Kiwi API Response Format

Information Technology — Application Programming Interfaces—

Kiwi API Response Format

This document was produced by the Kiwi Committee for Standardization.

For more information, visit https://www.mia.kiwi/projects/kiwi.mia.kcs.0001/

Table of Contents

Table of Figures

Index of Tables

0 Introduction

0.1 Conventions used in this document

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “NOT RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in BCP 14[1][2] when, and only when, they appear in all capitals, as shown here.
The grammatical rules in this document are to be interpreted as described in RFC 5234[3]. The core rules defined in RFC 5234, Appendix B apply.

0.2 Systematic review

The Kiwi API response format (hereinafter referred to as the “Format”, or “KAPIR”) SHALL undergo systematic review on YYYY-MM-DD.

1 Scope

This document describes version VERSION of the Format.
This document does not define HTTP status codes, error codes, request formats, or authentication mechanisms.

2 Terms and definitions

For the purposes of the Format, the following terms and definitions SHALL apply:

2.1 Member

A named key-value pair within a value of type OBJECT.

3 Types

3.1 Null

The NULL type SHALL represent the explicit absence of a value and SHALL be distinct from empty values – e.g. an empty STRING, OBJECT, or ARRAY.

3.2 String

The STRING type SHALL represent a sequence of zero or more Unicode characters. A value of type STRING with zero characters SHALL be considered empty.

3.3 Boolean

The BOOLEAN type SHALL represent a logical state that can either be TRUE or FALSE.

3.4 Number

The NUMBER type SHALL represent a numeric value, which MUST be an integer or a floating-point value.

3.5 Array

The ARRAY type SHALL represent an ordered collection of values where each value is identified by its position within the collection. The ARRAY type collection SHALL only contain values of types a) NULL, b) STRING, c) BOOLEAN, d) NUMBER, e) ARRAY, and f) OBJECT. The order of the values SHALL be preserved. A value of type ARRAY with zero collection values SHALL be considered empty. A value of type ARRAY MUST NOT contain

itself directly or indirectly within its collection of values. A value of type ARRAY whose collection only holds unique values SHALL be considered unique. A value of type ARRAY whose collection only holds values of a single type SHALL be considered uniform, conversely, a value of type ARRAY whose collection holds values of different types SHALL be considered mixed.

3.6 Object

The OBJECT type SHALL represent an unordered collection of zero or more members. A value of type OBJECT SHALL NOT contain multiple members with the same member name. A value of type OBJECT with zero members SHALL be considered empty. The order of the members SHALL NOT be significant. A value of type OBJECT MUST NOT contain itself directly or indirectly within its collection of members.

3.6.1 Members

A member SHALL consist of a name and a value. The member name SHALL be of type STRING and SHALL uniquely identify the member within the containing OBJECT. The member value SHALL be of any type. A member SHALL NOT exist outside of an OBJECT.

3.7 Suberror

A value of type SUBERROR SHALL be an OBJECT with members named:

— “code”;
— “message”.

3.7.1 Code Member

The member named “code” SHALL have a non-empty value of type STRING. The code member SHALL NOT be omitted.

3.7.2 Message Member

The member named “message” MAY be omitted. If present, it SHALL have a non-empty value of type STRING.

3.8 Error

A value of type ERROR SHALL be an OBJECT with members named:

— “code”;
— “message”;
— “errors”.

3.8.1 Code Member

The member named “code” SHALL have a non-empty value of type STRING. The code member SHALL NOT be omitted.

3.8.2 Message Member

The member named “message” MAY be omitted. If present, it SHALL have a non-empty value of type STRING.

3.8.3 Errors Member

The member named “errors” SHALL have a value of type uniform ARRAY, the collection of which SHALL only hold values of type SUBERROR. If the values collection is empty, the member MAY be omitted. The member value SHALL NOT be of type NULL. The member collection values SHOULD be unique.

3.9 Status

A value of type STATUS SHALL represent a non-empty, case-sensitive STRING value precisely equal to “success” or “error”.

```
status         = success-status / error-status
; "success" / "error"

success-status = "success"
error-status   = "error"
```

Figure 1 — STATUS Type Values

4 KAPIR Response

A KAPIR Response (hereinafter also referred to as a “Response”) is a response that adheres to the Format, delivered to a client by a server through an API.
The server MAY deliver non-KAPIR response in cases where it is preferable and agreed upon by the server and client, but the server SHOULD prefer KAPIR Responses in general.
The server SHALL be responsible for delivering valid Responses when reasonably expected or as previously agreed.

4.1 Structure

A valid Response SHALL have the members (hereinafter referred to as “mandatory members”):

— “status”;
— “version”;
— “data”.

OPTIONALLY, or if required, a Response SHALL also have the members (hereinafter referred to as “optional members”):

— “message”;
— “error”;
— “meta”;
— “ext”.

The order of the members SHALL NOT impact the Response.
Additional members SHALL NOT be included unless introduced by extensions.

4.1.1 Status Member

The status member SHALL reflect the state of the request. A request SHALL be successful if it was fully processed according to its intended function and resulted in no errors. A request SHALL be failed if it was not fully processed, if it was not completed as intended, or if it resulted in one or more errors.
Requests designed to return zero or more resources based on non-unique criteria – i.e. filters – SHALL NOT treat the absence of matching resources as an error. Such requests are considered fully processed as intended and SHALL return a successful status provided that the request was otherwise processed successfully.
Requests designed to return exactly one resource using a unique identifier – e.g. a UUID, or a unique name – SHALL treat the absence of a matching resource as an error and SHALL return a failed status, notwithstanding the absence of other errors in processing the request.
A Response to a failed request SHALL have a status member value of “error”, a non-null error member value and present error member, and a null data member value.
A Response to a successful request SHALL have a status member value of “success”, a null error member value or absent error member, and MAY have a null data member value.

The status member SHALL NOT be omitted.

The status member name SHALL be “status”.

The status member value SHALL be of type STATUS

4.1.2 Version Member

The version member SHALL be used to indicate the version of the Format used in the Response.

The version member SHALL NOT be omitted.

The version member name SHALL be “version”.

The version member value SHALL be of type STRING.

4.1.3 Data Member

The data member SHALL contain the main content returned by the server in response to the request. If no content is provided, or if the request failed, the data member value SHALL be of type NULL.

The data member SHALL NOT be omitted.

The data member name SHALL be “data”.

The data member value SHALL be of type:

— NULL;

— STRING;

— BOOLEAN;

— NUMBER;

— ARRAY;

— OBJECT.

4.1.4 Message Member

The message member SHOULD be used to provide information directed at the consumer. The user SHOULD generally relay the message member value to the consumer. The message member SHOULD NOT be used for programmatic handling of responses. If no message is provided, the message member SHALL have a value of type NULL, or SHALL be omitted from the Response.
The message member name SHALL be “message”.
The message member value SHALL be of type NULL, or non-empty STRING.

4.1.5 Error Member

The error member SHALL be used to provide structured information about an error that prevented the request from being successful. The root code and messages SHOULD represent the high-level error that occurred, while the values of type SUBERROR in the value collection of the “errors” member value SHOULD be used to convey more specific or deeper error details.
The “code” member value of the member and suberrors SHOULD be persistent — i.e. be the same code value for the same error type.
The error member name SHALL be “error”.
If the request is successful, the error member SHALL have a value of type NULL, or SHALL be omitted from the Response.
The error member, if present and not of type NULL, SHALL have a value of type ERROR.

4.1.6 Metadata Member

The metadata member MAY be used to provide metadata about the request – e.g. time of reception, compute time, number of resources returned – or Response that are not directly related to the data or errors. The metadata member and the values within it

SHALL NOT be required for a client to correctly interpret the Response. If no metadata is provided, the metadata member SHALL have a value of type NULL, or SHALL be omitted from the Response.
The metadata member name SHALL be “meta”.
The metadata member value SHALL be of type OBJECT.

4.1.7 Extensions Member

The extensions member MAY be used to indicate response extensions used in the Response.
If no extension is used, the extensions member SHALL have an empty ARRAY as its value, or SHALL be omitted from the Response.
The extensions member name SHALL be “ext”.
The extensions member value SHALL be of type unique uniform ARRAY, the collection of which SHALL only hold unique, non-empty values of type STRING.

4.2 Fallback Values

Mandatory members SHALL NOT be omitted and therefore do not have fallback values. Responses missing mandatory members SHALL be treated as erroneous by clients.
Clients consuming Responses SHALL be able to handle omitted optional members.
If the message member is omitted, it SHALL be treated as having a value of type NULL.
If the error member is omitted and the request is successful, it SHALL be treated as having a value of type NULL.
If the metadata member is omitted, it SHALL be treated as having a value of type empty OBJECT.
If the extensions member is omitted, it SHALL be treated as having a value of type empty ARRAY.
This document does not prescribe fallback values for the data member value, but it is RECOMMENDED that clients treat missing data values as having a value of type NULL.

5 Format Extensions

The Format MAY be extended by adding new, non-standard members to a Response. Such additional members are referred to as “extensions”.

5.1 Identification

Each extension SHALL have a corresponding extension code, which SHALL be included in the extensions member of the Response.
Extension codes SHALL be communicated to the client prior to use and SHALL be unique across all Responses.

5.2 Usage

Extensions SHALL be used only when agreed upon with the client.
Clients SHALL NOT be required to process or understand any extension members in order to correctly interpret the Response.
Extensions SHALL NOT introduce mandatory members that would cause a standard client to fail if the extension is not present.
Clients SHALL ignore unknown or absent extensions.

5.3 Responsibility and Mediation

Extension authors are responsible for:

— describing the requirements and meaning of their extension members;
— defining the method for computing or generating the extension values;
— ensuring that the extension code is unique and does not conflict with other extensions.

The Kiwi Committee for Standardization reserves the right to mediate any conflicts arising from conflicting extension codes, and offensive or otherwise inappropriate extension codes.

Extension specification SHALL be documented externally and made available to all participating clients and servers.

Clients MAY ignore unknown extensions without error.

Servers SHALL NOT assume clients support any extension unless it was explicitly convened.

Normative references

[1] BRADNER, S. *Key words for use in RFCs to Indicate Requirement Levels*. Online. RFC Editor, 1997. Available from: https://doi.org/10.17487/rfc2119. [viewed 2025-10-17].

[2] LEIBA, B. *RFC 2119 Key Words: Clarifying the Use of Capitalization*. Online. RFC Editor, 2017. Available from: https://doi.org/10.17487/rfc8174. [viewed 2025-10-17].

[3] OVERELL, P. *Augmented BNF for syntax specifications: ABNF.* Online. D. CROCKER (ed.). RFC Editor, 2008. Available from: https://doi.org/10.17487/rfc5234. [viewed 2025-11-18].