EVOTIX

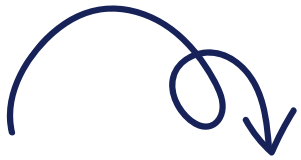
API Guide

Assure Customer API
User Guide: Getting Started

Evotix Ltd. Revision 2.0 (July 2024)

US +1 (872) 215 5913 **UK** +44 (0) 161 521 8490 **AU** +61 3 8595 5909 **Visit** evotix.com

EVOTIX



The Assure Customer API is the means by which Evotix provides integration capability to Assure for customers.

The Customer API is available to customers via the public internet and takes the form of a RESTful API. Using the Customer API you can automate processes such as managing users, org unit structure and exporting data for analysis. Making use of the Customer API requires a level of technical expertise so this is typically something that a company's IT function would handle.

This guide focusses on how to set up the Customer API and handle any errors you may come across. Separate guides exist for setting up each of the APIs.

EVOTIX

Contents

EVOTIX

Getting started with the Customer API

This getting start section covers getting setup to use the Customer API. This includes creating an API key and confirming that you can make a 'ping' request to the API. This guide should be followed before moving on to using the Customer API functionality such as User Management or Data Export.

Pre-requisites

1. Access is required to the Assure instance for which the Customer API is to be used. An Assure user account with permission to use the "System Settings -> API Access" is required. This means an account with the "Manage API" supervisor privilege.

2. Access is required to the system or platform from which the requests to the Customer API will be made. Ideally this should be the same system or platform environment that will be used to host the integration workflow. However, if this is not possible then a different system or platform environment can be used but this may mean that further testing will be needed to confirm access to the Customer API before using integration workflow in a 'production' setting.

Obtain an API Key

All requests to the Customer API require an API key. The API key authenticates access to the Customer. It is very important that API keys are stored securely and only accessible to authorised personnel as API keys grant access to all available functions and data (i.e. API keys basically have the same permissions as an Assure administrator user).

To obtain an API key follow the steps in the 'Adding and API key' section of the Customer Knowledge Base article on managing API keys.

Test connectivity

Having obtained an API key, connectivity to the Customer API can be tested using the 'ping' diagnostic API method, see the Customer API schema for more technical details of the 'ping' method. The 'ping' method is only to be used for diagnostic purposes such as during initial setup like this. The 'ping' API method does not require any data to be supplied and does not make any changes to Assure or reveal any data from within Assure. Hence it is a safe method to use for testing connectivity.

As mentioned above it is best to test connectivity from the actual system or platform where the integration workflow will run. The following details will be need to make the 'ping' API request (as well as the API key):

Detail	Notes
URL	The URL to be used takes the form: https://api.<stack>.sheassure.net/v1/ping Where <stack> is the name of the Assure stack (e.g. uk, uk2, anz, na, etc) which will be the same as used when you access the Assure Web UI.
HTTP method	HTTP methods are set of verbs which define the kind of action that the API request is to perform, see here for more details on HTTP verbs. The verb to use for the 'ping' request is GET.
API key header	The API key needs to be included in the HTTP headers of the request. The header that needs to be added is the x-api-key with its value being the API key.

The following sections show how to do the 'ping' request in a variety of software tools / platforms:

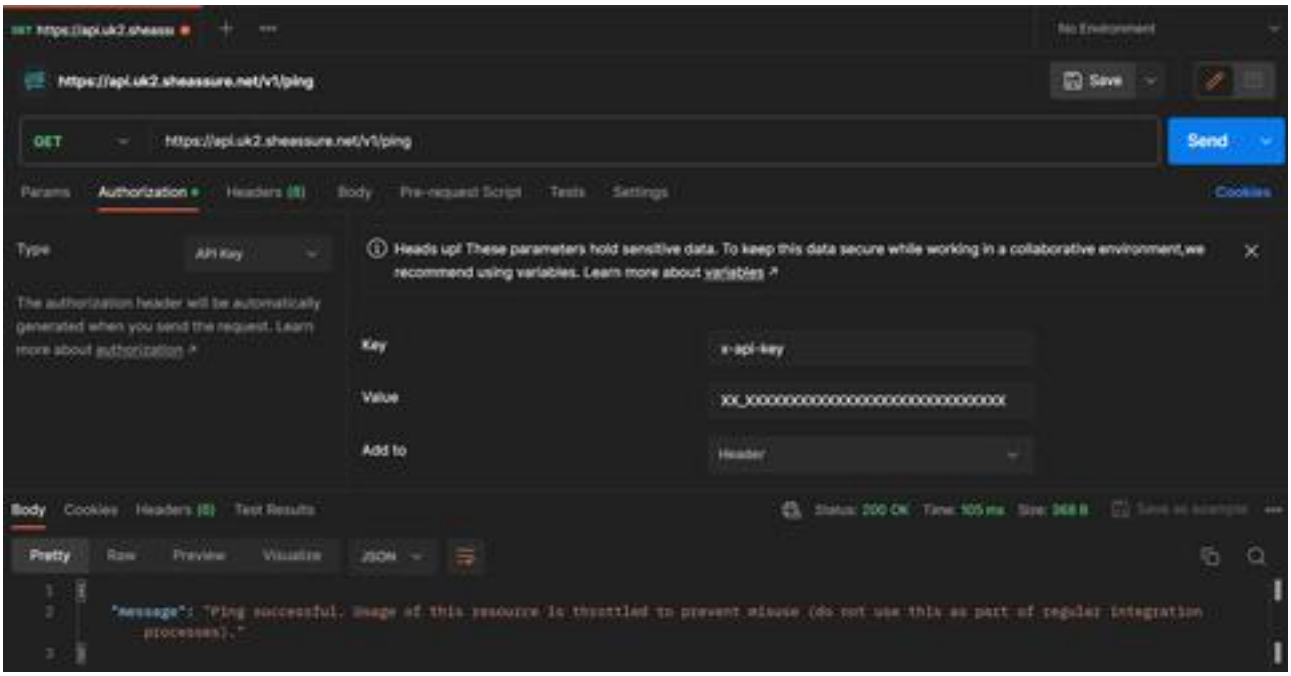
Whichever software tool or platform used the way to know that the 'ping' request has been **successful is that the reported HTTP response code will be 200** (sometimes reported as OK). If an error response is received then there may be a problem with the way the request is being made, or a problem with connectivity to the Customer API, or potentially a problem with the Customer API itself. See the next section (Handling errors from the Customer API) for more details.

Postman API platform

Postman API platform is a widely used tool for interacting with APIs. It is not a tool that would be used for creating integration workflows, however it is popular for exploring and experimenting with APIs. Postman can be used online or downloaded and run on a local machine. There is a guide from Postman showing how to create and send API requests.

The screenshot below shows a successful 'ping' request for a customer on the uk2 stack. Note the use of the 'Authorization' tab to configure the API key header. The XX_XXXXXXXXXXXXXXXXXXXXXXXXXXXX value is where the API key would be placed.

The response section at the bottom shows the result Status: 200 OK which indicates that the ping was successful (also displayed is the response message from the Customer API confirming the successful ping).



Windows Powershell

Windows Powershell is the command line shell provided with all modern versions of Windows. Powershell supports making API requests using the Invoke-WebRequest command and this can easily be used to 'ping' the Customer API. The code block below shows a successful 'ping' request for a customer on the uk2 stack. Note the use of the -Headers argument to include the API key header. The XXXXXXXXXXXXXXXXXXXX value is where the API key would be placed. The response from the command shows the StatusCode: 200 which indicates that the ping was successful (also displayed is the response message from the Customer API confirming the successful ping). NB: Some response lines have been removed for brevity.

```
PS C:\> Invoke-WebRequest -Headers @{'x-api-key' = 'XXXXXXXXXXXXXXXXXXXX'}
https://api.uk2.sheassure.net/v1/ping
StatusCode        : 200
StatusDescription : OK
Content           : {"message": "Ping successful. Usage of this resource is
throttled to prevent misuse (do not use
                    this as part of regular integration processes)."}
PS C:\>
```

EVOTIX

Python

Python is a popular programming language which is commonly used by IT teams when there is a need for more than just shell style scripting. The code block below shows the few lines of Python code required to make a ping request to the Customer API for a customer on the uk2 stack. The code throws an exception if the request fails, otherwise it prints out the response code to the console along with the content of the response from the Customer API. The XXXXXXXXXXXXXXXXXXXX value is where the API key would be placed.

```
# pip install requests
import requests

headers = { "x-api-key": "XXXXXXXXXXXXXXXXXXXX "}
r = requests.get("https://api.uk2.sheassure.net/v1/ping", headers=headers)

r.raise_for_status()
print(f"StatusCode={r.status_code}")
print(f"Body={r.content}")
```

The following shows the output from the above Python code being run where a successful response is generated. The StatusCode=200 indicates that the ping was successful.

```
StatusCode=200
Body=b'{"message": "Ping successful. Usage of this resource is throttled to prevent
misuse (do not use this as part of regular integration processes)."}'
```

EVOTIX

Handling errors from the Customer API

Inevitably errors will occur from time to time when making requests to an API. Therefore it is important to implement effective error handling so that the impact of errors can be minimised. The following sections give details to help understand error responses that might be encountered and guidance as to recommended handling of errors.

Understanding error responses

Broadly the error responses that could be encountered when making requests to the Customer API can be divided into errors relating to connectivity to the API, and errors that are generated by the API itself. The following sections explore these in more detail but it is important to make sure that error handling is in place for all types of errors that can occur.

Connectivity errors

The Customer API is available on the public internet which means that in order to be able to make a request to the API the software tool/platform needs access to the public internet for the purpose of making HTTPS requests to the API. The request also needs to be correctly setup in the software tool/platform so that it is addressing the correct API URL, using the correct API key, etc. When there is a problem with the setup or connectivity the error reporting is handled by the software tool/platform and the way this will be reported will be specific to the tool/platform. There may be a message in a log file, an exception may be thrown, or a function may return an error. It will be necessary to refer to the usage documentation for the software tool/platform or test the tool/platform by deliberately inducing errors to see how the result is reported.

The following table describes some common connectivity error conditions and what might cause them:

Condition	Possible cause	Notes
Failure to lookup API hostname	Incorrect setup/config	Make sure that the hostname part of the URL is correct in the tool/platform. Add logging to report the URL value just before the API request is made and verify that it is matches the expected value (double check against the documentation for what the hostname should be).

EVOTIX

Condition	Possible cause	Notes
	DNS service issue	Occasionally problems with the DNS servers supporting the server/platform being used can prevent hostname lookups. Verify this with a hostname lookup diagnostic tool and if the hostname fails to lookup then contact your IT support.
	Customer API platform issue	Although unlikely it is possible that a problem with the Assure infrastructure is preventing the hostname lookup working. Verify this with a hostname lookup diagnostic tool and if the hostname is definitely correct but fails to lookup then contact Evotix support.
Failure to connect to API hostname	Incorrect setup/config	Make sure that the hostname part of the URL is correct in the tool/platform. Add logging to report the URL value just before the API request is made and verify that it is matches the expected value (double check against the documentation for what the hostname should be).
	Firewall blocking the connection	This is a common issue especially on corporate IT networks. The Firewall is preventing connectivity to the API hostname. You will need to speak to the IT team about getting the firewall amended to allow connections.
	Customer API platform issue	Although unlikely it is possible that a problem with the Assure infrastructure is preventing the connection from working. If you have ruled out the other possibilities then contact Evotix support.
SSL certificate error	Incorrect setup/config	Make sure that the hostname part of the URL is correct in the tool/platform. Add logging to report the URL value just before the API request is made and verify that it is matches the expected value (double check against the documentation for what the hostname should be).
	Incorrect date/time on the servers/platform	Make sure the server/platform you are using has the correct date/time. Having the incorrect date/time can cause SSL certificate validation to fail as the certificate can incorrectly appear to be expired (or not valid yet).
	Customer API platform issue	Although unlikely it is possible that a problem with the Assure infrastructure is causing the SSL certificate error. If you have ruled out the other possibilities then contact Evotix support.
Connection timeout	IP routing issue	A connection that times out can indicate a routing issue between the server/platform generating the request and the public internet. If this is suspected then you will need to get help from your IT team.

Condition	Possible cause	Notes
	Firewall issue	Usually firewalls will actively fail connections that aren't permitted. But sometimes they can configured in a 'stealth' mode where the connection cannot get through but no error response is triggered. You will need to speak to the IT team about whether this might be the case and get the firewall amended to allow connections.
	Customer API platform issue	Although unlikely it is possible that a problem with the Assure infrastructure is causing connection timeout error. If you have ruled out the other possibilities then contact Evotix support.

HTTP error response

If a connection between the tool/platform and the Customer API is established then the result of the API request will be an HTTP status code indicating the outcome of the request.

- HTTP status codes 200-299 are defined as successful outcomes and any response in this range should be interpreted as a successful completion of the API request.
- HTTP status codes 300-399 are reserved for redirection requests, normally you will never receive these codes as they are handled transparently by the HTTP client library within your software/tool. If you do receive a 300-399 HTTP status code your HTTP client library is probably misconfigured or your tool/platform does not handle redirects in which case you will need to handle the redirection request in your integration process.
- The OpenAPI schema for the Customer API defines the expected HTTP status codes for each API method. This should always be consulted first and where the HTTP status is defined in the OpenAPI schema that definition should be used to understand the meaning of the error.

The following table defines in general terms the meaning of the HTTP status codes that might be encountered when requesting data from the Customer API.

IMPORTANT - The OpenAPI schema for the Customer API definitions for the HTTP status codes take precedence over the general definitions in the following table.

HTTP status code	Meaning	Notes
400	Bad request	The most common cause of this error is the JSON supplied to the API method not matching what is expected. This could be a missing mandatory field, or a field whose value is to long (or to short), or a field who's value doesn't match the expected pattern. Validate the JSON data against the OpenAPI schema to identify the problem. This error can also occur for situations where the request is not valid given the state of the data in the system. For example attempting to create a user with an email address that is already in use. Or trying to create a user when there is no licences remaining. For these situations the response body should contain a message explaining what the problem is.
401	Authentication failed	This is almost always caused by attempting to use an invalid API key or not having the API key in the correct HTTP header. It can also occur if the API URL is for the wrong stack and hence the API key is not valid (as API keys are only valid for the stack in which they were created).
404	Not found	This usually occurs when wrong URL suffix is being used and therefore is asking the API for a method which does not exist. Usually the response body will contain a message explaining what the problem is. Another common mistake which causes this error is using the wrong HTTP verb (e.g. using a POST when the API method expects a DELETE).
429	Too many requests	The Customer API implements throttling on a per customer basis. The number of requests per second is limited to 10 (with a burst to 20 per second). And the total number of requests per day is limited to 10,000. Receiving this error indicates that one of these limits has been reached.
500 - 599	API platform problem	Any error in the range 500-599 indicates a problem within the Assure Customer API platform. If you receive this error repeatedly then take a copy of the response message (if any) and contact Evotix support.

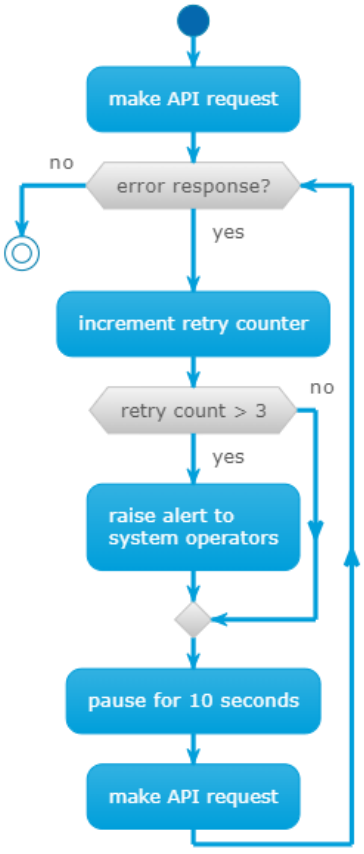
Retrying API requests

All systems will encounter errors occasionally and as such it is strongly recommended to implement strategies to try and handle these errors without aborting processing altogether. A simple mechanism is to implement a retry pattern around each API request. In simple terms this means whenever an error occurs the API request is tried for a certain number of times, typically with a short delay between attempts to give the error condition a chance to clear. The Customer API is designed to make retrying easy by having idempotent API methods. This means API methods can be called more than once

with the same data and the outcome in terms of our response and effect will be the same. This is very important as when an error occurs it is not always possible to be sure whether the request was actually completed or not (e.g. when receiving a timeout).

Our recommendation is that all API requests should be retried for ALL error conditions. There is no harm in retrying API requests and there is rarely any benefit in giving up after one error, even for errors that might appear to be permanent (most errors can be transient). A suggested pattern is to make at least 3 attempts for an API request with a short pause (e.g. 10 seconds) between attempts. After 3 attempts the process should do something to alert operators of the error condition (see the Alerting and logging section below). After this the behaviour depends on the process, if there is a possibility that the error is being caused by bad data then the process could quarantine the record being processed before moving onto to try other records. There are lots of different strategies that could be employed and it will be down to the developers of the process to determine the most appropriate strategy that ensures that processing can resume automatically and requiring as little operator intervention as possible.

The following activity diagram to shows an example retry handling flow.



EVOTIX

Alerting and logging

When problems do occur the first thing to do is let the system operators know so that they can take action to diagnose and resolve the issue. Any process making requests to the Customer API should have a means to raise an alert to the system operators if a recurring error condition is detected, or if bad data is detected. Once system operators have been alerted the first thing they will need to do is identify the root cause of the problem, effective logging is absolutely key to enabling swift resolution.

Every log line should include the date and time (with millisecond accuracy) that the line was generated. If the system/tool uses multiple threads or processes that write to the same log line then the thread and/or process id should be included.

Never record the API key in the log files, nor any other secrets, passwords, or other credentials.

The integration process should keep as detailed a log of its operation as is practical. At a minimum the process should log:

- Before making each API request a log line should be generated recording the reason for the API request, the URL being requested, the HTTP method being used and the system record or data entity that it relates to (e.g. if adding a user you would include the username as it is the primary identifier of the user record). It is very useful if the request content can also be logged.
- In the event that the API request is successful a log line should be generated confirming the successful completion including the system record or data entity that it relates to (e.g. if adding a user you would include the username as it is the primary identifier of the user record).
- In the event that the API fails a log line should be generated recording the failure. This should include the URL that was requested, the HTTP method that was used. The response body from the API must be included and also the content of the request body that was sent to the API request.
- Any exceptions that occur whilst calling the API should be caught and the details of the exception recorded in a log line along with the URL that was requested, the HTTP method that was used and the content of the request body.

Example logging

Here is an example of the logging from an integration process that is creating/updating users from a queue of records. The log shows a successful request followed by a request

that fails (and gets retried). Note the raising of an alert to system operators when the maximum retries is reached. This example process moves the failing record to the back of the processing queue in case the error is being caused by the user data in the record. This allows the process to continue trying to handle any other available user records that need creating/updating, whilst the system operators investigate the failure.

```
2023-10-31 14:35:54.101 - Generated userPOSTRequest JSON for 'example.apiuser'
{\r\n  \"username\": \"example.apiuser\",\r\n  \"fullname\": \"Example
APIUser\",\r\n  \"email\": \"example.apiuser@evotix.com\",\r\n
\"defaultOrgUnitExternalId\": \"REGION_NW\"\r\n}

2023-10-31 14:35:54.102 - About to create/update user 'example.apiuser' using POST
https://api.uk2.sheassure.net/v1/user

2023-10-31 14:35:54.877 - 200 response. Successfully created/updated user
'example.apiuser'

2023-10-31 14:35:55.228 - Generated userPOSTRequest JSON for 'another.user' {\r\n
\"username\": \"another.user\",\r\n  \"fullname\": \"Another User\",\r\n
\"email\": \"another.user@evotix.com\",\r\n  \"defaultOrgUnitExternalId\":
\"REGION_SW\"\r\n}

2023-10-31 14:35:55.510 - About to create/update user 'another.apiuser' using POST
https://api.uk2.sheassure.net/v1/user

2023-10-31 14:35:57.044 - 500 response. Failed to create/update user
'another.apiuser', response body {\r\n \"message\": \"Internal server error\"},
request body {\r\n  \"username\": \"another.user\",\r\n  \"fullname\": \"Another
User\",\r\n  \"email\": \"another.user@evotix.com\",\r\n
\"defaultOrgUnitExternalId\": \"REGION_SW\"\r\n}

2023-10-31 14:35:57.200 - Pausing for 10 seconds. Retry count 1 for
'another.apiuser'

2023-10-31 14:36:07.821 - About to create/update user 'another.apiuser' using POST
https://api.uk2.sheassure.net/v1/user

2023-10-31 14:36:09.935 - 500 response. Failed to create/update user
'another.apiuser', response body {\r\n \"message\": \"Internal server error\"},
request body {\r\n  \"username\": \"another.user\",\r\n  \"fullname\": \"Another
User\",\r\n  \"email\": \"another.user@evotix.com\",\r\n
\"defaultOrgUnitExternalId\": \"REGION_SW\"\r\n}

2023-10-31 14:36:10.020 - Pausing for 10 seconds. Retry count 2 for
'another.apiuser'

2023-10-31 14:36:20.131 - About to create/update user 'another.apiuser' using POST
https://api.uk2.sheassure.net/v1/user

2023-10-31 14:36:22.509 - 500 response. Failed to create/update user
'another.apiuser', response body {\r\n \"message\": \"Internal server error\"},
request body {\r\n  \"username\": \"another.user\",\r\n  \"fullname\": \"Another
```

EVOTIX

User\",\r\n \"email\": \"another.user@evotix.com\",\r\n
\"defaultOrgUnitExternalId\": \"REGION_SW\"\r\n}

2023-10-31 14:36:22.764 - Retry count 3 for 'another.apiuser'. Raising alert and moving 'another.apiuser' to the back of the processing queue.