRoot.size",
33             "unit": "$inputRoot.unit",
34             "downPaymentAmount": $inputRoot.downPayment
35           }
36       passthroughBehavior: "when_no_match"
37       type: "aws"
38
39 components:
40   schemas:
41     costCalculatorRequest:
42       type: object
43       properties:
44         price:
45           type: number
46         size:
47           type: number
48         unit:
49           type: string
50         downPayment:
51           type: number
```

On line 27 we have **requestTemplates object**, which specifies mapping templates for a request payload of the specified MIME types.

On lines 28 to 35, the first statement `#set($inputRoot = $input.path('$'))` uses a JSONPath expression and returns an object representation of the result. This allows you to access and manipulate elements of the payload natively in Apache Velocity Template Language (VTL).

After the `inputRoot` variable is assigned to the root of the request, we can map the value of price, size and unit into the appropriate fields.

Notice
The last statement is mapping the value of `downPayment` to a new attribute called `downPaymentAmount`.

The downstream lambda function uses the `downPaymentAmount` to calculate the total price. (i.e. `totalPrice = price + downPaymentAmount`).

3. Deploy the project again using SAM to verify that the **requestTemplate** was added to your API Integration Request.

```
1  sam build && sam deploy
```

4. Check if the stack was successfully updated and open the **Amazon API Gateway console**.

5. Choose your REST API named, `module-2-my-first-api`.

6. In Resources click in /pricepermeter POST

7. Click in **Integration Request** and then scroll down to the **Mapping Templates** section to see that the mapping template was added to your integration request.

Mapping templates (1)

Create template

application/json

Edit Delete

```
1 #set($inputRoot = $input.path('$'))
2 {
3   "price": "$inputRoot.price",
4   "size": "$inputRoot.size",
5   "unit": "$inputRoot.unit",
6   "downPaymentAmount": $inputRoot.downPayment
7 }
8
```

The next step is to map the integration response. An API Gateway response is identified by a response type defined by API Gateway. The response consists of an HTTP status code, a set of additional headers that are specified by parameter mappings, and a payload that is generated by a VTL mapping template

Transform Response Payload

Similar to the request mapping, create the mapping template for the response payload. Perform the following:

1. Go back to the AWS Cloud9 console.
2. Add a new **requestTemplates object** to the integration in your file `openapi.yaml`.

```
1  openapi: "3.0.1"
2  info:
3    title: "module-2-my-first-api"
4    description: "First API with IaC example"
5    version: "1.0"
6
7  paths:
8
9    /pricepermeter:
10     post:
11       consumes:
12         - "application/json"
13       produces:
14         - "application/json"
15       responses:
16         "200":
17           description: "200 response"
18       x-amazon-apigateway-integration:
19         httpMethod: "POST"
20         credentials:
21           Fn::GetAtt: [LambdaExecutionRole, Arn]
22         url:
23           Fn::Sub: "arn:aws:apigateway:${AWS::Region}:lambda:path/2015-03-31/functions/${CostCalculator.Arn}/invocations"
24       responses:
25         default:
26           statusCode: "200"
27       responseTemplates:
28         application/json:
29           #set($allParams = $input.params())
30           {
31             "body-json": "$input.json('$')",
32             "params": {
33               #foreach($stage in $allParams.keySet())
34                 #set($params = $allParams.get($stage))
35                 "stage": {
36                   #foreach($paramName in $params.keySet())
37                     "$paramName": "$util.escapeJavaScript($params.get($paramName))"
38                   #if($foreach.hasNext),#end
39                 }
40               #if($foreach.hasNext),#end
41             },
42             "stage-variables": {
43               #foreach($key in $stageVariables.keySet())
44                 "$key": "$util.escapeJavaScript($stageVariables.get($key))"
45               #if($foreach.hasNext),#end
46             },
47             "context": {
48               "account-id": "$context.identity.accountId",
49               "api-id": "$context.apiId",
50               "api-key": "$context.identity.apiKey",
51               "authorizer-principal-id": "$context.authorizer.principalId",
52               "caller": "$context.identity.caller",
53               "cognito-authentication-provider": "$context.identity.cognitoAuthenticationProvider",
54               "cognito-authentication-type": "$context.identity.cognitoAuthenticationType",
55               "cognito-identity-id": "$context.identity.cognitoIdentityId",
56               "cognito-identity-pool-id": "$context.identity.cognitoIdentityPoolId",
57               "http-method": "$context.httpMethod",
58               "stage": "$context.stage",
59               "source-ip": "$context.identity.sourceIp",
60               "user": "$context.identity.user",
61               "user-agent": "$context.identity.userAgent",
62               "user-arn": "$context.identity.userArn",
63               "request-id": "$context.requestId",
64               "resource-id": "$context.resourceId",
65               "resource-path": "$context.resourcePath"
66             }
67           }
68       requestTemplates:
69         application/json:
70           #set($inputRoot = $input.path('$'))
71           {
72             "price": "$inputRoot.price",
73             "size": "$inputRoot.size",
74             "unit": "$inputRoot.unit",
75             "downPaymentAmount": $inputRoot.downPayment
76           }
77       passthroughBehavior: "when_no_match"
78       type: "aws"
79
80 components:
81   schemas:
82     costCalculatorRequest:
83       type: object
84       properties:
85         price:
86           type: number
87         size:
88           type: number
89         unit:
90           type: string
91         downPayment:
92           type: number
```

Now on line 27 we have **responseTemplates object**, which defines **GatewayResponse** mapping templates, as a string-to-string map of key-value pairs, for a given gateway response.

On lines 29 to 70, we are defining the template that will be applied to the integration response, this template can be customized to add transformations according to your needs.

3. Deploy the project again using SAM to verify that the **requestTemplate** was added to your API Integration Request.

```
1  sam build && sam deploy
```

4. Check if the stack was successfully updated and open the **Amazon API Gateway console**.

5. Choose your REST API named, `module-2-my-first-api`.

6. In Resources click in /pricepermeter POST

7. Click in **Integration Response** and then scroll down to the **Mapping Templates** section to see that the mapping template was added to your integration response.

Create resource

Client

Method request

Integration response

Method response

Test

Integrat...

Method request

Integration request

Integration response

Method response

Test

Integration responses

Create response

Default - Response

Edit Delete

HTTP status code

Content handling

Method response status code

Default mapping

Header mappings

No header mappings defined

Mapping templates

Create template

Edit Delete

```
1 #set($allParams = $input.params())
2 {
3   "body-json": "$input.json('$')",
4   "params": {
5     #foreach($stage in $allParams.keySet())
6       #set($params = $allParams.get($stage))
7       "stage": {
8         #foreach($paramName in $params.keySet())
9           "$paramName": "$util.escapeJavaScript($params.get($paramName))"
10         #if($foreach.hasNext),#end
11       }
12     #if($foreach.hasNext),#end
13   },
14   "stage-variables": {
15     #foreach($key in $stageVariables.keySet())
16       "$key": "$util.escapeJavaScript($stageVariables.get($key))"
17     #if($foreach.hasNext),#end
18   },
19   "context": {
20     "account-id": "$context.identity.accountId",
21     "api-id": "$context.apiId",
22     "api-key": "$context.identity.apiKey",
23     "authorizer-principal-id": "$context.authorizer.principalId",
24     "caller": "$context.identity.caller",
25     "cognito-authentication-provider": "$context.identity.cognitoAuthenticationProvider",
26     "cognito-authentication-type": "$context.identity.cognitoAuthenticationType",
27     "cognito-identity-id": "$context.identity.cognitoIdentityId",
28     "cognito-identity-pool-id": "$context.identity.cognitoIdentityPoolId",
29     "http-method": "$context.httpMethod",
30     "stage": "$context.stage",
31     "source-ip": "$context.identity.sourceIp",
32     "user": "$context.identity.user",
33     "user-agent": "$context.identity.userAgent",
34     "user-arn": "$context.identity.userArn",
35     "request-id": "$context.requestId",
36     "resource-id": "$context.resourceId",
37     "resource-path": "$context.resourcePath"
38   }
39 }
```

Test the API

In the response mapping, we have chosen to pass through the response content from lambda without making any further modifications. The method request passthrough is a pre-defined template that maps the response headers, body, parameters and context. The `$context` variable holds all the contextual information of your API call.

1. Go back to your `module-2-my-first-api` API and click on the /pricepermeter POST method to test your API.
2. Click on the **Test** option to provide a sample message.
3. Copy and Paste the following JSON Sample in the **Request Body** section

```
{
  "price": "400000",
  "size": "1600",
  "unit": "sqft",
  "downPayment": "20"
}
```

Create resource

/ /pricepermeter

POST

Test

Headers

Enter a header name and value separated by a colon (:) Use a new line for each header.

header1:value1

header2:value2

Client certificate

No client certificates have been generated.

Request body

1 {
2 "price": "400000",
3 "size": "1600",
4 "unit": "sqft",
5 "downPayment": "20"
6 }
7

Test

4. Click on **Test**

Create resource

/ /pricepermeter

POST

Test

Request

/pricepermeter

Latency

354

Status

200

Response body

```
{
  "body-json": "{ \"statusCode\":200, \"body\": \"({\\\"pricePerUnit\\\":\\\"250.00\\\",\\\"medianPrice\\\":\\\"320000.00\\\",\\\"totalCost\\\":\\\"400020.00\\\"},\\\"headers\\\":{\\\"X-Powered-By\\\":\\\"AWS API Gateway & Lambda Serverless\\\",\\\"isBase64Encoded\\\":false}, \\\"params\\\": { \\\"path\\\": { \\\"queryString\\\": { \\\"header\\\": { \\\"stage-variables\\\": { \\\"context\\\": { \\\"account-id\\\": \\\"[REDACTED]\\\", \\\"api-id\\\": \\\"[REDACTED]\\\", \\\"api-key\\\": \\\"test-invoke-api-key\\\", \\\"authorizer-principal-id\\\": \\\"\\\", \\\"cognito-authentication-provider\\\": \\\"\\\", \\\"cognito-authentication-type\\\": \\\"\\\", \\\"cognito-identity-id\\\": \\\"\\\", \\\"cognito-identity-pool-id\\\": \\\"\\\", \\\"http-method\\\": \\\"POST\\\", \\\"stage\\\": \\\"test-invoke-stage\\\", \\\"source-ip\\\": \\\"test-invoke-source-ip\\\", \\\"user\\\": \\\"[REDACTED]\\\"}}}}}}}}\", \\\"stage-variables\\\": { \\\"context\\\": { \\\"account-id\\\": \\\"[REDACTED]\\\", \\\"api-id\\\": \\\"[REDACTED]\\\", \\\"api-key\\\": \\\"test-invoke-api-key\\\", \\\"authorizer-principal-id\\\": \\\"\\\", \\\"cognito-authentication-provider\\\": \\\"\\\", \\\"cognito-authentication-type\\\": \\\"\\\", \\\"cognito-identity-id\\\": \\\"\\\", \\\"cognito-identity-pool-id\\\": \\\"\\\", \\\"http-method\\\": \\\"POST\\\", \\\"stage\\\": \\\"test-invoke-stage\\\", \\\"source-ip\\\": \\\"test-invoke-source-ip\\\", \\\"user\\\": \\\"[REDACTED]\\\"}}}}}}\""}",
  "headers": {
    "X-Powered-By": "AWS API Gateway & Lambda Serverless"
  },
  "params": {
    "path": {
      "queryString": {
        "header": {
          "stage-variables": {
            "context": {
              "account-id": "[REDACTED]",
              "api-id": "[REDACTED]",
              "api-key": "test-invoke-api-key",
              "authorizer-principal-id": "",
              "cognito-authentication-provider": "",
              "cognito-authentication-type": "",
              "cognito-identity-id": "",
              "cognito-identity-pool-id": "",
              "http-method": "POST",
              "stage": "test-invoke-stage",
              "source-ip": "test-invoke-source-ip",
              "user": "[REDACTED]"
            }
          }
        }
      }
    }
  },
  "stage-variables": {
    "context": {
      "account-id": "[REDACTED]",
      "api-id": "[REDACTED]",
      "api-key": "test-invoke-api-key",
      "authorizer-principal-id": "",
      "cognito-authentication-provider": "",
      "cognito-authentication-type": "",
      "cognito-identity-id": "",
      "cognito-identity-pool-id": "",
      "http-method": "POST",
      "stage": "test-invoke-stage",
      "source-ip": "test-invoke-source-ip",
      "user": "[REDACTED]"
    }
  }
}
```

Notice
The `downPayment` amount was added to the price to calculate the `totalCost` output. The response also contains the header and context attributes.

Select your cookie preferences

We use essential cookies and similar tools that are necessary to provide our site and services. We use performance cookies to collect anonymous statistics, so we can understand how customers use our site and make improvements. Essential cookies cannot be deactivated, but you can choose "Customize" or "Decline" to decline performance cookies.

If you agree, AWS and approved third parties will also use cookies to provide useful site features, remember your preferences, and display relevant content, including relevant advertising. To accept or decline all non-essential cookies, choose "Accept" or "Decline". To make more detailed choices, choose "Customize".

Accept

Decline

Customize

