

Solution Recommendation

Template for APIs

<customer\_name>

v1.0, 2017-02-14

# Purpose

This document describes API recommendations for <project\_name> and the functional and nonfunctional requirements/designs for each API identified.

# Intended Audience

This document’s audience primarily includes architects, developers and testers engaged in architecting, designing, developing, and testing API-based solutions.

# Taxonomy

The general terms and definitions referenced in the glossary or taxonomy document.

For the purposes of this template, any occurrences of text between ‘<’ and ‘>’ should be replaced with relevant information.

*Text provided in this format is meant to guide the writer in providing the appropriate content for the section. All examples of this text should be removed by the writer prior to delivery to the customer.*

# Solution Overview

*This section should summarize the functional requirements of the solution as a whole. If the solution requirements are lengthy or stored externally, it is reasonable to simply provide a brief summary or a link to where the requirements are stored.*

## API Identification

*The API Identification section documents the APIs that have been discovered during review of the solution requirements. This list should provide a summary of the APIs identified as well as a summary of the API’s capabilities. When possible, link a solution requirement to each API identified.*

The solution requires two APIs:

* <first\_API>
* <second\_API>

## Business Process Viewpoint

*The business process viewpoint demonstrates the alignment of business process with the APIs/integrations that have been identified for the solution. When possible, it should also provide clarity on process-to-application interaction and how the overall business process leverages the data from the APIs/integrations.*

*Diagrams are encouraged over raw text and the following diagram standards are recommended for defining the business process viewpoint:*

* *BPMN 2.0 Diagram*
* *UML Activity or Sequence Diagram*
* *ArchiMate Business Process Cooperation or Application Usage Viewpoint*

## Logical Viewpoint

*The logical viewpoint describes the relationships between APIs and integrations in terms of the information flows between them, or in terms of the services they offer and use. This viewpoint is useful in designing the internal composition of the APIs/integrations within the solution.*

*Diagrams are encouraged over raw text and the following diagram standards are recommended for defining the logical viewpoint:*

* *UML Component Diagram*
* *ArchiMate Application Cooperation or Application Structure Viewpoint*

## Deployment Viewpoint

*The deployment viewpoint defines the relationship between software components and how they are packaged and deployed within the physical infrastructure. To be more specific, with deployment viewpoint constructs a physical model of how software components are deployed on hardware components.*

*Diagrams are encouraged over raw text and the following diagram standards are recommended for defining the deployment viewpoint:*

* *UML Deployment Diagram*
* *ArchiMate Implementation and Deployment Viewpoint*

# 

# 

# <first\_API>

## Functional Requirements

*This section should address the functional requirements exclusive to the API (i.e. the capabilities that the API will provide once developed). This should not simply be a regurgitation of a single project’s business requirements since this document is intended to address an API’s requirements across multiple projects. The language should be such that it allows test cases to be developed for API functionality validation.*

## Functional Design

### Business Process Viewpoint

*Unless explicitly defined in the Solution Business Process Viewpoint above, define how the API will support the overall business process.*

### Resources

The <first\_API> provides two resources:

* <first\_resource>
* <second\_resource>

#### <first\_resource>

The <first\_resource> resource provides access to one or more <objects> from <system>:

##### Resource Locators

* To retrieve all <first\_resource>:

GET {BASE\_URI}/example/{version}/<first\_resource>

* To retrieve a single <first\_resource>:

GET {BASE\_URI}/example/{version}/<first\_resource>/{<first\_resource>\_id}

##### HTTP Status Codes

Possible HTTP status codes for the response include:

* 200 OK - for successful <first\_resource> retrieval
* 401 Unauthorized - for errors in API authentication
* 403 Forbidden - for errors in API authorization
* 500 Internal Server Error - for <first\_resource> retrieval failures

##### Resource Locator Examples

* To retrieve all <first\_resource>:

GET http://<server>:<port>/example/v1.0.0/<first\_resource>

* To retrieve a single <first\_resource> by ID:

GET http://<server>:<port>/example/v1.0.0/<first\_resource>/12345

##### Information Structure

*The information structure shows the structure of the information used in the API/integration application, in terms of data types or class structures.*

*Diagrams are encouraged over raw text and the following diagram standards are recommended for defining the information structure:*

* *Data Structure Diagrams (DSD)*
* *ArchiMate Information Structure Viewpoint*

#### <second\_resource>

The <second\_resource> resource provides access to one or more <objects> from <system>:

##### Resource Locator

* To retrieve a <second\_resource>:

GET {BASE\_URI}/example/{version}/<second\_resource>

The unique identifier is specified as a query parameter:

* id - unique id for the <first\_resource>

##### HTTP Status Codes

Possible HTTP status codes for the response include:

* 200 OK - for successful <second\_resource> retrieval
* 401 Unauthorized - for errors in API authentication
* 403 Forbidden - for errors in API authorization
* 404 Not Found - when specified <second\_resource> is not found
* 500 Internal Server Error - for <second\_resource> retrieval failures

##### Resource Locator Example

* To retrieve the <second\_resource> specified by the ID 1234:

GET http://<server>:<port>/example/v1.0.0/<second\_resource>?id=1234

##### Information Structure

*The information structure shows the structure of the information used in the API/integration application, in terms of data types or class structures.*

*Diagrams are encouraged over raw text and the following diagram standards are recommended for defining the information structure:*

* *Data Structure Diagrams (DSD)*
* *ArchiMate Information Structure Viewpoint*

### Logical Viewpoint

*Unless explicitly defined in the Logical Viewpoint above, define the relationships between this API and other APIs, integrations, or systems. For example:*

* *For a System API, how does the API communicate with the backend system and what data is provided?*
* *For a Process API, what underlying APIs or systems does it communicate with to execute the orchestration?*
* *For an Experience API, what data and data formats are required by the specific channel?*

### Deployment Viewpoint

*Unless explicitly defined in the Solution Deployment Viewpoint above, define the relationship between the API’s software components and how they are packaged and deployed within the physical infrastructure.*

## Nonfunctional Requirements

### Availability

* *Specify Availability targets within agreed hours, expressed as percentages.*
* *Specify how Availability will be measured and reported, and over what periods.*
* *Specify required “uptime” or Mean Time to Failure (MTTF) targets.*

### Reliability

* *Number of API breaks or Mean-Time-Between-Failures (MTBF).*
* *Specify any Mean Time to Repair (MTTR) targets if known.*
* *Definition of what constitutes a “break” and how these will be monitored and recorded.*

### Throughput

* *Provide an indication of likely traffic volumes. For example:*
  + *Number of transactions to be processed*
  + *Number of concurrent users*
  + *Amount of data to be transmitted*

### Response Time

* *Target times for average or maximum response times, expressed as a percentile: e.g., 95% within 2 seconds.*

### Potential Reuse

* *Target times for average or maximum response times, expressed as a percentile: e.g., 95% within 2 seconds.*

### Security

## *Identify the security requirements of the API according to the* *following table.*

|  |  |
| --- | --- |
| **Requirement** | **Implementation** |
| User/Application Authentication | *Basic Auth? API Keys?* |
| User/Application Authorization | *OAuth 2.0?* |
| Message Integrity | *Data Encryption?* |
| Message Confidentiality | *TLS?* |
| Auditing | *Correlation IDs?* |

# 

# <second\_API>

## Functional Requirements

*This section should address the functional requirements exclusive to the API (i.e. the capabilities that the API will provide once developed). This should not simply be a regurgitation of a single project’s business requirements since this document is intended to address an API’s requirements across multiple projects. The language should be such that it allows test cases to be developed for API functionality validation.*

## Functional Design

### Business Process Viewpoint

*Unless explicitly defined in the Solution Business Process Viewpoint above, define how the API will support the overall business process.*

### Resources

The <second\_API> provides two resources:

* <first\_resource>
* <second\_resource>

#### <first\_resource>

The <first\_resource> resource provides access to one or more <objects> from <system>:

##### Resource Locators

* To retrieve all <first\_resource>:

GET {BASE\_URI}/example/{version}/<first\_resource>

* To retrieve a single <first\_resource>:

GET {BASE\_URI}/example/{version}/<first\_resource>/{<first\_resource>\_id}

##### HTTP Status Codes

Possible HTTP status codes for the response include:

* 200 OK - for successful <first\_resource> retrieval
* 401 Unauthorized - for errors in API authentication
* 403 Forbidden - for errors in API authorization
* 500 Internal Server Error - for <first\_resource> retrieval failures

##### Resource Locator Examples

* To retrieve all <first\_resource>:

GET http://<server>:<port>/example/v1.0.0/<first\_resource>

* To retrieve a single <first\_resource> by ID:

GET http://<server>:<port>/example/v1.0.0/<first\_resource>/12345

##### Information Structure

*The information structure shows the structure of the information used in the API/integration application, in terms of data types or class structures.*

*Diagrams are encouraged over raw text and the following diagram standards are recommended for defining the information structure:*

* *Data Structure Diagrams (DSD)*
* *ArchiMate Information Structure Viewpoint*

#### <second\_resource>

The <second\_resource> resource provides access to one or more <objects> from <system>:

##### Resource Locator

* To retrieve a <second\_resource>:

GET {BASE\_URI}/example/{version}/<second\_resource>

The unique identifier is specified as a query parameter:

* id - unique id for the <first\_resource>

##### HTTP Status Codes

Possible HTTP status codes for the response include:

* 200 OK - for successful <second\_resource> retrieval
* 401 Unauthorized - for errors in API authentication
* 403 Forbidden - for errors in API authorization
* 404 Not Found - when specified <second\_resource> is not found
* 500 Internal Server Error - for <second\_resource> retrieval failures

##### Resource Locator Example

* To retrieve the <second\_resource> specified by the ID 1234:

GET http://<server>:<port>/example/v1.0.0/<second\_resource>?id=1234

##### Information Structure

*The information structure shows the structure of the information used in the API/integration application, in terms of data types or class structures.*

*Diagrams are encouraged over raw text and the following diagram standards are recommended for defining the information structure:*

* *Data Structure Diagrams (DSD)*
* *ArchiMate Information Structure Viewpoint*

### Logical Viewpoint

*Unless explicitly defined in the Logical Viewpoint above, define the relationships between this API and other APIs, integrations, or systems. For example:*

* *For a System API, how does the API communicate with the backend system and what data is provided?*
* *For a Process API, what underlying APIs or systems does it communicate with to execute the orchestration?*
* *For an Experience API, what data and data formats are required by the specific channel?*

### Deployment Viewpoint

* *Unless explicitly defined in the Solution Deployment Viewpoint above, define the relationship between the API’s software components and how they are packaged and deployed within the physical infrastructure.*

## Nonfunctional Requirements

### Availability

* *Specify Availability targets within agreed hours, expressed as percentages.*
* *Specify how Availability will be measured and reported, and over what periods.*
* *Specify required “uptime” or Mean Time to Failure (MTTF) targets.*

### Reliability

* *Number of API breaks or Mean-Time-Between-Failures (MTBF).*
* *Specify any Mean Time to Repair (MTTR) targets if known.*
* *Definition of what constitutes a “break” and how these will be monitored and recorded.*

### Throughput

* *Provide an indication of likely traffic volumes. For example:*
  + *Number of transactions to be processed*
  + *Number of concurrent users*
  + *Amount of data to be transmitted*

### Response Time

* *Target times for average or maximum response times, expressed as a percentile: e.g., 95% within 2 seconds.*

### Potential Reuse

* *Target times for average or maximum response times, expressed as a percentile: e.g., 95% within 2 seconds.*

### Security

## *Identify the security requirements of the API according to the* *following table.*

|  |  |
| --- | --- |
| **Requirement** | **Implementation** |
| User/Application Authentication | *Basic Auth? API Keys?* |
| User/Application Authorization | *OAuth 2.0?* |
| Message Integrity | *Data Encryption?* |
| Message Confidentiality | *TLS?* |
| Auditing | *Correlation IDs?* |