**HTTP/2 Header Sanitization for AWS Lambda - User Guide**

This guide provides step-by-step instructions for implementing the HTTP/2 header sanitization solution for your AWS Lambda functions behind an Application Load Balancer (ALB).

**Table of Contents**

1. [Quick Start](https://claude.ai/chat/beb2149b-d03b-484b-8f02-ea8ef435b5e3#quick-start)
2. [Option 1: Lambda Layer Approach](https://claude.ai/chat/beb2149b-d03b-484b-8f02-ea8ef435b5e3#option-1-lambda-layer-approach)
3. [Option 2: Python Wrapper Approach](https://claude.ai/chat/beb2149b-d03b-484b-8f02-ea8ef435b5e3#option-2-python-wrapper-approach)
4. [Verifying the Solution](https://claude.ai/chat/beb2149b-d03b-484b-8f02-ea8ef435b5e3#verifying-the-solution)
5. [Troubleshooting](https://claude.ai/chat/beb2149b-d03b-484b-8f02-ea8ef435b5e3#troubleshooting)
6. [FAQs](https://claude.ai/chat/beb2149b-d03b-484b-8f02-ea8ef435b5e3#faqs)

**Quick Start**

**Prerequisites**

* AWS CLI installed and configured
* Existing Lambda function(s) behind an ALB with HTTP/2 enabled
* Symptoms: HTTP/2 PROTOCOL\_ERROR or broken responses with HTTP/2 clients

**Choose Your Implementation Method**

We offer two approaches:

1. **Lambda Layer** (recommended): No code changes, works with any runtime
2. **Python Wrapper**: Simple code addition, Python-specific

**Option 1: Lambda Layer Approach**

Follow these steps to use our pre-built Lambda Layer:

**Step 1: Add the Lambda Layer**

**Using AWS Console**

1. Navigate to your Lambda function in the AWS Console
2. Scroll down to the Layers section
3. Click "Add a layer"
4. Select "Specify an ARN"
5. Enter the ARN of our published layer: arn:aws:lambda:us-east-1:ACCOUNT\_ID:layer:CustomLambdaWebAdapter:1
   * Replace ACCOUNT\_ID with your AWS account ID
   * Use the latest version number available

**Using AWS CLI**

aws lambda update-function-configuration \

--function-name YOUR\_FUNCTION\_NAME \

--layers arn:aws:lambda:us-east-1:ACCOUNT\_ID:layer:CustomLambdaWebAdapter:1

**Step 2: Add Environment Variable**

**Using AWS Console**

1. Navigate to your Lambda function in the AWS Console
2. Scroll down to the Environment variables section
3. Click "Edit"
4. Add a new environment variable:
   * Key: AWS\_LAMBDA\_EXEC\_WRAPPER
   * Value: /opt/extensions/bootstrap
5. Click "Save"

**Using AWS CLI**

aws lambda update-function-configuration \

--function-name YOUR\_FUNCTION\_NAME \

--environment "Variables={AWS\_LAMBDA\_EXEC\_WRAPPER=/opt/extensions/bootstrap}"

**Step 3: Deploy and Test**

1. Trigger a new deployment if needed
2. Test with an HTTP/2 client:
3. curl --http2 -v https://your-alb-dns-name.region.elb.amazonaws.com/your-path

**Option 2: Python Wrapper Approach**

For Python Lambda functions only, you can add a simple wrapper to your handler code:

**Step 1: Add Sanitization Function**

Add this to your Lambda handler file:

def sanitize\_http2\_headers(response):

"""Sanitize HTTP/2 disallowed headers"""

# List of disallowed headers in HTTP/2

disallowed\_headers = [

"connection",

"keep-alive",

"proxy-connection",

"transfer-encoding",

"upgrade"

]

# Remove disallowed headers (case-insensitive)

if "headers" in response and response["headers"]:

sanitized\_headers = {}

for header\_name, header\_value in response["headers"].items():

if header\_name.lower() not in disallowed\_headers:

sanitized\_headers[header\_name] = header\_value

# Replace headers with sanitized version

response["headers"] = sanitized\_headers

return response

**Step 2: Modify Your Handler**

Wrap your response with the sanitization function:

def handler(event, context):

# Your original handler code

response = {

"statusCode": 200,

"headers": {

"Content-Type": "text/plain",

# Headers like Connection and Keep-Alive might be added automatically

},

"body": "Your response content"

}

# Apply sanitization before returning

return sanitize\_http2\_headers(response)

**Step 3: Deploy and Test**

1. Deploy your updated Lambda function
2. Test with an HTTP/2 client:
3. curl --http2 -v https://your-alb-dns-name.region.elb.amazonaws.com/your-path

**Verifying the Solution**

To verify the solution is working:

1. **Check HTTP/2 Protocol Status**:
2. curl --http2 -v https://your-alb-dns-name.region.elb.amazonaws.com/your-path

Look for Using HTTP/2 in the output and confirm no PROTOCOL\_ERROR messages.

1. **Examine Response Headers**:
2. curl --http2 -v https://your-alb-dns-name.region.elb.amazonaws.com/your-path 2>&1 | grep -i "connection\|keep-alive"

No Connection or Keep-Alive headers should appear in the response.

1. **CloudWatch Logs**:
   * For Lambda Layer approach: Look for "AWS Lambda Web Adapter with HTTP/2 header sanitization starting"
   * For Python wrapper: Add logging to your sanitization function to verify it's being called

**Troubleshooting**

**Common Issues**

**1. HTTP/2 Errors Still Occurring**

Check:

* Verify your ALB has HTTP/2 enabled
* Ensure you're testing with HTTPS (HTTP/2 requires HTTPS)
* Check that the Lambda Layer is correctly attached
* Verify the environment variable is set correctly

**2. Lambda Takes Longer to Initialize**

The Lambda Layer adds a small overhead to cold starts. This is expected and usually minimal (100-200ms).

**3. Layer Not Found Error**

If you see "Layer version arn:aws:lambda:..." not found" error:

* Confirm you're using the correct region in the ARN
* Verify you have access to the layer
* Try publishing the layer to your own account using the provided scripts

**4. "Cannot execute binary file" Error**

This indicates an architecture mismatch. Rebuild the layer for the correct architecture (Linux x86\_64).

**FAQs**

**Q: Will this affect HTTP/1.1 clients?**

No. HTTP/1.1 clients can safely include the Connection and Keep-Alive headers. Our solution only removes the headers for HTTP/2 compatibility.

**Q: Does this modify the request headers?**

No. The solution only modifies response headers. Request headers are untouched.

**Q: What's the performance impact?**

Minimal. The Lambda Layer adds a small memory overhead (~10MB) and negligible processing time (<1ms per request).

**Q: Will this break my existing Lambda function?**

No. The solution is designed to be non-invasive and only affects HTTP/2-incompatible headers.

**Q: Do I need to modify my framework or application code?**

No. With the Lambda Layer approach, your application code remains unchanged.

**Q: How do I update to a newer version of the Layer?**

Simply update your Lambda function's configuration to use the latest layer version:

aws lambda update-function-configuration \

--function-name YOUR\_FUNCTION\_NAME \

--layers arn:aws:lambda:us-east-1:ACCOUNT\_ID:layer:CustomLambdaWebAdapter:NEW\_VERSION

**Q: Can I use this solution with any Lambda runtime?**

Yes, the Lambda Layer approach works with any runtime supported by AWS Lambda.

**Q: What if I'm not using an ALB?**

This solution specifically addresses an issue with ALB + Lambda + HTTP/2. If you're using API Gateway or CloudFront, they already handle HTTP/2 headers correctly.