Audit Logging Service Integration Guide

Overview

Our team has developed a new functionality that enables saving request and response data for later analysis. This document provides a detailed guide on how to integrate and use the Audit Logging Service. The service is designed to capture HTTP requests and responses, persist them in storage, and enable the generation of reports. These reports provide insights into API usage patterns, helping teams make data-driven decisions, optimize feature adoption, and proactively address needs.

How the Audit Logging Service Works

The Audit Logging Service operates on an API-based model. It accepts a `POST` request and returns a status code `204`, indicating that the logging request has been successfully consumed and will be processed. Currently, the service supports Kafka as the target. The service is instrumental in capturing product metrics, which are critical for understanding feature value, driving future investments, and supporting real-time dashboards for operational insights.

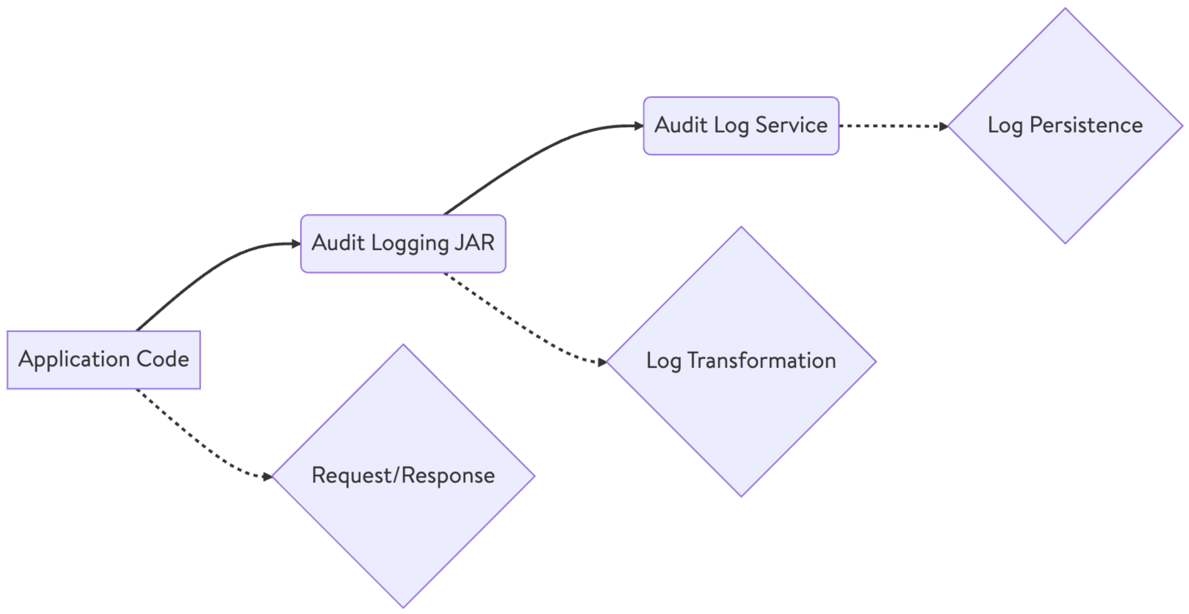
Sample cURL Request

Audit log service cURL

```
curl --location 'https://us.stg.proxy-gateway.walmart.com/api-proxy/service/audit/api-logs-srv/v1/logs/api-
requests' \
--header 'WM_SVC.NAME: AUDIT-API-LOGS-SRV' \
--header 'WM_SVC.ENV: stg:1.0.0' \
--header 'WM_SVC.VERSION: 1.1.0' \
--header 'WM_CONSUMER.ID: fda7cddb-b0ea-451e-9d2a-b090a08290a' \
--header 'Content-Type: application/json' \
--header 'WM_SEC.AUTH_SIGNATURE:
--header 'WM_CONSUMER.INTIMESTAMP: 1741855876885' \
--data '{
    "request_id": "70e0b563-f618-43bb-90c5-8be8bf40test",
    "service_name": "NRT",
    "endpoint_name": "transactionHistory",
    "version": "v1",
    "path": "/store/1998/gtin/00030772056301/transactionHistory",
    "method": "GET",
    "request_body": {
        "messageId": "dr-test-post-dr-1",
        "eventType": "ARRIVAL",
        "storeNbr": 45,
        "lineInfo": [
            {
                "gtin": "00044444444446",
                "secondaryItemIdentifier": {
                    "type": "ItemNbr",
                    "value": "553352632"
                },
                "destinationLocation": {
                    "locationArea": "STORE",
                    "location": "A142-002",
                    "lpn": "1234567890"
                },
                "quantity": 1,
                "expiryDate": "2023-09-30"
            }
        ],
        "documentInfo": [
            {
                "docType": "INVOICE",
                "docNbr": "10077",
                "docDate": 1676902620356
            }
        ],
        "userId": "testuser",
        "reasonDetails": [
            {
                "reasonCode": "xyz",
                "reasonDesc": "xyz"
            }
        ],
        "vendorNbr": "1234567",
        "eventCreationTime": 1676902620356
    },
    "response_body": {},
    "error_reason": "",
    "response_code": 200,
    "request_ts": 1739948540,
    "supplier_company": "luminate_company_id",
    "response_ts": 1739948548,
    "request_size_bytes": 2000,
    "response_size_bytes": 16,
    "created_ts": 1739948552,
    "trace_id": "182589fa7f75585af5b05b1ff556f213",
    "headers": {
        "wm-site-id": "1694066566785477000",
        "wm_consumer.id": "18260647337f525a4699eb25e05test"
    }
}'
```

****Note:**** ***To enable the Audit Logging Service, please raise a ticket with our team to onboard your respective consumer ID for API access.***

### Integration Flow Diagram : Below is a high-level flow diagram illustrating the integration process:



Integration Steps

1. **Add Dependency to `pom.xml`**

Add the following dependency to your `pom.xml` file:

Dependency of audit log jar - jdk 11

```
<dependency>
   <groupId>com.walmart</groupId>
   <artifactId>dv-api-common-libraries</artifactId>
   <version>0.0.52</version> <!-- For JDK 11 -->
</dependency>
```

Dependency of audit log jar - jdk 17

```
<dependency>
   <groupId>com.walmart</groupId>
   <artifactId>dv-api-common-libraries</artifactId>
   <version>0.0.51</version> <!-- For JDK 17 -->
</dependency>
```

****Tip:**** ***While specifying the version is optional (as the latest JAR is backward compatible), it is strongly recommended to use a specific version.***

2. **Configure Spring Boot Application**

In the main Spring Boot application class, specify the packages to scan for JAR utilities by adding the following to the `scanBasePackages` attribute:

Springboot Application configuration

```
@SpringBootApplication(scanBasePackages = {
        "com.walmart.dv.filters",
        "com.walmart.dv.services"
    })
```

3. **Build the Project**

Run the following Maven command to include the JAR in your external dependencies: *mvn clean install.*

4. **Configure CCM Properties and Feature Flag Configuration**

The JAR provides custom configurations for the service applications. The values mentioned in the below block has to be configured in the *CCM* of application which will be passed through the Jar to publish the AUDIT LOGS/Messages.

Feature Flag Configuration - The first configuration block controls whether the audit logging functionality is active:

CCM config - feature flag

```
  featureFlagConfig:
    description: "Feature Flag for NRT"
    resolutionPaths:
      - default: "/envProfile/envName"
    properties:
      isAuditLogEnabled:
        defaultValue: true
        description: "Enable or Disable the Audit Logging Endpoint"
        kind: "SINGLE"
        type: "BOOLEAN"
```

This configuration contains a simple boolean flag (`isAuditLogEnabled`) that acts as a master switch:

- When set to `true`: The Audit Logging service will be active and capture API requests/responses
- When set to `false`: The Audit Logging service will be completely disabled
- The `defaultValue` of `true` means logging is enabled by default unless explicitly turned off

Audit Logging Configuration - The second part of the configuration contains specific settings that control how the audit logging behaves:

CCM config - audit logging

```
auditLoggingConfig:
    description: "Audit log related Configs"
    resolutionPaths:
      - default: "/envProfile/envName"
    properties:
      wmConsumerId:
        defaultValue: "fda7cddb-b0ea-451e-9d2a-b090a08290ae"
        description: "WM_CONSUMER.ID request header value used to access Audit Logs"
        kind: "SINGLE"
        type: "STRING"
      auditLogURI:
        defaultValue: "https://us.stg.proxy-gateway.walmart.com/api-proxy/service/audit/api-logs-srv/v1/logs
/api-requests"
        description: "URI value used to access Audit Logs"
        kind: "SINGLE"
        type: "STRING"
      enabledEndpoints:
        defaultValue: "transactionHistory = ^\\/store\\/\\d+\\/gtin\\/\\d+\\/transactionHistory$,nrt_available
= ^\\/store\\/\\d+\\/gtin\\/\\d+\\/available$,nrt_storeInbounds = ^\\/store\\/\\d+\\/gtin\\/\\d+\\
/storeInbound$,nrt_status = ^\\/store\\/inventory\\/status$,nrt_directshipment = ^\\/store\\/directshipment$,
nrt_dc = ^\\/dc\\/inventory\\/status\\?itemIdentifier=wmItemNbr$,inventoryActions = ^\\/store\\
/inventoryActions$",
        description: "Enable or Disable the endpoints"
        kind: "SINGLE"
        type: "STRING"
      isResponseLoggingEnabled:
        defaultValue: true
        description: "Enable or Disable the response for audit log"
        kind: "SINGLE"
        type: "BOOLEAN"
      keyVersion:
        defaultValue: "1"
        description: "key version"
        kind: "SINGLE"
        type: "STRING"
      auditPrivateKeyPath:
        defaultValue: "/etc/secrets/audit_log_private_key.txt"
        description: "private key for audit log"
        kind: "SINGLE"
        type: "STRING"
      serviceApplication:
        defaultValue: "NRT"
        description: "Service application name for audit logging"
        kind: "SINGLE"
        type: "STRING"
```

Details of the configuration parameter that control the detailed behavior of the audit logging

wmConsumerId: This is the unique identifier for the application that will be using the Audit Logging Service.

auditLogURI: This is the URL of the Audit Log Service. It specifies the network location where the application will send its audit logs. This URL will likely be of API Proxy environment.

enabledEndpoints: This is a crucial configuration that determines which API endpoints will be audited. It's a list of endpoint names and their corresponding regular expressions. The format is a string of key-value pairs separated by equals signs (=), and individual endpoint configurations are separated by commas.

*****Example**: `"nrt_transactionHistory = ^\\/store\\/\\d+\\/gtin\\/\\d+\\/transactionHistory$"`.*

The endpoint named "nrt_transactionHistory" will be audited for any URL that matches the regular expression `^\\/store\\/\\d+\\/gtin\\/\\d+\\/transactionHistory$`.

*****Note**** *: Regular expressions can be tricky. Make sure they accurately match the URLs you want to audit. The ^ and $ characters in the regex indicate the beginning and end of the string, respectively, ensuring that the entire URL matches the pattern. `\\d+` matches one or more digits.*

isResponseLoggingEnabled: A boolean flag (true/false) that controls whether the response body of the API calls should be included in the audit logs. Be cautious about enabling this , as it can significantly increase log volume and potentially expose sensitive data.

keyVersion: The version of the encryption key used by the application. This is likely related to security and key rotation practices.

auditPrivateKeyPath: The file path to the application's private key. This key is used to authenticate the application to the Audit Logging Service. The text correctly advises against storing the private key directly in the application. Instead, it should be stored securely (e.g., in AKeyless) and mounted at runtime.

serviceApplication: The name of the application using the Audit Logging Service. This helps identify the source of the logs.

Summary

To integrate the Audit Logging Service:
1. Add the JAR dependency (choose the appropriate version for your JDK).
2. Configure package scanning in your Spring Boot application.
3. Add the required CCM configurations.
4. Build the project using Maven.

By following these steps, your application will be onboarded to the Audit Logging Service, enabling you to capture and analyze API usage effectively.

*****Happy Logging!*****