API Documentation Blueprint

(Section 13 from developer Playbook Outline)

The intent is for this document to be inserted into Section 13 of the Developer Playbook.

# Documentation Blueprint Overview

This Blueprint is both a guide and a full specification of the API Blueprint format for development of APIs within the VA.

The intent of this document is to provide technical development staff a guideline for researching, planning, developing and testing an API within the VA.

## What is in Scope of the Documentation Blueprint

The API Blueprint document is a deliverable produced by the VA project’s technical team to identify the API structure and format.

This blueprint can only be created after the system, application, and database designs are in place, as it relies on those structures for the API.

This document follows closely to the API Blueprint produced by ApriBlueprint.org. If follows Markdown Syntax for Object Notation (MSON), which represents data structures in a ‘mostly’ human readable plain text form.

This document presents a format structure to follow for API development. It serves as a developer’s reference.

## What is out of Scope of the Documentation Blueprint

This document does not teach a systems engineer or developer to write an API.

This document is not for project managers or business leads to define user stories, business cases, or project scopes.

## Intended Naming Conventions

This document follows and recommends that developers follow the accepted naming conventions for MSON. Where a situation exists that creates the requirement for adding proprietary or non-standard naming conventions, these should be clearly identified as exceptions and unit tested separately.

A naming and syntax Section is detailed separately.

# Core API Design Blueprint Document

The following Sections are the recommended Blueprint (structure, nomenclature and format) for a VA API Design.

## API Project Goal

List the project goal and overarching business case as defined by the project team.

Overarching Project Charter \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

API Project Goal \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## API Design Purpose

Clearly define the purpose of the API, and its use at the system level.

Purpose of the API \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

In defining the purpose, it is important to include the type(s) of data and how the data is intended to be used.

## API Design Scope

Expand on the purpose to cover the scope of the API in moving, copying, replicating, extracting or other function, that aids in technical staff and security personal in understanding the API function.

Design Scope \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## API Blueprint Language

These Sections provide details on the language structure of the API and how that melds into a blueprint for API development for VA teams.

### Introduction

This API Blueprint language follows closely in structure, and exactly in parameters to those defined in by APIBlueprint.Org. This Blueprint’s intent is to provide a format, rules, and structure to allow an API to work both within VA and to inter-systems all built on the same accepted blueprint, and free of vendor specific formatting.

This does not mean that some vendor specific functions will not be required for a specific project.

### API Blueprint

API Blueprint is a documentation-oriented web API description language. The API Blueprint is essentially a set of semantic assumptions laid on top of the Markdown syntax used to describe a web API.

In addition to the regular Markdown syntax, API Blueprint conforms to the GitHub Flavored Markdown syntax.

### API Blueprint document

An API Blueprint document – a blueprint – is a plain text Markdown document describing a Web API in whole or in part. The document is structured into logical sections. Each section has its distinctive meaning, content and position in the document.

General section definition and structure is discussed in detail later in the Blueprint section chapter.

All of the blueprint sections are optional. However, when present, a section must follow the API Blueprint document structure.

Blueprint document structure

* Metadata section
* API Name & overview section
* Resource sections
  + URI Parameters section
  + Attributes section
  + Model section
    - Headers section
    - Attributes section
    - Body section
    - Schema section
  + Action sections
    - Relation section
    - URI Parameters section
    - Attributes section
    - Request sections
      * Headers section
      * Attributes section
      * Body section
      * Schema section
    - Response sections
      * Headers section
      * Attributes section
      * Body section
      * Schema section
* Resource Group sections
  + Resource sections (see above)
* Data Structures section

NOTE: The number prior to a section name denotes the allowed number of the section occurrences.

NOTE: Refer to Sections Reference for description of a specific section type.

### Blueprint section

A Section represents a logical unit of an API Blueprint. For example: an API overview, a group of resources or a resource definition.

In general a section is defined using a keyword in a Markdown entity. Depending on the type of section the keyword is written either as a Markdown header entity or in a list item entity.

A section definition may also contain additional variable components such as its identifier and additional modifiers.

Metadata section

API Name & overview section

Resource sections

* + URI Parameters section
  + Attributes section
  + Model section
    - Headers section
    - Attributes section
    - Body section
    - Schema section
  + Action sections
    - Relation section
    - URI Parameters section
    - Attributes section
    - Request sections
      * Headers section
      * Attributes section
      * Body section
      * Schema section
    - Response sections
      * Headers section
      * Attributes section
      * Body section
      * Schema section

Resource Group sections

* + Resource sections (see above)

Data Structures section

NOTE: There are two special sections that are recognized by their position in the document instead of a keyword: The Metadata section and the API Name & Overview section. Refer to the respective section entry for details on its definition.

#### Section types

There are several types of API Blueprint sections. You can find the complete listing of the section types in the Section Reference.

The Blueprint section chapter discusses the section syntax in general. A specific section type may conform only to some parts of this general syntax. Always refer for respective section reference for details on its syntax.

#### Section structure

A general structure of an API Blueprint section defined by a keyword includes an identifier (name), section description and nested sections or a specifically formatted content.

#### Keywords

Following reserved keywords are used in section definitions:

Header keywords

* Group
* Data Structures
* HTTP methods (e.g. GET, POST, PUT, DELETE...)
* URI templates (e.g. /resource/{id})
* Combinations of an HTTP method and URI Template (e.g. GET /resource/{id})

List keywords

* Request
* Response
* Body
* Schema
* Model
* Header & Headers
* Parameter & Parameters
* Values
* Attribute & Attributes
* Relation

NOTE: Avoid using these keywords in other Markdown headers or lists

NOTE: With the exception of HTTP methods keywords the section keywords are case insensitive.

#### Identifier

A section definition may or must include an identifier of the section. An identifier is any non-empty combination of any character except [, ], (, ) and newline characters.

An identifier must not contain any of the keywords.

#### Description

A section description is any arbitrary Markdown-formatted content following the section definition.

It is possible to use any Markdown header or list item in a section description as long as it does not clash with any of the reserved keywords.

NOTE: It is considered good practice to keep the header level nested under the actual section.

#### Nested sections

A section may contain another nested section(s).

Depending on the nested section type, to nest a section simply increase its header level or its list item indentation. Anything between the section start and the start of following section at the same level is considered to be part of the section.

What sections can be nested and where depends upon the section in case, as described in the relevant section's entry.

NOTE: While not necessary it is a good habit to increase the level of a nested section markdown-header.

NOTE: A markdown-list section is always considered to be nested under the preceding markdown-header section.

## Sections Reference

Sections marked as "Abstract" serve as a base for other sections and as such they cannot be used directly.

### Abstract

#### Named section

Named section is the base section for most of the API Blueprint sections. It conforms to the general section and as such it is composed of a section name (identifier), description and nested sections or specific formatted content.

Defined by a keyword followed by an optional section name - identifier in a Markdown header or list entity.

Key attributes

* Abstract
* Parent sections: vary, see descendants
* Nested sections: vary, see descendants
* Markdown entity: header, list
* Inherits from: none

#### Asset section

The asset section is the base section for atomic data in API Blueprint. The content of this section is expected to be a pre-formatted code block.

Defined by a keyword in Markdown list entity.

Key attributes

* Abstract
* Parent sections: vary, see descendants
* Nested sections: none
* Markdown entity: list
* Inherits from: none

#### Payload section

Payload sections represent the information transferred as a payload of an HTTP request or response messages. A Payload consists of optional meta information in the form of HTTP headers and optional content in the form of an HTTP body.

Furthermore, in API Blueprint context, a payload includes its description, description of its message-body attributes and a message-body validation schema.

A payload may have its media type associated. A payload's media type represents the metadata received or sent in the form of a HTTP Content-Type header. When specified a payload should include nested Body section.

This section should include at least one of the following nested sections:

* Headers section
* Attributes section
* Body section
* Schema section

If there is no nested section the content of the payload section is considered as content of the Body section.

Key attributes

* Abstract
* Parent sections: vary, see descendants
* Nested sections: 0-1 Headers section, 0-1 Attributes section, 0-1 Body section, 0-1 Schema section
* Markdown entity: list
* Inherits from: Named section

### Section Basics

#### Metadata section

Metadata keys and their values are tool-specific. Refer to relevant tool documentation for the list of supported keys.

Key-value pairs. Each key is separated from its value by a colon (:). One pair per line. Starts at the beginning of the document and ends with the first Markdown element that is not recognized as a key-value pair.

Key attributes

* Parent sections: none
* Nested sections: none
* Markdown entity: special
* Inherits from: none

#### API name & overview section

Name and description of the API

Defined by the first Markdown header in the blueprint document, unless it represents another section definition.

Key attributes

* Parent sections: none
* Nested sections: none
* Markdown entity: special, header
* Inherits from: Named section

#### Resource group section

This section represents a group of resources (Resource Sections). May include one or more nested Resource Sections.

Defined by the Group keyword followed by group name (identifier).

Key attributes

* Parent sections: none
* Nested sections: 0+ Resource section
* Markdown entity: header
* Inherits from: Named section

#### Resource section

An API resource as specified by its URI or a set of resources (a resource template) matching its URI template.

This section should include at least one nested Action section and may include following nested sections:

* URI parameters section - URI parameters defined in the scope of a Resource section apply to any and all nested Action sections except when an URI template has been defined for the Action.
* Attributes section - Attributes defined in the scope of a Resource section represent Resource attributes. If the resource is defined with a name these attributes may be referenced in Attributes sections.
* Model section - Additional Action sections

A blueprint document may contain multiple sections for the same resource (or resource set), as long as their HTTP methods differ. However it is considered good practice to group multiple HTTP methods under one resource (resource set).

Key attributes

* Parent sections: none, Resource group section
* Nested sections: 0-1 Parameters section, 0-1 Attributes section, 0-1 Model section, 1+ Action section
* Markdown entity: header
* Inherits from: Named section

#### Resource model section

A resource manifestation - one exemplary representation of the resource in the form of a payload.

##### Referencing

The payload defined in this section may be referenced in any response or request section in the document using parent section's identifier. You can refer to this payload in any of the following Request or Response payload sections using the Markdown implicit reference syntax.

Defined by the Model keyword followed by an optional media type.

#### Schema section

Specifies a validation schema for the HTTP message-body of parent payload section.

Example - Following example uses Body section to provide an example of an application/json payload, and Schema section to provide a JSON Schema describing all possible valid shapes of the payload.

Defined by the Schema keyword in Markdown list entity.

Key attributes

* Parent sections: Payload section
* Nested sections: none
* Markdown entity: list
* Inherits from: Asset section

#### Action section

Definition of at least one complete HTTP transaction as performed with the parent resource section. An action section may consist of multiple HTTP transaction examples for the given HTTP request method.

This section may include one nested URI parameters section describing any URI parameters specific to the action – URI parameters discussed in the scope of an Action section apply to the respective Action section ONLY.

This section may include one nested Attributes section defining the input (request) attributes of the section. If present, these attributes should be inherited in every Action's Request section unless specified otherwise.

Action section should include at least one nested Response section and may include additional nested Request and Response sections.

Nested Request and Response sections may be ordered into groups where each group represents one transaction example. The first transaction example group starts with the first nested Request or Response section. Subsequent groups start with the first nested Request section following a Response section.

Multiple Request and Response nested sections within one transaction example should have different identifiers.

Defined by an HTTP request method:

-- or --

Defined by an action name (identifier) followed by an HTTP request method enclosed in square brackets [].

-- or --

Defined by an action name (identifier) followed by an HTTP request method and URI template enclosed in square brackets [].

Key attributes

* Parent sections: Resource section
* Nested sections: 0-1 Relation section, 0-1 URI parameters section, 0-1 Attributes section, 0+ Request section, 1+ Response section
* Markdown entity: header
* Inherits from: Named section

#### Request section

One HTTP request-message example – payload.

Defined by the Request keyword followed by an optional identifier.

Key attributes

* Parent sections: Action section
* Nested sections: Refer to payload section
* Markdown entity: list
* Inherits from: Payload section

#### Response section

One HTTP response-message example – payload.

Defined by the Response keyword. The response section definition should include an HTTP status code as its identifier.

Key attributes

* Parent sections: Action section
* Nested sections: Refer to payload section
* Markdown entity: list
* Inherits from: Payload section

#### UNIFORM RESOURCE IDENTIFIERS (URI) parameters section

Discussion of URI parameters in the scope of the parent section.

This section must be composed of nested list items only. This section must not contain any other elements. Each list item describes a single URI parameter. The nested list items subsections inherit from the Named section and are subject to additional formatting.

Where:

* <parameter name> is the parameter name as written in Resource Section's URI (e.g. "id").
* <description> is any optional Markdown-formatted description of the parameter.
* <additional description> is any additional optional Markdown-formatted description of the parameter.
* <default value> is an optional default value of the parameter – a value that is used when no value is explicitly set (optional parameters only).
* <example value> is an optional example value of the parameter (e.g. 1234).
* <type> is the optional parameter type as expected by the API (e.g. "number", "string", "boolean"). "string" is the default.
* Members is the optional enumeration of possible values. <type> should be surrounded by enum[] if this is present. For example, if enumeration values are present for a parameter whose type is number, then enum[number] should be used instead of number to.
* <enumeration value n> represents an element of enumeration type.
* required is the optional specifier of a required parameter (this is the default)
* optional is the optional specifier of an optional parameter.

NOTE: This section should only contain parameters that are specified in the parent's resource URI template, and does not have to list every URI parameter.

Defined by the Parameters keyword written in a Markdown list item.

Key attributes

* Parent Sections: Resource section | Action section
* Nested Sections: none
* Markdown entity: list
* Inherits from: none, special

#### Attributes section

This section describes a data structure using the Markdown Syntax for Object Notation (MSON). Data structures defined in this section may refer to any arbitrary data structures defined in the Data Structures section as well as to any data structures defined by a named resource attributes description.

Resource Attributes description is the description of the resource data structure.

If defined in a named Resource section, this data structure may be referenced by other data structures using the resource name.

Defined by the Attributes keyword followed by an optional MSON Type Definition enclosed in parentheses.

<Type Definition> is the type definition of the data structure being described. If the <Type Definition> is not specified, an object base type is assumed. See MSON Type Definition for details.

Key attributes

* Parent sections: Resource section | Action section | Payload section
* Nested sections: See Markdown Syntax for Object Notation
* Markdown entity: list
* Inherits from: none

##### Action Attributes description

Description of the default request message-body data structure.

If defined, all the Request sections of the respective Action section inherits these attributes unless specified otherwise.

##### Payload Attributes description

Description of payload (request, response, model) message-body attributes.

Not every attribute has to be described. However, when an attribute is described, it should appear in the respective Body section example, if a Body section is provided.

If defined, the Body section may be omitted and the example representation should be generated from the attributes description.

The description of message-body attributes may be used to describe message-body validation if no Schema section is provided. When a Schema section is provided, the attributes description should conform to the schema.

#### Headers section

Specifies the HTTP message-headers of the parent payload section. The content this section is expected to be a pre-formatted code block with the following syntax: <HTTP header name>: <value>

Defined by the Headers keyword in Markdown list entity.

Key attributes

* Parent sections: Payload section
* Nested sections: none
* Markdown entity: list
* Inherits from: none

#### Body section

Specifies the HTTP message-body of a payload section.

Defined by the Body keyword in Markdown list entity.

Key attributes

* Parent sections: Payload section
* Nested sections: none
* Markdown entity: list
* Inherits from: Asset section

### Data Structures section

This section holds arbitrary data structures definitions defined in the form of MSON Named Types.

Data structures defined in this section may be used in any Attributes section. Similarly, any data structures defined in a Attributes section of a named Resource Section may be used in a data structure definition.

Refer to the MSON specification for full details on how to define an MSON Named type.

Defined by the Data Structures keyword.

Key attributes

* Parent sections: none
* Nested sections: MSON Named Type definition (see below)
* Markdown entity: header
* Inherits from: none

### Relation section

This section specifies a link relation type for the given action as specified by RFC 5988.

NOTE: The link relation identifiers should be unique per resource in the blueprint document.

Defined by the Relation keyword written in a Markdown list item followed by a colon (:) and a link relation identifier.

Key attributes

* Parent sections: Action section
* Nested Sections: none
* Markdown entity: list
* Inherits from: none

## Uniform Resource Identifiers (URI) Templates

Based on RFC 6570 https://tools.ietf.org/html/rfc6570

### URI Path Segment

At its simplest form – without any variables – a path segment of an URI Template is identical to an URI path segment:

/path/to/resources/42

### URI Template Variable

Variable names are case-sensitive. The variable name may consists of following characters only:

* ASCII alpha numeric characters (a-z, A-Z)
* Decimal digits (0-9)
* Percent-encoded characters
* \_
* .

Multiple variables are separated by the comma without any leading or trailing spaces. A variable(s) must be enclosed in braces – {} without any additional leading or trailing whitespace.

Operators

The first variable in the braces might be preceded by an operator. API Blueprint currently supports the following operators:

* # – fragment identifier operator
* + – reserved value operator
* ? – form-style query operator
* & – form-style query continuation operator
* Examples
* {var}
* {var1,var2,var3}
* {#var}
* {+var}
* {?var}
* {?var1,var2}
* {?%24var}
* {&var}

NOTE: The explode variable modifier is also supported. Refer to RFC6570 for its description.

Variable Reserved Values

Following characters are reserved in variable values:

: / / / ? / # / [ / ] / @ / ! / $ / & / ' / ( / ) / \* / + / , / ; / =

### Path Segment Variable

Simple path segment component variable is defined without any operator:

/path/to/resources/{var}

With var := 42 the expansion is /path/to/resources/42.

NOTE: RFC6570 – Level 1

### Fragment Identifier Variable

URI Template variables for fragment identifiers are defined using the crosshatch (#) operator:

/path/to/resources/42{#var}

With var := my\_id the expansion is /path/to/resources/42#my\_id.

NOTE: RFC6570 – Level 2

### Variable with Reserved Characters Values

To define URI Template variables with reserved URI characters, use the plus (+) operator:

/path/{+var}/42

With var := to/resources the expansion is /path/to/resources/42.

NOTE: RFC6570 – Level 2

### Form-style Query Variable

To define variables for a form-style query use the question mark (?) operator

/path/to/resources/{varone}{?vartwo}

With varone := 42 and vartwo = hello the expansion is /path/to/resources/42?vartwo=hello.

To continue a form-style query use the ampersand (&) operator:

/path/to/resources/{varone}?path=test{&vartwo,varthree}

With varone := 42, vartwo = hello, varthree = 1024 the expansion is /path/to/resources/42?path=test&vartwo=hello&varthree=1024.

NOTE: RFC6570 – Part of Level 3

## API Unit Testing List

Define how the unit testing will be conducted based on the subsections below.

### Blueprint based Unit Testing

### Vendor specific or non-standard unit testing

## API Error Handling and options

Describe how error handling is intended to be handled.

## API Logging

Describe how API Logging is intended to be function.

## API Interface agreements

This Section provides a list of interface agreements that the developer must reference for security, connectivity and/or access management. These agreements may be in the form of:

Interconnection Security Agreements (i.e. NIST 800-47)

Interface Capabilities Document

# Glossary of Terms

A brief list of terms as used in the API Blueprint context.

**Action**

An HTTP transaction (a request-response transaction). Actions are specified by an HTTP request method within a resource.

**API**

An HTTP Application programming interface. Might refer to an API description. See API Blueprint.

**API Blueprint**

The API Blueprint language. A format used to describe API in an API blueprint file.

**Asset**

Atomic data. Most often representing one resource representation in the form of message-body or its validation schema.

**Attribute**

Based on the context, attribute (property) of a message-body data structure, or attribute of a resource, or an input attribute of a transition – Action.

**Blueprint**

An API description. A blueprint file (or a set of files) that describes an API using the API Blueprint language.

**Data Structure**

A particular data organization, or a description of it. In API Blueprint, data structures and their Attributes are described using the Markdown Syntax for Object Notation – MSON.

**Entity**

Entity being transferred in a payload.

**Header**

A message-header.

**Method**

An HTTP Request Method.

**Message**

An HTTP transaction message.

**Message body**

An asset representing HTTP transaction message body.

**Message header**

An asset representing HTTP transaction message header.

**Parameter**

An URI template variable.

**Payload**

An HTTP transaction message including its discussion and any additional assets such as entity-body validation schema. A payload may have an identifier – a string for a request payload or an HTTP status code for a response payload.

**Property**

An entity field (attribute).

**Request**

A payload containing one specific HTTP Request.

**Response**

A payload containing one specific HTTP Response.

**Resource**

An API resource specified by its URI. It can also refer to a set of resources matching one URI template.

**Resource Model**

One manifestation of a resource in the form of a payload. A resource model is an example representation of its resource. Can be referenced later in the place of a payload.

**Resource Set**

A set of API resources. Its URI matches one specific URI template.

**Trait**

A quality or characteristic of an API Blueprint SECTION.

**Schema**

A validation schema in a form of an asset used to validate (or describe) a message-body.

**URI template**

A compact sequence of characters for describing a range of Uniform Resource Identifiers through variable expansion, see RFC 6570.