Successware   
API   
Documentation

Table of Contents

[Overview 2](#_Toc118472339)

[Assumptions 2](#_Toc118472340)

[Prerequisites 2](#_Toc118472341)

[Authentication and Token Generation API Endpoint 2](#_Toc118472342)

[GraphQL Console URL – To Run Queries and Mutations 3](#_Toc118472343)

[GraphQL Console – Required, Optional, Enum Values in Comments 4](#_Toc118472344)

[GraphQL Console – Query Example 4](#_Toc118472345)

[GraphQL Console – Query with Parameters Example 5](#_Toc118472346)

[GraphQL Console – Mutation Example 8](#_Toc118472347)

[Invoke Tenant Registration 10](#_Toc118472348)

[Invoke Credential Confirmation 11](#_Toc118472349)

# Overview

This Successware API document describes the process and steps to use the API endpoints with a few examples: tenant registration and credential confirmation. To use the Successware API, one would require credentials. The exchanging of credentials with the third-party integrators is automated with a handshake in the Successware Web Application using the Add-On Manager wizard.

## Assumptions

As this document targets the technical and development team, it is assumed that the team has the proper credentials (shared manually via email, etc.) to authenticate, execute the steps, and run the samples provided in the below sections.

## Prerequisites

A set of tools are required to perform the steps and run the examples.

* Postman or any other preferred tool to send and receive the RESTful API endpoints
* Chrome or any preferred Web Browser for GraphQL to execute queries and mutations

The latest version of the tools would be appropriate.

# Authentication and Token Generation API Endpoint

The first step in using the Successware API starts by generating a valid token using the following RESTful endpoint and Postman or other preferred tools.

In the Postman, fill the following:

1. Use the login endpoint <https://publicapi-dev.successwareg2.com/api/login> with POST
2. Enter the following JSON in the Body, update the credential information.

{

    "username": "addonname@tenantid.com",

    "password": "Password"

}

Assumption: Agent username and password is provided and available.

1. The service response would include a valid token that is used for further API calls

Graphical user interface, text, application, email

Description automatically generated

**Note**:

* Copy the “access\_token” value in the response. Example Postman script
* The “error\_message” would have a value in case of error
* The “expires\_in” would be in seconds and the token is valid up to an hour

# GraphQL Console URL – To Run Queries and Mutations

Successware provides and shares a GraphQL Console with third-party integrators that exposes various queries and mutations required for custom integrations. Please use the GraphQL Console URL to perform a query or mutation based on the environment.

Login URL List:

|  |  |
| --- | --- |
| Environment | URL |
| DEV Login URL | https://[publicapi-dev.successwareg2.com/api/login](https://publicapi-dev.successwareg2.com/api/login) |
| QA Login URL | https://[publicapi-qa.successwareg2.com/api/login](https://publicapi-qa.successwareg2.com/api/login) |
| UAT Login URL | https://[publicapi-uat.successwareg2.com/api/login](https://publicapi-uat.successwareg2.com/api/login) |
| RC Login URL | https://publicapi-rc.successwareg2.com/api/login |
| Prod Login URL | https://publicapi.successwareg2.com/api/login |

SW G2API URL List:

|  |  |
| --- | --- |
| Environment | URL |
| DEV API URL | https://publicapi-dev.successwareg2.com/api/graphql |
| QA API URL | https://[publicapi-qa.successwareg2.com/api/graphql](https://publicapi-qa.successwareg2.com/api/graphql) |
| UAT API URL | https://[publicapi-uat.successwareg2.com/api/graphql](https://publicapi-uat.successwareg2.com/api/graphql) |
| RC API URL | https://[publicapi-rc.successwareg2.com/api/graphql](https://publicapi-rc.successwareg2.com/api/graphql) |
| Prod API URL | https://publicapi.successwareg2.com/api/graphql |

GraphQL Console/Playground URL List:

|  |  |
| --- | --- |
| Environment | URL |
| DEV GraphQL Console | <https://publicapi-dev.successwareg2.com/> |
| QA GraphQL Console | https://[publicapi-qa.successwareg2.com/](https://publicapi-qa.successwareg2.com/) |
| UAT GraphQL Console | [https://[publicapi-uat.successwareg2.com/](https://publicapi-uat.successwareg2.com/)](https://swgraphql-docs-uat.successwareg2.com/v2/new) |

## GraphQL Console – Required, Optional, Enum Values in Comments

Graphical user interface, application

Description automatically generated

The “DOCS” drawer would show the available GraphQL queries and mutations. Selecting a query would show the expected inputs and response types and selecting the type would provide the details and expected values.

In GraphQL, the required input fields are noted as “jobClass: String!” and optional are without “!”. Graphical user interface, application

Description automatically generated

## GraphQL Console – Query Example

To run a query in the GraphQL Console, follow the following steps.

1. Open the GraphQL Console in the browser using the URL <https://publicapi-dev.successwareg2.com/>
2. Use the generated token from the previous “Authentication and Token Generation API Endpoint” section.
3. This token authenticates all the API endpoints on the GraphQL layer. First, in the GraphQL console, in the HTTP Header section, add an "authorization" tag and append the token (generated in the previous section) with the "Bearer " prefix in the header of the request. As shown below.

Graphical user interface, application, Word

Description automatically generated

1. In the query pane of the GraphQL Console, use the query tag as in the example below:

query {

findAllVisitTypes {

id

type

description

estimatedManHour

isActive

}

}

1. Once the required query statement is ready, click on the "Run/Execute" Graphical user interface, application, Teams

   Description automatically generated button, and in the results pane, we can see the service response.

Graphical user interface, application, Teams

Description automatically generated

## GraphQL Console – Query with Parameters Example

The GraphQL Console also gives the ability to execute a query with parameters. Following is an example of how a query with parameters can be executed.

Query:

query SearchAgreement($input: searchViewAgreementRequest!, $sort: [String], $page: Int!, $size: Int!) {  
searchAgreement(input: $input, sort: $sort, page: $page, size: $size) {  
totalElements  
totalPages  
pageSize  
pageNumber  
numberOfElements  
content {  
id  
agreementNumber  
agreementTypeName  
perpetual  
createdDate  
saleDate  
startDate  
endDate  
totalPrice  
taxBill  
deposit  
billFrequency  
billAmount  
collectPerVisit  
renewal  
activated  
fulfilled  
canceled  
terminated  
lastName  
companyName  
address1  
locationId  
billingCustomerId  
salesPerson  
year  
visitsLeft  
nextVisitMonth  
nextVisitYear  
lastBilling  
nextBilling  
billDayOfMonth  
methodOfPayment  
creditCardExpireMonth  
creditCardExpireYear  
compBillingCount  
terminateDate  
city  
state  
zipCode  
phoneNumber  
extension  
email  
preferredTech  
invoiceNumber  
invoiceDate  
subTotal  
taxAmount  
totalAmount  
paymentAmount  
netBalance  
posted  
adjusting  
creditCardName  
type  
renewalNoticesCount  
lastNotice  
projectedMonth  
projectedYear  
estimatedHours  
callId  
visitPrice  
visitCanceled  
visitComplete  
visitInvoiced  
locationEquipAge  
billingMethod  
cover  
recurringId  
paymentType  
xchargePaymentEnabled  
achTransactionId  
approved  
balance  
renewalAccepted  
totalNumberBilling  
saleTypeCode  
renewalCreated  
\_\_typename  
}  
\_\_typename  
}  
}

Query variable:

{

"input": {  
"agreementNumber": null,  
"agreementTypeName": null,  
"fromDate": "",  
"toDate": "",  
"viewType": null,  
"acceptedRenewals": ""  
},  
"sort": "agreementNumber asc",  
"page": 0,  
"size": 10

}

|  |
| --- |
| A screenshot of a computer  Description automatically generated |

## GraphQL Console – Mutation Example

Following is an example of a mutation in the GraphQL Console.

Mutation alterRefBooks($input: ReferenceBookRequest!) {  
alterRefBooks(input: $input) {  
pageNumber  
pageSize  
tableName  
totalDataRow  
totalPages  
crudOperation {  
id  
status  
type  
\_\_typename  
}  
\_\_typename  
}  
}

{  
"input": {  
"tableName": "Zones",  
"rowData": {  
"rowIndex": 0,  
"column1": "BRANDI",  
"column2": "asa",  
"column3": "#200d0d",  
"column4": "true"  
},  
"crudOperation": "update"  
}  
}

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

# Invoke Tenant Registration

Successware will create a third-party integrator-specific user for a tenant, generate a random password and share these credentials with the integrator. These credentials will be saved in the Successware database and used to validate the registration process. Integrator can use the following service to authenticate and start using Successware APIs:

<https://swintegrate-dev.successwareg2.com/web/bedrock/login>

This service expects the following payload:

{

userName: "<AddonName@tenantid.com>"

password: "<password>"

}

To test your endpoint for Tenant-Registration, submit the Invoke Tenant Registration

In case of a successful authentication, you will receive a token in response similar to one below (do not copy the below token. It’s an example):

eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6InN3Z2VuMiJ9.eyJ0ZW5hbnRfaWQiOjE0NjI2OTQxNDI4NjcxNzY4LCJlcnJvcl9tZXNzYWdlIjpudWxsLCJ1c2VyX25hbWUiOiJ0ZXN0QHRlc3QuY29tIiwiaXNzIjoiaHR0cDovL2JlZHJvY2stdWFhLXNlcnZpY2Uuc3diZWRyb2NrLWRldiIsInRpbWVfem9uZSI6IkVTVDVFRFQiLCJjbGllbnRfaWQiOiJiZWRyb2NrX3dlYiIsIndhcm5pbmdfbWVzc2FnZSI6bnVsbCwicGFzc3dvcmQiOiJoYXNoKO-\_vVx1MDAwRl1SXHUwMDE377-9b--\_ve-\_.I\_\_cAA-euwYiaRefVQIG\_GE\_r5\_5iZZBIQv0D49SzgvHFjE5byXvL2-KMgJfEf\_0Q0VLaQGjMCfxWNb7aSCjHgImo5uR12lh5Mf7eOoatZ36Hk2HUSuepIHJPs2FHvw-R\_cYOpBQrVl7Xv-n15MaJrWHsM6TAVIhWR6ufDMG9bk

In case of invalid credentials, login failed message would appear.

Graphical user interface, text, application, email

Description automatically generated

# Invoke Credential Confirmation

Assumptions:

* The third-party integrator has shared the test account credentials with Successware.
* The third party addon has been published in Successware.
* The test account credentials have been saved in Successware database.

The third-party integrator must establish an endpoint that Successware can call to Validate the credentials.

The following query (example) will be used to call the third-party endpoint to the validate the credentials.

query {  
  validateCredentials(addonname:"CompanyCam"){  
    confirmed  
    message  
    redirectLink  
  }  
}

where addonName is the third party addon name (in the above example it is CompanyCam.)

Note: Tenant-id is extracted from the token.

Successware will send the Account-Confirmation payload to the third-party integrator’s endpoint.

    "confirmAccountResponse": {

        "confirmed": true,

        "message": "Client account confirmed",

        "redirectLink": ""

    }

}

Graphical user interface, text, application, email

Description automatically generated