833-653-6618 | www.g2risksolutions.com

Merchant Data Management (MDMX) API Implementation Guide

Last updated: October 2024

Table of contents

Introduction

The G2 Risk Solutions (G2RS) Merchant Data Management (MDMX) API provides customers the ability to manage their G2RS merchant monitoring portfolio via API. This includes submitting merchants to G2RS for monitoring, updating merchant attributes, and removing merchants from monitoring. API integration allows customers the flexibility to integrate the process of providing and updating merchants in G2RS' system within their own internal platforms/tools and can increase risk analyst efficiency by eliminating the portal-based file upload process.

Quick start

The most detailed documentation for the MDMX API is available via a Swagger UI. You can visualize and interact with the APIs by navigating to:

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

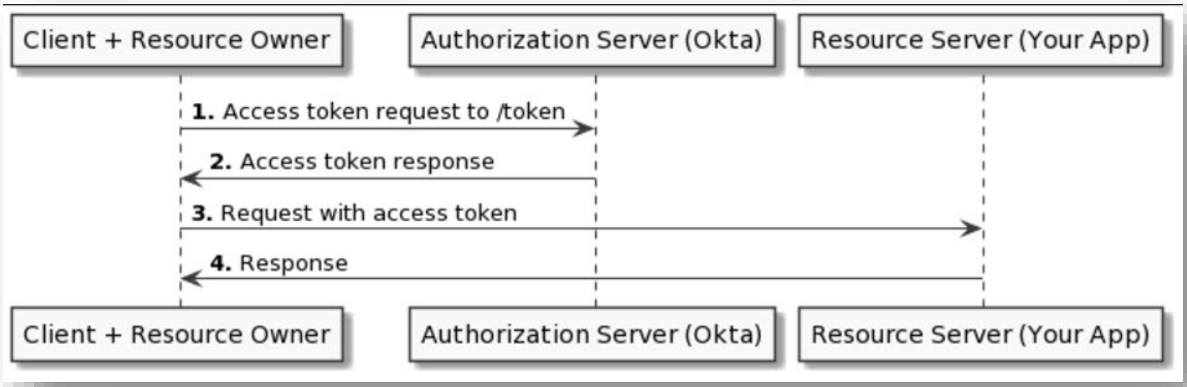
Access to API endpoints is via access tokens only. In addition to a production environment, G2RS also provides a sandbox environment for testing your integration. Both environments use separate access endpoints and tokens.

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

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

Authentication

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



1. Customers will receive a client id and a client secret during API onboarding. 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> (replace <> with specific credentials)
3. A token can then be retrieved from Okta with the following request:
   Sandbox (post): https://sso.g2risksolutions.com/oauth2/aus1owavhv6lOeErx1d8/v1/token
   Production (post): https://sso.g2risksolutions.com/oauth2/aus1owaqit51cBAHO1d8/v1/token
   Content-Type: application/x-www-form-urlencoded
   Authorization: Basic <BASE64_ENCODED> (replace <> with the string obtained in step 2)
   Body: grant_type: client_credentials
4. This will return a token, which should be used for all MDMX API requests and should be provided in the authorization header:
   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
authorizationGrantType
    ) {
        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
InMemoryReactiveOAuth2AuthorizedClientService(clientRegistrations);
        AuthorizedClientServiceReactiveOAuth2AuthorizedClientManager authorizedClientManager =
new AuthorizedClientServiceReactiveOAuth2AuthorizedClientManager(clientRegistrations,
clientService);
        ServerOAuth2AuthorizedClientExchangeFilterFunction oauth = new
ServerOAuth2AuthorizedClientExchangeFilterFunction(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 MDMX 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 endpoints that are used to interact with G2RS' MDMX API. Additional documentation including required parameters and parameter descriptions can be found on

Swagger via the link in the “Quick Start” section of this document. The high-level descriptions of each endpoint are as follows:

Method	Action
POST merchants	Add a new merchant to G2RS monitoring (single request)
POST merchants/bulk	Submit a file of merchants to G2RS monitoring (bulk request)
PUT merchants	Update attributes of an existing merchant in G2RS’ system
GET merchant-search	Search for a merchant within your G2RS merchant dataset
GET merchant-profile	Retrieve merchant details provided to G2RS
GET merchant-groups	Retrieves list of active merchant groups (projects)
GET merchant-groups/{merchantGroupId}/merchants	Retrieves all of the merchants within a specific merchant group

Data dictionary

Field	Definition
merchantGroupId	ID that corresponds to a specific merchant group (collection of merchants within your portfolio).
urlString	Merchant’s URL
mid	Merchant ID associated with your merchant
merchantName	Merchant’s legal name
billingDescriptor	Merchant’s billing descriptor
merchantDBA	Merchant’s DBA
mcc	Merchant’s MCC
mastercardICA	Merchant’s MasterCard ICA
visaBIN	Merchant’s Visa BIN
phoneNumber	Merchant’s phone number
emailAddress	Merchant’s email address
streetAddress1, streetAddress2, city, regionState, country, postalZip	Merchant’s address data
principalName	Merchant’s principal name
additionalPrincipalNames	Array of additional principal names for the merchant if >1

dataUpdateAction	For updating merchant principal names (if multiple), dataUpdateAction must be provided as “ADD” or “REPLACE” to indicate whether new principal name values should be added to or replace existing principal name values
principalPhone	Principal’s phone number (first principal provided)
principalEmail	Principal’s email address (first principal provided)
principalAddress1, principalAddress2, principalCity, principalRegionState, principalCountry, principalZip	Principal’s address data (first principal provided)
primaryMerchantContactName	Primary merchant contact name
data1, data2, data3	Custom data fields to capture additional data as desired by client

Status codes: general (see A 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": "" }

G2RS merchant definition

G2RS defines a merchant with a combination of attributes to accommodate various customer merchant data models. The G2RS data model can be summarized with the following rules:

- A unique merchant in G2RS’ data model is defined as the combination of merchantId and urlId. A merchantId may be associated with multiple urlIds, but these are treated as separate “merchant records” for the purposes of monitoring.
    - The merchantId described in this document refers to a unique G2RS identifier. This is not equivalent to the merchant MID. In MDMX API responses, the alpha-numeric identifier assigned by customers is returned as the field: “mid”.
    - A merchantId is generated even for merchants without a “mid” provided by customers. In this case, the “mid” assigned by G2RS is a system-generated value and will be flagged as such in the MDMX API endpoint responses that include merchant information.
- If a merchant (with the same MID) is in more than one merchant group/project, a separate merchantId record will be created for each merchant group. However, they will all have the same “mid” value.
- A merchant group (project) may only contain one “enabled” (actively in monitoring) record for a given URL (urlId)

- Endpoints that require G2RS ID values for a merchant (merchantId + urlId) as parameters include:
    - PUT merchants
    - GET merchant-profile

These ID values are returned in the following endpoint responses:

- POST/PUT merchants
- GET merchant-search
- GET merchant-profile

Common request/response samples

This section provides sample requests/responses from the MDMX API for the most common customer use cases, including creating/updating merchants in G2RS' system and removing merchants from monitoring.

Creating a merchant—bulk request

Post: https://api-sandbox.g2netview.com/mdmx/merchants/bulk

Request body

- Consumes multi-part form data:
    - bulkMerchantRequest (key name): A JSON object containing merchantGroupId and ingestType.
        - Example: {"merchantGroupId": 65, "ingestType": "ADD"}
        - IngestType values:
            - "ADD" - merchants in file will be **added** to the merchant group's current set of merchants in monitoring.
            - "REMOVE" - merchants in file will **replace** the merchant group's current set of merchants in monitoring.

- file (key name): The CSV file containing merchant data. File must follow standard PMM/TLD client upload template format, which can be downloaded from the G2RS client portal.

- Please note:
  - The CSV file may not exceed 20,000 rows.
  - The system prevents the submission of multiple files for the same merchant group if an ingestion process is already underway. Please use the /merchants/bulk/{trackingId} API described in Swagger to check file processing status and submit the next (if multiple), when a success message is received.

Creating a merchant—single request

Post: https://api-sandbox.g2netview.com/mdmx/merchants

Request body


```
{
  "merchantName": "New Merchant",

  "urlString": "www.abc123456defg.com",
  "mid": "1111111",
  "merchantGroupId": 501,
  "billingDescriptor": "BD123",
  "merchantDBA": "Merchant New",
  "mcc": "5532",
  "mastercardICA": "1234",
  "visaBIN": "5678",
  "streetAddress1": "1122 W Craig Pl",
```

```
  "streetAddress2": "Apt 12",
  "city": "Dallas",
  "regionState": "TX",
  "country": "USA",
  "postalZip": "55120",
  "phoneNumber": "256-111-1234",
  "emailAddress": "test@email.com",
  "principalName": "John Smith",
  "principalAddress1": "99913 E Craig",
  "principalCity": "Dallas",
  "principalRegionState": "TX",
  "principalCountry": "US",
  "principalZip": "55123",
  "principalPhone": "215-123-2345",
  "principalEmail": "principal@gmail.com",

  "dataField1": "Data 1",
  "dataField2": "Data 2",
  "dataField3": "Data 3",
  "primaryMerchantContactName": "Jane Smith"
}
```


Response body


```
{
```

"msjDescription": "New merchant created with the following information. ",

"merchantCreated": true,

"inMonitoring": true,

"merchantGroupId": 501,

"merchantId": 63861440,

"mid": "1111111",

"systemGeneratedMid": false,

"urlString": "www.abc123456defg.com",

"urlId": 643504639,

"businessId": 59754601,

"merchantName": "New Merchant",

"merchantDBA": "Merchant New",

"billingDescriptor": "BD123",

"mcc": "5532",

"mastercardICA": "1234",

"visaBIN": "5678",

"merchantAddress": {

"streetAddress1": "1122 W Craig Pl",

"streetAddress2": "Apt 12",

"city": "Dallas",

"regionState": "Tx",

"country": "Usa",

"postalZip": "55120"

},

```
  "phoneNumber": "256-111-1234",
  "emailAddress": "test@email.com",
  "principalName": "John Smith",
  "principalAddress": {
    "streetAddress1": "99913 E Craig",
    "streetAddress2": "",
    "city": "Dallas",
    "regionState": "Tx",
    "country": "Us",
    "postalZip": "55123"
  },
  "principalPhone": "215-123-2345",
  "principalEmail": "principal@gmail.com",
  "primaryMerchantContactName": "Jane Smith",
  "dataField1": "Data 1",

  "dataField2": "Data 2",
  "dataField3": "Data 3"
}
```


Updating a merchant

Put: https://api-sandbox.g2netview.com/mdmx/merchants

Request body (using same merchant that was created above)


```
{
 "merchantId": "63861440",
 "urlId": "643504639",
 "principalName":"New Principal Name"
}
```


Response body


```
{
   "msjDescription": "Merchant update processed. Current values for the merchant are”,
   "inMonitoring": true,
   "merchantGroupId": 501,
   "merchantId": 63861440,
   "mid": "1111111",
   "systemGeneratedMid": false,

   "urlId": "643504639",
   "urlString": "www.abc123456defg.com",
   "businessId": "59754601",
   "merchantName": "New Merchant",
   "merchantDBA": "Merchant New",
   "billingDescriptor": "BD123",
   "mcc": "5532",
```

"mastercardICA": "1234",

"visaBIN": "5678",

"merchantAddress": {

"streetAddress1": "1122 W Craig Pl",

"streetAddress2": "Apt 12",

"city": "Dallas",

"regionState": "Tx",

"country": "Usa",

"postalZip": "55120"

},

"principalAddress": {

"streetAddress1": "99913 E Craig",

"streetAddress2": "",

"city": "Dallas",

"regionState": "Tx",

"country": "Us",

"postalZip": "55123"

},

"phoneNumber": "256-111-1234",

"emailAddress": "test@email.com",

"principalName": "New Principal Name",

"principalPhone": "215-123-2345",

"principalEmail": "principal@gmail.com",

"primaryMerchantContactName": "Jane Smith",

```
  "dataField1": "Data 1",
  "dataField2": "Data 2",
  "dataField3": "Data 3"
}
```


Removing a merchant from G2RS monitoring

Put: https://api-sandbox.g2netview.com/mdmx/merchants

Request body (using the same merchant that was created above)


```
{
 "merchantId": "63861440",
 "urlId": "643504639",
 "inMonitoring": false
}
```


Response body


```
{
  "msjDescription": "Merchant update processed. Current values for the merchant are",
  "inMonitoring": false,
  "merchantGroupId": 501,
  "merchantId": 63861440,
  "mid": "1111111",
  "systemGeneratedMid": false,
```

```
"urlId": "643504639",
"urlString": "www.abc123456defg.com",
"businessId": "59754601",
"merchantName": "New Merchant",
"merchantDBA": "Merchant New",
"billingDescriptor": "BD123",
"mcc": "5532",
"mastercardICA": "1234",
"visaBIN": "5678",
"merchantAddress": {
    "streetAddress1": "1122 W Craig Pl",
    "streetAddress2": "Apt 12",
    "city": "Dallas",
    "regionState": "Tx",
    "country": "Usa",

    "postalZip": "55120"
},
"principalAddress": {
    "streetAddress1": "99913 E Craig",
    "streetAddress2": "",
    "city": "Dallas",
    "regionState": "Tx",
    "country": "Us",
    "postalZip": "55123"
```

```
  },
  "phoneNumber": "256-111-1234",
  "emailAddress": "test@email.com",
  "principalName": "New Principal Name",
  "principalPhone": "215-123-2345",
  "principalEmail": "principal@gmail.com",
  "primaryMerchantContactName": "Jane Smith",
  "dataField1": "Data 1",
  "dataField2": "Data 2",
  "dataField3": "Data 3"
}
```


Performance

Rate limit	5000 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.
Downtime	Service availability is expected to remain above 99.5%. Planned maintenance downtime will be communicated to clients beforehand.

Sandbox environment

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

Features:

- Authentication

- Parameter validation
- Isolated mock data set, which can also be used to test CC/TL API integration
- Replicate production functionality, including submitting a new merchant to G2RS, updating merchant attributes, and removing a merchant from monitoring
- Response and status code validation

Contact us

Email: clientservices@g2risksolutions.com