# External Error Responses

A main error message is defined, with JSON structure with the following fields:

* A field "status", which can be identified in the response as a standard code from list of Hypertext Transfer Protocol (HTTP) response status codes.
* A unique error "code", which can be identified and traced for more details. It must be human readable; therefore, it must not be a numeric code. In turn, to achieve a better location of the error, you can reference the value or field that is causing it, and include it in the message.
* A detailed description of "message"

A JSON error structure is proposed below:

|  |
| --- |
| {  "status": 400,  "code": "INVALID\_ARGUMENT",  "message": "A human readable description of what the event represent"  } |

In error handling different cases must be considered, even at the functional level that it is possible to modify the error message returned to the API consumer. For this error handling there are two possible alternatives listed below:

* Error handling done with custom policies in the API admin tool.
* Error management performed in a service queried by API.

The essential requirements to consider would be:

* Error handling should be centralized in a single place, regardless of whether they are APIs that are proxies to end services or APIs that invoke an integrator service on the ESB.
* Customization of the generated error based on the error content returned by the final core service should be contemplated.
* Latency should be minimized in its management.

In the following, we elaborate on the existing client errors. In particular, we identify the different error codes and cluster them into separate tables, depending on their nature: i) syntax exceptions, ii) service exceptions, and iii) server errors.

**Client Errors**

**Syntax Exceptions**

|  |  |
| --- | --- |
| Error code | Description |
| INVALID ARGUMENT | A specified resource duplicate entry found. |
| CONFLICT | A specified resource duplicate entry found. |
| OUT\_OF\_RANGE | Client specified an invalid range. |
| PERMISSION\_DENIED | Client does not have sufficient permissions to perform this action. |
| ABORTED | Concurrency conflict. |
| ALREADY\_EXISTS | The resource that a client tried to create already exists. |

**Service Exceptions**

|  |  |
| --- | --- |
| Error code | Description |
| UNAUTHENTICATED | Request not authenticated due to missing, invalid, or expired credentials. |
| NOT\_FOUND | The specified resource is not found. |
| TOO\_MANY\_REQUESTS | Either out of resource quota or reaching rate limiting. |
| AUTHENTICATION\_REQUIRED | New authentication is required. |

**Server Exceptions**

|  |  |
| --- | --- |
| Error code | Description |
| FAILED\_PRECONDITION | Request cannot be executed in the current system state. |
| DATA\_LOSS | Unrecoverable data loss or data corruption. |
| INTERNAL | Unknown server error. Typically a server bug. |
| BAD\_GATEWAY | An upstream internal service cannot be reached. |
| UNAVAILABLE | Request timeout exceeded. |
| TIMEOUT | Request timeout exceeded. |
| NOT\_IMPLEMENTED | This functionality is not implemented yet |
| METHOD\_NOT\_ALLOWED | The requested method is not allowed/supported on the target resource. |
| NOT\_ACCEPTABLE | The server cannot produce a response matching the content requested by the client through Accept-\* headers. |
| UNSUPPORTED\_MEDIA\_TYPE | The server refuses to accept the request because the payload format is in an unsupported format. |

***Note: When no login has been performed or no authentication has been assigned, a non-descriptive generic error will always be returned in all cases, a UNAUTHENTICATED 401 “Request not authenticated due to missing, invalid, or expired credentials.” is returned, whatever the reason.***