Technical Documentation for the integration of the Appointments API

econocom
Products & Solutions

Index

1. Introduction

This document provides a detailed guide for implementing the creation of appointments in your system using a call to our API. The client must authenticate the user and send a POST request to create an appointment in our system. This guide includes authentication, obtaining the token and using that token to make authenticated requests.

Enviroments

Initially we will have a test environment whose access URL will be *https://dev03.api.altabox.net*. Once the tests and possible improvements that have arisen have been completed, the final production URL *https://altabox.realdooh.com* will be used.

Test enviroment	*https://dev03.api.altabox.net*
Prod enviroment	*https://altabox.realdooh.com/nexus*

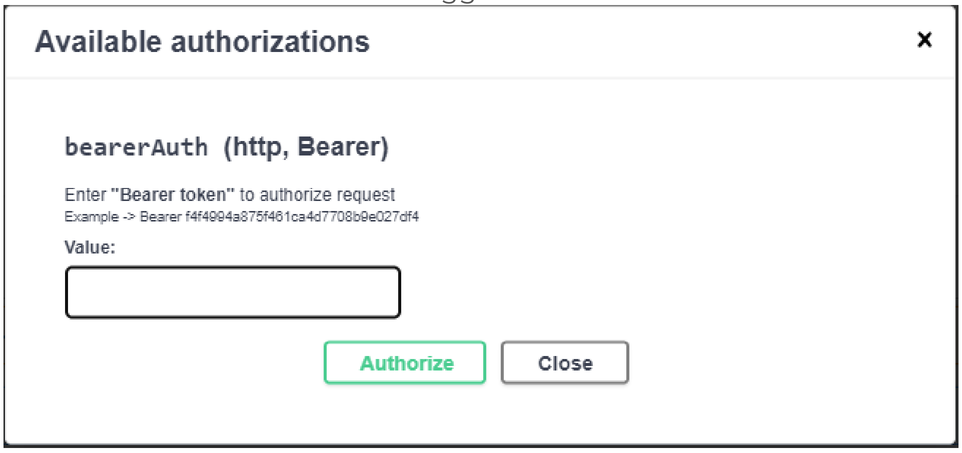
Online documentation

RealDooh Swagger will be available for the Appointments and Sentinel APIs in the test environment at the url: Swagger UI

Credentials: desarrollo / 1lt1b4x

In order to test the requests, it will be necessary to provide a valid authentication token. This token is generated by using the client_id credentials: "api" and client_secret: "apiSecret" this will be explained further in the documentation.

Table 1 - Swagger authorization



Available authorizations

bearerAuth (http, Bearer)

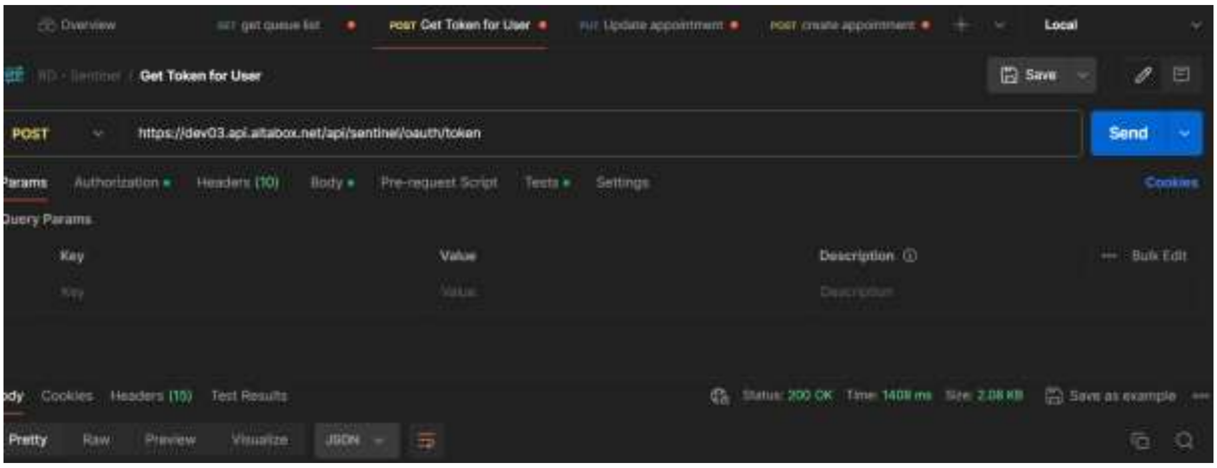
Enter "Bearer token" to authorize request
Example -> Bearer f4f4994a875f461ca4d7708b9e027df4

Value:

2. Authentication

To make any request to our API, it is necessary to first authenticate and obtain an access token (JWT). The authentication process is done through the Sentinel API, which is responsible for securing and managing users.

Endpoint: *https://dev03.api.altabox.net/api/sentinel/oauth/token*



POST https://dev03.api.altabox.net/api/sentinel/oauth/token

Status: 200 OK Time: 1408 ms Size: 2.08 KB

Table 2 - Body (x-www-form-urlencoded):

Name	Type	Description	Mandatory
username	string	The client's username.	Yes
password	string	The client's password.	Yes
grant_type	string	Must be "password".	Yes

Table 3 – Get token request in curl format

```
curl -X POST \
  https://dev03.api.altabox.net/api/sentinel/oauth/token \
  -H "Authorization: Basic <base64(client_id:client_secret)>" \
  -H "Content-Type: application/x-www-form-urlencoded" \
  -d "username=<username>&password=<password>&grant_type=password"
```

The client_id credentials: "*api*" and client_secret: "*apiSecret*" serve to allow initial access to perform the user search and authenticate against our user management API it is the combination of the credential codified in base64.

Table 4 - Get token successful response

```
{
"access_token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX25hbWUiOiJUZXN0Iiwic2NvcGUi
OlsiYWxsIl0sImFwZWxsaWRvIjoiTW9yZW5vIE1hcnTDrW4iLCJjb3JyZW8iOiJwb3N0bWFuQG
Vjb25vY29tLmNvbSIsImV4cCI6MTcxNjk5NTY5OCwibm9tYnJlIjoiQW50b25pbyAiLCJhdXRo
b3JpdGllcyI6WyJWTV9VU0VSIiwiQlNUX0FETUlOIl0sImp0aSI6IjY1OWQwNTg0LTdmNDAtND
gzNi1iMDk1LTQxMmM5OThhZTBhMCIsImNsaWVudF9pZCI6ImFwcGNsaWVudGUifQ.gUbRGOonv
le9P_YFu6qH-ih-NSTO6-BE5c8QTzif2AU",
"token_type":                "bearer",                  "refresh_token":"
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX25hbWUiOiJ0ZXN0Iiwic2NvcGUiO
lsiYWxsIl0sImFwZWxsaWRvIjoiTW9yZW5vIE1hcnTDrW4iLCJjb3JyZW8iOiJwb3N0bWFuQGV
jb25vY29tLmNvbSIsImF0aSI6IjY1OWQwNTg0LTdmNDAtNDgzNi1iMDk1LTQxMmM5OThhZTBhM
CIsImV4cCI6MTcxNjk5NTY5OCwibm9tYnJlIjoiQW50b25pbyAiLCJhdXRob3JpdGllcyI6WyJ
WTV9VU0VSIiwiQlNUX0FETUlOIl0sImp0aSI6IjEyYWZkMzNmLWNlNWYtNDk1ZS04YmY4LWJjM
zU4YjQ3MjljNyIsImNsaWVudF9pZCI6ImFwcGNsaWVudGUifQ.VjKCIlsvsWsGXPHVm3TAthgK
WK6AV2LjlY7kXV6Fmqg",
"expires_in": 86399,
"scope": "all",
"apellido": "Test ",
"correo": "postman@econcom.com",
"nombre": "Antonio ",
"jti": "659d0584-7f40-4836-b095-412c998ae0a0"
}
```

Token usage

Once the token (access_token) is obtained, it will be used in the header of all subsequent requests to authenticate the user.

Table 5 - Headers for Authenticated Requests

```
-H "Authorization: Bearer <access_token>"
-H "Content-Type: application/json"
```

3. Appointment class

The appointment object could include the following fields, depending on the implementation:

Field	Type	Description	Mandatory
initialTime	LocalDateTime	Initial date and time of the appointment	No
inTime	LocalDateTime	Entry date and time.	Yes
arrivalTime	LocalDateTime	Date and time of arrival.	No
calledInTime	LocalDateTime	Date and time of the incoming call.	No
calledOutTime	LocalDateTime	Date and time of the outgoing call.	No
outTime	LocalDateTime	Output date and time.	Yes
queueId	Long	ID of the queue (1)	Yes
servicePointId	Long	ID of the service point. (1)	Yes
userId	Long	User ID.	Yes
codUser	String	User code.	No
ticketId	Long	ID of the ticket.	No
appointmentStatus	String	Status of the appointment	No
reservedTime	LocalDateTime	Reserved date and time.	No
verification_code	String	Verification code.	No
mode	Integer	Appointment mode.	No
smsRemiderDate	LocalDateTime	Date and time of the SMS reminder.	No

All the LocalDateTime fields have YYYY-MM-DDTHH:MM:SS. format.
The appointment object that we use will also have an extra attribute called variables, which is an array of variable type objects that will have two attributes. One of them is value and the other is variableType, with value being the attribute where we will report the vehicle license plate for the appointment and variableType a unique id (11001 for the license plate and 11005 for the owner's full name) that will help us identify the information of the owner of the appointment(*Required).

4. Create an appointment

To create an appointment, we need some mandatory fields, one of them is the queueid or the store id. To do this, we need to make a call to the bwin API to obtain the list of stores and be able to select and inform the store where we want to register the appointment.

Endpoint:
https://dev03.api.altabox.net/api/appointments/api/appointments/multiple

Table 6 – Create appointment parameters

Name	Type	Description	Mandatory
mode	string	The mode of the appointment.	No
maxNumAppointments	boolean	Boolean to define the maximum number of appointments. (true)	No
numStations	integer	Number of stations.	No
verifySlot	boolean	Boolean to verify the slot.	No

Table 7 – Create appointment body

Name	Type	Description	Mandatory
appointment	Appointment	JSON object representing the appointment.	Yes

This is an example of the minimum and mandatory fields that an appointment object must have.

Table 8 – Appointment minimum mandatory fields on JSON format

```
{
        "inTime": "2023-06-25T14:30:00",
        "onTime": "2023-06-25T15:30:00",
        "servicePointId": 1,
        "queueId": 1,
        "userId": 428,
        "appointmentStatus": "RESERVATION",
         "variables": [{ "value": "Jhon Doe – Appointment",
                         "variableType":{"id":11005}
                      },
                      {"value": "9573BXD",
                         "variableType":{"id":11001}
                      }
                      ]
 }
```

The parameter servicePointId will always be the same but you'll need to obtain the list of all the queues of the system to indicate in which store the appointment will be created.

Get 1ueue list (stores)

To get the stores the client must send a GET request.
Endpoint: *https://dev03.api.altabox.net/api/bewin/api/queue/list*

Table 9 - Get list of stores request in curl format

```
curl -X GET\
  https://dev03.api.altabox.net/api/bewin/api/queue/list \
  -H "Authorization: Bearer <access_token>" \
```

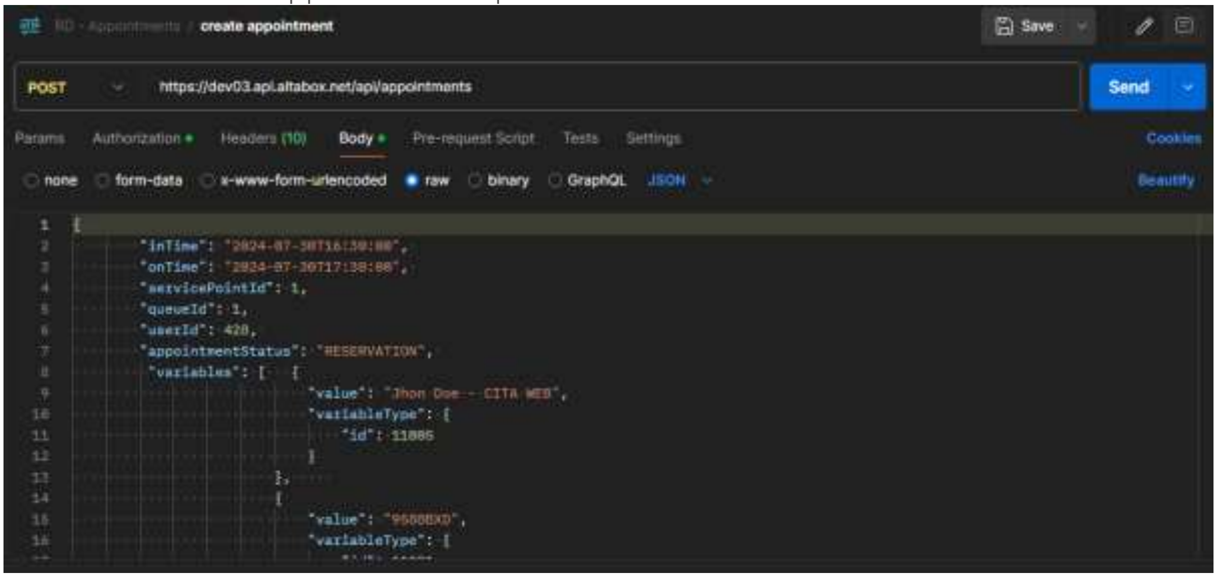
Table 10 - Get list of stores successful response

```
{
    "data": [
        [
            {
                "createdAt": "2024-05-27T10:26:52",
                "createdBy": 1,
                "active": true,
                "id": 1,
                "name": "Oficina de pruebas",
                "description": "Oficina de pruebas",
                "shortName": "ODP",
                "openingHours": "09:00 - 19:00",
                "resetQueueHour": 1,
                "numberOfStations": 0,
                "email": "sat@client.com",
                "phone": "900000000",
                "internalCode": "1",
                "idCode": "00000000",
                "stations": []
            }
        ]
    ],
    "achieved": true,
    "details": null
}
```

Table 11 – Create an Appointment request in curl format

```
curl -X POST \
https://dev03.api.altabox.net/api/appointments/api/appointments/multiple
  -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d ' {
        "inTime": "2023-06-25T14:30:00",
        "outTime": "2023-06-25T15:30:00",
        "servicePointId": 1,
        "queueId": 1,
        "userId": 428,
        "appointmentStatus": "RESERVATION",
         "variables": [{ "value": "Jhon Doe – Appointment",
                         "variableType":{"id":11005}
                      },
                      {"value": "9573BXD",
                         "variableType":{"id":11001}
                      }
                      ]
      }'
```

Table 12 - Create an Appointment request in Postman



RD - Appointments / create appointment

Save

POST https://dev03.api.altabox.net/api/appointments

Send

Params Authorization Headers (10) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
{
        "inTime": "2024-07-30T16:30:00",
        "onTime": "2024-07-30T17:30:00",
        "servicePointId": 1,
        "queueId": 1,
        "userId": 428,
        "appointmentStatus": "RESERVATION",
         "variables": [   {
                          "value": "Jhon Doe - CITA WEB",
                          "variableType": {
                              "id": 11005
                          }
                       },
                       {
                          "value": "9588BXD",
                          "variableType": {
```

Table 13 - Create an Appointment successful response

```
{
   "createdAt": "2024-05-28T12:49:43",
   "active": true,
   "id": 286714,
   "inTime": "2024-05-29T14:30:00",
   "outTime": "2024-05-29T15:30:00",
   "queueId": 1,
   "servicePointId": 1,
   "userId": 428,
   "appointmentStatus": "RESERVATION",
   "variables": []
}
```

Expected Responses

- 200 OK: Appointment created successfully.
- 409 Conflict: Duplicate quote.
- 412 Precondition Failed: Syntax error.
- 400 Bad Request: Bad request.
- 428 Precondition Required: Required token.
- 500 Internal Server Error: Internal server error.

5. Update an appointment

To modify an appointment, the client must send a PUT request to the corresponding endpoint.

Endpoint *https://dev03.api.altabox.net/api/appointments/api/appointments*

Table 14 - Appointment update parameters

Name	Type	Description	Mandatory
mode	string	The mode of the appointment.	No

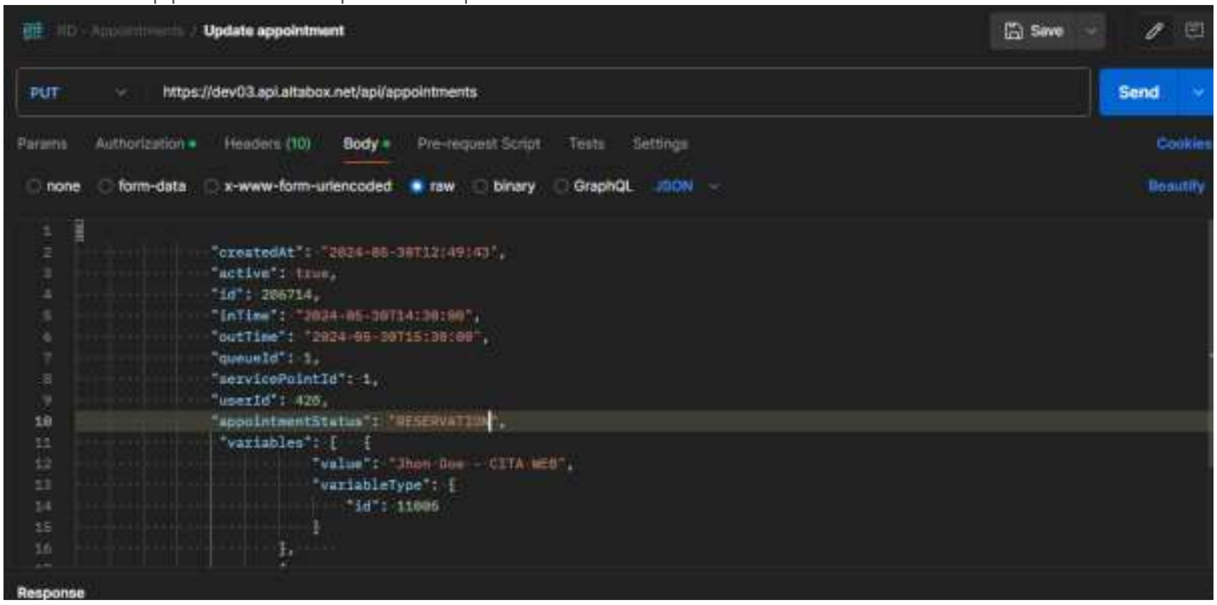
Table 15 - Appointment update body

Name	Type	Description	Mandatory
appointment	Appointment	JSON object representing the appointment.	Yes

Table 16 – Update appointment dates request in curl format

```
curl -X PUT \
   https://dev03.api.altabox.net/api/appointments/v1/appointments \
  -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
  -d ' {
        "inTime": "2023-06-30T14:30:00",
        "outTime": "2023-06-30T15:30:00",
        "servicePointId": 1,
        "id": 286714,
        "queueId": 1,
        "userId": 428,
        "appointmentStatus": "RESERVATION",
         "variables": [{ "value": "Jhon Doe – Appointment",
                        "variableType":{"id":11005}
                     },
                     {"value": "9573BXD",
                        "variableType":{"id":11001}
                     }
                     ]
      } '
```

Table 17 - Appointment update request in Postman



PUT https://dev03.api.altabox.net/api/appointments

```
{
        "createdAt": "2024-05-28T12:49:43",
        "active": true,
        "id": 286714,
        "inTime": "2024-05-29T14:30:00",
        "outTime": "2024-05-29T15:30:00",
        "queueId": 1,
        "servicePointId": 1,
        "userId": 428,
        "appointmentStatus": "RESERVATION",
         "variables": [   {
                    "value": "Jhon Doe - CITA WEB",
                    "variableType": {
                        "id": 11006
                    }
                },
```

Table 18 - Appointment update succesful response

```
{
    "createdAt": "2024-05-28T12:49:43",
    "updatedAt": "2024-05-28T13:32:57",
    "active": true,
    "id": 286714,
    "inTime": "2024-05-29T14:30:00",
    "outTime": "2024-05-29T15:30:00",
    "queueId": 1,
    "servicePointId": 1,
    "userId": 428,
    "appointmentStatus": "CANCEL",
    "variables": []
}
```

Expected Responses

- 200 OK: Appointment successfully modified.
- 409 Conflict: Duplicate quote.
- 412 Precondition Failed: Syntax error.
- 400 Bad Request: Bad request.
- 428 Precondition Required: Required token.
- 500 Internal Server Error: Internal server error.

6. Delete an appointment

To delete an appointment, the client must send a DELETE request to the corresponding endpoint.

Endpoint: *https://dev03.api.altabox.net/api/appointments/api/appointments.*

Parameters:

Name	Type	Description	Mandatory
id	number	Id of the appointment	Yes

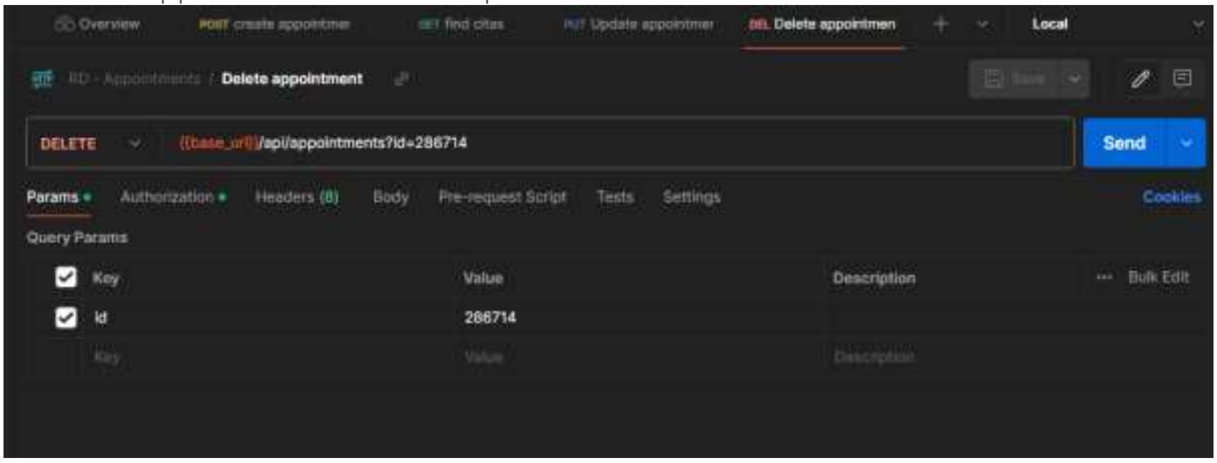
Table 19 - Appointment deletion request in curl format

```
curl -X DELETE \

https://dev03.api.altabox.net/api/appointments/api/appointments?id=286714\

  -H "Authorization: Bearer <access_token>" \
  -H "Content-Type: application/json" \
```

Table 20 - Appointment deletion request on Postman



DELETE {{base_url}}/api/appointments?id=286714

Query Params

Key	Value	Description
id	286714	

Expected Responses

- 200 OK: Appointment successfully deleted.
- 500 Internal Server Error: Internal server error.

7. Implementation in different languages

The following section provides examples of how to make these requests in several common programming languages.

Example in JavaScript (Using Fetch API)

```
// Authentication
fetch(https://dev03.api.altabox.net/api/sentinel/oauth/token', {
  method: 'POST',
  headers: {
    'Authorization': 'Basic ' + btoa(client_id + ':' + client_secret),
    'Content-Type': 'application/x-www-form-urlencoded'
  },
  body: new URLSearchParams({
    'username': username,
    'password': password,
    'grant_type': 'password'
  })
})
.then(response => response.json())
.then(data => {
  const token = data.access_token;

  // Appointment creation
  return fetch('https://dev03.api.altabox.net /api/appointments/multiple', {
    method: 'POST',
    headers: {
      'Authorization': 'Bearer ' + token,
      'Content-Type': 'application/json'
    },
    body: JSON.stringify({
      appointmentDate: '2023-06-25T14:30:00',
      userId: 12345,
      description: 'Cita de prueba'
    })
  });
})
.then(response => response.json())
.then(data => console.log('Cita creada:', data))
.catch(error => console.error('Error:', error));
```

Example in Java

```
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.Base64;
import java.io.OutputStream;
import java.io.InputStreamReader;
```

```java
import java.io.BufferedReader;
import org.json.JSONObject;
// Authentication
URL url = new URL("https://dev03.api.altabox.net/api/sentinel/oauth/token");
HttpURLConnection conn = (HttpURLConnection) url.openConnection();
conn.setRequestMethod("POST");
conn.setRequestProperty("Authorization", "Basic " +
Base64.getEncoder().encodeToString((clientId + ":" + clientSecret).getBytes()));
conn.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");
conn.setDoOutput(true);

String params = "username=" + username + "&password=" + password +
"&grant_type=password";
OutputStream os = conn.getOutputStream();
os.write(params.getBytes());
os.flush();
os.close();

BufferedReader br = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
String response = br.readLine();
br.close();

JSONObject jsonResponse = new JSONObject(response);
String token = jsonResponse.getString("access_token");

// Appointment creation
url = new URL("https://dev03.api.altabox.net/api/appointments/multiple");
conn = (HttpURLConnection) url.openConnection();
conn.setRequestMethod("POST");
conn.setRequestProperty("Authorization", "Bearer " + token);
conn.setRequestProperty("Content-Type", "application/json");
conn.setDoOutput(true);

JSONObject appointment = new JSONObject();
appointment.put("appointmentDate", "2023-06-25T14:30:00");
appointment.put("userId", 12345);
appointment.put("description", "Cita de prueba");

os = conn.getOutputStream();
os.write(appointment.toString().getBytes());
os.flush();
os.close();

br = new BufferedReader(new InputStreamReader(conn.getInputStream()));
response = br.readLine();
br.close();

System.out.println("Cita creada: " + response);
```

8. Conclusion

This documentation provides all the details necessary for the customer to integrate appointment creation into their system using our API. If you have any questions or need additional assistance, please feel free to contact our technical support team (LD_COM_REALWINEPS@econocom.com).