Scoreboard Numbers API - Integration Guide

The Scoreboard Numbers API is developed by Align Technologies Corp.

Contents

Overview

The Scoreboard Numbers API provides secure, scalable endpoints for managing company metrics data. Built on AWS infrastructure with multi-layer security, the API enables clients to retrieve, create, and update metrics with comprehensive validation and monitoring.

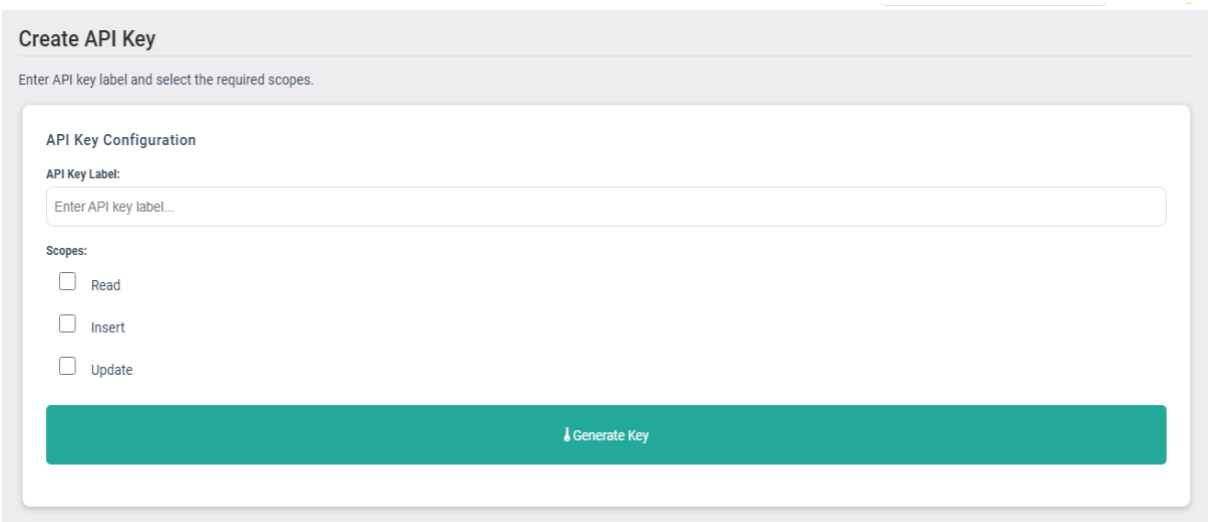
Admin Section

As an Admin User in the software, you have access to the API Credentials.

Getting Started

Generate API Credentials

1. **Login to the application**
    - Log in
    - Navigate to **Administrator → Manage API Keys → Create API Key**



2. **Create API Key**
    - Enter a descriptive label for your key
    - Select required scopes:
        - Read: GET operations
        - Insert: POST operations
        - Update: PUT operations
    - Click **"Generate New API Key."**

3. **Download Credentials**
    - Click **"Generate Key"** to create your credentials
    - Download the JSON file containing:


```
{
 "label": "key label 14 7 1",
 "access_token": "eyJraWQiOiJ4aHd0Q2c0VFgwK0daUzJ....",
 "refresh_token": "eyJjdHkiOiJKV1QiLCJlbmMiOiJBMjU2R0NNIiwiYW...",
 "api_key": "ak_a1dCS2UZUkdSRn1UBW50XVNaX1..",
 "scopes": "read, insert, update",
 "api_key_expiry": "7/14/2026 9:24:39 AM",
 "generated_at": "7/14/2025 9:24:41 AM"
}
```


⚠ **Security Warning**: Store this file securely and never commit it to version control.

Access Token - This is the OAuth authentication token that needs to be sent in the http header. Refer below for samples in setting this parameter

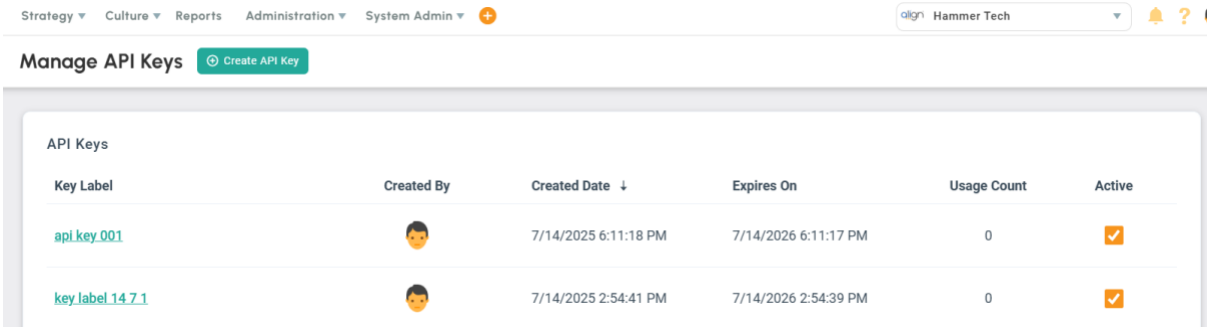
Refresh Token - This token should be used to get the new token when the current access token has expired, refer to the "Generate new access token" section related to this.

API Key - This key needs to be sent in the http header. Refer below for samples in setting this parameter

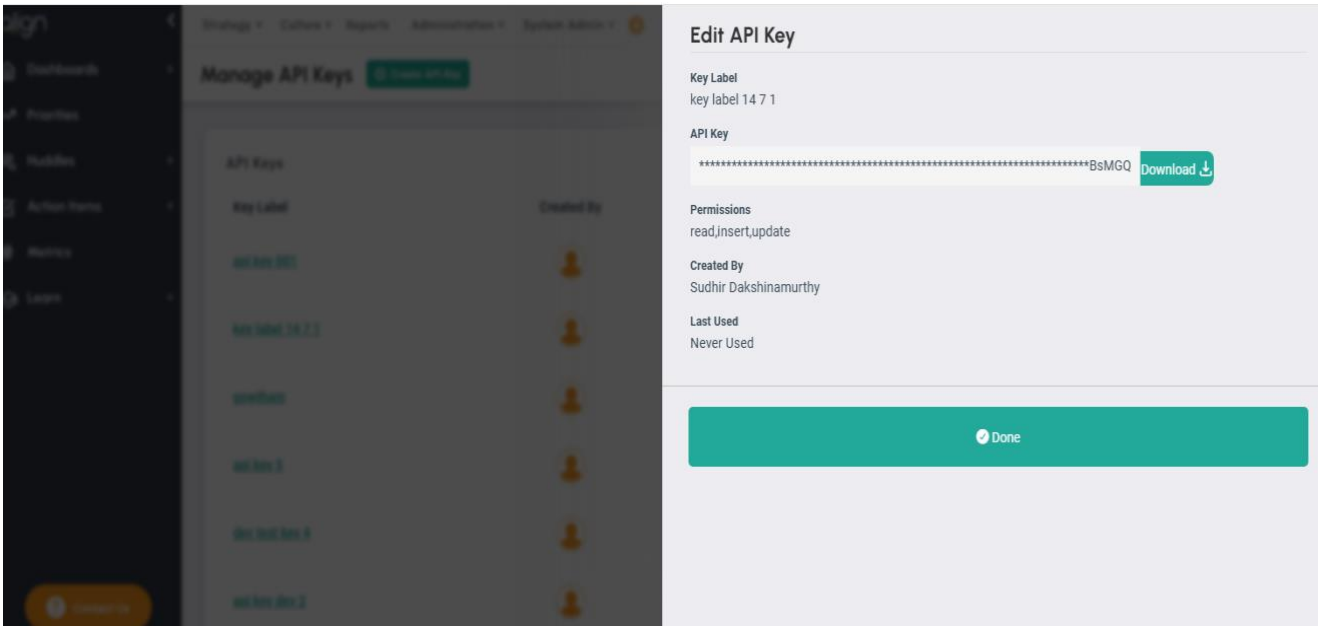
Scopes - Refer to the operation that is allowed for this key

API Expiry Date - Expiry date of the API key

4. After Generating key you see the Manage API Keys page where you will see and manage all the key generated



- You can Active or Inactive key by clicking on the check box
- Usage Count gives you the how many times the key is used
- If the key is lost, you can retrieve it by clicking on the key label and selecting 'Download'. you will get the JSON file



Note:

The downloaded JSON file needs to be shared with your development team to invoke Scoreboard API.

Developer Section

Scoreboard API - Base URL

- **BaseURL**: https://api.aligntoday.com

Step 2: Test Your Connection

```
curl -i  -X GET \
  "{BaseURL}/api/v1/metrics" \
  -H "Authorization: Bearer {your_access_token}" \
  -H "X-API-Key: {your_api_key}"
```

Note :

Access tokens expire every 24 hours. When you receive a 401 Unauthorized error, generate a new access token. Refer to Generate new access token using refresh token

Authentication

Required Headers

All API requests must include both in the http headers:

Authorization: Bearer {access_token}

X-API-Key: {api_key}

Example:

```
GET \
  "{BaseURL}/api/v1/metrics" \
  -H "Authorization: Bearer {your_access_token}" \
  -H "X-API-Key: {your_api_key}"
```

API Endpoints Reference

1. Get All Metrics

Retrieve a list of metrics with optional filtering.

Endpoint: GET /api/v1/metrics

Query Parameters:

Parameter	Type	Required	Description	Example
fromDate	string	No	Filter metrics from date (yyyy-MM-dd)	2024-01-01
toDate	string	No	Filter metrics to date (yyyy-MM-dd)	2024-12-31
owner	string	No	Filter by owner name (partial match)	John Doe

Example Request:

```
curl -X GET \
  "{BaseURL}/api/v1/metrics?fromDate=2024-01-01&toDate=2024-12-31" \
  -H "Authorization: Bearer {your_access_token}" \
  -H "X-API-Key: {your_api_key}"
```

Success Response (200 OK):

```
{
  "Success": true,
  "Message": "Metrics retrieved successfully",
  "Data": [
    {
      "Name": "Revenue",
      "Value": "1000000",
      "Description": "Monthly revenue",
      "Comments": "Q1 target met",
      "Owner": "John Doe",
      "Status": "Active",
```

```
    "LastUpdatedUTC": "2024-01-15T10:30:00Z",
    "UniqueID": "REV-2024-001"
   }
 ]
}
```


Validation Errors (400 Bad Request):


```
{
 "Success": false,
 "Errors": [
  {
   "Field": "FromDate",
   "Issue": "Invalid format. Please use yyyy-MM-dd format (e.g., 2024-01-15)"
  },
  {
   "Field": "FromDate",
   "Issue": "Cannot be later than toDate."
  }
 ]
}
```


2. Get Single Metric

Retrieve a specific metric by its unique ID.

Endpoint: GET /api/v1/metrics/{id}

Note: {id} should be URL encoded.
Example: actual: metric/23-from-api-2, encoded: metric%2F23-from-api-2

Path Parameters:

Parameter	Type	Required	Description
id	string	Yes	Unique ID of the metric

Example Request:


```
curl -X GET \
 "{BaseURL}/api/v1/metrics/REV-2024-001" \
 -H "Authorization: Bearer {your_access_token}" \
 -H "X-API-Key: {your_api_key}"
```

Success Response (200 OK):


```
{
  "Success": true,
  "Message": "Metric retrieved successfully",
  "Data": {
    "Name": "Revenue",
    "Value": "1000000",
    "Description": "Monthly revenue",
    "Comments": "Q1 target met",
    "Owner": "John Doe",
    "Status": "Active",
    "LastUpdatedUTC": "2024-01-15T10:30:00Z",
    "UniqueID": "REV-2024-001"
  }
}
```


Error Response (404 Not Found):


```
{
  "Success": false,
  "Error": "Metric not found"
}
```


3. Create Metrics

Create one or multiple metrics in a single request.

Endpoint: POST /api/v1/metrics

Request Body:


```
[
  {
    "Name": "Revenue",
    "Value": "1000000",
    "Description": "Monthly revenue",
    "Owner": "John Doe",
    "Status": "Active"
  }
```

]

Field Specifications:

Field	Type	Required	Validation Rules	Default
Name	string	**Yes**	Cannot be empty	-
Value	string	No	Any string value	"0"
Description	string	No	Any string	-
Owner	string	No	Must match existing person (case-insensitive)	Current user
Status	string	No	Active, Inactive, Draft, Deprecated	"Active"

Example Request:

```
curl -X POST \
  "{BaseURL}/api/v1/metrics" \
  -H "Authorization: Bearer {your_access_token}" \
  -H "X-API-Key: {your_api_key}" \
  -H "Content-Type: application/json" \
  -d '[
    {
      "Name": "Revenue",
      "Value": "1000000",
      "Description": "Monthly revenue",
      "Owner": "John Doe",
      "Status": "Active"
    },
    {
      "Name": "Customer Count",
      "Value": "500",
      "Description": "Total active customers"
    }
  ]'
```

Complete Success (201 Created):

```
{
  "Success": true,
```

```
  "Message": "All metrics created successfully",
  "Data": [
    {
      "Name": "Revenue",
      "Status": "CREATED",
      "UniqueID": "REV-2024-001"
    },
    {
      "Name": "Customer Count",
      "Status": "CREATED",
      "UniqueID": "CUST-2024-001"
    }
  ]
}
```


Partial Success (207 Multi-Status):


```
{
  "Success": true,
  "Message": "1 of 2 metrics created successfully",
  "Data": [
    {
      "Name": "Revenue",
      "Status": "CREATED",
      "UniqueID": "REV-2024-001"
    },
    {
      "Name": "",
      "Status": "ERROR",
      "ErrorMessage": "Metric Name is required"
    }
  ]
}
```


Complete Failure (422 Unprocessable Entity):

```
{
 "Success": true,
 "Message": "No metrics could be created",
 "Data": [
  {
   "Name": "",
   "Status": "ERROR",
   "ErrorMessage": "Metric Name is required"
  }
 ]
}
```


4. Update Metrics

Update one or multiple existing metrics.

Endpoint: PUT /api/v1/metrics

Request Body:


```
[
 {
  "UniqueID": "REV-2024-001",
  "Name": "Updated Revenue",
  "Value": "1200000",
  "Description": "Updated monthly revenue",
  "Owner": "Jane Smith",
  "Status": "Active"
 }
]
```


Field Specifications:

Field	Type	Required	Validation Rules
UniqueID	string	**Yes**	Must exist in database
Name	string	No	Cannot be empty if provided
Value	string	No	Any string value
Description	string	No	Any string

Owner	string	No	Must match existing person
Status	string	No	Valid status values

Example Request:

```
curl -X PUT \
  "{BaseURL}/api/v1/metrics" \
  -H "Authorization: Bearer {your_access_token}" \
  -H "X-API-Key: {your_api_key}" \
  -H "Content-Type: application/json" \
  -d '[
    {
      "UniqueID": "REV-2024-001",
      "Name": "Updated Revenue",
      "Value": "1200000",
      "Description": "Updated monthly revenue"
    }
  ]'
```

Success Response (201 Created):

```
{
  "Success": true,
  "Message": "All metrics updated successfully",
  "Data": [
    {
      "UniqueID": "REV-2024-001",
      "Status": "UPDATED"
    }
  ]
}
```

5. Add Historical Values

Add historical time-series data for an existing metric.

Endpoint: POST /api/v1/metrics/pastValues

Request Body:

```
{
  "UniqueID": "REV-2024-001",
  "Values": [
    {
      "Date": "2024-01-01",
      "Value": "950000"
    },
    {
      "Date": "2024-01-02",
      "Value": "980000"
    }
  ]
}
```

Field Specifications:

Field	Type	Required	Validation Rules
UniqueID	string	**Yes**	Must exist in database
Values	array	**Yes**	Cannot be empty
Values[].Date	string	**Yes**	yyyy-MM-dd format, cannot be a future date
Values[].Value	string	**Yes**	Cannot be null or empty

Example Request:

```
curl -X POST \
  "{BaseURL}/api/v1/metrics/pastValues" \
  -H "Authorization: Bearer {your_access_token}" \
  -H "X-API-Key: {your_api_key}" \
  -H "Content-Type: application/json" \
  -d '{
    "UniqueID": "REV-2024-001",
    "Values": [
      {
        "Date": "2024-01-01",
```

```
      "Value": "950000"
    },
    {
      "Date": "2024-01-02",
      "Value": "980000"
    }
  ]
}'
```

Success Response (201 Created):

```
{
  "Success": true,
  "Message": "All values updated successfully",
  "Data": [
    {
      "Date": "2024-01-01",
      "Value": "950000",
      "Status": "CREATED"
    },
    {
      "Date": "2024-01-02",
      "Value": "980000",
      "Status": "CREATED"
    }
  ]
}
```

Partial Success (207 Multi-Status):

```
{
  "Success": true,
  "Message": "1 of 2 values updated successfully",
  "Data": [
    {
      "Date": "2024-01-01",
```

```
    "Value": "950000",
    "Status": "CREATED"
   },
   {
    "Date": "2025-12-31",
    "Value": "1000000",
    "Status": "ERROR",
    "ErrorMessage": "Cannot set value for future date"
   }
  ]
}
```

Error Handling

HTTP Status Codes

Status Code	Status Name	Category	Description	When It Occurs
200	OK	Success	GET requests completed successfully	Successful retrieval of metrics
201	Created	Success	All CREATE/UPDATE operations successful	All metrics/values created/updated successfully
207	Multi-Status	Partial Success	Partial success in bulk operations	Some items succeeded, some failed
400	Bad Request	Client Error	Pre-API validation failures	Missing required fields, invalid formats
401	Unauthorized	Auth Error	Missing/invalid API key or token	Authentication failures
403	Forbidden	Auth Error	HTTP method not permitted	Insufficient permissions
404	Not Found	Client Error	Resource not found	Requested metric doesn't exist
422	Unprocessable Entity	Business Error	All items in bulk operation failed	None of the metrics/values could be processed
429	Too Many Requests	Rate Limit	Rate limit exceeded	API key exceeded rate limits

500	Internal Server Error	Server Error	Unexpected server errors	Unexpected system errors

Error Response Formats

Authentication Errors (401):

```
{
  "Message": "API key is missing. Add it as 'X-API-Key' header or 'api_key' query parameter."
}
```

Permission Errors (403):

```
{
  "Message": "HTTP method POST not allowed for this API key"
}
```

Validation Errors (400):

```
{
  "Success": false,
  "Errors": [
    {
      "Field": "Name",
      "Issue": "The Name field is required."
    }
  ]
}
```

Business Logic Errors (404, 422):

```
{
  "Success": false,
  "Error": "Metric not found"
}
```

Rate Limiting Errors (429):

```
{
  "Message": "Rate limit exceeded. Please try again later."
}
```

Rate Limiting

Limits

- **Per Minute**: 60 requests per API key
- **Per Day**: 10,000 requests per API key

Rate Limit Headers

When approaching limits, responses include headers:

X-RateLimit-Limit-Minute: 60

X-RateLimit-Remaining-Minute: 45

X-RateLimit-Limit-Day: 10000

X-RateLimit-Remaining-Day: 8500

Handling Rate Limits

When you receive a 429 Too Many Requests response:

1. **Parse the response** to understand which limit was exceeded
2. **Implement exponential backoff** starting with a 1-minute delay
3. **Monitor the reset time** and adjust your request timing
4. **Consider request batching** to reduce API calls

Example Rate Limit Response:

```
{
  "Message": "Rate limit exceeded: 61/60 requests per minute. Try again after 2024-01-15T10:31:00 UTC."
}
```

6. Generate new access token using refresh token

Access tokens expire every 24 hours. When you receive a 401 Unauthorized error, refresh your token:

Endpoint: POST /api/v1/auth/token/refresh

Request:

POST /api/v1/auth/token/refresh?refreshToken={refresh_token}

Success Response:

```
{
  "Success": true,
  "Data": "eyJraWQiOiJ4aHd0Q2c0VFgwK0da...",
  "Message": "Token refreshed successfully"
}
```

Error Response:

```
{
  "Success": false,
  "Message": "Invalid or expired refresh token"
}
```

Authentication Errors

Status Code	Error	Description
401	Missing API Key	"API key is missing. Add it as 'X-API-Key' header or 'api_key' query parameter."
401	Invalid API Key	"Invalid API key"
401	Missing Authorization	"Authorization header is missing. Add it as 'Authorization: Bearer <token>' header."
401	Invalid Token	"Invalid authorization token"
403	Method Not Allowed	"HTTP method GET not allowed for this API key"

Security Best Practices

1. Credential Management

```
#  Good - Use environment variables
export ALIGN_API_KEY="{your_api_key}"
export ALIGN_ACCESS_TOKEN="{your_access_token}"

#  Bad - Hardcoded in source code
const apiKey = "ak_a1dCS2UZUkdSRn1UBW50XVBbU1VjW10..."
```

2. Secure Storage

- **Use secure vaults**: AWS Secrets Manager, Azure Key Vault, HashiCorp Vault
- **Separate environments**: Different API keys for dev, staging, production
- **Regular rotation**: Rotate API keys and tokens periodically

- **Access control**: Limit who can access API credentials

3. Network Security

- **HTTPS only**: All requests must use HTTPS
- **IP monitoring**: Monitor for unusual IP access patterns
- **Request logging**: Log all API requests for audit purposes
- **Error handling**: Don't expose sensitive information in error messages

4. Development Practices

```
//  Good - Proper error handling
try {
  const response = await fetch(apiUrl, {
    headers: {
      'Authorization': `Bearer ${process.env.ALIGN_ACCESS_TOKEN}`,
      'X-API-Key': process.env.ALIGN_API_KEY
    }
  });

  if (!response.ok) {
    // Handle specific error cases
    if (response.status === 401) {
      await refreshToken();
      // Retry request
    }
  }
} catch (error) {
  // Log error without exposing sensitive data
  console.error('API request failed:', error.message);
}
```

5. Token Management

- **Automatic refresh**: Implement automatic token refresh logic
- **Secure transmission**: Never log or transmit tokens in plain text

- **Expiration handling**: Always check token expiration before requests
- **Revocation**: Immediately revoke compromised tokens

Testing & Validation

Sample Test Cases

```
// Test Case 1: Valid API Call
const testValidCall = async () => {
  const response = await fetch(`${baseUrl}/api/v1/metrics`, {
    headers: {
      'Authorization': `Bearer ${accessToken}`,
      'X-API-Key': apiKey
    }
  });

  expect(response.status).toBe(200);
  expect(response.headers.get('content-type')).toContain('application/json');
};

// Test Case 2: Missing API Key
const testMissingApiKey = async () => {
  const response = await fetch(`${baseUrl}/api/v1/metrics`, {
    headers: {
      'Authorization': `Bearer ${accessToken}`
      // Missing X-API-Key header
    }
  });

  expect(response.status).toBe(401);
  const body = await response.json();
  expect(body.Message).toContain('API key is missing');
};
```

```
// Test Case 3: Rate Limit Testing
const testRateLimit = async () => {
  const promises = [];

  // Send 65 requests rapidly (exceeds 60/minute limit)
  for (let i = 0; i < 65; i++) {
    promises.push(
      fetch(`${baseUrl}/api/v1/metrics`, {
        headers: {
          'Authorization': `Bearer ${accessToken}`,
          'X-API-Key': apiKey
        }
      })
    );
  }

  const responses = await Promise.all(promises);
  const rateLimitedResponses = responses.filter(r => r.status === 429);
  expect(rateLimitedResponses.length).toBeGreaterThan(0);
};
```

Troubleshooting

Common Issues and Solutions

1. Authentication Failures

Problem: 401 Unauthorized - API key is missing

{

"Message": "API key is missing. Add it as 'X-API-Key' header or 'api_key' query parameter."

}

Solution:

- Verify the header name is exactly X-API-Key (case-sensitive)
- Ensure the API key value doesn't have extra spaces or characters
- Check that the API key hasn't expired

Problem: 401 Unauthorized - Invalid authorization token

{

"Message": "Invalid authorization token"

}

Solution:

- Check if the access token has expired (valid for 24 hours)
- Refresh the token using the refresh endpoint
- Verify the Bearer prefix is included in the Authorization header

2. Permission Issues

Problem: 403 Forbidden - HTTP method POST not allowed

{

"Message": "HTTP method POST not allowed for this API key"

}

Solution:

- Check the API key scopes in the Scoreboard dashboard
- Ensure the required scope (Insert for POST, Update for PUT, Read for GET) is enabled
- Generate a new API key with the correct permissions if needed

3. Rate Limiting

Problem: 429 Too Many Requests

{

"Message": "Rate limit exceeded: 61/60 requests per minute. Try again after 2024-01-15T10:31:00 UTC."

}

Solution:

- Implement exponential backoff retry logic
- Wait until the specified time before making new requests

- Consider batching multiple operations into single requests
- Monitor your request patterns and optimize for efficiency

4. Validation Errors

Problem: 400 Bad Request - Invalid date format


```
{
  "Success": false,
  "Errors": [
    {
      "Field": "FromDate",
      "Issue": "Invalid format. Please use yyyy-MM-dd format (e.g., 2024-01-15)"
    }
  ]
}
```


Solution:

- Ensure date parameters use the exact format yyyy-MM-dd
- Validate date ranges (fromDate cannot be later than toDate)
- Check that all required fields are included in the request body

5. Token Refresh Issues

Problem: Token refresh fails


```
{
  "Success": false,
  "Message": "Invalid or expired refresh token"
}
```


Solution:

- Check if the refresh token has expired (valid for 5 years)
- Verify the refresh token is being passed correctly in the query parameter
- If the refresh token has expired, generate new API credentials

Debugging Steps

1. Check API Key Status

- Log in to the Scoreboard platform
- Go to Administration → Manage API Keys
- Verify key is active and hasn't expired

2. Validate Request Format
    - Use tools like Postman or cURL to test requests
    - Check that JSON payload is properly formatted
    - Verify all required headers are included
3. Monitor Rate Limits
    - Track your API usage in the dashboard
    - Implement logging to monitor request frequency
    - Use rate limit headers to adjust request timing

Diagnostic Tools

cURL Example:

```
# Test basic connectivity
curl -v -X GET \
  "{BaseURL}/api/v1/metrics" \
  -H "Authorization: Bearer YOUR_TOKEN" \
  -H "X-API-Key: {your_api_key}"

# Test with verbose output to see full HTTP exchange
curl -v --trace-ascii trace.txt -X GET \
  "{BaseURL}/api/v1/metrics"
```

Postman Collection:

```
{
  "info": {
    "name": "Align Metrics API",
    "schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json"
  },
  "variable": [
    {
      "key": "baseUrl",
      "value": "{BaseURL}"
    },
    {
```

```
      "key": "accessToken",
      "value": "{accessToken}"
    },
    {
      "key": "apiKey",
      "value": "{apiKey}"
    }
  ],
  "auth": {
    "type": "bearer",
    "bearer": [
      {
        "key": "token",
        "value": "{accessToken}"
      }
    ]
  }
}
```

SDK & Code Examples

1.JavaScript/Node.js

```
class AlignMetricsClient {
  constructor(apiKey, accessToken, baseUrl) {
    this.apiKey = apiKey;
    this.accessToken = accessToken;
    this.baseUrl = baseUrl || '{BaseURL}';
    this.refreshToken = null;
  }

  // Set refresh token for automatic token refresh
  setRefreshToken(refreshToken) {
```

```
  this.refreshToken = refreshToken;
}

// Make authenticated request with automatic retry
async request(endpoint, options = {}) {
  const url = `${this.baseUrl}${endpoint}`;
  const headers = {
    'Authorization': `Bearer ${this.accessToken}`,
    'X-API-Key': this.apiKey,
    'Content-Type': 'application/json',
    ...options.headers
  };

  try {
    const response = await fetch(url, {
      ...options,
      headers
    });

    // Handle token expiration
    if (response.status === 401 && this.refreshToken) {
      await this.refreshAccessToken();
      // Retry request with new token
      headers['Authorization'] = `Bearer ${this.accessToken}`;
      return fetch(url, { ...options, headers });
    }

    // Handle rate limiting
    if (response.status === 429) {
      const retryAfter = response.headers.get('Retry-After') || 60;
      throw new Error(`Rate limit exceeded. Retry after ${retryAfter} seconds.`);
    }
```

```
    return response;
  } catch (error) {
    console.error('API request failed:', error);
    throw error;
  }
}

// Refresh access token
async refreshAccessToken() {
  if (!this.refreshToken) {
    throw new Error('No refresh token available');
  }

  const response = await fetch(
    `${this.baseUrl}/api/v1/auth/token/refresh?refreshToken=${this.refreshToken}`,
    { method: 'POST' }
  );

  if (!response.ok) {
    throw new Error('Failed to refresh token');
  }

  const data = await response.json();
  this.accessToken = data.Data;
}

// Get all metrics
async getMetrics(filters = {}) {
  const params = new URLSearchParams();
  if (filters.fromDate) params.append('fromDate', filters.fromDate);
  if (filters.toDate) params.append('toDate', filters.toDate);
```

```
  if (filters.owner) params.append('owner', filters.owner);

  const queryString = params.toString();
  const endpoint = `/api/v1/metrics${queryString ? `?${queryString}` : ''}`;

  const response = await this.request(endpoint);
  return response.json();
}

// Get single metric
async getMetric(id) {
  const response = await this.request(`/api/v1/metrics/${encodeURIComponent(id)}`);
  return response.json();
}

// Create metrics
async createMetrics(metrics) {
  const response = await this.request('/api/v1/metrics', {
    method: 'POST',
    body: JSON.stringify(metrics)
  });
  return response.json();
}

// Update metrics
async updateMetrics(metrics) {
  const response = await this.request('/api/v1/metrics', {
    method: 'PUT',
    body: JSON.stringify(metrics)
  });
  return response.json();
}
```

```
  // Add historical values
  async addPastValues(uniqueId, values) {
    const response = await this.request('/api/v1/metrics/pastValues', {
      method: 'POST',
      body: JSON.stringify({
        UniqueID: uniqueId,
        Values: values
      })
    });
    return response.json();
  }
}

// Usage Example
const client = new AlignMetricsClient(
  '{your_api_key}',
  '{your_access_token}'
);
client.setRefreshToken('your_refresh_token');

// Get all metrics
try {
  const metrics = await client.getMetrics({
    fromDate: '2024-01-01',
    toDate: '2024-12-31'
  });
  console.log('Metrics:', metrics);
} catch (error) {
  console.error('Error fetching metrics:', error);
}
```

```
// Create new metrics
try {
  const newMetrics = await client.createMetrics([
    {
      Name: 'Revenue',
      Value: '1000000',
      Description: 'Monthly revenue',
      Owner: 'John Doe',
      Status: 'Active'
    }
  ]);
  console.log('Created metrics:', newMetrics);
} catch (error) {
  console.error('Error creating metrics:', error);
}
```

2.Python

```
import requests
import json
import time
from urllib.parse import urlencode
from typing import Dict, List, Optional

class AlignMetricsClient:
    def __init__(self, api_key: str, access_token: str, base_url: str = None):
        self.api_key = api_key
        self.access_token = access_token
        self.base_url = base_url or '{BaseURL}'
        self.refresh_token = None
        self.session = requests.Session()

    def set_refresh_token(self, refresh_token: str):
        """Set refresh token for automatic token refresh"""
```

```
    self.refresh_token = refresh_token

def _get_headers(self) -> Dict[str, str]:
    """Get standard headers for API requests"""
    return {
        'Authorization': f'Bearer {self.access_token}',
        'X-API-Key': self.api_key,
        'Content-Type': 'application/json'
    }

def _make_request(self, endpoint: str, method: str = 'GET', data: Dict = None, params: Dict = None):
    """Make authenticated request with automatic retry logic"""
    url = f"{self.base_url}{endpoint}"
    headers = self._get_headers()

    try:
        response = self.session.request(
            method=method,
            url=url,
            headers=headers,
            json=data,
            params=params
        )

        # Handle token expiration
        if response.status_code == 401 and self.refresh_token:
            self._refresh_access_token()
            headers = self._get_headers()
            response = self.session.request(
                method=method,
                url=url,
                headers=headers,
```

```
            json=data,
            params=params
        )

        # Handle rate limiting
        if response.status_code == 429:
            retry_after = int(response.headers.get('Retry-After', 60))
            raise Exception(f"Rate limit exceeded. Retry after {retry_after} seconds.")

        response.raise_for_status()
        return response.json()

    except requests.exceptions.RequestException as e:
        print(f"API request failed: {e}")
        raise

def _refresh_access_token(self):
    """Refresh access token using refresh token"""
    if not self.refresh_token:
        raise Exception("No refresh token available")

    url = f"{self.base_url}/api/v1/auth/token/refresh"
    params = {'refreshToken': self.refresh_token}

    response = requests.post(url, params=params)
    response.raise_for_status()

    data = response.json()
    self.access_token = data['Data']

def get_metrics(self, from_date: str = None, to_date: str = None, owner: str = None):
    """Get all metrics with optional filtering"""
```

```
        params = {}
        if from_date:
            params['fromDate'] = from_date
        if to_date:
            params['toDate'] = to_date
        if owner:
            params['owner'] = owner

        return self._make_request('/api/v1/metrics', params=params)

    def get_metric(self, metric_id: str):
        """Get single metric by ID"""
        return self._make_request(f'/api/v1/metrics/{metric_id}')

    def create_metrics(self, metrics: List[Dict]):
        """Create one or more metrics"""
        return self._make_request('/api/v1/metrics', method='POST', data=metrics)

    def update_metrics(self, metrics: List[Dict]):
        """Update one or more metrics"""
        return self._make_request('/api/v1/metrics', method='PUT', data=metrics)

    def add_past_values(self, unique_id: str, values: List[Dict]):
        """Add historical values for a metric"""
        data = {
            'UniqueID': unique_id,
            'Values': values
        }
        return self._make_request('/api/v1/metrics/pastValues', method='POST', data=data)

# Usage Example
if __name__ == "__main__":
```

```
client = AlignMetricsClient(
    api_key='{your_api_key}',
    access_token='{your_access_token}'
)
client.set_refresh_token('your_refresh_token')

try:
    # Get all metrics
    metrics = client.get_metrics(
        from_date='2024-01-01',
        to_date='2024-12-31'
    )
    print(f"Retrieved {len(metrics['Data'])} metrics")

    # Create new metric
    new_metrics = client.create_metrics([
        {
            'Name': 'Python Revenue',
            'Value': '2000000',
            'Description': 'Revenue tracked via Python',
            'Status': 'Active'
        }
    ])
    print("Created metrics:", new_metrics)

    # Add historical values
    if new_metrics['Success'] and new_metrics['Data']:
        unique_id = new_metrics['Data'][0]['UniqueID']
        past_values = client.add_past_values(unique_id, [
            {'Date': '2024-01-01', 'Value': '1800000'},
            {'Date': '2024-01-02', 'Value': '1900000'}
        ])
```

```
        print("Added past values:", past_values)

    except Exception as e:
        print(f"Error: {e}")
```

3.C#/.NET

```
using System;
using System.Collections.Generic;
using System.Net.Http;
using System.Text;
using System.Text.Json;
using System.Threading.Tasks;
using System.Web;

public class AlignMetricsClient
{
    private readonly HttpClient _httpClient;
    private readonly string _baseUrl;
    private string _apiKey;
    private string _accessToken;
    private string _refreshToken;

    public AlignMetricsClient(string apiKey, string accessToken, string baseUrl = null)
    {
        _apiKey = apiKey;
        _accessToken = accessToken;
        _baseUrl = baseUrl ?? "{BaseURL}";
        _httpClient = new HttpClient();
    }

    public void SetRefreshToken(string refreshToken)
    {
```

```
        _refreshToken = refreshToken;
    }

    private void SetHeaders()
    {
        _httpClient.DefaultRequestHeaders.Clear();
        _httpClient.DefaultRequestHeaders.Add("Authorization", $"Bearer {_accessToken}");
        _httpClient.DefaultRequestHeaders.Add("X-API-Key", _apiKey);
    }

    private async Task<T> MakeRequestAsync<T>(string endpoint, HttpMethod method, object data = null)
    {
        SetHeaders();
        var url = $"{_baseUrl}{endpoint}";

        using var request = new HttpRequestMessage(method, url);

        if (data != null)
        {
            var json = JsonSerializer.Serialize(data);
            request.Content = new StringContent(json, Encoding.UTF8, "application/json");
        }

        try
        {
            var response = await _httpClient.SendAsync(request);

            // Handle token expiration
            if (response.StatusCode == System.Net.HttpStatusCode.Unauthorized &&
!string.IsNullOrEmpty(_refreshToken))
            {
                await RefreshAccessTokenAsync();
```

```
            SetHeaders();

            // Retry request
            using var retryRequest = new HttpRequestMessage(method, url);
            if (data != null)
            {
                var retryJson = JsonSerializer.Serialize(data);
                retryRequest.Content = new StringContent(retryJson, Encoding.UTF8, "application/json");
            }
            response = await _httpClient.SendAsync(retryRequest);
        }

        // Handle rate limiting
        if (response.StatusCode == System.Net.HttpStatusCode.TooManyRequests)
        {
            var retryAfter = response.Headers.RetryAfter?.Delta?.TotalSeconds ?? 60;
            throw new Exception($"Rate limit exceeded. Retry after {retryAfter} seconds.");
        }

        response.EnsureSuccessStatusCode();
        var responseContent = await response.Content.ReadAsStringAsync();
        return JsonSerializer.Deserialize<T>(responseContent);
    }
    catch (HttpRequestException ex)
    {
        throw new Exception($"API request failed: {ex.Message}", ex);
    }
}

private async Task RefreshAccessTokenAsync()
{
    if (string.IsNullOrEmpty(_refreshToken))
```

```
            throw new Exception("No refresh token available");

        var url = $"{_baseUrl}/api/v1/auth/token/refresh?refreshToken={HttpUtility.UrlEncode(_refreshToken)}";

        using var request = new HttpRequestMessage(HttpMethod.Post, url);
        var response = await _httpClient.SendAsync(request);
        response.EnsureSuccessStatusCode();

        var responseContent = await response.Content.ReadAsStringAsync();
        var tokenResponse = JsonSerializer.Deserialize<TokenRefreshResponse>(responseContent);
        _accessToken = tokenResponse.Data;
    }

    public async Task<MetricsResponse> GetMetricsAsync(string fromDate = null, string toDate = null, string
owner = null)
    {
        var queryParams = new List<string>();
        if (!string.IsNullOrEmpty(fromDate)) queryParams.Add($"fromDate={HttpUtility.UrlEncode(fromDate)}");
        if (!string.IsNullOrEmpty(toDate)) queryParams.Add($"toDate={HttpUtility.UrlEncode(toDate)}");
        if (!string.IsNullOrEmpty(owner)) queryParams.Add($"owner={HttpUtility.UrlEncode(owner)}");

        var queryString = queryParams.Count > 0 ? "?" + string.Join("&", queryParams) : "";
        return await MakeRequestAsync<MetricsResponse>($"/api/v1/metrics{queryString}", HttpMethod.Get);
    }

    public async Task<SingleMetricResponse> GetMetricAsync(string id)
    {
        return await MakeRequestAsync<SingleMetricResponse>($"/api/v1/metrics/{HttpUtility.UrlEncode(id)}",
HttpMethod.Get);
    }

    public async Task<CreateMetricsResponse> CreateMetricsAsync(List<CreateMetricRequest> metrics)
    {
```

```
        return await MakeRequestAsync<CreateMetricsResponse>("/api/v1/metrics", HttpMethod.Post, metrics);
    }

    public async Task<UpdateMetricsResponse> UpdateMetricsAsync(List<UpdateMetricRequest> metrics)
    {
        return await MakeRequestAsync<UpdateMetricsResponse>("/api/v1/metrics", HttpMethod.Put, metrics);
    }

    public async Task<PastValuesResponse> AddPastValuesAsync(string uniqueId, List<MetricValue> values)
    {
        var request = new PastValuesRequest
        {
            UniqueID = uniqueId,
            Values = values
        };
        return await MakeRequestAsync<PastValuesResponse>("/api/v1/metrics/pastValues", HttpMethod.Post,
request);
    }

    public void Dispose()
    {
        _httpClient?.Dispose();
    }
}

// Data Models
public class MetricsResponse
{
    public bool Success { get; set; }
    public string Message { get; set; }
    public List<Metric> Data { get; set; }
}
```

```
public class Metric
{
    public string Name { get; set; }
    public string Value { get; set; }
    public string Description { get; set; }
    public string Comments { get; set; }
    public string Owner { get; set; }
    public string Status { get; set; }
    public DateTime LastUpdatedUTC { get; set; }
    public string UniqueID { get; set; }
}

public class CreateMetricRequest
{
    public string Name { get; set; }
    public string Value { get; set; }
    public string Description { get; set; }
    public string Owner { get; set; }
    public string Status { get; set; }
}

public class UpdateMetricRequest
{
    public string UniqueID { get; set; }
    public string Name { get; set; }
    public string Value { get; set; }
    public string Description { get; set; }
    public string Owner { get; set; }
    public string Status { get; set; }
}
```

```
// Usage Example
class Program
{
    static async Task Main(string[] args)
    {
        var client = new AlignMetricsClient(
            "{your_api_key}",
            "{your_access_token}"
        );
        client.SetRefreshToken("your_refresh_token");

        try
        {
            // Get metrics
            var metrics = await client.GetMetricsAsync(
                fromDate: "2024-01-01",
                toDate: "2024-12-31"
            );
            Console.WriteLine($"Retrieved {metrics.Data.Count} metrics");

            // Create new metric
            var newMetrics = await client.CreateMetricsAsync(new List<CreateMetricRequest>
            {
                new CreateMetricRequest
                {
                    Name = "C# Revenue",
                    Value = "3000000",
                    Description = "Revenue tracked via C#",
                    Status = "Active"
                }
            });
            Console.WriteLine($"Created metrics: {newMetrics.Message}");
```

```
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error: {ex.Message}");
        }
        finally
        {
            client.Dispose();
        }
    }
}
```

4.PHP

```
<?php

class AlignMetricsClient {
    private $apiKey;
    private $accessToken;
    private $refreshToken;
    private $baseUrl;

    public function __construct($apiKey, $accessToken, $baseUrl = null) {
        $this->apiKey = $apiKey;
        $this->accessToken = $accessToken;
        $this->baseUrl = $baseUrl ?: '{BaseURL}';
    }

    public function setRefreshToken($refreshToken) {
        $this->refreshToken = $refreshToken;
    }

    private function getHeaders() {
```

```
        return [
            'Authorization: Bearer ' . $this->accessToken,
            'X-API-Key: ' . $this->apiKey,
            'Content-Type: application/json'
        ];
    }

    private function makeRequest($endpoint, $method = 'GET', $data = null) {
        $url = $this->baseUrl . $endpoint;
        $headers = $this->getHeaders();

        $curl = curl_init();
        curl_setopt_array($curl, [
            CURLOPT_URL => $url,
            CURLOPT_RETURNTRANSFER => true,
            CURLOPT_CUSTOMREQUEST => $method,
            CURLOPT_HTTPHEADER => $headers,
            CURLOPT_TIMEOUT => 30
        ]);

        if ($data !== null) {
            curl_setopt($curl, CURLOPT_POSTFIELDS, json_encode($data));
        }

        $response = curl_exec($curl);
        $httpCode = curl_getinfo($curl, CURLINFO_HTTP_CODE);
        $error = curl_error($curl);
        curl_close($curl);

        if ($error) {
            throw new Exception("cURL Error: " . $error);
        }
```

```
    // Handle token expiration
    if ($httpCode === 401 && $this->refreshToken) {
        $this->refreshAccessToken();
        return $this->makeRequest($endpoint, $method, $data);
    }

    // Handle rate limiting
    if ($httpCode === 429) {
        throw new Exception("Rate limit exceeded. Please retry after 60 seconds.");
    }

    if ($httpCode >= 400) {
        throw new Exception("HTTP Error: " . $httpCode . " - " . $response);
    }

    return json_decode($response, true);
}

private function refreshAccessToken() {
    if (!$this->refreshToken) {
        throw new Exception("No refresh token available");
    }

    $url = $this->baseUrl . '/api/v1/auth/token/refresh?refreshToken=' . urlencode($this->refreshToken);

    $curl = curl_init();
    curl_setopt_array($curl, [
        CURLOPT_URL => $url,
        CURLOPT_RETURNTRANSFER => true,
        CURLOPT_CUSTOMREQUEST => 'POST',
        CURLOPT_TIMEOUT => 30
```

```
        ]);

        $response = curl_exec($curl);
        $httpCode = curl_getinfo($curl, CURLINFO_HTTP_CODE);
        curl_close($curl);

        if ($httpCode !== 200) {
            throw new Exception("Failed to refresh token");
        }

        $data = json_decode($response, true);
        $this->accessToken = $data['Data'];
    }

    public function getMetrics($fromDate = null, $toDate = null, $owner = null) {
        $params = [];
        if ($fromDate) $params['fromDate'] = $fromDate;
        if ($toDate) $params['toDate'] = $toDate;
        if ($owner) $params['owner'] = $owner;

        $queryString = !empty($params) ? '?' . http_build_query($params) : '';
        return $this->makeRequest('/api/v1/metrics' . $queryString);
    }

    public function getMetric($id) {
        return $this->makeRequest('/api/v1/metrics/' . urlencode($id));
    }

    public function createMetrics($metrics) {
        return $this->makeRequest('/api/v1/metrics', 'POST', $metrics);
    }
```

```
    public function updateMetrics($metrics) {
        return $this->makeRequest('/api/v1/metrics', 'PUT', $metrics);
    }

    public function addPastValues($uniqueId, $values) {
        $data = [
            'UniqueID' => $uniqueId,
            'Values' => $values
        ];
        return $this->makeRequest('/api/v1/metrics/pastValues', 'POST', $data);
    }
}

// Usage Example
try {
    $client = new AlignMetricsClient(
        '{your_api_key}',
        '{your_access_token}'
    );
    $client->setRefreshToken('your_refresh_token');

    // Get metrics
    $metrics = $client->getMetrics('2024-01-01', '2024-12-31');
    echo "Retrieved " . count($metrics['Data']) . " metrics\n";

    // Create new metric
    $newMetrics = $client->createMetrics([
        [
            'Name' => 'PHP Revenue',
            'Value' => '4000000',
            'Description' => 'Revenue tracked via PHP',
            'Status' => 'Active'
```

```
        ]
    ]);
    echo "Created metrics: " . $newMetrics['Message'] . "\n";

} catch (Exception $e) {
    echo "Error: " . $e->getMessage() . "\n";
}
?>
```

Support

Contact Information

Primary Support Channel

- **Email**: support@aligntoday.com
- **Response Time**: Within 48 business hours for standard issues
- **Escalation**: Critical issues will be prioritized

Getting Help

Before Contacting Support

1. Check this documentation for common solutions
2. Verify your API key status in the Scoreboard Application -> Administration -> Manage API Keys section
3. Review error messages and HTTP status codes

When Contacting Support, Include:

- API key label (not the actual key)
- Timestamp of the issue
- Full error message or HTTP status code
- Sample request (without sensitive data)
- Expected vs. actual behavior

Version History

- **v1.0** (Current): Initial Metrics API release

Appendix

Quick Reference

Base URL: https://api.aligntoday.com

Required Headers:

Authorization: Bearer {access_token}

X-API-Key: {api_key}

Rate Limits:

- 60 requests/minute per API key
- 10,000 requests/day per API key

Key Endpoints:

- GET /api/v1/metrics - Get all metrics
- GET /api/v1/metrics/{id} - Get single metric
- POST /api/v1/metrics - Create metrics
- PUT /api/v1/metrics - Update metrics
- POST /api/v1/metrics/pastValues - Add historical values
- POST /api/v1/auth/token/refresh - Refresh access token

Status Code Quick Reference

Code	Meaning	Action
200	Success	Continue
201	Created	Continue
207	Partial Success	Check individual results
400	Bad Request	Fix request format
401	Unauthorized	Check credentials/refresh token
403	Forbidden	Check API key permissions
404	Not Found	Verify resource exists
422	Unprocessable	Fix validation errors
429	Rate Limited	Wait and retry

500	Server Error	Contact support

Complete Error Messages:

Complete HTTP Status Code Reference

Status Code	Status Name	Category	Description	When It Occurs
200	OK	Success	GET requests completed successfully	Successful retrieval of metrics
201	Created	Success	All CREATE/UPDATE operations successful	All metrics/values created/updated successfully
207	Multi-Status	Partial Success	Partial success in bulk operations	Some items succeeded, some failed in bulk operations
400	Bad Request	Client Error	Pre-API validation failures, invalid request format	Missing required fields, invalid date formats, empty request body
401	Unauthorized	Auth Error	Missing/invalid API key or authorization token	API key missing, invalid, expired, or authorization token issues
403	Forbidden	Auth Error	HTTP method not permitted for API key	API key lacks permission for specific HTTP method (GET/POST/PUT)
404	Not Found	Client Error	Resource not found (business logic)	Requested metric does not exist
422	Unprocessable Entity	Business Logic Error	All items in bulk operation failed	None of the metrics/values could be processed successfully
429	Too Many Requests	Rate Limit Error	Rate limit exceeded	API key exceeded minute or daily rate limits
500	Internal Server Error	Server Error	Unexpected server errors	Unexpected system errors during processing

Detailed Response Code Breakdown

🔐 **Authentication & Authorization Errors (401, 403, 429)**

401 Unauthorized - API Key Issues

Error Scenario	Message	Description
Missing API Key	"API key is missing. Add it as 'X-API-Key' header or 'api_key' query parameter."	No API key provided in request
Invalid Format	"API key format is invalid or corrupted."	API key structure is malformed
Key Not Found	"API key does not exist or has been deleted."	API key not found in database
Key Inactive	"API key has been deactivated."	API key exists but is disabled
Company Mismatch	"API key does not belong to the specified company."	API key company ID doesn't match
User Mismatch	"API key does not belong to the specified user."	API key user ID doesn't match
Key Expired	"API key expired on [date] UTC."	API key has passed expiration date
Unable to Decode	"Unable to decode API key information."	API key cannot be decoded/parsed

401 Unauthorized - Authorization Token Issues

Error Scenario	Message	Description
Missing Authorization	"Authorization header is missing. Add it as 'Authorization: Bearer <token>' header."	No Bearer token in Authorization header
Token Not Found	"Access token is required"	Token is required but not provided
Invalid Token Format	"Token format is invalid"	Token structure is malformed
Token Expired	"Token expired on [date] UTC"	JWT token has expired
Invalid Signature	"Token signature is invalid"	Token signature verification failed
Invalid Issuer	"Token issuer is invalid"	Token issuer doesn't match expected
Token Secret Mismatch	"Token does not belong to specific secret key."	Token doesn't match API key's AWS token

403 Forbidden - Permission Issues

Error Scenario	Message	Description
Method Not Allowed	"API key does not have permission for [permission]."	API key lacks permission for HTTP method
No Permissions	"No permissions configured for this API key."	API key has no permissions assigned
Unsupported Method	"HTTP method '[method]' is not supported."	HTTP method is not supported by system

429 Too Many Requests - Rate Limiting

Error Scenario	Message	Description
Minute Limit	"Rate limit exceeded: [current]/[limit] requests per minute. Try again after [time] UTC."	Per-minute rate limit exceeded
Daily Limit	"Daily rate limit exceeded: [current]/[limit] requests per day. Resets at [date] 00:00:00 UTC."	Per-day rate limit exceeded

📝 Business Logic Errors (400, 404, 422)

400 Bad Request - Validation Errors

Context	Error Scenario	Message	Description
Get All Metrics	Date Format	"Invalid format. Please use yyyy-MM-dd format (e.g., 2024-01-15)"	Date parameter not in yyyy-MM-dd format
Get All Metrics	Date Range	"Cannot be later than toDate" / "Cannot be earlier than fromDate"	fromDate > toDate validation
Get Single Metric	Missing ID	"Metric ID is required"	ID parameter is empty or null
Create/Update Metrics	Empty Request	"Request body cannot be empty"	Request body is null or empty array
Create/Update Metrics	Model Validation	Field-specific validation messages	Data annotation validation failures
Add Past Values	Empty Request	"Request body is required"	Request body is null
Add Past Values	Missing UniqueID	"UniqueID is required"	UniqueID field is empty

Context	Error Scenario	Message	Description
Add Past Values	Missing Values	"At least one value is required"	Values array is empty
Add Past Values	Metric Not Found	"Metric Not Found for unique ID: '[ID]'"	Referenced metric doesn't exist

404 Not Found - Resource Errors

Context	Error Scenario	Message	Description
Get Single Metric	Metric Not Found	"Metric not found"	Requested metric ID doesn't exist

422 Unprocessable Entity - Business Logic Failures

Context	Error Scenario	Message	Description
Create Metrics	All Failed	"No metrics could be created"	All items in batch failed validation/creation
Update Metrics	All Failed	"No metrics could be updated"	All items in batch failed validation/update
Add Past Values	All Failed	"No values could be updated"	All values in batch failed validation/creation

✅ **Success Responses (200, 201, 207)**

200 OK - Successful Retrieval

Context	Message	Description
Get All Metrics	"Metrics retrieved successfully"	Successfully returned metrics list
Get Single Metric	"Metric retrieved successfully"	Successfully returned single metric

201 Created - Successful Creation/Update

Context	Message	Description
Create Metrics	"All metrics created successfully"	All metrics in batch created successfully

Context	Message	Description
Update Metrics	"All metrics updated successfully"	All metrics in batch updated successfully
Add Past Values	"All values updated successfully"	All values in batch added successfully

207 Multi-Status - Partial Success

Context	Message Pattern	Description
Create Metrics	"[X] of [Y] metrics created successfully"	Some metrics created, some failed
Update Metrics	"[X] of [Y] metrics updated successfully"	Some metrics updated, some failed
Add Past Values	"[X] of [Y] values updated successfully"	Some values added, some failed

🚨 **Server Errors (500)**

500 Internal Server Error - System Failures

Context	Message	Description
Get All Metrics	"An unexpected error occurred while retrieving metrics"	System error during retrieval
Get Single Metric	"An unexpected error occurred while retrieving the metric"	System error during single metric retrieval
Create Metrics	"An unexpected error occurred while creating metrics"	System error during creation
Update Metrics	"An unexpected error occurred while updating metrics"	System error during update
Add Past Values	"An unexpected error occurred while adding past values"	System error during value addition
Authentication	"An error occurred while validating the API key."	System error during authentication

FAQs

Q: Can I generate multiple API keys?
A: Yes

Q: What is the max API rate?
A: 60 requests per minute per API and 10000 requests per day. (numbers can change)

Q: How do I revoke a key?
A: Use the Scoreboard App > Manage API Key > Select the Inactive option against the API Key.

Q: Can I automate token refresh?
A: Yes, integrate the refresh-token API in your app's auth flow.

Last Updated: July 2025
Version: 1.0
© 2025 Align- All Rights Reserved