# ITS WEB API SPECIFICATION

Version 1.30 ● 16 July 2024



Table of Contents

[ITS WEB API SPECIFICATION 1](#_Toc172188742)

[Overview 3](#_Toc172188743)

[Specification Overview 3](#_Toc172188744)

[Using the Web Services Interface 3](#_Toc172188745)

[Authentication 4](#_Toc172188746)

[Supported Formats 4](#_Toc172188747)

[Error Handling 5](#_Toc172188748)

[HTTP Status Code Summary 5](#_Toc172188749)

[Error Types 5](#_Toc172188750)

[Error Response Structure 6](#_Toc172188751)

[Sample Response for an Invalid Argument 6](#_Toc172188752)

[Other Error Responses 7](#_Toc172188753)

[API Parameter Guidelines 7](#_Toc172188754)

[Date Parameters 7](#_Toc172188755)

[Resource Objects 7](#_Toc172188756)

[Sample Resource Object 7](#_Toc172188757)

[Multiple Values 7](#_Toc172188758)

[Pagination 8](#_Toc172188759)

# Overview

## Specification Overview

This specification document defines the application programming interface (API) for server-to-server interfaces with Internet Testing Systems (ITS) resources. The API assumes the following:

* Every web API consumer has an ITS user account, an OAUTH2 API key, and an OAUTH2 API secret.
* The remote application will only communicate using HTTPS.
* It supports TLS 1.2 only with these strong ciphers:
  + TLS\_ECDHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256
  + TLS\_ECDHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384
  + TLS\_ECDHE\_RSA\_WITH\_CHACHA20\_POLY1305\_SHA256

## Using the Web Services Interface

ITS provides the web services interface on our staging and production environments. The interface is available using HTTPS protocol through a server URL that is specific to each environment. Multiple versions of web services are supported, and a new version is created whenever functionality is changed for an existing call.

# Authentication

The ITS API uses the OAUTH2 client credential flow to authenticate and authorize clients.

**Obtaining Web API Bearer Token**

1. Every ITS Web API requires a bearer token for authentication.  The steps below walk you through obtaining a bearer token.
2. POST *grant type, client id*, *client secret*, and *scope* to ITS’ identity server token endpoint.
3. ITS’ Token Staging End Point = <https://identity-staging.testsys.io/connect/token>
4. grant\_type = client\_credentials
5. client\_id = Program Team will supply.
6. client\_secret = Program Team will supply.
7. Scope
8. Escaped =  https%3A%2F%2Fapi-staging.testsys.io
9. Plain text = [https://api-staging.testsys.io](https://api-staging.testsys.io/)
10. Copy the access token value returned.  This will be a JWT.
11. Add an authorization header to your Web API request in the form Bearer <token>.  Note that the token will expire after one hour.

# Supported Formats

The ITS API supports JSON by default. API calls that also exist in the legacy API support both XML and JSON. This specification defines which formats are supported by an API call.

Clients should use the Content-Type and Accept headers to specify the expected format. If no headers are supplied, the API expects JSON in the request and returns JSON in the response.

* Use the Content-Type header to specify the format of the data in the request body.
  + If Content-Type: application/json is supplied, the API expects JSON.
  + If Content-Type: application/xml is supplied, the API expects XML.
* Use the Accept header attribute to specify the format of the response data.
  + Accept: application/json returns a JSON response.
  + Accept: application/xml returns an XML response.

# Error Handling

ITS uses conventional HTTP response codes to indicate the success or failure of an API request. The API uses 200 to indicate success. Codes in the 4xx range indicate that a request failed given the information provided. For example, a required parameter may have been omitted or an invalid value may have been supplied for a parameter. Codes in the 5xx range indicate an error with ITS’s servers (these are rare).

## HTTP Status Code Summary

| HTTP Status Code | Status | Description |
| --- | --- | --- |
| 200 | OK | Everything worked as expected. |
| 400 | Bad Request | The request was unacceptable. The server did not understand the request due to malformed syntax. |
| 401 | Unauthorized | The request was not authorized by a valid bearer token, or the request was made on the wrong environment. |
| 403 | Forbidden | The API key doesn’t have permissions to perform the request (the client-id does not have access to the specific call or to the program/bank/institution requested). |
| 404 | Not Found | The requested resource doesn’t exist. |
| 405 | Method Not Allowed | The HTTP method is not supported for this endpoint. |
| 409 | Conflict | The request conflicts with another request. This may be a result of using the same idempotent key. |
| 422 | Unprocessable Entity | The request was unacceptable. The server understood the content type of the request entity, but it couldn’t process the request due to client error. For example, there may be invalid or missing parameters. |
| 429 | Too Many Requests | Too many requests reached the API too quickly. We recommend an exponential backoff of the requests. |
| 500, 502, 503, 504 | Server Errors | Something went wrong on ITS’s API end. These errors are rare. |

## Error Types

| Type | Description |
| --- | --- |
| internal-error | An internal error occurs when there is an unexpected, temporary problem with ITS’s servers. These types of errors are extremely rare. |
| invalid-arguments | An invalid argument error occurs when your request has invalid or missing parameters. |
| unsupported-media-type | An unsupported media type error occurs when the request payload is in an unsupported format. |
| deprecated-versions | A deprecated version error occurs when the version requested is no longer available. |

## Error Response Structure

The ITS error response structure follows the standard set forth in [IETF RFC 7807](https://datatracker.ietf.org/doc/html/rfc7807). The API returns a JSON or XML problem details object when it encounters an error. The table below lists the elements that could be returned in ITS’s problem details error response.

| Element | Description |
| --- | --- |
| errors | A list of all errors that occurred. |
| type | The type of error that occurred. This is one of four possible values: internal-error, invalid-arguments, unsupported-media-type, or startup-failure. See a detailed description of each type in [Error Types](#_Error_Types). |
| title | A short, human-readable summary of the error type. This is the same for each occurrence of the error, except for the purposes of localization. |
| status | The HTTP status code generated for this occurrence of the error. |
| detail | A human-readable explanation specific to this occurrence of the error. |
| instance | A URI reference that links to further details about this occurrence of the error. |

### Sample Response for an Invalid Argument

In the example below, a request to an endpoint was made with invalid values for the after-id, before-id, and exclude-flag parameters.

#### JSON

|  |
| --- |
| {  "errors": {  "after-id": [  "Cannot include BeforeId and AfterId."  ],  "before-id": [  "BeforeId must be a positive integer.",  "Cannot include BeforeId and AfterId."  ],  "exclude-flag": [  "The field ExcludeFlag must be between 0 and 4095."  ]  },  "type": "/error/invalid-arguments",  "title": "Your request had invalid arguments.",  "status": 422,  "detail": "See error property for details.",  "instance": "/error/instance/5A4FEC35-B04B-433F-9079-8ACAF113A558"  } |

#### XML

|  |
| --- |
| <problem>  <errors>  <after-id>Cannot include BeforeId and AfterId.</after-id>  <before-id>BeforeId must be a positive integer.</before-id>  <before-id> Cannot include BeforeId and AfterId.</before-id>  <exclude-flag>The field ExcludeFlag must be between 0 and 4095.</exclude-flag>  </errors>  <type>/error/invalid-arguments</type>  <title>Your request had invalid arguments.</title>  <status>422</status>  <detail>See error property for details.</detail>  <instance>/error/instance/5A4FEC35-B04B-433F-9079-8ACAF113A558</instance>  </problem> |

### Other Error Responses

Some 4xx errors do not return any information in the response body. We recommend using the solutions in the following table if you experience these errors.

|  |  |
| --- | --- |
| Type | Causes and Solutions |
| 401 Unauthorized | Refer to the WWW-Authenticate header for information on required credentials (see OAuth) and issues with the bearer token. |
| 403 Forbidden | Check your query string for typos. If you think the call you’re making should succeed, contact your PM. |
| 405 Method Not Allowed | Refer to the Allow header for a list of the supported HTTP methods. |

# API Parameter Guidelines

## Date Parameters

All API parameter dates are displayed in UTC in ISO date and time format: (yyyy-mm-ddTHH:mm:ssZ).

## Resource Objects

Resource objects are used in two ways. They can be posted to an Import, Create, or Update call, or they can be returned from a Query call. Every element in the resource object can be returned when queried. Only resource elements that have “Create/Update” set to true are honored by the Create and Update calls. If a resource element is posted to a Create or Update call and “Create/Update” is set to false, the element is ignored.

### Sample Resource Object

The table below depicts a sample resource object.

| Name | Type | Required | Description | Create/Update |
| --- | --- | --- | --- | --- |
| Identifiers | | | | |
| resource-id | string | No | The unique ITS database identifier. | No |
| program-resource-id | string | No | The external identifier of the resource. | Yes |

## Multiple Values

If a parameter is flagged with the ability to support multiple values, then multiple values can be supplied in the request parameter. The values must be comma delimited.

## Pagination

All resources support bulk queries that have the potential to return a significant amount of data. To limit the amount of data returned, the API supports pagination via three parameters: limit, before-id, and after-id.

ITS allows cursor-based pagination using the before-id and after-id parameters. Both parameters take an existing resource ID value and return objects in reverse chronological order. The before-id parameter returns objects listed before the named object. The after-id parameter returns objects listed after the named object. These parameters are mutually exclusive: only one of after-id or before-id may be used.