Monitoring API Implementation Guide (PMM/TLD)

Last updated: February 2024

Table of contents

Introduction

The G2 Risk Solutions (G2RS) Monitoring API supports Persistent Merchant Monitoring (PMM) and Transaction Laundering Detection (TLD) workflows via customer-facing API. This includes receiving PMM/TLD cases, actioning cases, requesting a TLD investigation, reporting back a test transaction decline, and searching the G2RS merchant case history for a monitored merchant. API integration allows customers the flexibility to view and action G2RS cases within their own internal platforms/tools and can increase risk analyst efficiency by reducing the number of systems that users need to log into.

Quick start

The most detailed documentation for the Monitoring (PMM/TLD) API is also available via a Swagger UI. You can visualize and interact with the APIs by navigating to:

https://api-sandbox.g2netview.com/pmmtld/swagger-ui/index.html

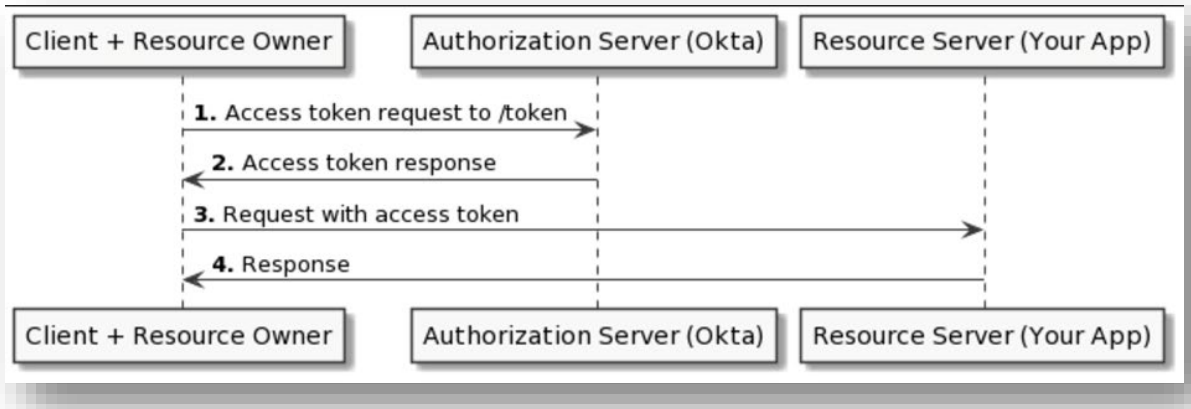
It is highly recommended to integrate and test first with our dynamic sandbox API for all requests your organization plans to utilize before moving to production. This is a precaution to avoid interfering with your production dataset. Please see the section at the end of this guide for more information on G2RS' PMM/TLD sandbox API.

Sandbox URL: https://api-sandbox.g2netview.com/pmmtld/

Production URL: https://api.g2netview.com/pmmtld/

Authentication

The G2RS Monitoring API facilitates machine-to-machine communication via OAuth 2.0 client credentials.



Client + Resource Owner
Authorization Server (Okta)
Resource Server (Your App)
1. Access token request to /token
2. Access token response
3. Request with access token
4. Response
Client + Resource Owner
Authorization Server (Okta)
Resource Server (Your App)

1. During API onboarding, customers will receive a client id and a client secret. This pair of credentials will be used to retrieve a valid jwt (token) via G2RS' Okta instance. Please store these somewhere securely.
2. Client credentials should be encoded in base64 using the following format: <ClientID>:<ClientSecret> (re-place <> with specific credentials)
3. A token can then be retrieved from Okta with the following request:
   a. Sandbox (post): https://sso.g2risksolutions.com/oauth2/aus1owavhv6lOeErx1d8/v1/token
   b. Production (post): https://sso.g2risksolutions.com/oauth2/aus1owaqit51cBAHO1d8/v1/token
   c. Content-Type: application/x-www-form-urlencoded
   d. Authorization: Basic <BASE64_ENCODED> (replace <> with the string obtained in step 2)
   e. Body: grant_type: client_credentials
4. This will return a token, which should be used for all Monitoring API requests and should be provided in the authorization header:
   a. Authorization: Bearer <jwt> (replace <> with the string obtained in step 3)
5. This token will be valid for one hour. We recommend implementing a caching mechanism to avoid requesting a new token along with every API call.

Sample using spring boot security with web client

```
@Configuration
public class OAuthClientConfiguration {
@Bean
ReactiveClientRegistrationRepository clientRegistrations(
@Value("${spring.security.oauth2.client.provider.okta.token-uri}") String token_uri,
@Value("${spring.security.oauth2.client.registration.okta.client-id}") String client_id,
@Value("${spring.security.oauth2.client.registration.okta.client-secret}") String
client_secret, @Value("${spring.security.oauth2.client.registration.okta.scope}") String
scope, @Value("${spring.security.oauth2.client.registration.okta.authorization-grant-
type}") String authoriza-
tionGrantType
) {
ClientRegistration registration = ClientRegistration
.withRegistrationId("okta")
.tokenUri(token_uri)
.clientId(client_id)
.clientSecret(client_secret)
.scope(scope)
.authorizationGrantType(new AuthorizationGrantType(authorizationGrantType))
.build();
return new InMemoryReactiveClientRegistrationRepository(registration);
}
@Bean
WebClient webClient(ReactiveClientRegistrationRepository clientRegistrations) {
InMemoryReactiveOAuth2AuthorizedClientService clientService = new
InMemoryReactiveOAuth2Autho- rizedClientService(clientRegistrations);
AuthorizedClientServiceReactiveOAuth2AuthorizedClientManager
authorizedClientManager = new Autho-
rizedClientServiceReactiveOAuth2AuthorizedClientManager(clientRegistrations,
clientService);
ServerOAuth2AuthorizedClientExchangeFilterFunction oauth = new
ServerOAuth2AuthorizedClientEx- changeFilterFunction(authorizedClientManager);
oauth.setDefaultClientRegistrationId("okta");
return WebClient.builder()
.filter(oauth)
.build();
}
}
```

```
@Autowired
private WebClient webClient;
@Scheduled(fixedRate = 5000) public void scheduledRequest() {
```

```
webClient.get()
.uri(“http://localhost:8081”)
.retrieve()
.bodyToMono(String.class)
.map(string
-> “Schedule request response: “ + string)
.subscribe(logger::info);
```

Request types summary

The G2RS Monitoring (PMM/TLD) API is based on REST-principles and requests are sent in JSON via standard HTTPS in UTF-8 format. The API also uses appropriate verbs for each action.

There are several calls that are used to interact with G2RS’ PMM and TLD products. Additionally, more detailed documentation including required parameters and parameter descriptions can be found on Swagger. The high-level descriptions of each are as follows:

Product	Method	Action
PMM	GET url-cases	Retrieve list of published PMM cases within provided filters
PMM	GET url-cases/{caseId}	Retrieve details of a specific PMM case
TLD	GET lead-cases	Retrieve a list of current TLD leads within provided filters
TLD	GET lead-cases/{caseId}	Retrieve the details of a specific TLD lead
PMM, TLD	GET action-reasons	Retrieves action and reasons catalog for PMM (CONTENT_COMPLIANCE), TLD (TRANSACTION_LAUNDERING), or both (ALL)
PMM, TLD	PUT action-cases/{caseId}	Apply a valid action and reason from the action-reason catalog for a PMM case (URLCase) or TLD lead
PMM, TLD	GET case-comments	Retrieve case comments for the provided caseId
PMM, TLD	POST case-comments	Create a new comment related to the provided caseId

PMM, TLD	PUT case-comments	Update an existing comment related to the provided caseId
PMM, TLD	DELETE case-comments	Remove an existing comment related to the provided caseId
PMM, TLD	GET case-screenshots	Retrieve screenshots, screenshot URL, and screenshot caption (if applicable) for the provided caseId. Screenshots are provided in encoded base64 format
TLD	GET credit-cards	Retrieve the current TLD card list
TLD	GET trace-cases	Retrieve list of published TLD traces within provided filters
TLD	GET trace-cases/{caseId}	Retrieve details of a specific TLD trace
TLD	POST trace-cases/{traceCaseId}/trace-hit	Report TLD test transaction declines and associated merchant details to G2
TLD	GET client-requested-investigation-cases	Retrieve a list of current TLD investigations (client-requested investigations) within provided filters
TLD	GET client-requested-investigation-cases/{caseId}	Retrieve the details of a specific TLD investigation
PMM, TL	POST client-requested-investigation-cases	Request a new TLD investigation (client-requested investigation)
PMM, TL	GET merchants-search	Search for a merchant within your G2RS merchant dataset
PMM, TL	GET merchant-profile	View merchant attributes provided to G2RS
PMM, TL	GET merchant-profile/history	Fetch PMM and TLD case history for merchant

G2RS recommends API users to post PMM and TLD actions back via the API as soon as cases are actioned within your system to ensure accurate data is provided to MasterCard MMP (if your organization is configured for MMP monthly reporting) and avoid data discrepancies in your merchant portfolio.

Status codes—general (see swagger for specific message examples)

Code	Description	Body
200	OK	{}
204	No Content	{}
201	Created	{}
400	Bad Request	{“timestamp”: “” “path”: ““, “status”: 400, “error”: “Bad Request”, “message”: “”, “requestId”: “” }
401	Unauthorized	{“timestamp”: “” “path”: ““, “status”: 401, “error”: “Unauthorized”, “message”: “”, “requestId”: “” }
403	Forbidden	{“timestamp”: “” “path”: ““, “status”: 403, “error”: “Forbidden”, “message”: “”, “requestId”: “” }
404	Not found	{“timestamp”: “” “path”: ““, “status”: 404, “error”: “Not Found”, “message”: “”, “requestId”: “” }
500	Internal Server Error	{“timestamp”: “” “path”: ““, “status”: 500, “error”: “Internal Server Error”, “message”: “”, “requestId”: “” }

Commonly grouped requests

Integrating some PMM/TLD workflow actions via API requires the grouping of certain requests. The two most common use cases for this are viewing the full details of a case (including screenshots/comments) and supplying merchant information while reporting a test decline, requesting a TLD investigation, or looking up a merchant's G2RS case history.

Case study

For performance purposes, some PMM/TLD case data components are split into multiple endpoints including key details, comments, and screenshots. In order to retrieve all relevant data for a single case, the same caseId should be used to invoke each of these endpoints. For example, to retrieve all details for a PMM URLCase, a request should be made with the target caseId to:

- GET url-cases/{caseId}
- GET case-comments
- GET case-screenshots

Supplying merchant information

Certain endpoints require the ids of merchants and associated data within a customer's G2RS monitoring dataset. This involves identifying the intended merchant via merchants-search and then providing the resulting ids to the relevant endpoint. Examples include:

- Reporting a test transaction decline/trace hit (POST trace-cases/{traceCaseId}/trace-hit)
  1. Search for cardNumber via GET trace-cases to identify the associated Trace caseId
  2. Look up merchant associated with the decline via GET merchants-search
  3. Send merchantId, businessId, and urlId returned via merchants-search to POST trace-cases/{traceCaseId}/trace-hit
- Requesting a TLD investigation (POST client-requested-investigation-cases)
  1. Look up merchant in monitoring via GET merchants-search
  2. Send merchantId, businessId, and urlId returned via merchants-search to POST client-requested-investigation-cases
- Retrieving a monitored merchant's information or G2RS case history (GET merchant-profile, GET merchant-profile/history, or GET merchant-profile/stats)
  1. Look up merchant in monitoring via GET merchants-search
  2. Send merchantId, businessId, and urlId returned via merchants-search to merchant-profile endpoints

PMM/TLD notifications webhook

Monitoring API customers can sign up to receive PMM and TLD notifications via webhook, which include the following events:

- PMM report published
- TLD lead published or updated
- TLD investigation published or updated

To integrate with the notifications webhook, customers need to provide a public URL/endpoint where G2RS will send the notification. G2RS will support token-based security with each notification message posted to this endpoint.

Configuring the webhook can be completed by sending a POST request to the ‘subscriptions’ endpoint (see Swagger documentation for more details). The request body should include the public URL/endpoint where G2RS will send notifications, the email address of a technical contact in the case that delivery attempts are unsuccessful, and a token or username/password that G2RS will send with the notification for security (optional).

Examples of webhook notification messages include:

PMM report published

```
{
   "type": " REPORT_CASE_PUBLISHED ",
   "entityId": 123456,
   "date": "2023-08-04T18:35:35.312+00:00",
   "body": " The Monitoring Report: Demo Client – 8/31/2023, ID: 123456 has been
             published."
}
```

TL lead published

```
{
  "type": “LEAD_CASE_STATUS_UPDATED",
  "entityId": 731309406,
  "date": "2023-08-04T18:35:35.312+00:00",
  "body": “A TL Lead, ID: 731309406 has been published for your organization.”
}
```

In the event that a notification is undeliverable after three attempts, G2RS will auto-generate an email to an address specified by the customer (required) to notify API customers of the notification failure.

Notifications can also be fetched with a GET notifications request if customers choose not to integrate with the webhook method or if notifications are undeliverable to the customer’s endpoint. G2RS’ recommended polling cadence, if integrated with the GET notifications method, is every 5 minutes if your organization has TLD, as leads and TLD investigation results are published in real-time following G2RS analyst review. If your organization has PMM only (no TLD), the recommended polling cadence is every hour.

**Please note: webhook and notification lookup functionality are not currently supported in the API sandbox.*

Performance

Rate limit	100 transactions per minute. Exceeding this limit will not result in request failure, but may cause delays in result delivery.
Response time	All API calls should return a response within 3 seconds. Queries that involve a large amount of merchant data (e.g., some scenarios of merchant search, retrieving a large range of URLCases) may result in a delayed response time.
Downtime	Service availability is expected to remain above 99.5%. Planned maintenance downtime will be communicated to clients beforehand.

Sandbox environment

The Monitoring API sandbox environment closely mirrors production and allows customers to test their PMM/TLD API integration. Customers must authenticate and will have access to a mock data set that may be manipulated with any Monitoring API endpoint.

Features:

- Authentication
- Parameter validation
- Isolated mock data set
- Replicate production functionality, including applying actions, requesting a TLD investigation, reporting a test transaction decline, etc.
- Response and status code validation

**Please note: webhook and notification lookup functionality are not currently supported in the API sandbox.*

Contact us

Email: clientservices@g2risksolutions.com