Product Metrics QA Sign off - Component Testing

Audit logging service

API Details

- **Endpoint:** `POST` `https://us.stg.proxy-gateway.walmart.com/api-proxy/service/audit/api-logs-srv/v1/logs/api-requests`
- **Functionality:** Saves API request and response logs for auditing purposes.
- **Contract:** https://engineering.dataventures.walmart.com/Data%20Ventures/Platform/Audit%20Logs/AUDIT-API-LOGS/save-api-log [need to be updated]

1. Positive Test Cases (Happy Path)

Test Case	Request Payload	Expected Response	Actual Response	Pass/Fail
Valid API Request	{ "request_id": "70e0b563-f618-43bb-90c5-8be8bf404143", "service_name": "NRT", "endpoint_name": "transactionHistory", "version": "v1", "path": "/store/2000/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" }, "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": {}, "response_code": 200, "request_ts": 1739948540, "response_ts": 1739948548, "request_size_bytes": 2000, "response_size_bytes": 16, "created_ts": 1739948552, "trace_id": "182589fa7f75585af5b05b1ff556f216", "headers": { "wm-site-id": "1694066566785477000", "wm_consumer.id":"test" } }	`204 OK`	`204 OK`	Pass

2. Negative Test Cases (Invalid Inputs)

Test Case	Request Payload	Expected Response	Actual Response	Pass/Fail

Missing Required Fields	Remove `request_id`, `service_name`, `endpoint_name`, `path`, `response_code`	`400 Bad Request`,
		
		{
		"type": "https://uri.walmart.com/errors/invalid-request",
		"title": "Request is not well-formed, syntactically incorrect, or violates schema.",
		"trace_id": "182bfaac5315c75af34b6c8c04c64817",
		"errors": [
		{
		"code": "https://uri.walmart.com/errors/invalid-property-value",
		"reason": "The value of the property is invalid.",
		"property": "/request_id/",
		"location": "body"
		}
		],
		"status": 400,
		"detail": "Invalid request parameters received",
		"instance": "/v1/logs/api-requests"
		}
			`400 Bad Request`	Pass

Invalid Data Types	`"response_code": "abc"` instead of integer	`400 Bad Request`	`400 Bad Request`	Pass
Out-of-Range Values	`"response_code": 600` (invalid HTTP status)	`400 Bad Request`,	`400 Bad Request`	Pass
Invalid Timestamp Order	`"request_ts": 1739948550`, `"response_ts": 1739948548` (response before request)	`400 Bad Request`, validation error	`204`	to be taken as enhancement
Malformed JSON	Missing closing `}` or incorrect commas	`400 Bad Request`, error message indicates malformed JSON	`400 Bad Request`	Pass
Missing Authorization Header	Remove `Authorization` header [missing info]	`400 Bad Request`	`400 Bad Request`	Pass
SQL Injection Attempt	`"service_name":1=1; ---"`	`400 Bad Request`, API should filter inputs	`204`	enhancement

Edge Cases (Boundary Testing)

Test Case	**Request Payload**	**Expected Response**	**Actual Response**	**Pass /Fail**
Minimum Lengths	`"trace_id": "12345678901234567890"` (<25 chars)	`400 Bad Request`	`400 Bad Request`	Pass
Maximum Lengths	**response_code: 600 (>520)**	`400 Bad Request`	`400 Bad Request`	Pass

Empty JSON Body	{}	400 Bad Request,
		
		{
		"type": "https://uri.walmart.com/errors/invalid-request",
		"title": "Request is not well-formed, syntactically incorrect, or violates schema.",
		"trace_id": "182bfbef922e895aa7aa3f4558155062",
		"errors": [
		{
		"code": "https://uri.walmart.com/errors/invalid-property-value",
		"reason": "The value of the property is invalid.",
		"property": "/request_id/",
		"location": "body"
		},
		{
		"code": "https://uri.walmart.com/errors/invalid-property-value",
		"reason": "The value of the property is invalid.",
		"property": "/endpoint_name/",
		"location": "body"
		},
		{
		"code": "https://uri.walmart.com/errors/invalid-property-value",
		"reason": "The value of the property is invalid.",
		"property": "/response_code/",
		"location": "body"
		},
		{
		"code": "https://uri.walmart.com/errors/invalid-property-value",
		"reason": "The value of the property is invalid.",
		"property": "/service_name/",
		"location": "body"
		},
		{
		"code": "https://uri.walmart.com/errors/invalid-property-value",
		"reason": "The value of the property is invalid.",
		"property": "/path/",
		"location": "body"
		}
		],
		"status": 400,
		"detail": "Invalid request parameters received",
		"instance": "/v1/logs/api-requests"
		}
			400 Bad Request	Pass

Different API Versions with exceeded length	"version": "v2", "v3", "v4" instead of "v1" (>=2 ch and <=2 ch)	204	204	Pass
High Frequency Requests	Send **1000 requests per minute**	204	204	Pass
Future Timestamp Values	"request_ts": 2732760750 (August 6, 2056, 04:12:30 UTC) (unrealistic future date)	?	?	enhance ment

Logging Different HTTP Methods	"method": "PUT" or "method": "PATCH" instead of POST
	
	{
	"request_id": "70e0b563-f618-43bb-90c5-8be8bf404143",
	"service_name": "NRT",
	"endpoint_name": "transactionHistory",
	"version": "v1",
	"path": "/store/2000/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"
	},
	"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": {},
	"response_code": 200,
	"request_ts": 1739948540,
	"response_ts": 1739948548,
	"request_size_bytes": 2000,
	"response_size_bytes": 16,
	"created_ts": 1739948552,
	"trace_id": "182589fa7f75585af5b05b1ff556f216",
	"headers": {
	"wm-site-id": "1694066566785477000",
	"wm_consumer.id":"test"
	}
	}
		405	405	Pass

Is Null value in the optional fields allowed		204	204	pass

why different response structure for Empty value {} in any field (opt/mandate)	{ "timestamp": "2025-03-12T07:25:02.317+00:00", "status": 400, "error": "Bad Request", "path": "/v1/logs/api-requests" }	400	400	pass
Testing API Downtime Handling	what response to expect	500	500	pass

GCS Sink Service

The GCS Sink Service is a Kafka consumer, processing data from the Logging Service and persisting valid data in the GCS database. We need to validate data filtering, schema validation, and message consumption behavior

Positive Test Cases (Happy Path)

Test Case	Test Description	Expected Outcome	Actual Outcome	Pass /Fail
Valid Data Flow	Ensure that valid messages (response_code `204`, correct schema, correct `wm-site-id`) are **consumed** and **persisted** in GCS.	Data appears in **GCS database**	Data appears in **GCS database**	pass
Valid Schema Handling	Messages with the correct schema format should be **processed** successfully.	Data is **consumed from Kafka** and **stored in GCS**	Data is **consumed from Kafka** and **stored in GCS**	pass

Negative Test Cases (Invalid Inputs)

Test Case	Test Description	Expected Outcome	Actual Outcome	Pass /Fail
Non-204 Response Code Data Handling	Send logs with **response_code 400**	Data **should NOT appear in Kafka**, so GCS Sink **does not receive it**	Data **should NOT appear in Kafka**,	pass
Incorrect Schema Field Name	Modify request body with an incorrect field name (e.g., `trace_ID` instead of `trace_id`)	Data is **consumed by GCS Sink and persisted in GCS but with trace_id having value as empty**	Data is **consumed by GCS Sink and persisted in GCS but with trace_id having value as empty**	pass
Invalid `wm-site-id` Handling	Send logs with `wm-site-id other than 1694066566785477000 (us-region)`	**Kafka lag increases**, but GCS Sink does **not consume the data**	**Kafka lag increases**, but GCS Sink does **not consume the data**	pass
Adding random fields in the request body	Add some random fields to the request payload, how the data is persisted in GCS?	**These random fields are dropped in Kakfa topic and only fields as per in schema are persisted in GCS**	**These random fields are dropped in Kakfa topic and only fields as per in schema are persisted in GCS**	pass

Edge Cases (Boundary Testing)

Test Case	Test Description	Expected Outcome	Actual Outcome	Pass /Fail
how duplicates req are handled	If api is hit with same payload multiple times, how data persists in GCS?	**only the record with latest timestamp persists in GCS**	**only the record with latest timestamp persists in GCS**	pass
Kafka Consumer Failure Handling	Stop GCS Sink service for some time and restart	GCS Sink **should resume processing without data loss**	GCS Sink **resume processing without data loss**	pass