هيئة الزكاة والضريبة والجمارك
Zakat, Tax and Customs Authority

API Development guidelines

General Information Technology Department

Digital Business Development - Integration Management

Version 1.0.0

2023-01-01

Document History

Version	Author	Date	Version Summary
1.0	Abdulaziz Aljaber	2023-01-10	First Version
1.1	Abdulaziz Aljaber	2023-01-16	including business error clarification and required codes.
1.2	Abdulaziz Aljaber	2023-03-14	Adding Attachment rules

Version	Approver	Date
1.2	Muhannad Almufarriji	2023-03-14

1. Table of Contents

2. Introduction

2.1 The purpose of this document:

The purpose of this document is to guide providers, consumers and integration developers on how to integrate with **ZATCA Integration Services**. The following topics are ZATCA Integration standardized where providers and consumers should follow when providing services or consumer them from/to ZATCA Integration.

3. System integrations

When an internal system needs to integrate with other systems, either internally or externally, the Integration team should be involved as a middleware layer between two or more systems. The middleware layer acts as a translator between the different systems, ensuring that they can communicate with each other effectively and securely.

The consumer of the middleware layer should follow the middleware layer's design, not the service provider's design. As middleware will do the required transformation of the services to meet ZATCA Integration needs and goals.

Any point-to-point Integration between two systems without Integration layer are not allowed and must be through ZATCA Integration only.

The middleware layer can perform a variety of transformations on the data and service, including and not limited to:

Transform from XML to Json.

Changes on the parameter’s names.

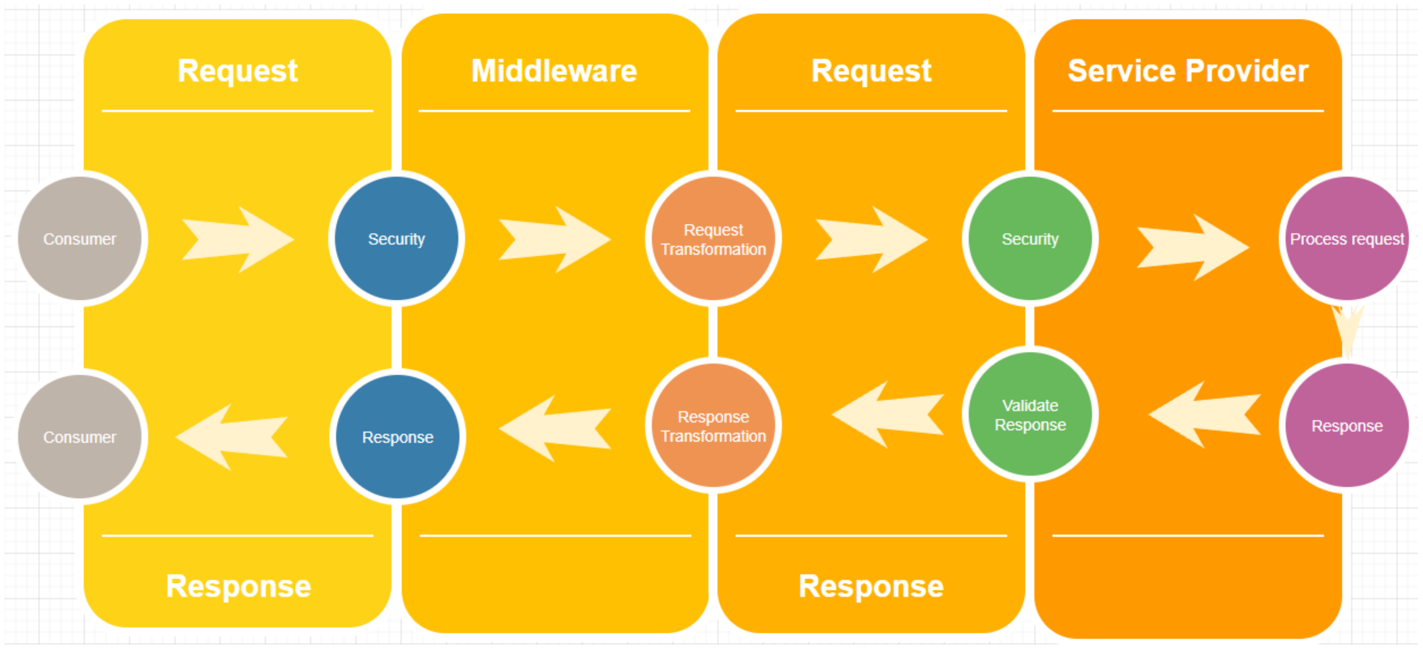
Adding different objects name.

Schema Design of the service’s response.

Schema Design of the service’s request.

These are just a few examples of the transformations that the middleware layer can perform.

The specific transformations that are performed will depend on the specific needs of the services based on middleware’s view and requirements.



3.1 As Provider:

Clients who become service providers to ZATCA, either internally or externally, must provide certain prerequisites in advance. These prerequisites should be shared exclusively in one-on-one emails with the ZATCA Integration team. These prerequisites are necessary to initiate the development process for the required services.

3.1.1 Prerequisite:

The prerequisites that are required from service providers will depend on the specific needs of the services, as determined by the Integration team's analysis. The most common items that are required are listed below. Any additional requirements will be raised by the Integration team and shared with the providers.

#	Item Name	Mandatory	Description
1	Integration documents and supportive files	Required	Integration documents should include and not limited to 1- **Brief of services description and what does it do:** This section should provide a brief overview of each of the services that are being integrated. This includes the service's purpose, the service's inputs and outputs, and the service's business logic related to the service. This is required to build the required transformation and validation of the business logic in the integration layer as well. For example, if certain parameter accepts certain value if other related value is equal to certain value. 2- **How to Integrate with and in which type (Rest or SOAP APIs):** This section should describe how to integrate with the services. This includes the protocols that are used, the endpoints that are used, and the authentication mechanisms that are used. 3- **All the Parameters found in a service should be clear in name, description, type, max/min/ length, values, mandatory or optional, and business scenario if any:** This section should provide a detailed description of all the parameters that are used by the services. This includes the parameter's name, description, type, maximum and minimum length, possible values, whether the parameter is mandatory or optional, and any business rules that apply to the parameter. 4- **URLs of the services for all environments:** This section should list the URLs of the services for all the environments that the services are deployed to. This includes the development environment, the staging environment, and the production environment. 5- **Operation type (GET, POST…):** This section should list the operation type for each of the services. This includes the HTTP methods that are used by the services, such as GET, POST, PUT, and DELETE. In addition to the information above, the below should be shared with the above details. • **In case if the services are in Rest:** Swagger file will be required. This file provides a detailed description of the RESTful APIs, including the endpoints, the parameters, and the response formats. • **In case if the services are in SOAP**: WSDIL file will be required. This file provides a detailed description of the SOAP APIs, including the messages, the bindings, and the namespaces.
2	Samples of Request and response	Required	Samples of request and response for all services found in the Provider's integration document.

- **Success scenario**: This is a sample request that would be successful. It should include the correct parameters and values, and it should produce the expected results.
- **Failed scenarios:** These are samples of requests that would fail. They may have incorrect parameters, invalid values, or technical issues.
- **Business error scenarios**: These are samples of requests that would fail due to a business error. For example, a request may fail if a certain value does not meet certain business condition.
- **Other specific/generic errors**: These are samples of requests that would fail due to other errors which might be technical, functional or validation.

Example Failed scenarios:

Request: Get https://base-url/endpoint/get-user-by-id?id=1

Response in case something went wrong:

```
{
  " errorCode ":”0001”,
  "message": ” something went wrong”

}
```

In this example, the code 0001 will always represent the error for something went wrong while processing the request.

Example Business scenarios:

Request: Get https://base-url/endpoint/get-bill-by-id?id=1

Response in case bill is found but can’t retrieve it as the bill is not yet paid:

```
{
  " errorCode ":”0002”,
  "message": ”Bill is not yet paid”

}
```

In this example, the code 0002 will always represent the error for bill not yet paid.

Example specific/generic scenarios:

Request: Get https://base-url/endpoint/get-all-users-by-protCode?portCode=99

Response in case port code is wrong:

```
{
  " errorCode ":”0003”,
  "message": ”Bill is not yet paid”

}
```

In this example, the code 0002 will always represent the error for port code is not correct.

3	Authentication	Required	Authentication details should be shared from the service provider in one-on-one email to integration team only for all environments.

4	Business Errors and other Error codes	Required	In case of Business or other types of errors during processing the request, the provider must return a clear error description and a one-to-one relationship error code. Dynamic messages are not allowed.
			• Use a clear and concise error message. The error message should be easy to understand and should provide enough information to understand the cause of the error.
			• Use a one-to-one relationship between the error code and the error message. This means that each error code should have a corresponding error message that describes the error in detail.
			For example:
			Request: Get https://base-url/endpoint/get-user-by-id?id=1
			Response in case user is not found:
			`{`
			`" errorCode ":"1234",`
			`  "message": " user not found in database"`
			`}`
			In this example, the code 1234 will always represent the error for user not found.

5	Test Data	Required	Service provider must provide enough test data to be tested by the team while developing. For both optional and required parameters.

3.1.2 API Design consideration

In case the services Includes the below mentioned points then providers should follow them when providing services to ZATCA Integration.

1. API with File attachments

If a service includes file attachments, then it is best to separate the file attachments into a separate service and must be in base-64, since file attachments can be large and can have a significant impact on the performance of the main service.

By separating the file attachments into a separate service/API, the main service can be kept lightweight and performant.

2. Large list of data

If a service is returning a large amount of data as an array "List", then it is best to include pagination in the service. This will allow clients to request data in smaller chunks, which can improve performance and usability. Pagination should be implemented from the service provider.

3.2 As Consumer:

The method of communication between ZATCA and the client will be via a Representational State Transfer (REST) API using JSON message format for the payload other ways if its changed client will be informed by Integration team. Clients who become service consumer with ZATCA, either internally or externally, must follow the required details and actions to consume the required service.

- Clients should be able to reach the integration network servers. Clients may be required to raise a ticket to the security team to open the connection between the client and the integration servers in all environments.
    a. The client should be able to reach the integration network servers. This means that the client's network should be able to communicate with the integration network servers.
    b. The client may be required to raise a request to the security team. This is because the security team may need to open firewall ports or make other changes to the network configuration in order to allow the client to connect to the integration servers.
    c. The request should be raised in all environments. This means that the request should be raised for the development environment, the staging environment, and the production environment.
- Clients should be able to access the Integration Developer Portal. To do this, they must first identify someone from their team as a team lead / admin. The team lead / admin will be responsible for maintaining access to the Integration Developer Portal for their team. Once they have been granted access, they will be able to perform the following tasks:
    a. **View products:** The administrator can view a list of all the products and services that are available in the Integration Developer Portal.
    b. **Create apps**: In a stage and development environment, the team lead / admin can create apps that will allow them to integrate their systems with the ZATCA Integration platform. However, in Production environment. The Integration Operation team will prepare the required to grant access to the required products and services.
    c. **Subscribe to products and services:** The administrator can subscribe to products and services that they need to integrate their systems with the ZATCA Integration platform.
    d. **Create API tokens and keys:** The administrator can create API tokens and keys that will allow them to access the services that they have subscribed to.
    e. **Download swagger documents:** The administrator can download swagger documents that describe the APIs that they are using.

f. **Test the services**: The administrator can test the services that they are using to ensure that they are working correctly.

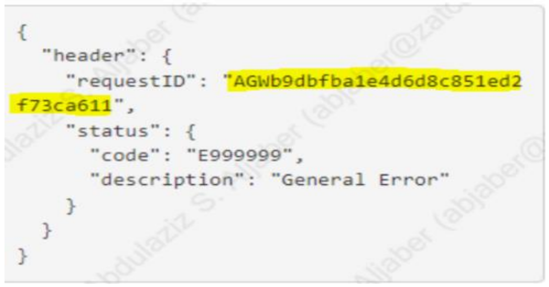
Here are some additional things to keep in mind:

- The administrator should only add members to organization which are belongs to the team.
- Tokens and API keys must not be shared in public or private to other individuals who are not belonging to their team, this include pasting them in public emails, threads, Jira or any other places, this includes Client-ID and Client-Secret in all available environments. Please note that sharing your credentials can put your keys and apps at risk. If your credentials are exposed, the Integration team will immediately delete them for safety. This will impact your ability to use your application and API keys until you receive new credentials from Integration team.

3.1Integration Support

In case support is needed or any errors while calling the APIs please contact ZATC Integration team zatcaIntegration@zatca.gov.sa

To investigate into your issue, you are kindly requested to provide the **requestID** found in the header of the response as shown.



```
{
  "header": {
    "requestID": "AGWb9dbfba1e4d6d8c851ed2
f73ca611",
    "status": {
      "code": "E999999",
      "description": "General Error"
    }
  }
}
```

We also recommend sharing a full log using the below format.

Request ID:	**AGWbef3a3b0c45449797738458dab4ba**
Environment:	**Staging**
Service Name:	**employees/search**
URL:	**-**
Timestamp	**-**
Error Code	E999999
Request: **-**	**Response:** `{ "header": { "requestID": "AGWbef3a3b0c45449797738458dab4ba", "status": { "code": "E999999", "description": "General Error" } } }`