Joni Ranta

Testing AWS hosted Restful APIs with Postman

Metropolia University of Applied Sciences

Master of Engineering

Information Technology

Master's Thesis

10th of January 2023

PREFACE

In this study I learned a lot about testing and Postman tool that I use in my daily work. With this new information about Postman, I do see new ways to use the tool and I most likely will in the future. Surprising to me was that, what at first seemed like a straight forward thing to do, turned out to be more complex what I imagined.

Big thanks to Sinch Finland QA team and my Operations team colleagues for support they provided.

Vantaa, 10th of January 2023

Joni Ranta

Abstract

Author:	Joni Ranta
Title:	Testing AWS hosted Restful APIs with Postman
Number of Pages:	34 pages
Date:	10th of January 2023
Degree:	Master of Engineering
Degree Programme:	Information Technology
Professional Major:	Networking and Services
Supervisors:	Jarmo Heikintalo, Development manager Ville Jääskeläinen, Principal Lecturer

This thesis is for creating an automated testing setup to be used after quarterly Version upgrade for production environments. This is to reduce manual testing required after each upgrade. The testing setup was limited to a setup that is reasonable to complete within a Masters´ thesis timeframe. Further development can be done after the initial setup has been completed.

To reduce overlap in tools for company landscape, Quality assurance team was interviewed for tools and for any possible scripts that they have and could be used for this production environment testing setup. Target was also to use as much as possible already existing internal landscape.

End product of this thesis is an automated (as much as is feasible) testing process for customer environments using tools and methods that are already in use within the Sinch Finland Systems Quality assurance and Operations team.

Keywords: AWS, Cognito, Postman, Restful API, HTTPS, Testing

Contents

List of Abbreviations

API	Application programming interface
AWS	Amazon Web Services
CP	Communications panel
CSRF or XSRF	Cross-site request forgery
IA	Infrastructure Admin
JWT	JSON Web Token
MFA	Multi-factor authentication
QA	Quality assurance
SC	System Configurator
SIP	Session Initiation Protocol
SMS	Short message service
VPN	Virtual Private Network

1 Introduction

Sinch Finland Systems Oy (Sinch being the parent company) provides contact center, named Sinch Contact Pro for customers to use. Contact center is a call center product with added wider communications functionalities. “Sinch Contact Pro is a true omnichannel solution, supporting all commonly used contact-center communication channels including telephony, email, chat, video, SMS, and messaging apps such as WhatsApp, Facebook Messenger, Viber, and more.” [1] SMS is short messaging service, that is used mainly by mobile phones to send text messages. This type of contact center product is aimed at companies to be used for their customer communication.

In this thesis functionality testing is focused on customer environments that are in active customer use, upkeep done by Sinch Operations team, and more specifically to test their basic functionality after new version has been rolled out to customer use. New version upgrades take place on quarterly windows, where upgrades are done on weekend time frame to minimize customer use impact.

To reduce overlapping tools and methods, same tools were used as for functionality testing in development phase. Quality Assurance (QA) team was interviewed for tools that they use and also for any possible existing scripts that could be used for Operations team testing.

Goal is also to automate, as much as possible, testing for production environment customer base after product update. Depending on the automation outcome, testing could also be run on the back ground and notify for any anomalies found.

1.1 Method and material

To find out what software and possible scripting QA team used at the moment of writing this thesis, a meeting was held on 10th of June 2022, with QA team manager.

Out of all testing tools used, Postman was selected during the meeting to be used for this new testing setup. Postman provides tools that can be used for testing API's [2] and is used for both QA and Operations team. Application programming interface (API) is used to fetch or modify existing data in other application. API´s can be seen as a way for different applications to communicate with each other.

Operations team uses, when writing this thesis, Postman for creating new customer instances in Amazon Web Services (AWS) cloud infrastructure but not for testing purposes. QA team has experience on using the Postman tool to test Contact Pro interfaces and could help creating the test patterns for production test use cases.

Scripted testing fills in the requirements for testing process to be used for production environment testing. "Whenever repeatability, objectivity, and auditability are important, scripted testing can be used" [3 pp. 188]

1.2 Thesis structure

In Chapter 2 current state analysis is conducted. This will give overview of the current production environment and testing methods used by Operations team. Improvement points and summary of current state are also included in this chapter.

Chapter 3 provides reader background information on topics that will be used in Chapter 4, where the Postman requests are created for testing. In Chapter 5 tests and analysis on test results, are conducted using the Postman collection created in Chapter 4. Final conclusions and further improvement points are in Chapter 6.

2 Current state analysis

This chapter will go through the current state of Contact Pro usage for both customers and internal Sinch usage, Operations teams testing for its usability and where current setup could be improved.

2.1 Contact Pro usage

Customers' end users use Contact Pro via user interface called Communications Panel. End users in service description and manuals are often referred as agents. "Sinch Contact Pro provides contact center agents with Communication Panel (CP), a responsive HTML5-based user interface that runs in popular web browsers is used without the need for any installations or browser add-ons"[4]. Communications panel provides the most commonly used user functions for the end users [4]. Customers have access to Contact Pro globally via internet and Contact Pro uses AWS cloud infrastructure for global reach.

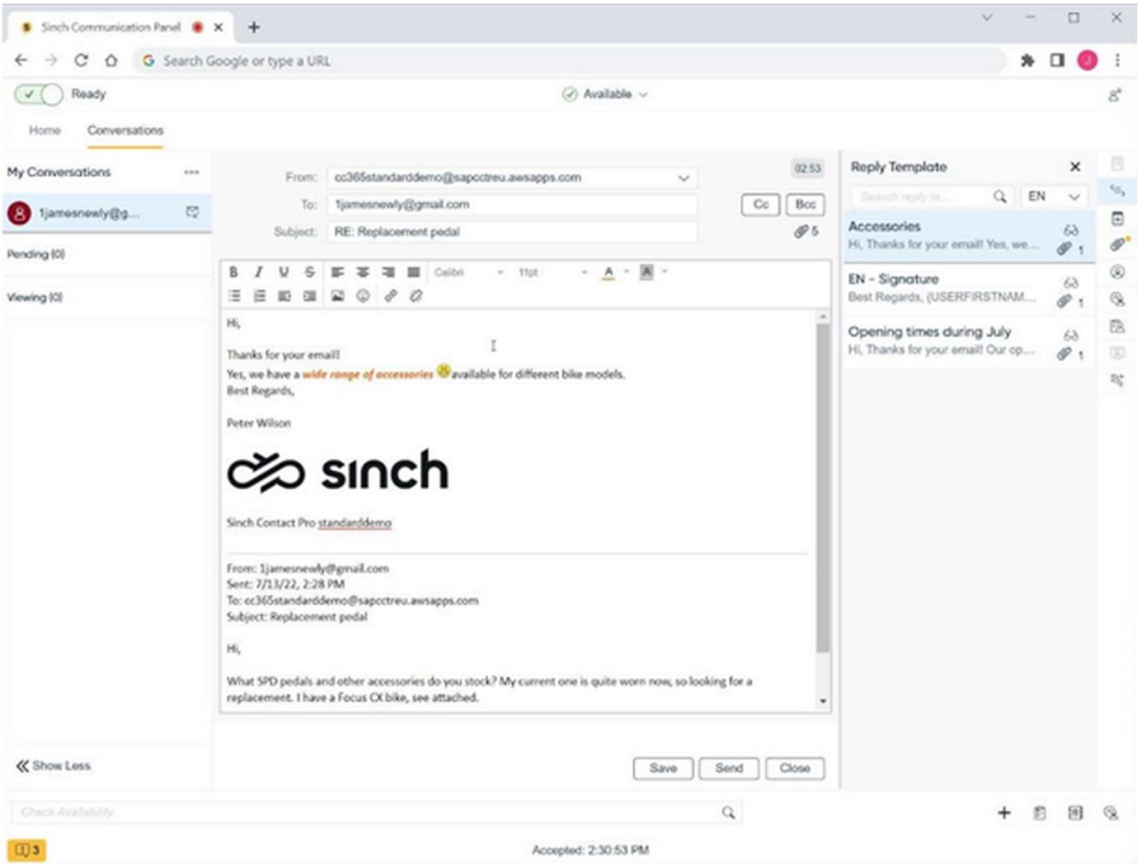

Figure 1. Sinch Contact Pro Communications panel, user answering to an email

Each customer has its own tenant. This is to separate customers and their users. Tenant environment contains its own virtual server for each customer on which the Contact Pro software is run on. Tenants are unique with their usage addresses and API´s.

For user creation and similar functions, Java-based configurator called System Configurator (SC) is used [5]. System configurator is mainly used by customers' power users, Sinch Service Desk and Project team or external partners that provide configuration services.

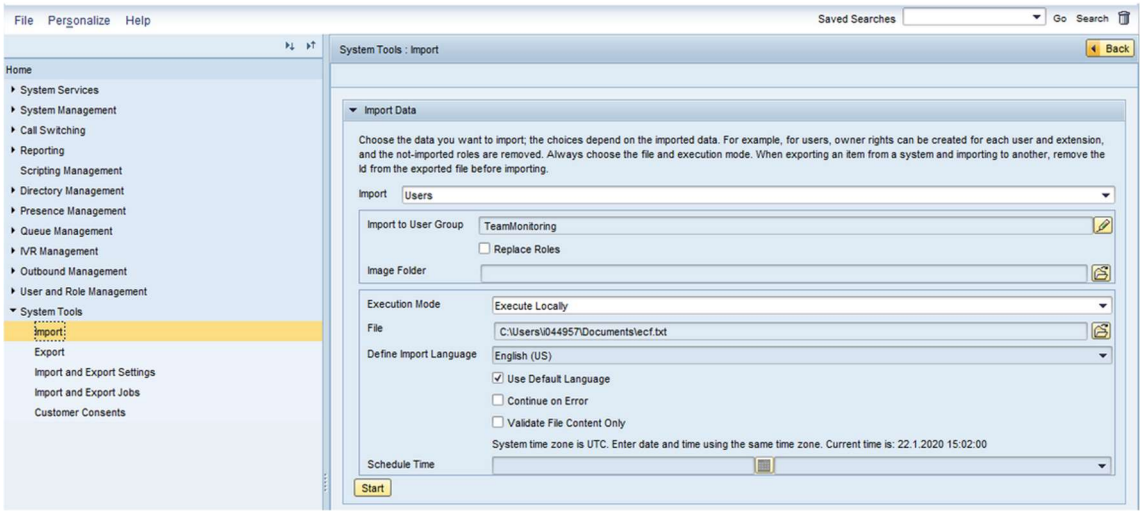

Figure 2. User importing user data in System Configurator [5]

Operations team also uses Java-based Infrastructure Admin (IA) tool which is used to configure Contact Pro connectivity and databases etc. Infrastructure Admin tool is not mentioned in either service description [4] or Contact Pro documentation [5]. This configuration work is entirety done by Sinch Operations team in Contact Pro landscape.

Some configuration work that customer has no access to, is also done in System Configuration level. Mainly this is for infrastructure related configuration such as

Email channel setup or Session Initiation Protocol (SIP) trunk setup. SIP trunks are used by phone operators to route calls.

Before mentioned Infrastructure Admin tool is accessed directly from customers tenant server and the usage is limited to Operations team. Tenant upgrades are done via the Infrastructure Admin tool so if it would not work, it would be seen during the upgrade. Because customers have no access to it and upgrades are done with it, IA tool can be left outside the scope of this thesis.

2.2 Contact Pro production environment interfaces

Production environment interfaces are shown in below picture.

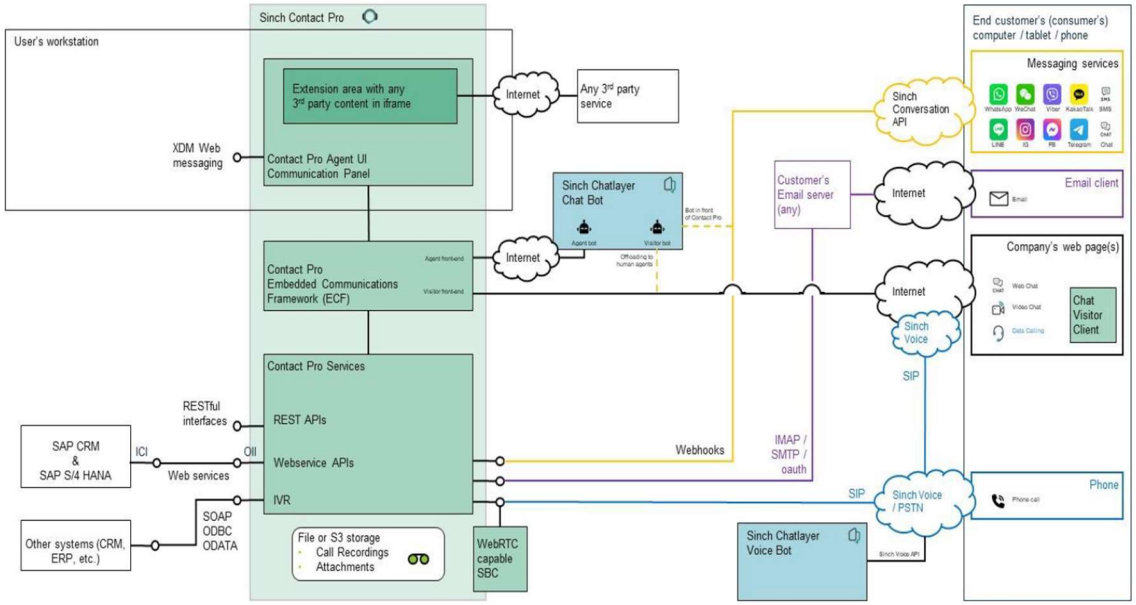

Figure 3. Sinch Contact Pro interfaces [4]

From the picture taken from service description, can be seen that customers' agents access Sinch Contact Pro through internet regardless of what user interface they use. Customer access is as such limited to internet and any usability tests should follow the same route via internet.

2.3 Operation teams' current testing

Operations teams' customer environment functionality testing after quarterly upgrade is currently limited to manual testing with either using same user interfaces as customers' users (logging in with Communications panel or similar interface), or with direct links to test certain API´s if they respond as they should. Direct API links require to be set up manually before testing and modifications are required per customer.

API´s in this test scenario are tested only to respond. To fully test an API, usually API secret and user id with password, are required to receive the requested information from API. MFA API authentication is supported by AWS [6] but is not used in Contact Pro APIs when writing this thesis.

Functionality testing is also done if customer reports an incident. Functionality is tested by either Operations team or Service Desk team before investigating the issue further from the Contact Pro logs. No active and alerting log reading tool is in use and logs are read only for incident solving and product development.

On operation system and on level higher, AWS cloud level, alerts per customer tenant exist but these usually do not translate well for how Contact Pro product operates within the virtual container. This, and the fact that no active log reading tool exists for Contact Pro logs, prevent using test triggers from such outputs.

Automation of any level on quarterly maintenance window would reduce manual workload for the persons doing the upgrades on those windows.

2.4 Information deposits

Information can roughly be split into two categories, one that customer has access to and one meant for Sinch personnel internal use, mainly the customer link page.

2.4.1 Databases

Contact Pro databases are located at data centers on AWS cloud services across the world on 4 different locations [4]. “By default, the tenant is provisioned in the data center closest to the customer’s end users”[4].

Customers access Contact Pro data via integration interfaces through internet. “They (integration interfaces) are published via AWS API gateway and access is controlled with user authentication and API keys”[4]. As such, usability tests to databases would need to follow the same route to mimic customer use efficiently.

2.4.2 Customer link page

Customer link page includes information where each customer tenant can be found, meant for Service Desk and Operations team use for faster access to customer tenants. Customers have no access to this link page.

Customer link page resides in AWS. Page itself is used with internet browsers and uses AWS Cognito authentication for user login.

Database for customer link page is not located in the same servers as customers databases are and is a separate database instance altogether.

For Sinch internal users’ customer link page uses AWS user pools (with Cognito authentication) for login with personal certificates into customer tenants. Customers use either user certificates or basic authentication as their login methods for Contact Pro. Basic authentication means username and password type of login. This login difference between Sinch internal use and customers’ users, creates a difference in login link naming. Both types of links are available in the link page for Sinch personnel to manually test accessibility also from customer perspective with a basic authentication test user.

Customer link page is the collective place where customer tenant names can be found and are upkept manually. This is the only collection where information is

available as a list format and it contains all the tenants. Separation exists between a customer region and production and test tenants on different tabs.

Other possible location instead of customer link page would be AWS, to fetch customer tenants in production use but customer tenant information is divided between AWS regions and would need to fetched from each region separately.

A customer link page does not currently contain customer API link information. This information can only be found from AWS, separated by regions. Customer API links contain information that is unique to each customer and can not be directly created with just customer tenant name.

2.5 Summary and improvement points

Any form of testing automation after quarterly upgrade is an improvement to current situation. Manual testing is error prone and requires a lot of time to do properly.

Customer link page should be used for fetching live production tenants because it is a single point to store such information and is maintained actively by Operations team. Link data can be fetched and used to test Communications panel interface and other interfaces that can added as a link information.

Requirement for manual link modification for each tenant for API testing is not maintained easily as each person would need to maintain their own collection or share collections with each other. If this information was available on customer link page, testing automation could use it from there. For faster access to API information, the information could be added to the customer link page if internal guidelines for this type of data allows it.

Depending on the API, also API secret and user credentials, can be required to fully test the interface. These are confidential information and customer specific. Most likely this information will not be added to the customer link page considering it is confidential and prone to change.

Usability tests should follow the same route as customers use the Contact Pro product. Tests should access interfaces and API’s through internet instead of any other possible routes that Sinch internal users have.

3 Background information

In this section background information is provided to give some understanding for the methods and software involved. In Chapter 4 where the testing setup is introduced in detail, these Chapter 3 topics will be deepened with each use case.

3.1 Testing practices and needs

Software testing can be done in multiple ways and forms. The book “A Practitioner’s Guide to Software Test design” written in 2003 by Lee Copeland mentions 11 different ways how software can be tested. Out of these 11 different ways of testing, scripted testing is the most prominent one to be used for this thesis.

As mentioned before in this thesis introductory chapter and quoted from Lee Copelands book, scripted testing can be used when repeatability, objectivity, and auditability are needed [3 pp 188]. Testing setup to be build needs to have these qualities.

Operations team does not do testing as part of their daily work. This means that repeatability is important regardless who from Operations team does the test execution. As Lee defines repeatability in his book “Repeatability means that there is a definition of a test -- at a level of detail sufficient other than the author to execute it in an identical way.” [3 pp 189]. Postman helps with this requirement since it enables test procedures to be shared among Operations team in exactly the same format [2]. Variables such as user credentials would need to be changed for each test user, but the test itself would be the same.

The next test requirement, objectivity, is described as such “Objectivity means that the test creation does not depend on the extraordinary -- skill of the person creating the test but is based on well understood test design princibles. [3 pp 189]”. Using Postman and its shared test procedure, this objectivity need is met.

“Auditability includes traceability from requirements, design, and code to the test cases and back again. This enables formal measures of testing coverage. [3]” Testing code is shared among testers in Javascript format and rest of the information about used APIs can be found from service configuration documentation [5]. Any user can see the test code used and modify it if needed.

3.2 Postman for testing use

Introductory chapter mentioned a meeting with QA team that scouted the testing software that was in use with QA. Out of software already used, Postman was selected to be used for this test setup. It also fulfils scripted testing practice needs.

3.2.1 What is Postman

Postman is software that is described in their website as API platform that can be used to build APIs and use them [2]. Further than this, Postman can be used to test said Postman APIs and other APIs as well [7].

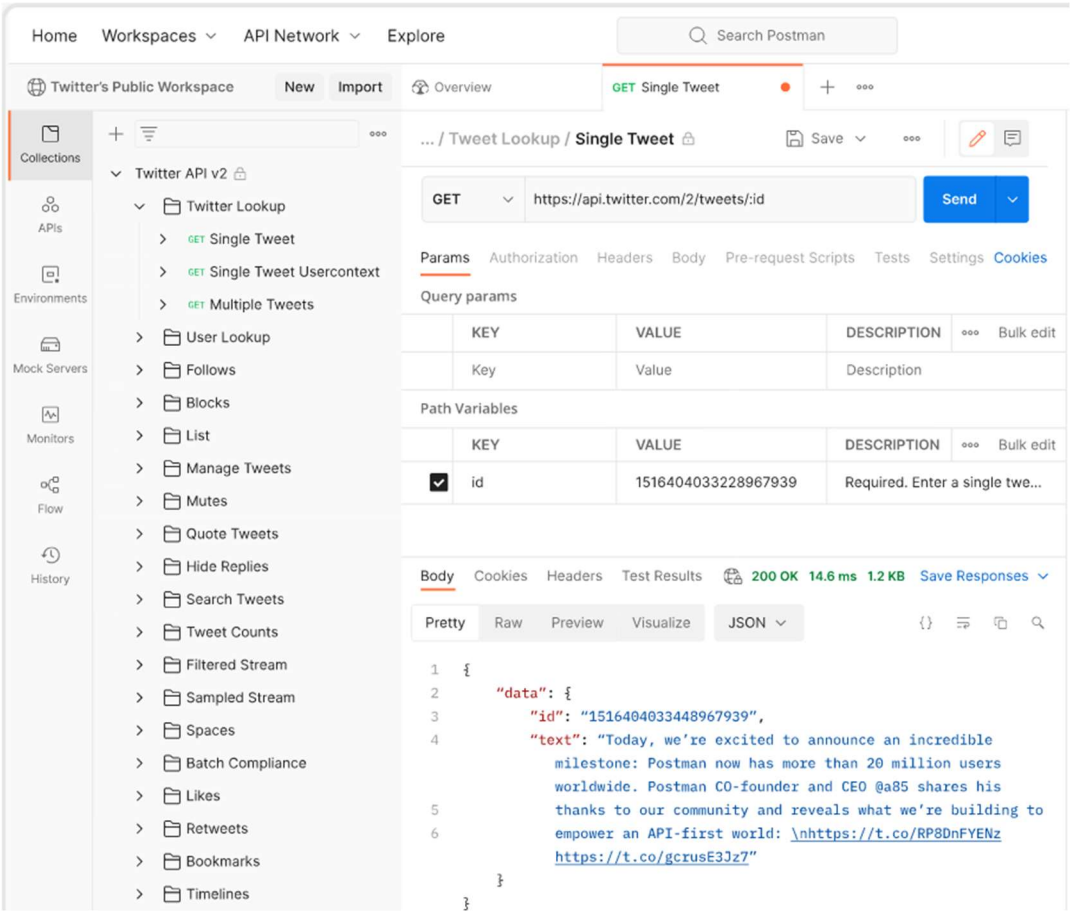

Figure 4. Postman user interface from Postman download page [8].

Postman software is either downloaded to the users´ computer (Windows, Mac and Linux operation systems are supported) or used from the web browser with Postman Web version. [8]

Postman itself is then used to send requests to APIs. “Requests can retrieve, add, delete or update data” [9]. Requests can be used to send parameters, login details, authorization information or any other body data that is needed [9].

These requests are then added to collections and collection can host multiple requests. Data that request receives is called response. [9]

3.2.2 Testing APIs

Test is a functionality in Postman that can be “added to individual responses, collections and folders in collection” [7]. Tests tab of individual request can be seen in user interface picture of Postman. Test scripts need to be written in programming language Javascript [7]. Javascript for testing scripts can be written manually or by using Snippets in the code editor. Snippets are API requests in code format [10].

Test scrips that Postman use “can use dynamic variables, carry out test assertions on response data, and pass data between requests.” [7] In practice this means that one test can lead to another depending on the received response and requests can use data received in previous response.

3.2.3 HTTPS page request

HTTP is technology commonly used in internet for browsing webpages. HTTP supports with the HTTP/1.1 specification GET, HEAD, POST, PUT, DELETE, CONNECT, OPTIONS and TRACE request methods. The combination of allowed methods can be configured per HTTP server. Server can support all of the possible request methods or just some of them. Methods are case sensitive. [11]

Used in customer link page is HTTPS version of HTTP technology. HTTPS is secured version of HTTP protocol as it encrypts the data on Transport layer. This is important because it protects sent data against different capture methods and false data provider identity type of attacks. Using HTTP on a website can be seen as deprecated technology and ”HTTPS is now used more often by web users than the original non-secure HTTP.” [12]

HTTPS requests can be made with Postman to fetch data available on a website. Received information can be further processed into new requests. [9]

3.3 AWS Cognito, identity pools and user pools

Amazon Cognito is way in AWS to implement user identification and sign up to AWS hosted web and mobile applications. Cognito supports different types of user identification and advanced security features. [13]

Cognito has two main components, user pools and identity pools. User pools, consists of user data, login credentials and such. It also provides different types of sign-in options if enabled to users. Identity pools are used to grant access to other AWS services. These pools can be used together or separately. Below is a picture of common AWS Cognito setup with both of these pools being used. [13]

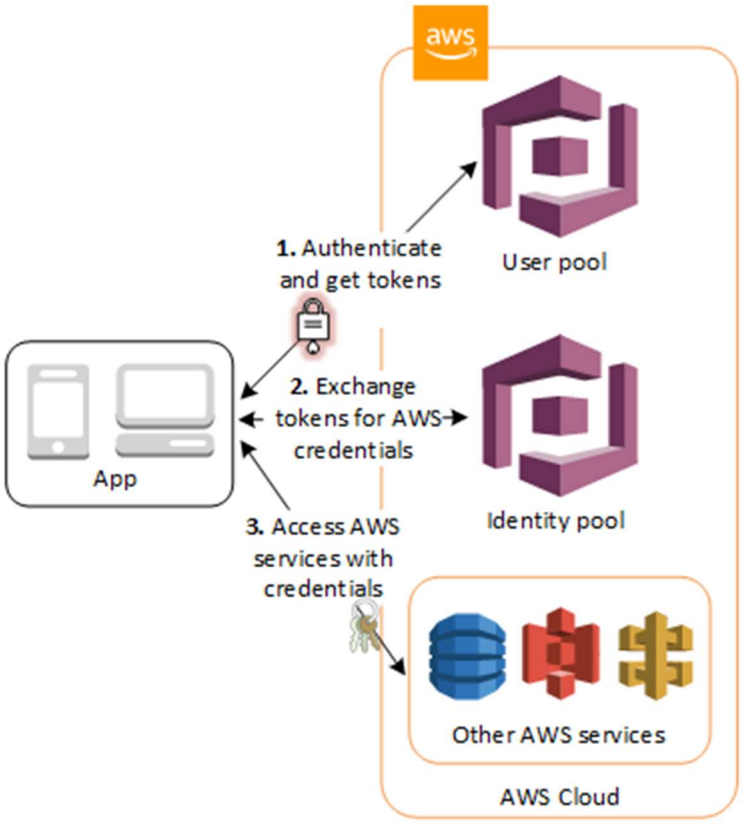

Figure 5. Amazon Cognito common setup from developer guide [13]

In a setup like this, user is authenticated and then granted access to another AWS service [13]. AWS Cognito developer guide [13] describes the flow in more details as follows:

1. In the first step the app user signs in through a user pool and receives user pool tokens after a successful authentication.

2. Next, the app exchanges the user pool tokens for AWS credentials through an identity pool.

3. Finally, the app user can then use those AWS credentials to access other AWS services such as Amazon S3 or DynamoDB.

This is the same setup as customer link page uses for user authentication. User login information will be a variable in Postman that is required to be changed for each user doing the test. Tokens that are changed to AWS credentials, will expire within a set time frame, meaning that cookies created with each authentication cycle most likely need to cleared after each full test run.

3.4 API: Restful interface

API can mean different things in terms of how each software communicates with each other. REST (Representational State Transfer) APIs however try to unison this by using a definitive architecture on how these REST APIs should work. This helps when creating a module to communicate with this interface since it is not completely black box on how it operates. [14]

Terms Restful API and REST API can be a bit confusing, Restful meaning the Restful web APIs but these terms Restful and REST can be used interchangeably. In common lingo, Restful API or REST API is usually meaning the same thing. [14]

When Restful client requires a resource, it connects to the server by using the Restful API. Restful API request and response will follow the basic flow below [14]:

1. Client sends a request to the server following API format.

2. Server authenticates the sender client and confirms that client has the rights for required access.

3. Server then starts processing the request if rights check is cleared.

4. Response is returned to the client from server. Response will tell if the request was successful or if any errors occurred. If the request was successful, the response will contain the information requested.

There can be slight variations on the Restful API functionality depending on the developer of the Restful API. [14]

Contact Pro uses different APIs for different functionality, usually for providing some information from Contact Pro to third party software. 8 of these interfaces are following Restful API architecture [15]. Restful interface is also important for Communication panel user interface since it uses the Restful APIs for monitoring and directory data. If these Restful interfaces do not work properly, it would be seen in Communication panel use as well.

4 Test setup creation

This chapter will cover the test setup creation in detail using Postman and deepen upon the background information topics in Chapter 3. Customer and Sinch production environment specific details in screen captures will be pixelated for data protection. QA team provided guidance for creating these test setup Postman requests.

4.1 GET request to customer link page

Authentication with Postman requires use of tokens and steps where data is fetched from both the site that is to be accessed and from Amazons' Cognito services. The process starts with HTTPS GET request from the internal customer link site.

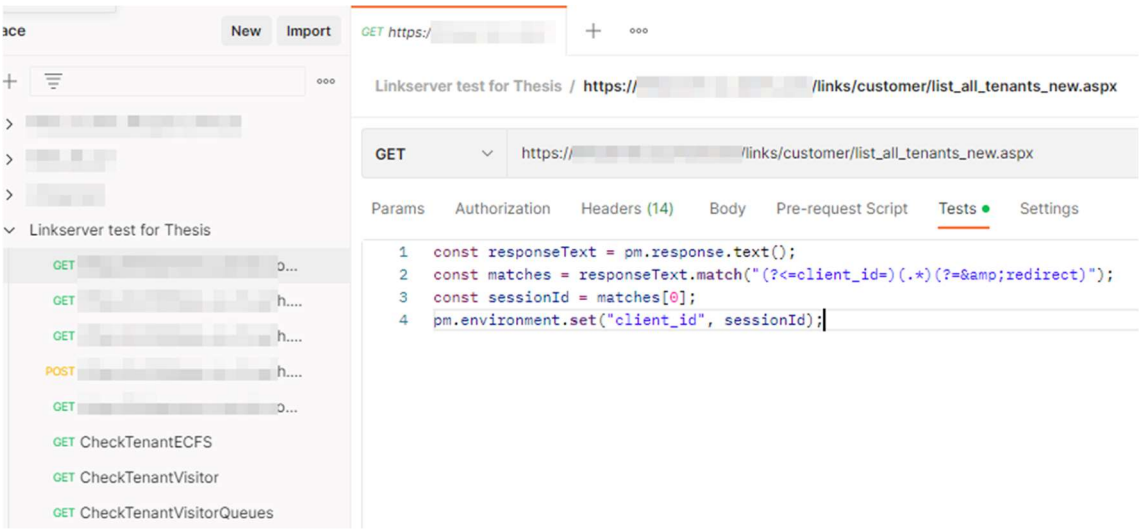

Figure 6. Postman GET request to customer link page

In Figure 6 we can see that Tests part of GET request contains modified fields. Indicator for this is the green dot next to Tests tab. Change indicator is the same in Postman regardless of the tab that contains modified fields or information. Scripts written in Tests tab are executed after the GET request has been made and target the response that was given as a reply to the GET request.

In the Tests tab are “const” declarations for creating constants. Also used is pm object which is Postman JavaScript API functionality. It is called with pm.* and it provides access to response and request data including the variables used.

Received response text is searched for client_id value and is set for variable “client_id” as value. This information is used in the next step.

4.2 Authorization request to Amazon Cognito

After client_id value has been fetched from the link server website, authorization request is made to Amazon Cognito.

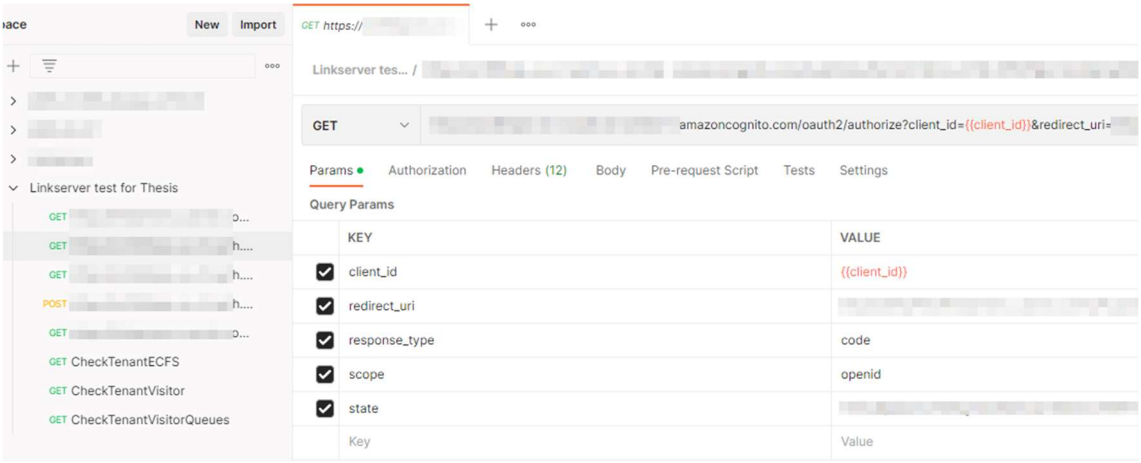

Figure 7. Postman authorization request to Amazon Cognito.

In this GET request, added information is sent using Params tab in Postman. Key values defined here are added to the get request to Amazon Cognito. This can be seen in Figure 7 request URL but exact values are pixelated. Noteworthy information here is that Amazon Cognito authorize service targeted and authentication method used is OAuth 2.0. Client_id defined in earlier request can be seen used here as first value.

Redirect_uri is defined address where the server sends the user after successful authentication. Response type and scope tells the authentication server how this request is handled. State is mainly used to prevent cross-site request forgery type

of an attack on web applications. All of these fields are required in OAuth 2.0 authentication flow.

4.3 GET request for CSRF token.

Once authorization request is done, Amazon Cognito login page will be requested to provide CSRF token. Token is server generated value, similar to that of the State key used by requests sent from Postman in this setup.

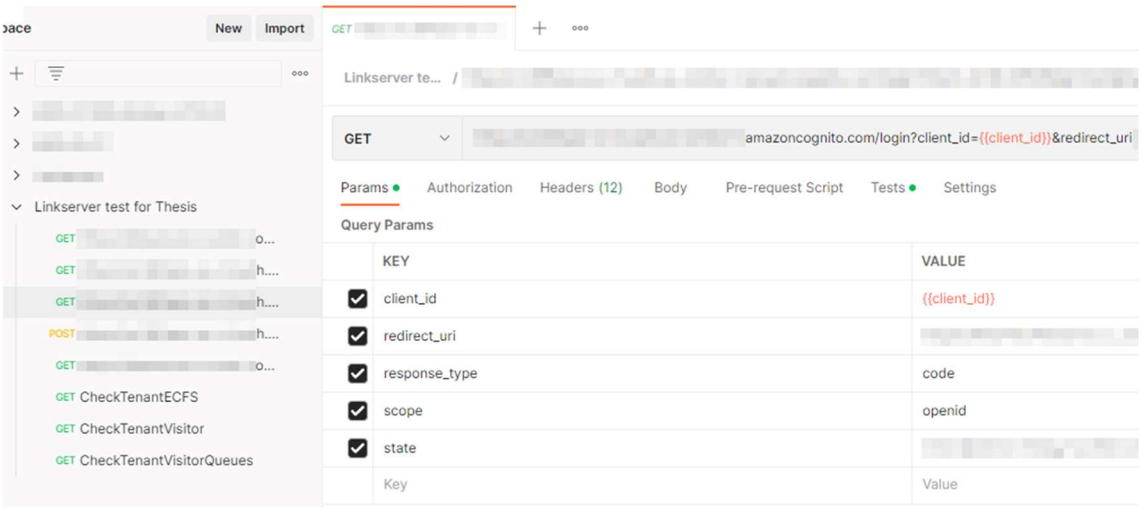

Figure 8. Postman request for generating CSRF token, Params tab.

In Figure 8 GET request the same values in Params tab are used as before but now Amazon Cognito login service is targeted instead of authorize service as in previous request. From the green dots we can see that in this request Tests tab also contains changed values. That is where CSRF token value is taken from received response.

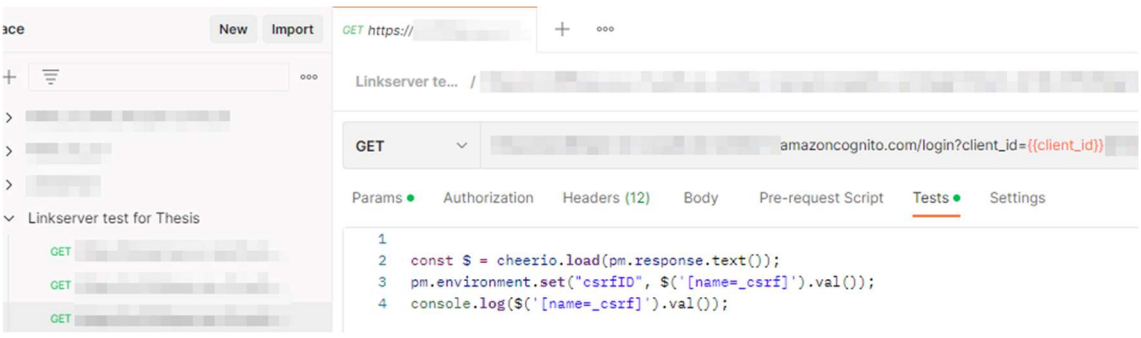

Figure 9. Postman request for generating CSRF token, Tests tab.

In the Test tab´s written code, Cheerio is used. Postman has this JavaScript enabled by default and it is used to parse HTML and XML type of data [16]. The code separates the CSRF token from the response received and prints the value in console log. Console log is tool that can be used to debug Postman commands should any issues arise. CSRF token value is added as environment variable csrfID. This will be used in the next Postman request.

4.4 POST request with login information

Now that CSRF token is received, instead of GET requests, the next request is of POST type. POST type requests are used to send information to a website or service.

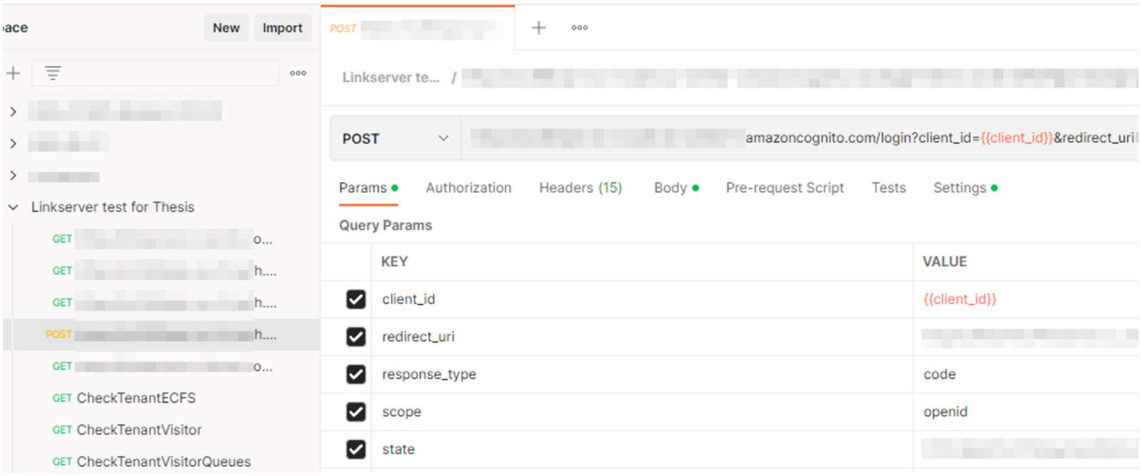

Figure 10. Postman post login request to Amazon Cognito, Params tab.

In Figure 10 Params tab information has the same values as in GET requests before and Amazon Cognito login service is targeted.

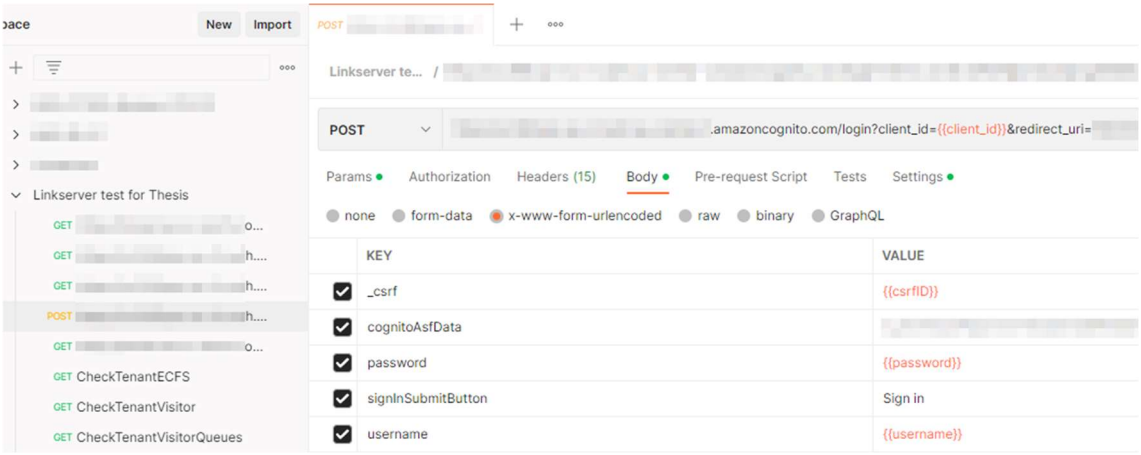

Figure 11. Postman POST login request to Amazon Cognito, Body tab.

In the Body tab csrfID, that was obtained earlier, is used. CognitoAsfData is Cognito access token generated outside Postman. The token is JSON Web Token (JWT) and it was generated using the *amazon-cognito-identity-js* JavaScript package [17]. Token includes information about the Amazon Cognito user pools to be used for authentication etc. Information is encypted and not in readable form but pixelated in picture for extra security measure. Token generated this way is not permanent and will need to be renewed within set time.

Password and username are user credentials used to login via Cognito. They are user specific and need to be changed for each Operations team member using this Postman collection. SignInSubmitButton contains information that this user is sign in. This can be visualised as a mouse click on Sign in - button.

On Settings tab, only change to default settings is, that Postman does not follow redirects when this Post request is run. Default setting is to allow redirects.

4.5 Fetching customer data after authentication

At this point, all required information has been gathered and input to Amazon Cognito to allow data to be fetched from customer link page.

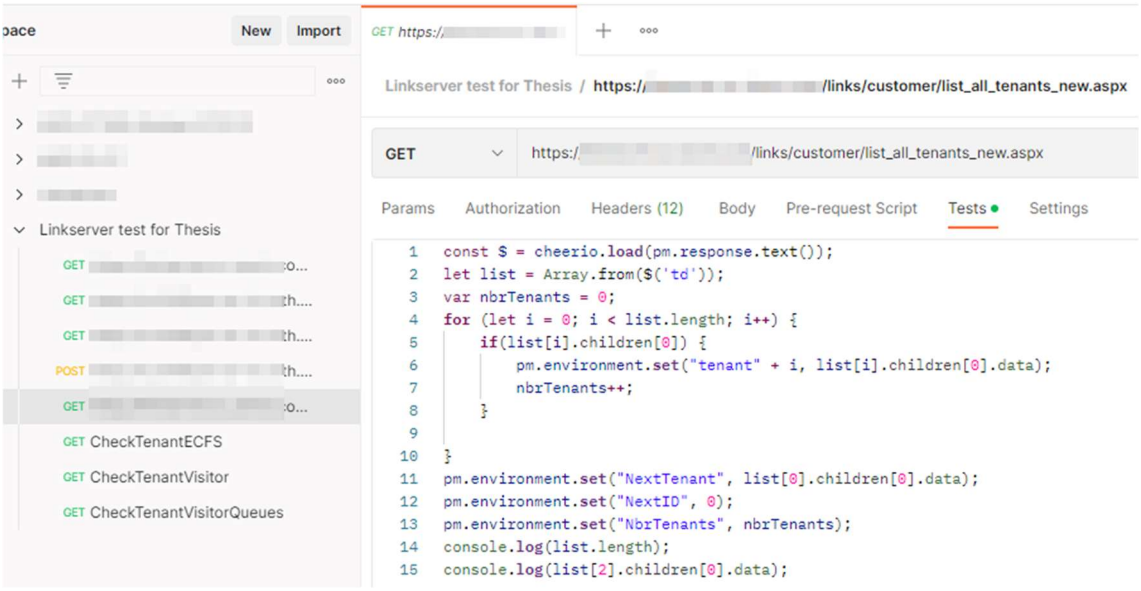

Figure 12. Fetching customer data from customer link page

Response to this GET request is a list of customer tenants in use. Code written in Tests tab in Figure 12 uses Cheerio to create an array of tenants from the response data received. This array will be used in the following test cases. Console log is used to print list length, meaning the number of tenants that was received, and to print one the tenant names in array for troubleshooting.

Access is allowed in this case since access tokens have been exchanged before and with this GET request, session cookies named AWSELBAuthCookie-0 and AWSELBAuthCookie-1, are created into Postman. Created cookies can be seen by clicking Cookies link in Postman, it provides view of all cookies created with this request flow.

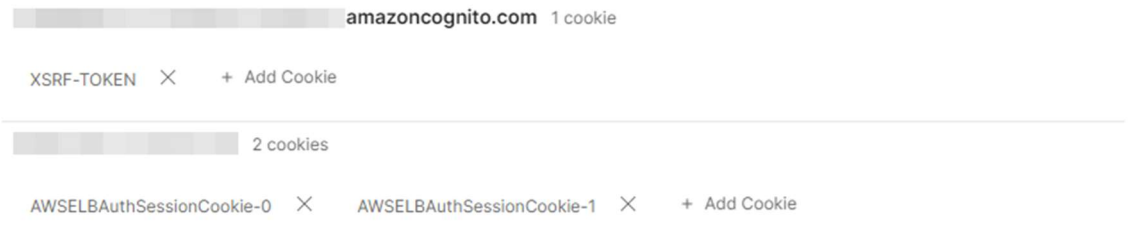

Figure 13. Cookies added and used with these requests.

XSRF-token cookie with amazoncognito.com name, is the CSRF token mentioned earlier in cookie form. Both abbreviations CSRF and XSRF mean the same thing, Cross-site request forgery.

4.6 Test cases

In Operations team meeting on 28th of October 2022 needed test cases were discussed. Out of all possible Restful interfaces queries, three different ones were selected to be tested and serve as test cases for this thesis. Test cases can be seen in previous Figures in Chapter 4 and they are named CheckTenantECFS, CheckTenantVisitor and CheckTenantVisitorQueues.

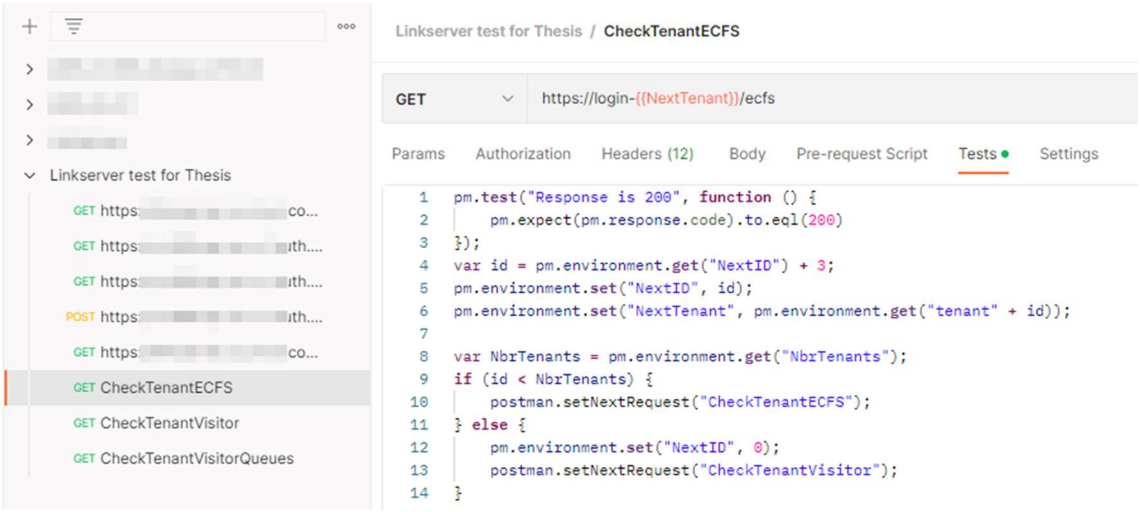

Figure 14. CheckTenantECFS Tests tab

On the GET row we can see that login page is targeted, customer name used and ECFS service is called. This service returns 200 OK if everything is as it should and can be seen as a way to check if customer tenants Restful Interface is responding properly. If 404 or other any result is given, this can be seen as failure to the normal functionality and investigation is needed on that customer tenant. Testing this way is high level test if Restful Interface is up. If it is not, no chat use or communications panel use could be done, to mention few examples.

Unlike in previous codes, where for loops can be seen, here test code uses if structure. If number of tenants received is smaller than id used, go to this test again. Every time this if structure is used, it adds 3 to the NextID value to be used. This is because data received in previous list all tenants´ response contains other information as well, not just customer tenant name. By skipping with 3, only customer tenant names are targeted. In the else part of if structure NextID value is returned to 0, to reset the pointer to the beginning. After this, Postman proceeds to the next test case named CheckTenantVisitor.

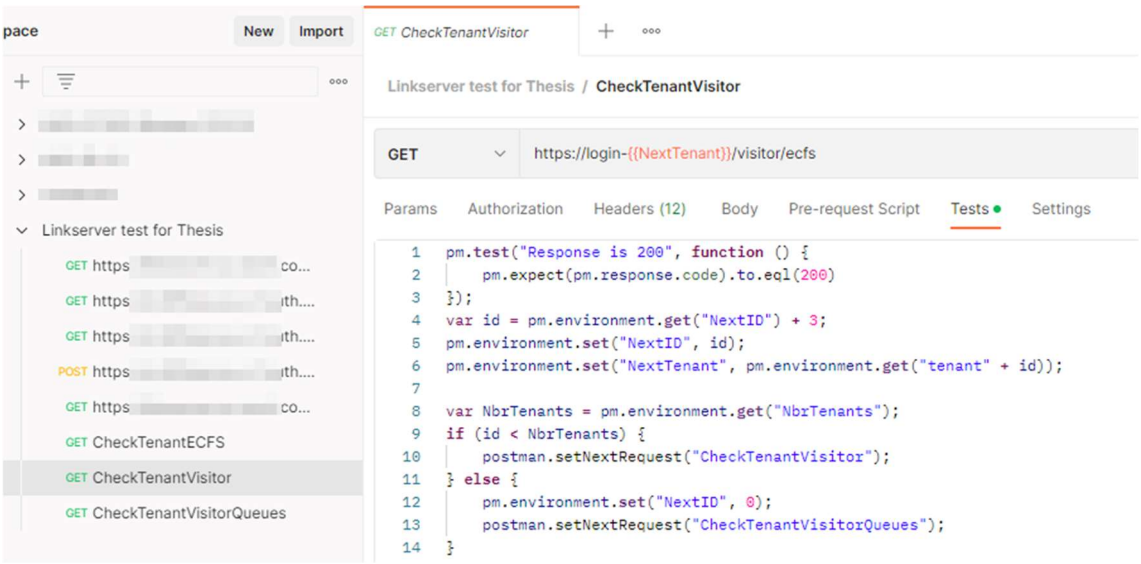

Figure 15. CheckTenantVisitor Tests tab

Here is similar structure than in previous test case but targeted with GET function is different address. Visitor and then ECFS is targeting Restful Interface used with chat capabilities and Communications Panel functionality. Return is the same 200 OK if test is successful. This is more detailed answer to functionality testing than previous test. If 404 or other value is returned, this would be seen in chat functionality not working properly and in Communications panel use as well. NextID value is set to 0 in the else part and then next test case is run, named CheckTenantVisitorQueues.

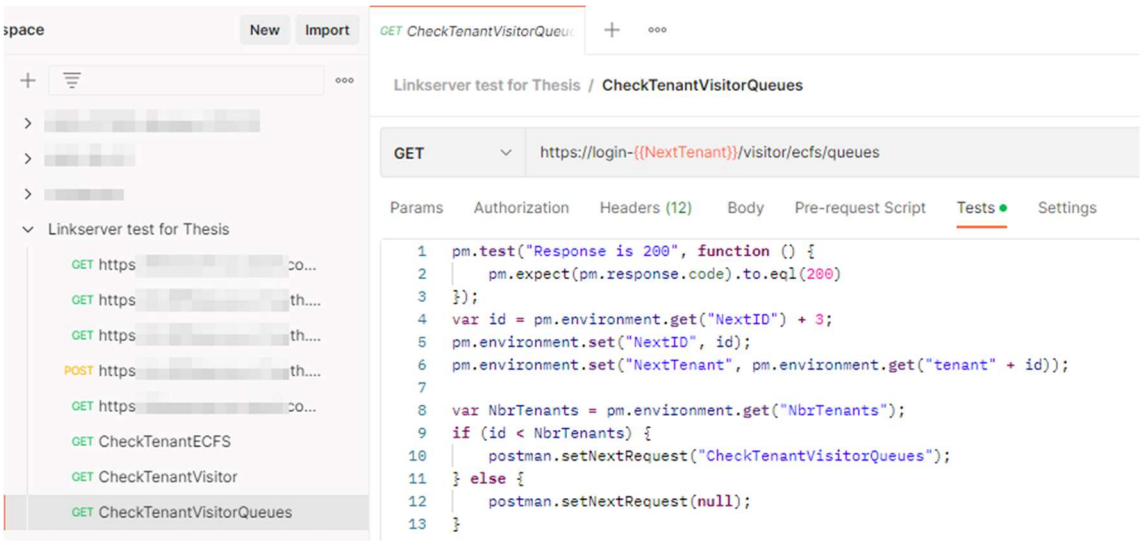

Figure 16. CheckTenantVisitorQueues Tests tab

Similar test structure than in previous tests but with GET targeted is queues part of ECFS service. This is to test if system returns queues list from ECFS service. Information is used in many customer integrations where queue statuses are needed. Communications panel uses this information as well. Returned here is again 200 OK and if not, cause would need to be investigated. Where in previous tests basic responsiveness is tested, this test fetches information deeper from the system, testing database connection and basic responsiveness at the same time. Simply viewing how long each query would take, is not good measure for functionality however, since customer tenants have different number of queues in their systems.

In the code when else part is run, null value is returned to setNextRequest. This ends tests cycle and no further tests are run after this.

To summarize what these three tests do:

CheckTenantECFS, tests high level Restful Interface functionality, basically if it is up and running or not.

CheckTenantVisitor, tests if connectivity to Restful Interface part where chat capabilities, among others, are run.

CheckTenantVisitorQueues, tests if queue data can be fetched from database. This tests both Restful Interface use and database connectivity.

5 Running tests with created Postman collection

In this chapter, Postman collection created in Chapter 4 is run and test cases analysed.

5.1 Postman collection Run options

Postman has build-in feature for running collections to test API or other functionality. This Run-option can be found from the collection level. Right mouse click on collections name will bring menu where option is listed. This menu screen will be split into two parts below for better visibility, run order part and how to run collection part.

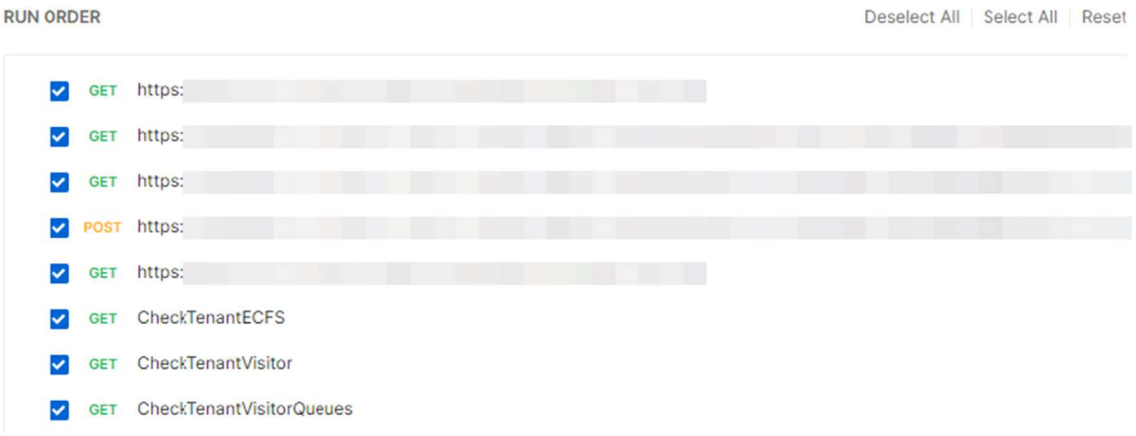

Figure 17. Postman Run order to be selected

Run order part contains the possibility to select or deselect part of the collection to run. In this test setup, all pixelated selections are required to run in order for the test setup to work. Tests, beginning with word “Check”, can be selected all or some or one of them. In Figure 17 full collection is run and so all of the tests will be run with this selection. This possibility is handy if only one or few tests are required to run and tester is short on time.

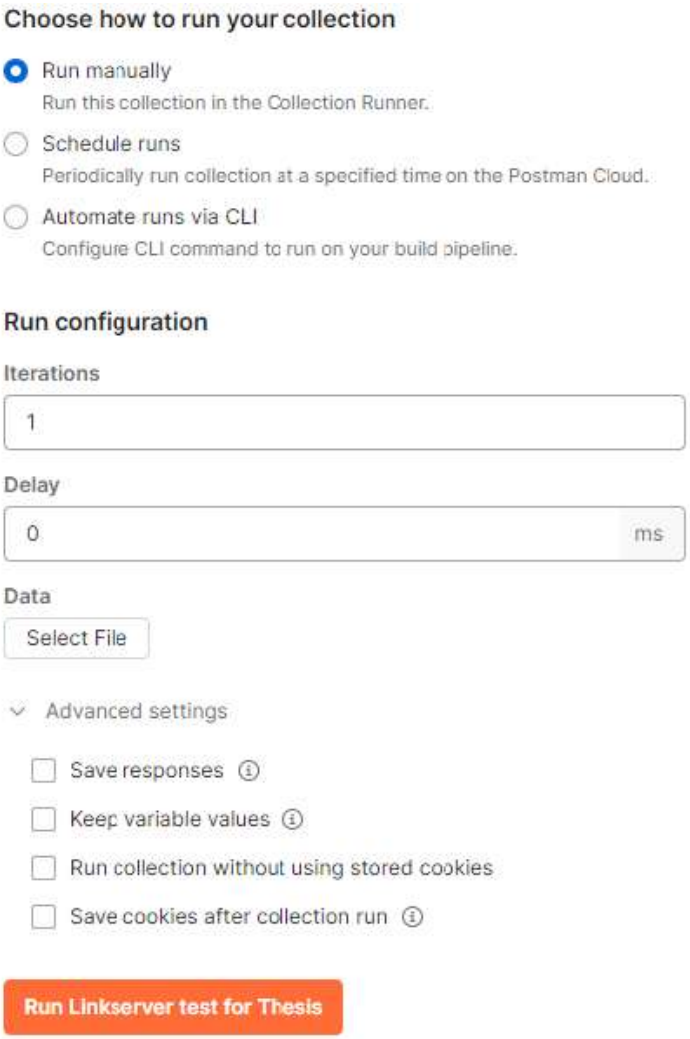

Figure 18. Postman details on how to run collection

How to run collection part contains the possibilities to change details on how collection is run and also has possibility to automate the collection run. Default settings are Keep variable values and Save cookies after collection run to be enabled. Because test setup created uses access tokens and cookies that are only valid for certain time, Save cookies option needs to be disabled. Keep variable values is disabled since in this test setup all values are wanted to be up-to-date and so refreshed each time collection is run.

For measuring each test running time efficiently, the run order part for was modified to contain one test and then collection was run 10 times for each test. In the end whole test setup was run also 10 times to see the time it needed for complete. From these 10 runs, also average values were calculated for each

step. All the tests were run on 6th of January 2023. Tests were run to 168 tenants. As the most high level of the tests, CheckTenantECFS was run first and then in order tests are in the Run order list.

5.2 CheckTenantECFS test run time

CheckTenantsECFS	Duration	Avg. Resp. Time
Run 1	2min 59s	705ms
Run 2	2min 16s	507ms
Run 3	1min 19s	171ms
Run 4	1min 14s	165ms
Run 5	1min 11s	144ms
Run 6	1min 10s	149ms
Run 7	1min 20s	184ms
Run 8	1min 20s	178ms
Run 9	1min 1s	127ms
Run 10	1min 15s	150ms
Average	1min 30s	248ms

Figure 19. CheckTenantECFS run times

In Figure 19 duration is the time that test took to run and Avg. Resp. Time is value provided by Postman and calculated average value from the response time each tenant had.

Here we have two run abnormalities, Run 1 and Run 2. Run 1 had two wrong data inputs from customer tenant list page. One of these was wrong tenant link for region, corrected by selecting the correct region for the tenant. Second wrong input was information about tenant that was no longer active. This wrong tenant information can be seen in Run 2 as well. Wrong tenant information was then corrected and not visible on Run 3 forward or with tests that come after this first one.

Average run time was calculated to be 1minute 30 seconds but Run 1 and Run 2 increase this amount. If average for Run 3 to Run 10 was calculated, the value would be 1 minute and 14 seconds. Similar value for average response time would be 159 milliseconds.

5.3 CheckTenantsVisitor test run time

CheckTenantsVisitor	Duration	Avg. Resp. Time
Run 1	1min 23s	179ms
Run 2	1min 15s	169ms
Run 3	1min 16s	166ms
Run 4	1min 17s	182ms
Run 5	1min 23s	179ms
Run 6	1min 25s	201ms
Run 7	1min 28s	177ms
Run 8	1min 43s	183ms
Run 9	1min 18s	159ms
Run 10	1min 11s	152ms
Average	1min 21s	175ms

Figure 20. CheckTenantsVisitor test run times

Since no failed tenants were found, values on Figure 20 CheckTenantsVisitor test run times, have less variation. The most variation against average value can be found from Run 8 where the run took 1 minute and 43 seconds to complete with average response time of 183 milliseconds. Average running time is similar to that of the test CheckTenantECFS.

5.4 Check TenantsVisitorQueues run time

CheckTenantsVisitorQueues	Duration	Avg. Resp. Time
Run 1	1min 32s	401ms
Run 2	1min	224ms
Run 3	51s	174ms
Run 4	1min 6s	193ms
Run 5	57s	208ms
Run 6	56s	207ms
Run 7	59s	223ms
Run 8	1min	194ms
Run 9	53s	168ms
Run 10	53s	185ms
Average	1min 1s	218ms

Figure 21. CheckTenantsVisitorQueues test run times

Where previous tests tested high level functionality TenantsVisitorQueues tests both API accessibility and database availability. Values in Figure 21 are interesting because test that seemed the most delicate is providing the fastest response times on average. With average run time of 1 minute and 1 second, and Run 1 taking more than 1 minute and 30 seconds, this test is the fastest to execute.

5.5 Full test run

Full Run	Duration	Avg. Resp. Time
Run 1	3min 10s	162ms
Run 2	2min 55s	151ms
Run 3	2min 45s	140ms
Run 4	2min 53s	147ms
Run 5	2min 56s	156ms
Run 6	3min 31s	184ms
Run 7	3min 40s	179ms
Run 8	3min 36s	194ms
Run 9	3min 15s	145ms
Run 10	3min 3s	145ms
Average	3min 11s	160ms

Figure 22. Full test run, running times

As could be estimated from run times in previous test runs, in Figure 22 we can see that these three tests take average of 3 minutes and 11 seconds to complete with average response time of 160ms. With 168 tenants, the average response time is calculated from total of 504 individual connectivity tests.

Point to take here is that, if these connectivity tests have failures, the time it takes to run is increased. Postman waits, by default, 60 seconds for response. In Run 1 where two failures were found, run took 120 seconds more time to complete. If full test collection execution was made in this scenario, containing three tests cases to be tested, the time is increased threefold. Running time would have increased by 360 seconds. In scenario where a lot of tenants have connectivity issues it would be sensible to run only the first test case CheckTenantECFS and end full run before it finishes.

6 Conclusions and further improvement points

The created test collection can be considered to be success. Test collection can be run by anyone on Operations team so running the tests is not limited to just few persons. Only limitation, outside of Operations teams know-how, comes from the fact that in Chapter 4, CognitoAsfData access token was created using *amazon-cognito-identity-js* JavaScript package. This token was created with QA teams help and running such packages is not something Operations team does. Token is long lived, active for one year once created, but replacement is required in time.

Full test run taking less than 4 minutes, when 504 individual tests are executed on 168 tenants. Active tenants are always fetched from customer link page when collection is run so up-to-date information is used in execution. Fast execution time also means that at this point all tenants can be tested at once. If tenant amount increases by a lot, regional runs may need to be created in order to test only the region that has been upgraded.

Improvement points for the future use would be:

1. Adding Communications panel login test to this Postman collection would be beneficial.
2. Generate new CognitoAsfData access token automatically, or atleast add notification when this is about to expire.
3. Automate this Postman collection run to run in background, either using Postman's own tools or with other means. Notifications about possible failures would be needed to fully utilize this automation.
4. Regional runs, instead of testing all tenants, if full test run time increases to tens of minutes.

References

1 Sinch. Contact Pro | Sinch's Omnichannel Cloud Contact Center [Internet]. Stockholm: Sinch; 2022 [Cited 2022 Oct 25]. Available from: https://www.sinch.com/products/customer-engagement/contact-pro/

2 Postman. Postman API Platform [Internet]. San Francisco: Postman; 2022 [Cited 2022 Oct 10]. Available from: https://www.postman.com/product/what-is-postman/

3 Lee Copeland. A Practitioner's Guide to Software Test design [Internet]. London: Artech House; 2003 [cited 2023 Nov 10]. Chapter 12. Available from: https://ebookcentral.proquest.com/lib/metropolia-ebooks/detail.action?docID=227688

4 Sinch Contact Pro Service Description [Internet]. Sinch; 2022. [Cited 2023 Jan 10]. [40 p]. Available from: https://docs.cc.sinch.com/cloud/service-description/en/Service_Description.pdf

5 Sinch. Service Configuration [Internet]. Stockholm: Sinch; 2022. [Cited 2022 Oct 25]. Available from: https://docs.cc.sinch.com/cloud/service-configuration/en/index.html visited

6 Amazon Web Services. Configuring MFA-protected API access – AWS Identity and Access Management [Internet]. Seattle: Amazon Web Services; 2022. [Cited 2022 Nov 26]. Available from: https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa_configure-api-require.html

7 Postman. Writing tests | Postman Learning Center [Internet]. San Francisco: Postman; 2022. [Cited 2022 Dec 05]. Available from: https://learning.postman.com/docs/writing-scripts/test-scripts/

8 Postman. Download Postman | Get Started for Free [Internet]. San Francisco: Postman; 2022. [Cited 2022 Dec 05]. https://www.postman.com/downloads/

9 Postman. Building requests | Postman Learning Center [Internet]. San Francisco: Postman; 2022. [Cited 2022 Dec 05]. Available from: https://learning.postman.com/docs/sending-requests/requests/

10 Postman. Generating client code | Postman Learning Center [Internet]. San Francisco: Postman; 2022. [Cited 2022 Dec 05]. Available from: https://learning.postman.com/docs/sending-requests/generate-code-snippets/

11 Wikipedia. Hypertext Transfer Protocol – Wikipedia [Internet]. San Francisco: Wikipedia; 2022. [Cited 2022 Dec 29]. Available from: https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

12 Wikipedia. HTTPS [Internet]. San Francisco: Wikipedia; 2022. [Cited 2022 Dec 05]. Available from: https://en.wikipedia.org/wiki/HTTPS

13 Amazon Web Services. What is Amazon Cognito? – Amazon Cognito [Internet]. Seattle: Amazon Web Services; 2022. [Cited 2022 Dec 06]. Available from: https://docs.aws.amazon.com/cognito/latest/developerguide/what-is-amazon-cognito.html

14 Amazon Web Services. What is RESTful API? – RESTful API Beginner's guide [Internet]. Seattle: Amazon Web Services; 2022. [Cited 2022 Dec 10]. Available from: https://aws.amazon.com/what-is/restful-api/

15 Sinch. Contact Pro Documentation [Internet]. Stockholm: Sinch; 2022. [Cited 2022 Dec 25]. Available from: https://docs.cc.sinch.com/cloud/api.html

16 Open source. Cheerio [Internet]. Global: Open source; 2022. [Cited 2022 Dec 28]. Available from: https://cheerio.js.org/

17 Amazon Web Services. Integrating Amazon Cognito with web and mobile apps – Amazon Cognito [Internet]. Seattle: Amazon Web Services; 2022. [Cited 2022 Dec 28]. Available from: https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-integrate-apps.html

18 Amazon Web Services. AuthenticateCognitoActionConfig – Elastic Load Balancer [Internet]. Seattle: Amazon Web Services; 2022. [Cited 2022 Dec 29]. Available from: https://docs.aws.amazon.com/elasticloadbalancing/latest/APIReference/API_AuthenticateCognitoActionConfig.html