Understand and Troubleshoot HAR Logs for WxCalling

Contents

Introduction

This document describes understanding and troubleshooting HAR logs.

Feature Overview

A **HAR Log** (Short for HTTP Archive Logging) is a log of all the network activity between your browser and a website during a specific session.

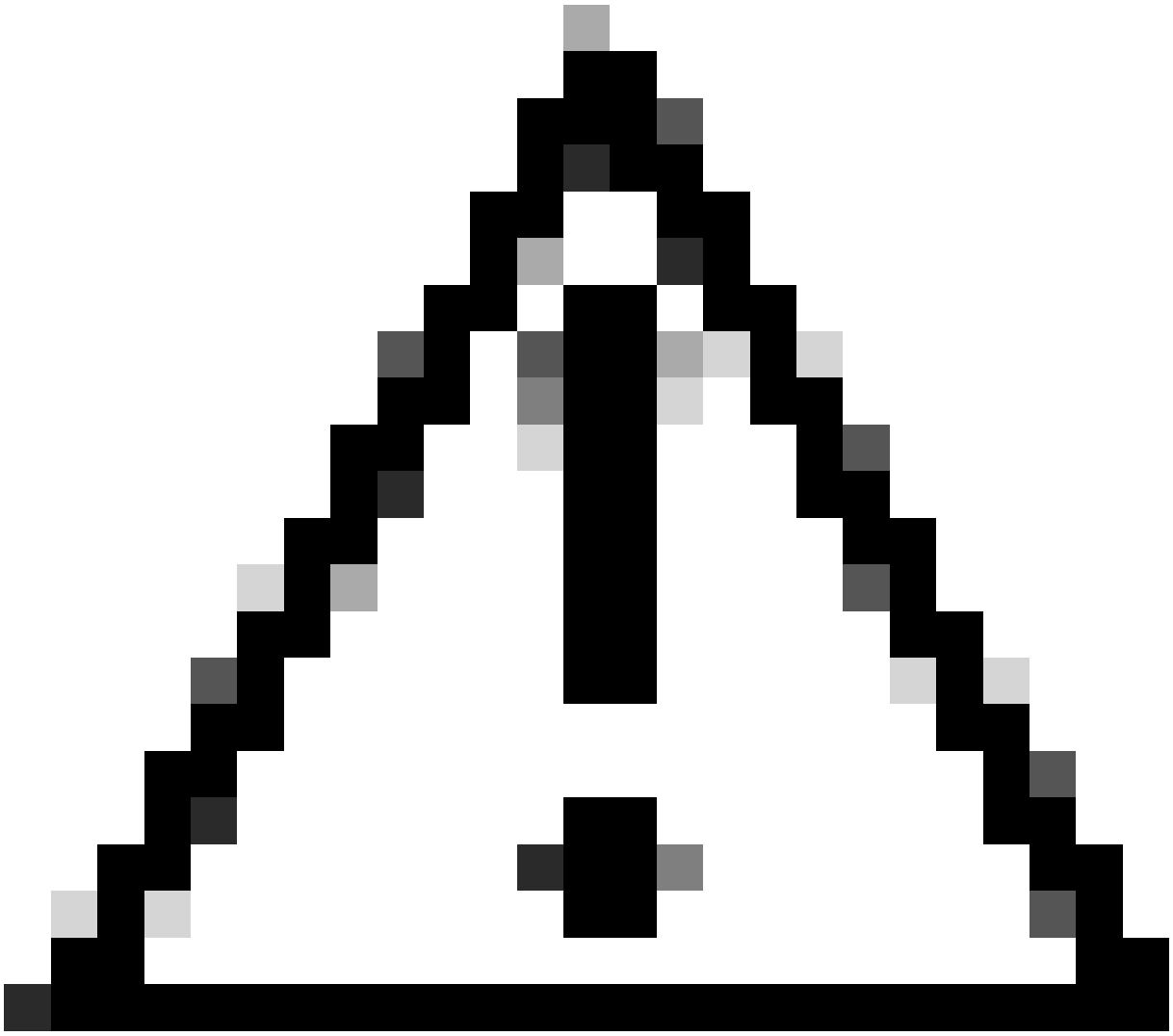
When you visit a website, your browser sends requests to the server and receives responses. The HAR file captures all those requests, responses, and the timing information.

They are essential for diagnosing web performance issues, troubleshooting network errors, and analyzing API transactions.

HAR logs help in identifying slow-loading resources, failed requests, and tracking user interactions with web applications.

Limitations

- HAR files can be large and difficult to parse manually.
- Sensitive data can be logged; ensure data sanitization before sharing.
- Some security policies can block full HAR capture.



Caution: The HAR file includes the entire content of the requests and responses in clear text (including passwords, session ID, cookies, etc). Therefore, you must not share a HAR file taken in

---

a session to a publicly available service. If the Network tab was opened when a password was entered, it is in clear text in the HAR file. You want to remove these credentials before sharing the HAR file. Open the saved HAR file in a text editor, find the password and replace it with a random text like "". Do not modify the JSON formatting of the file while making this change as it is then not parsed properly for review.

HAR Sanitizer Tool

History

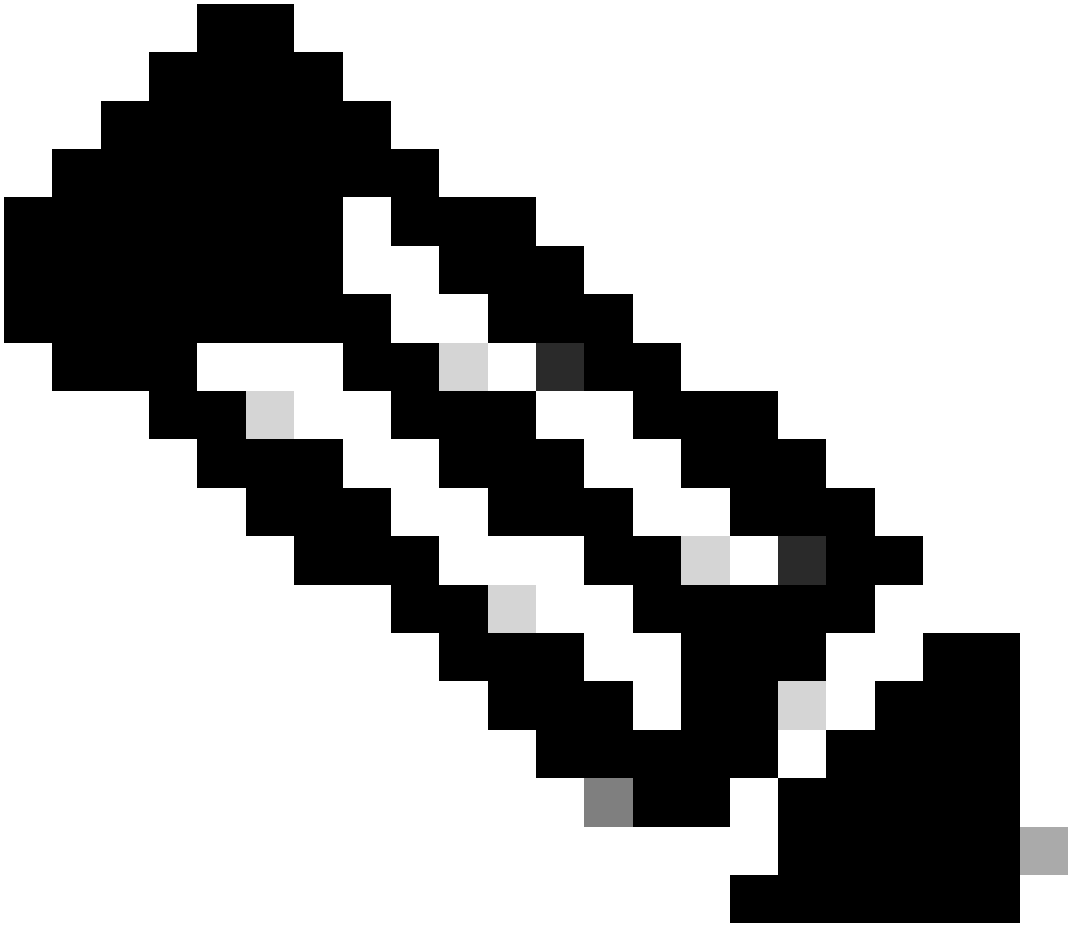
In 2023, Okta had a security breach where HAR files were stolen from their support case management systems. The attackers used the access token and cookies from those files to get access to their customer accounts. In response Okta implemented a sanitization tool.

Present State

Currently there is an ongoing automation in place that tries to strip out the sensitive materials from the traces, but leaves the rest intact to preserve the ability to troubleshoot and analyze your issue.

When you upload a HAR file, the tool automatically:

1. Detects that a HAR file has been uploaded
2. Sanitizes the HAR file using https://har-sanitize.cisco.com
3. Uploads the sanitized file to the case
4. Encrypts the original HAR file using GPG
5. Uploads the encrypted HAR file to the case
6. Deletes the original HAR file from the case
7. Adds a note the case explaining what has been done and a link to this tool



Note: If you cannot solve the issue using the sanitized version of the HAR file you can file an exception as explained in the next sections and get a copy of the original HAR file back.

How to Extract HAR Log

Generate a HAR File in Your Browser

Refer to: Generate a HAR File in Your Browser.

Extract from the document:

- Chrome:
  Open **Developer Tools (F12) > Network Tab > Preserve Log > Start Recording > Reproduce Issue > Export HAR File**.
- Firefox:
  Open **Developer Tools (F12) > Network Tab > Engine Icon > Check “Persist Logs” > Start Recording > Save HAR**.

Useful Information Before Analyzing HAR Logs

HTTP Methods

HTTP methods (also known as HTTP verbs) define the type of operation a client wants to perform on a given resource (such as data or a web page).

Each method has a specific purpose and semantics.

The most common HTTP methods are: **GET, POST, PUT, PATCH, DELETE, HEAD, OPTIONS**, and **TRACE**.

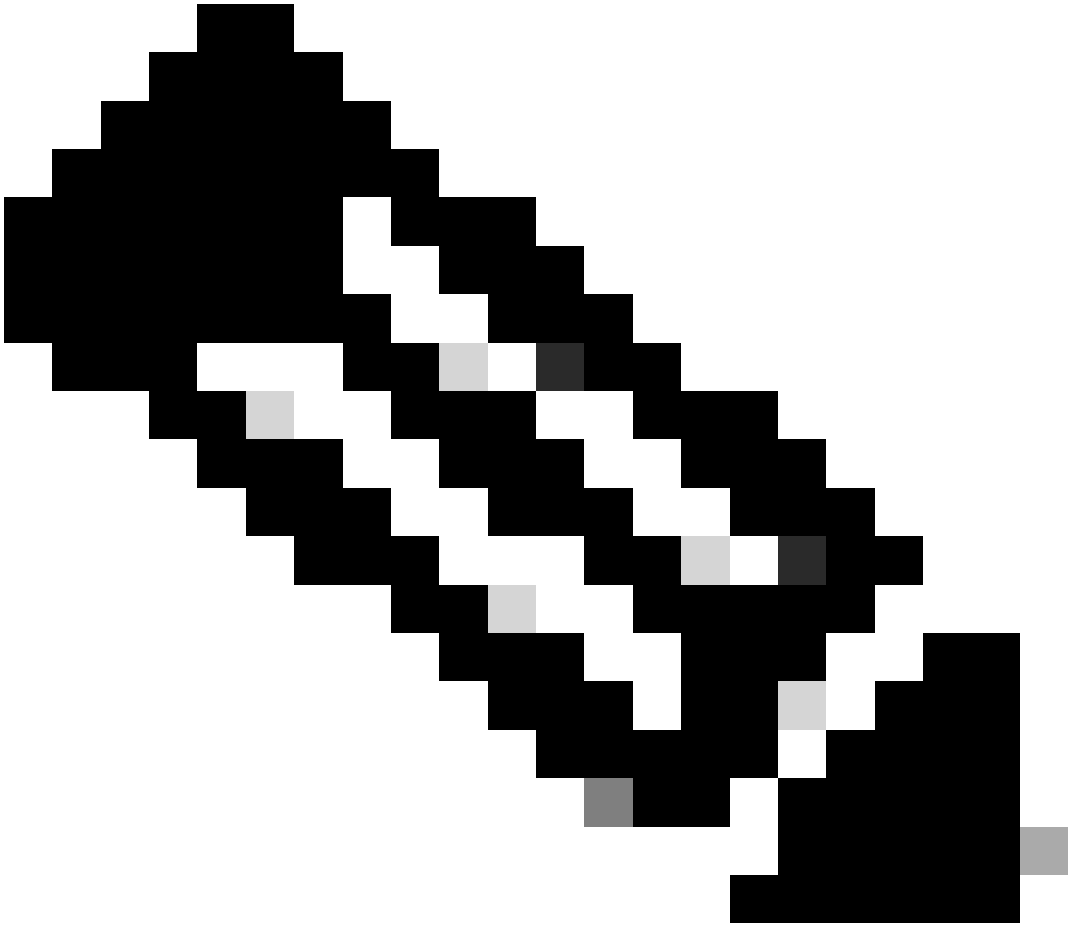
Method	Safe	Idempotent	Typical Use	Request Body	Response Body
GET	Yes	Yes	Retrieve data	No	Yes
POST	No	No	Create resource/submit data	Yes	Yes
PUT	No	Yes	Replace resource	Yes	Yes
PATCH	No	Yes*	Partially update resource	Yes	Yes
DELETE	No	Yes	Remove resource	No	Optional
HEAD	Yes	Yes	Retrieve headers	No	No
OPTIONS	Yes	Yes	Discover methods/features	No	Optional
TRACE	Yes	Yes	Diagnostic testing	No	Yes

* PATCH is generally idempotent, but it depends on implementation.

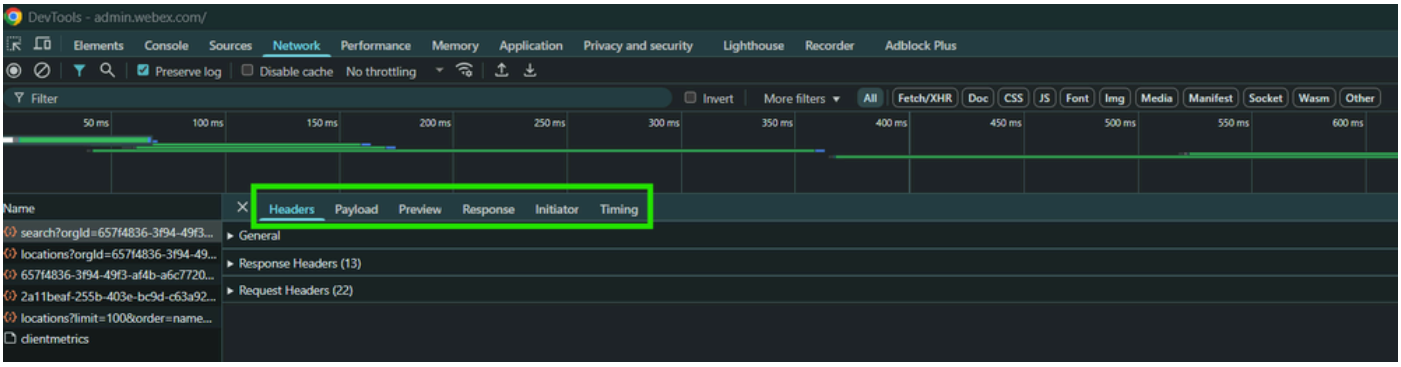
Explanation of Columns

- Safe: Indicates whether the HTTP method is considered safe, meaning it does not modify resources or server state Safe methods are intended only for retrieval, not for making changes. They are typically used for read-only operations.
- Idempotent: Specifies whether repeating the same HTTP request multiple times has the same effect as making it once. An idempotent method means that no matter how many times the request is repeated, the result remains unchanged (after the first successful request)
- Typical Use: Describes the common purpose or scenario for which the HTTP method is use. This gives context for when and why you would use each method in web development or API design.
- Request Body: Indicates whether the HTTP request typically includes a body (data sent to the server). The request body is often used to send data (such as JSON, XML, form data) to the server, such as when creating or updating resources.
- Response body: Specifies whether the HTTP response usually contains a body (data returned by the server). The response body is the data sent back from the server to the client, such as the requested resource, status messages, or confirmation of an action.

Anatomy of a HAR Log



Note: All the example images in this section, were taken when trying to load the locations in Control HUB for an organization



Headers

Captures metadata exchanged between the client (browser) and the server during HTTP requests and responses.

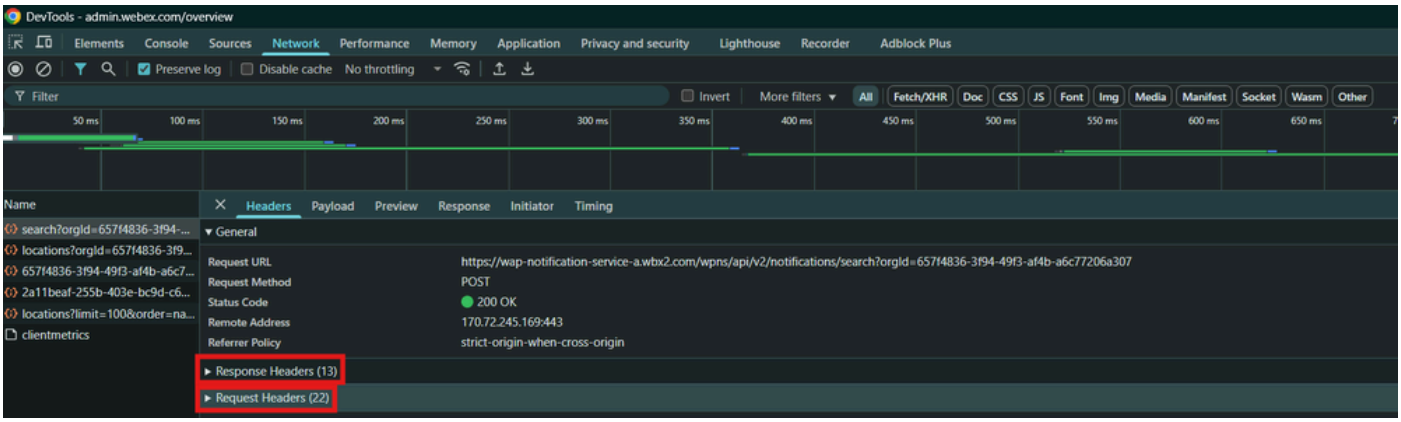
Headers provide context about the data being sent or received, such as the type of content, authentication information, caching policies, and more.

Headers are included in these sections of an HAR log:

1. Request Headers: These are sent from the browser (client) to the server as part of the HTTP request.
2. Response Headers: These are returned by the server to the browser in response to the request.

Headers are stored as arrays of key-value pairs, where:

- Key: The name of the header.
- Value: The value associated with the header.



Common Request Headers

1. Host: Specifies the domain name of the server (such as example.com) and the port number.

- Example: Host: www.example.com

2. User-Agent: Identifies the browser or client making the request, along with its version and operating system.

- Example: User-Agent: Mozilla/5.0 \(Windows NT 10.0; Win64; x64\)

3. Accept: Indicates the types of content the client can handle (such as HTML, JSON, images).

- Example: Accept: text/html,application/xhtml+xml

4. Accept-Encoding: Specifies the types of encoding (such as gzip, deflate) the client can decode.

- Example: Accept-Encoding: gzip, deflate

5. Authorization: Contains credentials for authentication, such as tokens or basic auth credentials.

- Example: Authorization: Bearer <token>

6. Cookies: Includes cookies sent by the client to the server.

- Example: Cookie: sessionId=12345; userPref=darkMode

7. Content-Type: Indicates the type of data being sent in the request body (for POST/PUT requests).

- Example: Content-Type: application/json

8. Referer: Identifies the URL of the page that referred the client to the current resource.

- Example: Referer: https://www.example.com/

Common Response Headers

1.Content-Type: Specifies the MIME type of the resource (such as text/html, application/json).

- Example: Content-Type: application/json

2. Content-Length: Indicates the size of the response body in bytes.

- Example: Content-Length: 1234

3. Cache-Control: Specifies caching policies for the resource (such as whether it is cacheable, and for how long).

- Example: Cache-Control: no-cache, no-store, must-revalidate

4. Server: Identifies the server software/version.

- Example: Server: Apache/2.4.29

5. Set-Cookie: Contains cookies that the server wants the client to store.

- Example: Set-Cookie: sessionId=67890; Path=/; Secure

6. Date: The date and time when the server generated the response.

- Example: Date: Tue, 10 Oct 2023 12:00:00 GMT

7. Location: Used in redirections, indicating the URL the client must navigate to.

- Example: Location: https://www.example.com/new-page

8. ETag: A unique identifier for the resource, often used for caching and versioning.

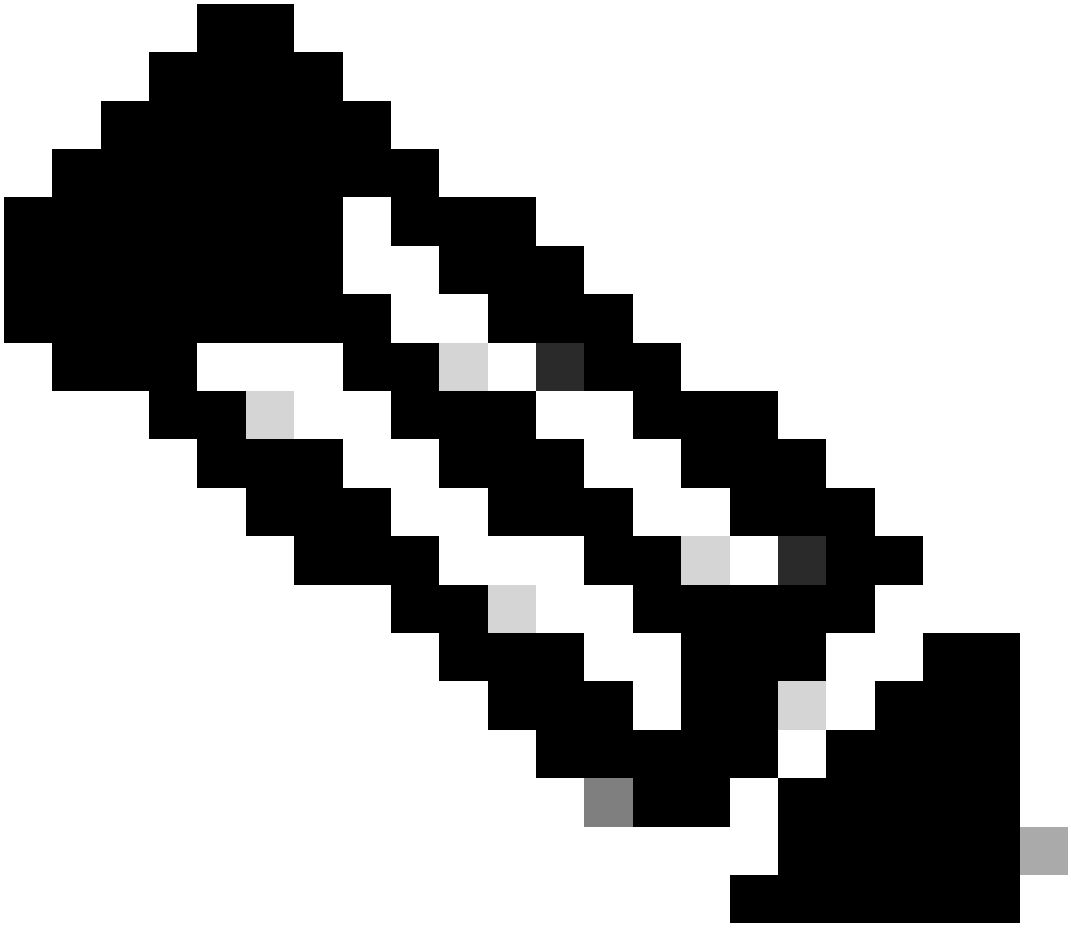
- Example: ETag: 12345abcd

9. Content-Encoding: Indicates how the response body is encoded (such as gzip, deflate).

- Example: Content-Encoding: gzip

10. Access-Control-Allow-Origin: Specifies which origins are allowed to access resources (used in CORS).

- Example: Access-Control-Allow-Origin: *



Note: Most relevant for Webex Calling is the Trackingid Header, as this is the ID that you can look up in the LMA Tool.

Payload

Data sent or received in the body of an HTTP request or response.

It is most commonly associated with POST, PUT, or PATCH requests, where the client sends data to the server, such as form submissions, file uploads, or JSON data for APIs.

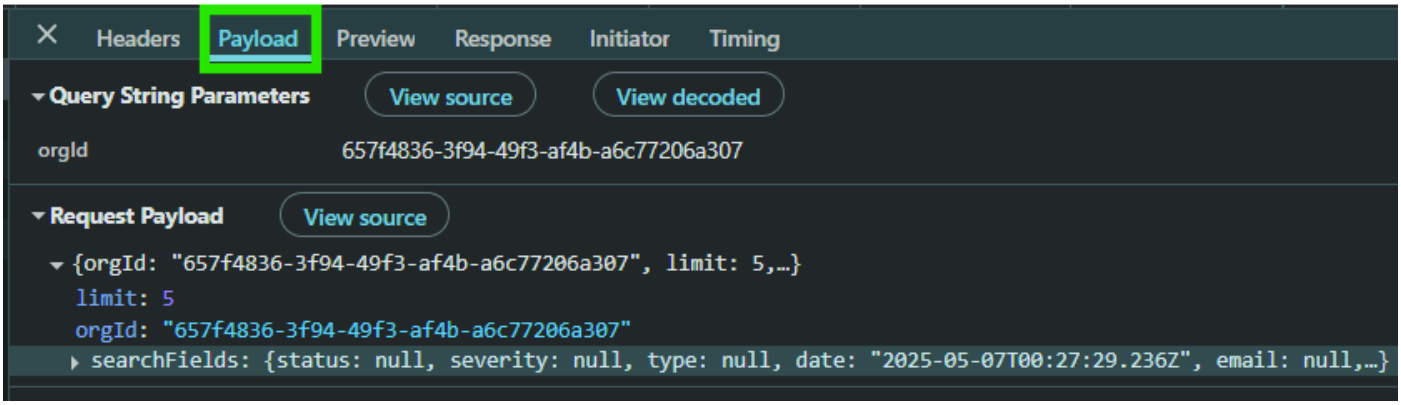
The payload can also exist in HTTP responses, which contain data returned by the server, such as HTML, JSON, or binary content (such as images, files).

Where Payload Appears in HAR Logs

The **Payload i**s typically found in two main sections of the HAR log:

1. Request Payload: Data sent from the client (browser) to the server in the body of an HTTP request.

2. Response Payload: Data returned by the server to the client in the body of an HTTP response.

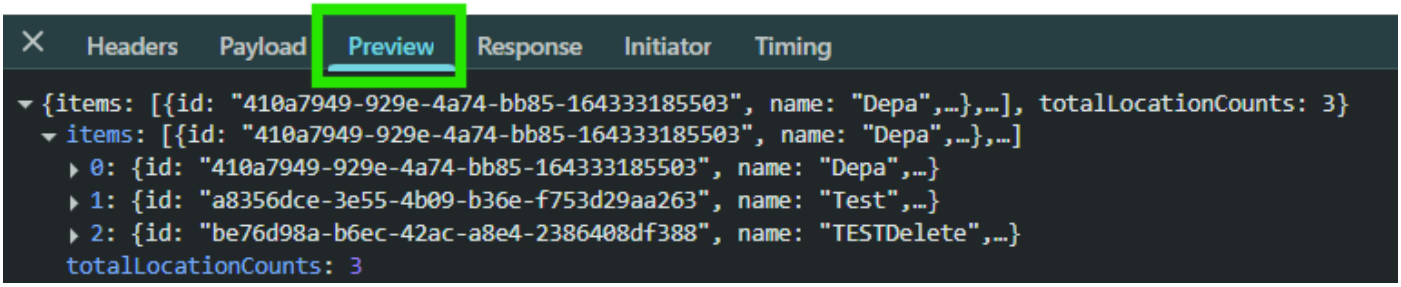


Preview

Is part of the response.content object and provides a representation of the data returned by the server in a structured and human-readable format, if available.

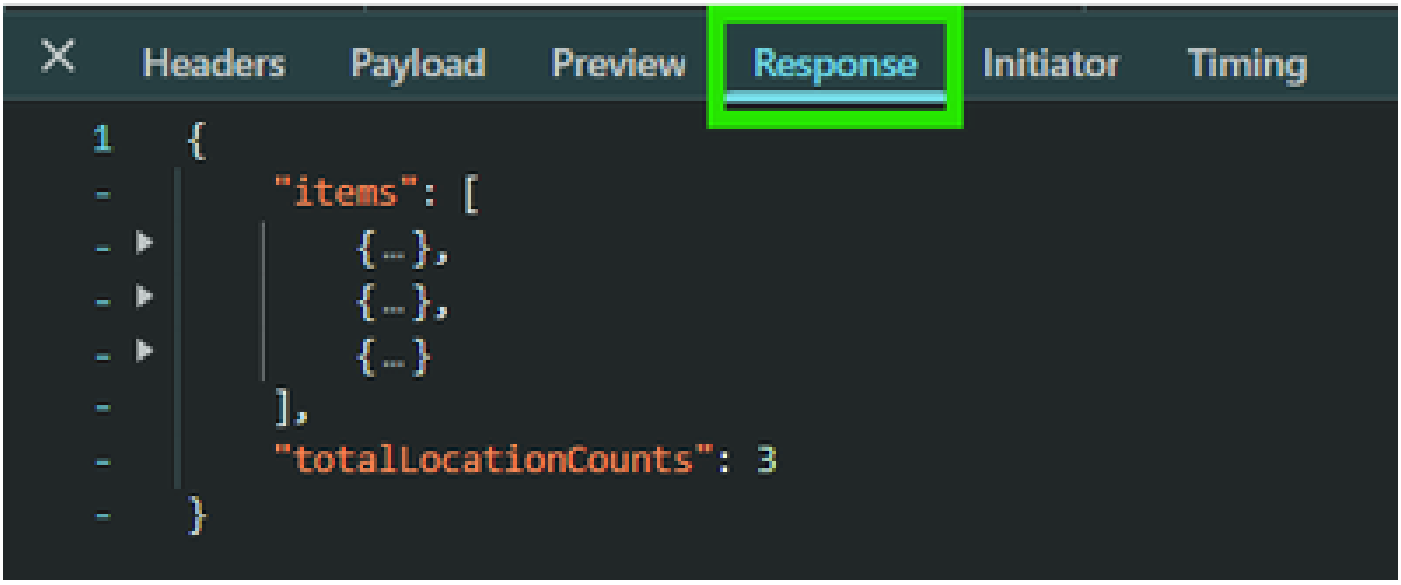
The **Preview** is typically used to display parsed or structured data from the response body in a user-friendly way, such as JSON, XML, or other formats.

This section is particularly useful for debugging APIs, inspecting returned data, or understanding the structure of the server response.



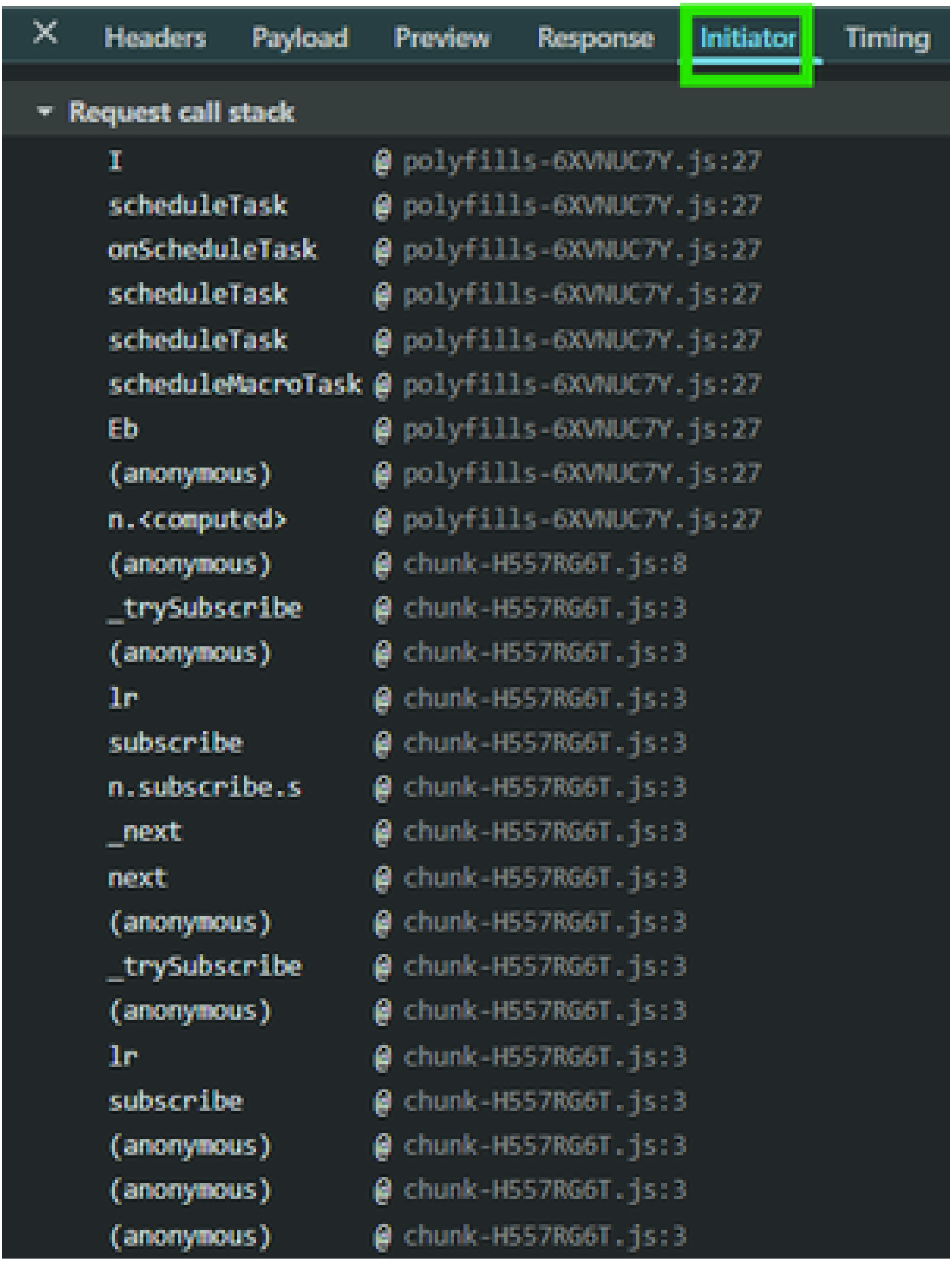
Response

The **Response** provides detailed information about the HTTP response sent by the server to the client (browser) for a specific request. This section contains metadata, headers, content details, and other critical data that can help you understand the behavior of the server during the request-response cycle. It provides a detailed snapshot of the server reply to an HTTP request.



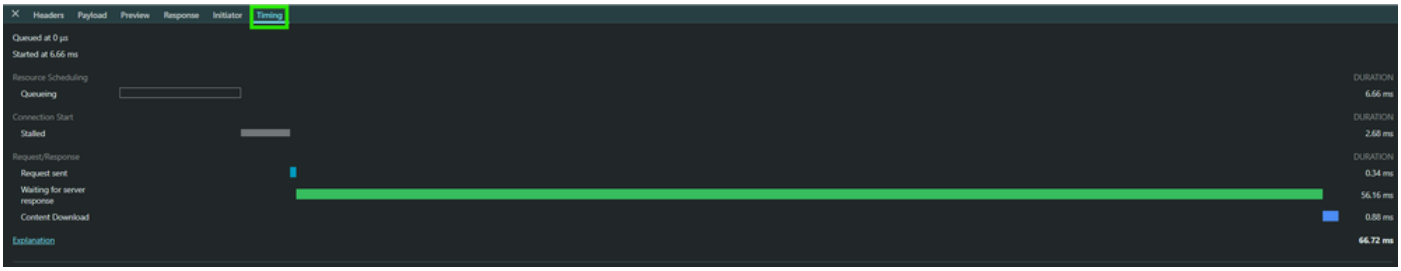
Initiator

The **Initiator** provides insight into what triggered a specific HTTP request during the loading of a webpage. It identifies the source or cause of a network request, helping developers understand the chain of events that led to the request. The initiator also helps trace the origin of a request and can point to the exact line of code or resource responsible for making it.


Headers
Payload
Preview
Response
Initiator
Timing
Request call stack
I polyfills-6XVNUC7Y.js:27
scheduleTask polyfills-6XVNUC7Y.js:27
onScheduleTask polyfills-6XVNUC7Y.js:27
scheduleTask polyfills-6XVNUC7Y.js:27
scheduleTask polyfills-6XVNUC7Y.js:27
scheduleMacroTask polyfills-6XVNUC7Y.js:27
Eb polyfills-6XVNUC7Y.js:27
(anonymous) polyfills-6XVNUC7Y.js:27
n.<computed> polyfills-6XVNUC7Y.js:27
(anonymous) chunk-H557RG6T.js:8
_trySubscribe chunk-H557RG6T.js:3
(anonymous) chunk-H557RG6T.js:3
lr chunk-H557RG6T.js:3
subscribe chunk-H557RG6T.js:3
n.subscribe.s chunk-H557RG6T.js:3
_next chunk-H557RG6T.js:3
next chunk-H557RG6T.js:3
(anonymous) chunk-H557RG6T.js:3
_trySubscribe chunk-H557RG6T.js:3
(anonymous) chunk-H557RG6T.js:3
lr chunk-H557RG6T.js:3
subscribe chunk-H557RG6T.js:3
(anonymous) chunk-H557RG6T.js:3
(anonymous) chunk-H557RG6T.js:3
(anonymous) chunk-H557RG6T.js:3

Timing

Timing provides a detailed breakdown of the various stages involved in processing an HTTP request and response. It helps developers understand how long each step of the request-response cycle takes, from initiating the connection to receiving the final response. Timing also tracks the sequence and duration of events that occur when a browser makes a request to a server and receives a response. It includes detailed metrics for DNS resolution, establishing the connection, sending the request, waiting for the server to respond, and downloading the response data.



Troubleshooting

Internal tool to View HAR Logs

Navigate to **EasyLmaSearch > Import HAR/Saz file**.

General Troubleshooting Steps

1. Open the HAR file in a .
2. Identify failed requests (such as HTTP 4xx/5xx errors).

3. Check response times and slow-loading elements.

4. Analyze request/response headers for authentication and CORS issues.

5. Look for tracking IDs in network requests and .
6. Cross-check failures against responses if posible.

Example Troubleshooting from Tickets

Slow-loading Elements

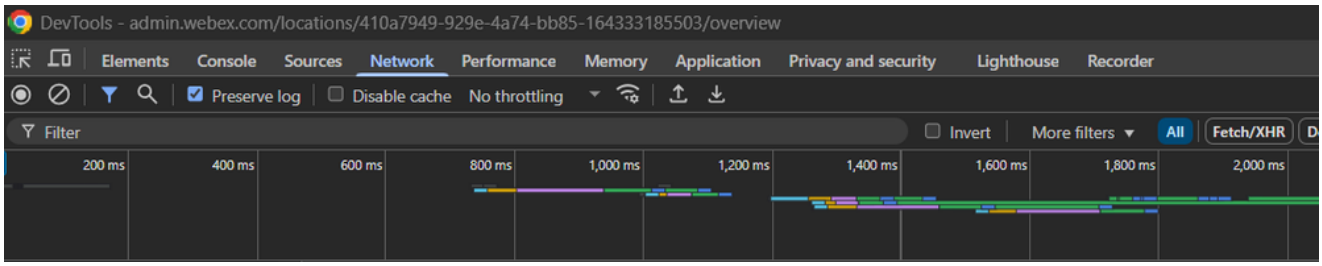
Example:

- Control Hub is taking long time to load the Numbers for a Location

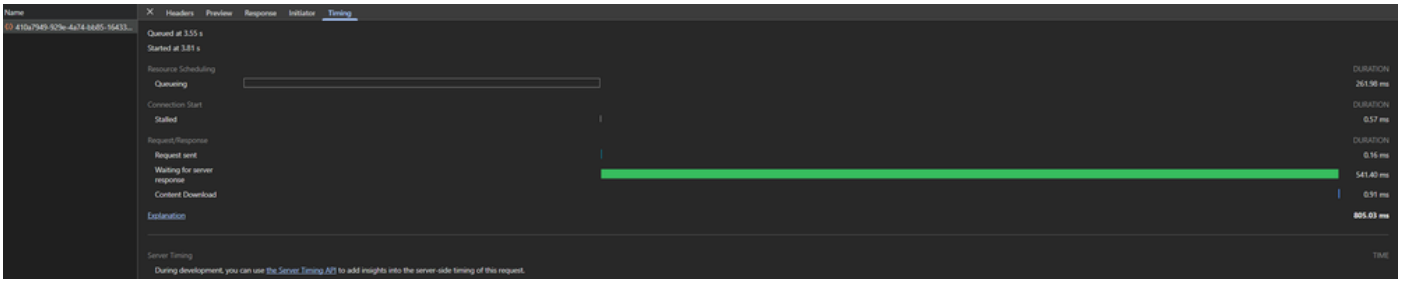
Troubleshooting:

- Request a video of the issue (trying to load the numbers page for a Location)
- Request a HAR log while trying to load the numbers for a location

1. Open the HAR file in a HAR Viewer or browser Developer Tools.
2. Identify the request in the **Waterfall** view



3. Click in the **Request** that is taking long to load.
4. Review the **Timing** tab of the log:



5. Check response times and slow-loading elements.

6. Gather the Trackingid for that **Header**.

7. Open EasyLMA and search with the trackingid.

Timing Breakdown Phases Explained

Here is more information about each of the phases you can see in the**Timing**tab:

- Queueing. The browser queues requests before connection start and when:
  - There are higher priority requests. Request priority is determined by factors such as the type of a resource, as well as its location within the document. For more information, read the resource priority section of the fetchpriority guide.
  - There are already six TCP connections open for this origin, which is the limit. (Applies to HTTP/1.0 and HTTP/1.1 only.)
  - The browser is briefly allocating space in the disk cache.
- Stalled. The request could be stalled after connection start for any of the reasons described in Queueing.
- DNS Lookup. The browser is resolving the request IP address.
- Initial connection. The browser is establishing a connection, including TCP handshakes or retries and negotiating an SSL.
- Proxy negotiation. The browser is negotiating the request with a proxy server.
- Request sent. The request is being sent.
- ServiceWorker Preparation. The browser is starting up the service worker.
- Request to ServiceWorker. The request is being sent to the service worker.
- Waiting (TTFB). The browser is waiting for the first byte of a response. TTFB stands for Time To First Byte. This timing includes 1 round trip of latency and the time the server took to prepare the response.
- Content Download. The browser is receiving the response, either directly from the network or from a service worker. This value is the total amount of time spent reading the response body. Larger than expected values could indicate a slow network, or that the browser is busy performing other work which delays the response from being read.

Resource Not Available

Example:

"I have enabled SNR for my user on admin control hub but I do not see the option to setup SNR number when I log into the user.webex.com portal.
Could you please check my org and user to see why I am unable to see it on user hub?"

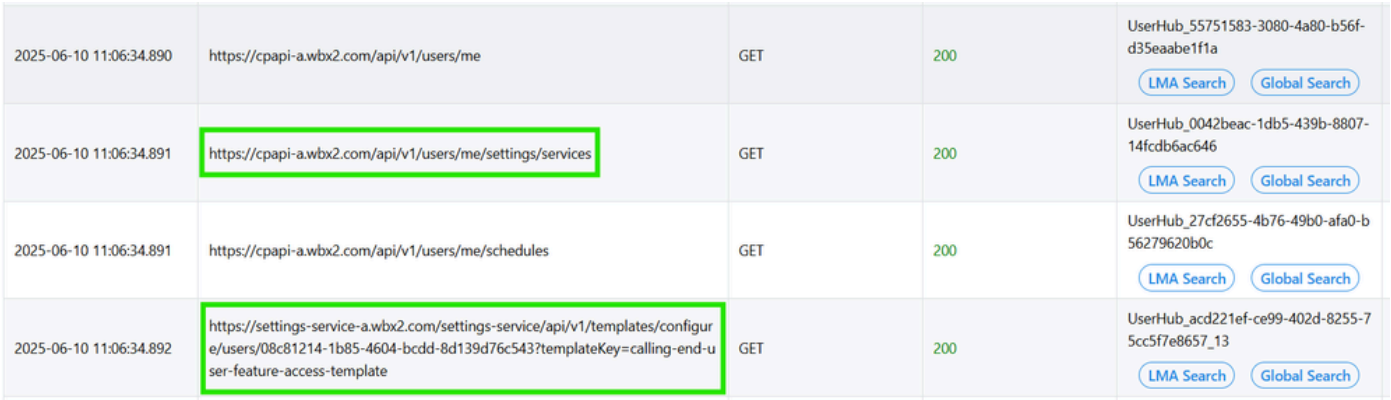
Troubleshooting:

1. Confirm what the user is seeing by asking for a screenshot of a working user versus a non working user.

2. Request a HAR Log while loading the options for the user.

Next steps:

1. Open the HAR file in a HAR Viewer or browser Developer Tools.
2. Identify requests:



Service Makes a Request (GET) to Know the Services of the User

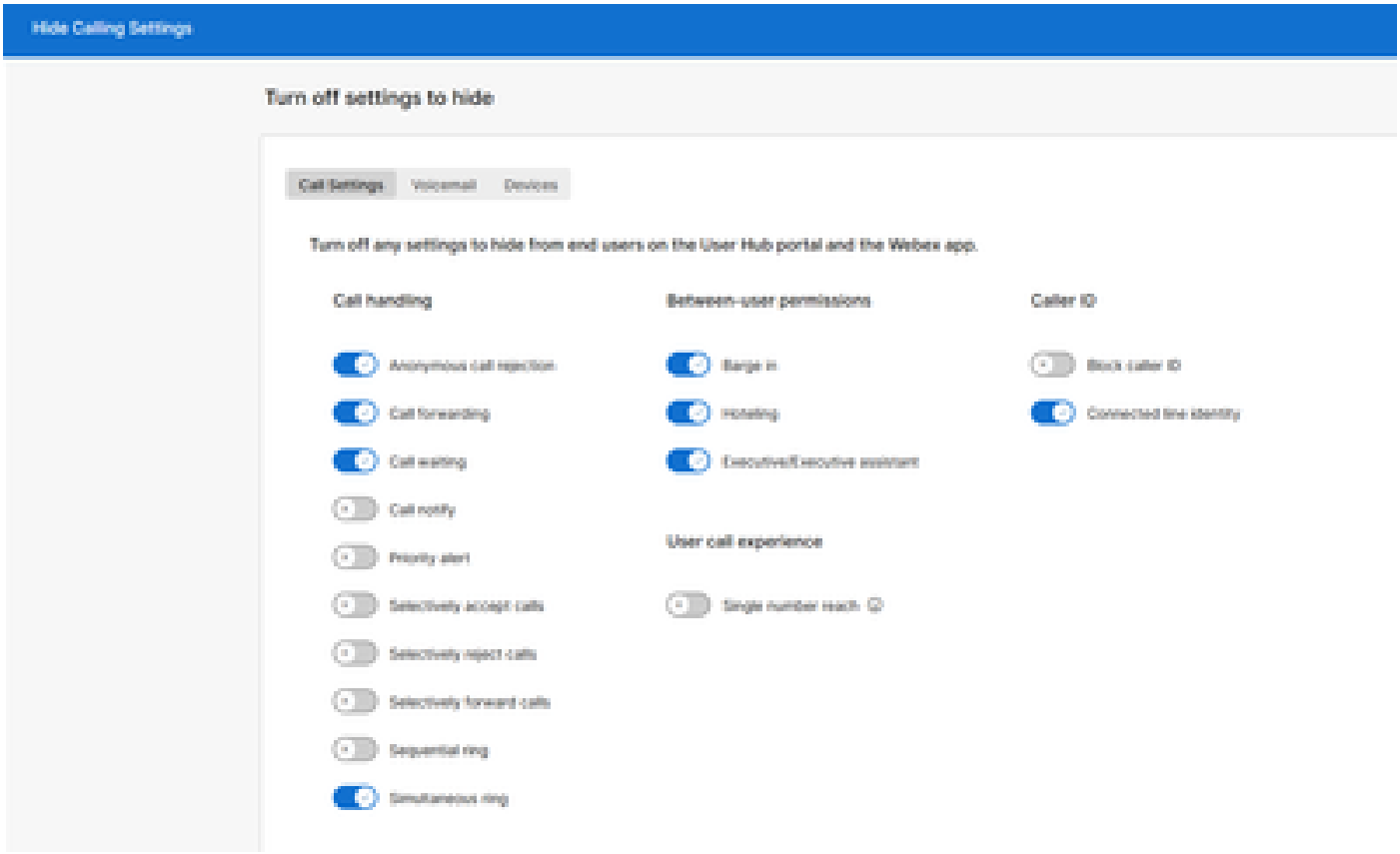
Calling End User Feature Access Template request (GET) the template of the services that are shown to the user:

1. Analyze request/response headers.

The "No Access" means the template is hiding those options for the user.

2. You need to review the template for the ORG and turn on the **Single number reach** for the user to be able to see it in the **User Hub**.

Example:



Can Not Enable a Feature

Example with call recording:

When trying to enable call recording for a user, you receive an error message: "Call recording change failed".

To troubleshoot:

1. Confirm the error by requesting the complete error message text.

2. Ask for a screenshot of the error.

3. Request a HAR log while trying to enable "Call Recording" in the user affected .

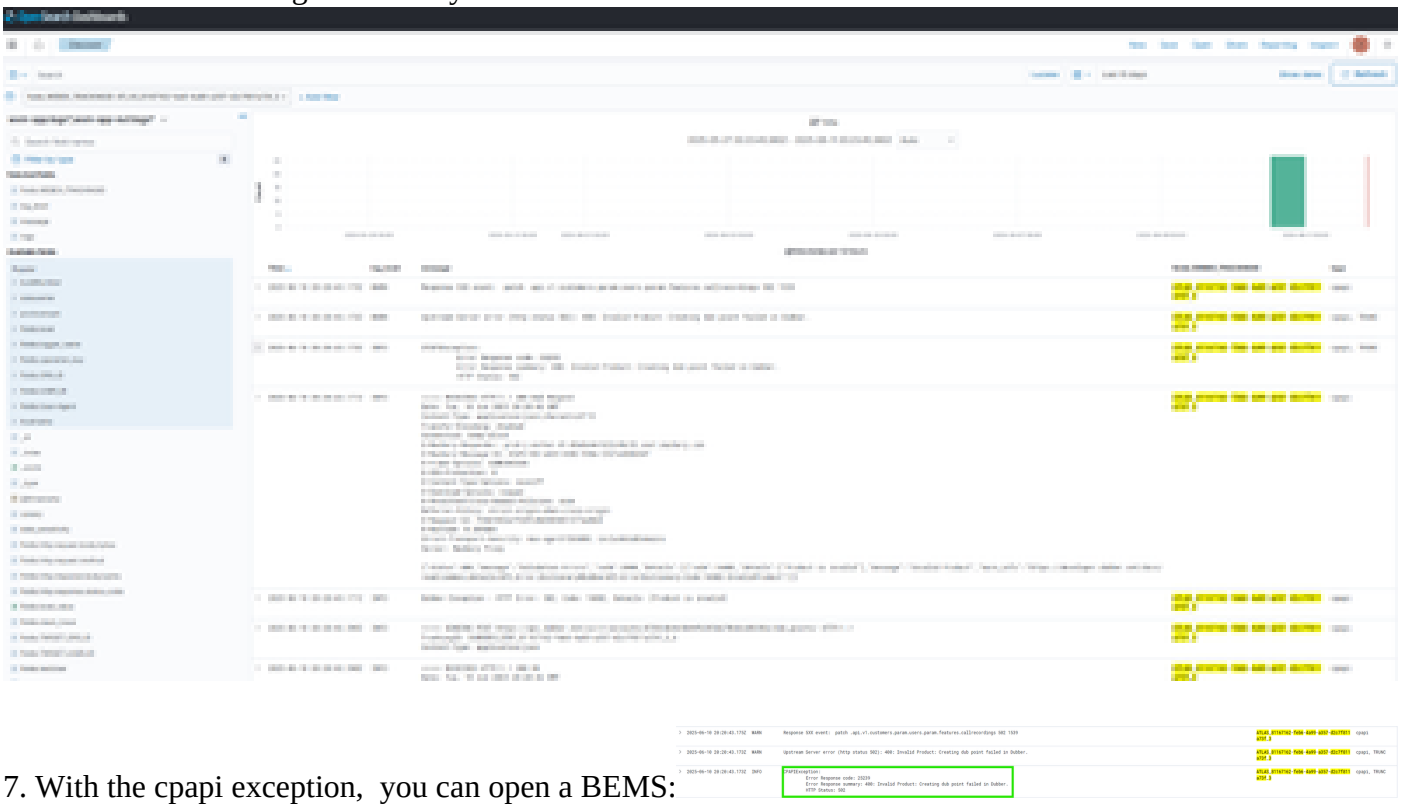
4. Open the HAR file in a HAR Viewer or browser Developer Tools.

5. Identify failed requests (such as HTTP 4xx/5xx errors).

POST Only | HAR Network Viewer

Started Datetime	URL	Method	Code	Tracking ID	Error	Org
2025-06-10 15:20:36.927	https://wap-notification-service-a.wbx2.com/wpns/api/v2/notifications/search?orgId=0769cbb4-c9b4-4fb3-95da-79bdce89296e	POST	200	ATLAS_81167162-feb6-4a99-a357-d2c7f811a73f_0 LMA Search Global Search		
2025-06-10 15:20:37.216	https://settings-service-r.wbx2.com/settings-service/api/v1/templates/configure/users/4229487a-4793-41a3-9257-232d20ce80ca?includeAllOrgTemplates=true	GET	200	ATLAS_81167162-feb6-4a99-a357-d2c7f811a73f_1 LMA Search Global Search		
2025-06-10 15:20:37.217	https://cpapi-r.wbx2.com/api/v1/customers/0769cbb4-c9b4-4fb3-95da-79bdce89296e/users/4229487a-4793-41a3-9257-232d20ce80ca/features/callrecordings	GET	200	ATLAS_81167162-feb6-4a99-a357-d2c7f811a73f_2 LMA Search Global Search		Ricart
2025-06-10 15:20:41.445	https://cpapi-r.wbx2.com/api/v1/customers/0769cbb4-c9b4-4fb3-95da-79bdce89296e/users/4229487a-4793-41a3-9257-232d20ce80ca/features/callrecordings	PATCH	502	ATLAS_81167162-feb6-4a99-a357-d2c7f811a73f_3 LMA Search Global Search	**400: Invalid Product: Creating dub point failed in Dubber.**	Ricart
2025-06-10 15:21:59.893	https://admin-batch-service-r.wbx2.com/api/v1/customers/0769cbb4-c9b4-4fb3-95da-79bdce89296e/jobs	GET	200	ATLAS_81167162-feb6-4a99-a357-d2c7f811a73f_5 LMA Search Global Search		Ricart

6. Look for the tracking IDs in EasyLMA.



7. With the cpapi exception, you can open a BEMS:

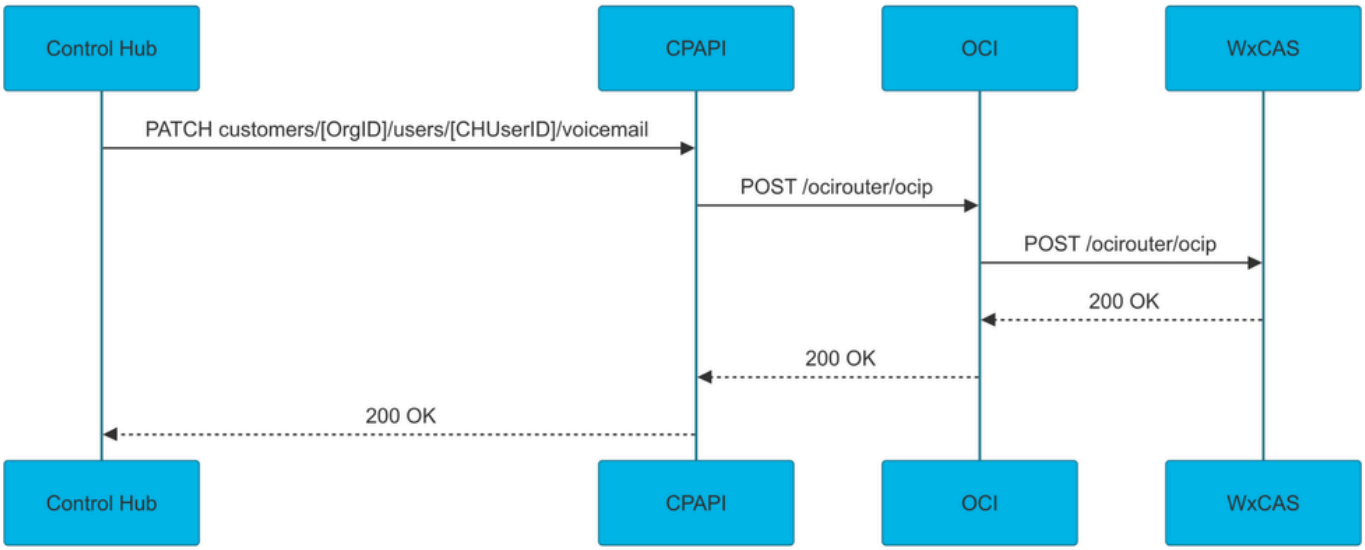
8. Open a BEMS with the information gathered:

- Screenshots
- Complete error Message
- TrackingID
- cpapi Exception

9. Ask in the Dubber space or to the BU team to review with the Dubber team the error:
"Error Response summary: 400: Invalid Product: Creating dub point failed in Dubber. HTTP Status: 502"

FAX Messaging

Here is the diagram illustrating the provisioning flow as it moves through microservices:



When troubleshooting fax messaging provisioning issues within Control Hub, it is essential to collect a HAR trace to gather detailed insights into the nature of the problem and understand the reasons behind provisioning failures.

When enabling the fax messaging feature, the HAR trace captures and display the relevant request from CH to CPAPI. This captured request follows a specific format.

From the CH → CPAPI:

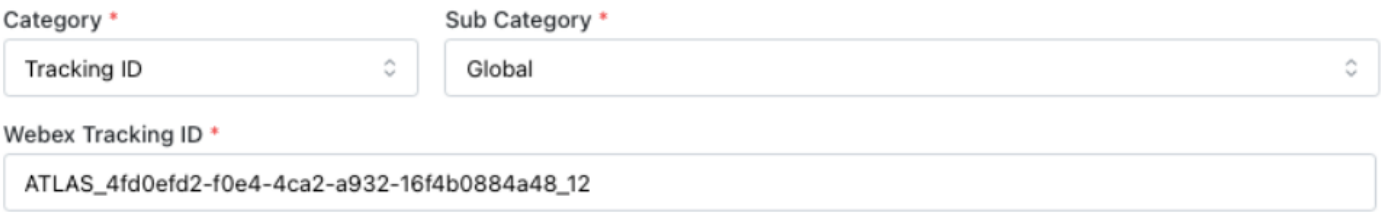
PATCH
Request URL:https://cpapi-r.wbx2.com/api/v1/customers/[OrgID]/users/[CHUserID]/voicemail
TrackingID ATLAS_4fd0efd2-f0e4-4ca2-a932-16f4b0884a48_12

Post Data {
"enabled": true,
"notifications": {
"enabled": true,
"destination": "lazoclaudiafi+faxmessaging@gmail.com"
},
"sendAllCalls": {
"enabled": false
},
"sendBusyCalls": {
"enabled": true,
"greeting": "DEFAULT"
},
"sendUnansweredCalls": {
"enabled": true,
"greeting": "DEFAULT",
"maxRings": 3
},
"transferToNumber": {
"enabled": false
},
"emailCopyOfMessage": {
"enabled": true,
"emailId": "lazoclaudiafi+faxmessaging@gmail.com"
},

"faxMessage": {
"enabled": false,
"phoneNumber": "+12099193323",
"extension": null
},
"messageStorage": {
"mwiEnabled": true,
"storageType": "INTERNAL",
"externalEmail": "lazoclaudiafi+faxmessaging@gmail.com"
}

To effectively track this information in EasyLMA, please refer to the detailed guidance provided here:

Category: TrackingID
Sub Category: Global
Webex Tracking ID: ATLAS_4fd0efd2-f0e4-4ca2-a932-16f4b0884a48_12



You can find the logs provided here:

From the CPAPI → OCI router:

SENDING POST https://ocirouter-rialto.broadcloudpbx.com:443/ocirouter/ocip HTTP/1.1
X-BroadWorks-Target: id=10f0e34e-7a42-46e7-9bb6-993bcd638f7d;type=enterprise
X-BroadWorks-Protocol-Version: 1.0
Content-Type: application/xml
TrackingID: CPAPI_4fd0efd2-f0e4-4ca2-a932-16f4b0884a48_12_0

OCIROUTER_4fd0efd2-f0e4-4ca2-a932-16f4b0884a48_12_0]: Rx [http] 10.71.101.37:80 -> ch3-bwks-v-ocir01-bc StatusCode=200

From the OCI Router → WXCAS:

10.71.128.200:37514
<?xml version="1.0" encoding="UTF-8"?>
<BroadsoftDocument xmlns="C" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance" protocol="OCI">
<externalUserIdentity xmlns="">
<id>159128f9-0758-46ac-85ff-120fae29c9ed</id>
<organizationId>10f0e34e-7a42-46e7-9bb6-993bcd638f7d</organizationId>
<role>Administrator</role>
</externalUserIdentity>
<trackingId xmlns="">CPAPI_4fd0efd2-f0e4-4ca2-a932-16f4b0884a48_12_1</trackingId>
<command xmlns="" xsi:type="UserVoiceMessagingUserModifyVoiceManagementRequest">
<userId>5849cbde-8ac7-43d6-8726-b5e0678a7904</userId>

<isActive>true</isActive>
<processing>Unified Voice and Email Messaging</processing>
<voiceMessageDeliveryEmailAddress>lazoclaudiafi+faxmessaging@gmail.com</voiceMessageDeliveryEmailAdd
<usePhoneMessageWaitingIndicator>true</usePhoneMessageWaitingIndicator>
<sendVoiceMessageNotifyEmail>true</sendVoiceMessageNotifyEmail>
<voiceMessageNotifyEmailAddress>lazoclaudiafi+faxmessaging@gmail.com</voiceMessageNotifyEmailAddress>
<sendCarbonCopyVoiceMessage>true</sendCarbonCopyVoiceMessage>
<voiceMessageCarbonCopyEmailAddress>lazoclaudiafi+faxmessaging@gmail.com</voiceMessageCarbonCopyE
<transferOnZeroToPhoneNumber>false</transferOnZeroToPhoneNumber>
<alwaysRedirectToVoiceMail>false</alwaysRedirectToVoiceMail>
<busyRedirectToVoiceMail>true</busyRedirectToVoiceMail>
<noAnswerRedirectToVoiceMail>true</noAnswerRedirectToVoiceMail>
</command>
<command xmlns="" xsi:type="UserVoiceMessagingUserModifyGreetingRequest20">
<userId>5849cbde-8ac7-43d6-8726-b5e0678a7904</userId>
<busyAnnouncementSelection>Default</busyAnnouncementSelection>
<noAnswerAnnouncementSelection>Default</noAnswerAnnouncementSelection>
<noAnswerNumberOfRings>3</noAnswerNumberOfRings>
</command>
<command xmlns="" xsi:type="UserFaxMessagingModifyRequest">
<userId>5849cbde-8ac7-43d6-8726-b5e0678a7904</userId>
<isActive>false</isActive>
<phoneNumber>+12099193323</phoneNumber>
<extension xsi:nil="true"/>
</command>
</BroadsoftDocument>

Escalation Information

- HAR Log File
- Screenshots of Errors
- Steps to Reproduce the Issue
- Timestamp of Incident
- LMA logs

Please answer these questions before opening the BEMS escalation as this helps with the further troubleshooting:

- What error do you see?
- What Tracking IDs do you see?
- Did you review the Tracking ID in LMA?
- What do you see in LMA?
- Is this BEMS really necessary?