



UG 30

Bidding Upload/Download (BUD) APIs User's Guide

Issued: August 2026

Version: 1.0

Effective Date: 08/27/2026

Prepared By: IT Reliability Technologies

New York Independent System Operator
10 Krey Boulevard
Rensselaer, NY 12144
(518) 356-6060
www.nyiso.com

Disclaimer: The information contained within this guide, along with other NYISO guides, is intended to be used for information purposes only, and is subject to change. The NYISO is not responsible for the user's reliance on these publications, or for any erroneous or misleading material.

Table of Contents

TABLE OF FIGURES	VI
REVISION HISTORY	VII
1. IMPORTANT USAGE NOTES AND INTRODUCTION	1
1.1.Important Usage Notes.....	1
1.2.. Introduction.....	1
1.3.Using the BUD APIs	2
1.3.1. General Usage Guidelines.....	2
1.3.3. Location of BUD API Service Endpoints.....	3
2. AUTHENTICATION.....	4
2.1.How to Obtain Access Credentials	4
2.2.HTTP Authorization Header Format.....	4
2.3.NAESB Digital Certificate (TLS Client Certificate).....	4
2.4.Security Roles and Endpoint Access.....	5
2.5.Data Formatting	5
2.5.1. Formatting Numbers	5
2.5.2. Formatting Dates and Times	6
2.5.3. Response requestType Echo Behavior	6
2.5.4. Worked 25-Hour DST Example (JSON POST/load-bid).....	6
2.5.5. JSON Payload Structure	8
2.6.HTTP Status Codes for Responses.....	9
2.6.1. Standard 400 Bad Request Response Shape	9
2.6.2. Standard 401 Unauthorized Response Shape.....	9
2.6.3. Standard 403 Forbidden Response Shape	10
2.6.4. Standard 500 Internal Server Error Response Shape	10
2.7.Interaction Methods Overview.....	10
3. OPTION 1: REST JSON API	12
3.1.Overview & When to Use.....	12
3.2.General Workflow.....	12
3.3.Repeatability, Retries, and Error Handling	13
3.4.Operations-Specific Reference.....	13
4. OPTION 2: CSV LEGACY ENDPOINT	14
4.1.Overview & When to Use.....	14

4.2.How the BUD CSV Endpoint Works	14
4.3.Supported CSV Upload Operations (BID_TYPE values).....	14
4.4.Supported CSV Download Operations (QUERY_TYPE values)	15
4.5.CSV Template Format Overview	15
4.6.Worked Examples.....	16
5. OPTION 3: WEB GUI (BUD UI)	17
5.1.Overview & When to Use.....	17
5.2.Accessing the BUD UI.....	17
5.3.Uploading a CSV File	17
5.4.Accepted File Types.....	18
5.5.Interpreting Results.....	18
6. COMPARISON: CSV ENDPOINT VS. JSON API.....	20
6.1.Feature Comparison Table	20
6.2.Pros and Cons	20
7. COMPLETE ENDPOINT REFERENCE	21
7.1.Load Endpoints	22
7.1.1. GET /load-detail – Retrieve Load Details.....	22
7.1.2. GET /load-sch – Retrieve Load Schedules	25
7.1.3. POST /load-bid – Submit Load Bids	29
7.1.4. POST /delete-load-bid – Delete Load Bids	35
7.2.Transaction Endpoints	38
7.2.1. GET /tran-sch – Retrieve Transaction Schedules	38
7.2.2. GET /tran-contract – Retrieve Transaction Contracts	41
7.2.3. POST /tran-bid – Submit Transaction Bids.....	44
7.2.4. POST /confirm-tran-bid – Confirm Transaction Bids	49
7.2.5. POST /delete-tran-bid – Delete Transaction Bids	52
7.3.Generator Endpoints.....	56
7.3.1. GET /gen-detail – Retrieve Generator Details	56
7.3.2. GET /gen-sch – Retrieve Generator Bids & Schedules	60
7.4.CSV Endpoint.....	69
7.4.1. POST /csv – Legacy CSV Upload/Download.....	69
APPENDIX A: FIELD VALIDATION REFERENCE	A

APPENDIX B: COMMON ERROR MESSAGESC

APPENDIX C: GLOSSARY.....E

Table of Figures

Figure 1: BUD UI Login screen17

Figure 2: BUD UI Home Screen showing the drag-and-drop upload area.....18

Figure 3: BUD UI Result page showing the raw BUD response.19

Revision History

Version	Effective Date	Revisions
1.0	08/27/2026	Initial Release

1. Important Usage Notes and Introduction

1.1. Important Usage Notes

The Bidding Upload/Download (BUD) system replaces the legacy Upload/Download Batch Procedures described in Section 8 of the NYISO Market Participant User's Guide (MPUG) for bidding data, available at

<https://www.nyiso.com/manuals-tech-bulletins-user-guides>

All Market Participants currently using the MIS Upload/Download Batch Procedures for bid submission, deletion, confirmation, and retrieval will need to transition to the BUD system.

The BUD system provides **three interactive methods for entering and receiving data into MIS**:

1. **REST JSON API** — A modern, programmatic interface for submitting, deleting, confirming, and retrieving bids using structured JSON payloads over HTTPS.
2. **CSV Endpoint** — A backward-compatible endpoint that accepts the same CSV template format used by the legacy MIS Upload/Download Batch Procedures (see MPUG Section 8).
3. **Web GUI** — A browser-based drag-and-drop interface for uploading CSV files.

The CSV endpoint is **backward-compatible** with existing MIS CSV templates as described in Section 8 of the Market Participant User's Guide (MPUG).

Market Participant Responsibility: Each Market Participant is responsible for entering bid information into the MIS accurately, completely, and in a timely fashion. Each Market Participant is responsible for the results, intended or otherwise, of its individual bidding strategies. See Section 1 of the Market Participant User's Guide (MPUG). for additional discussion of Market Participant Responsibilities and Controls available on the NYISO Web site at the following URL:

<https://www.nyiso.com/manuals-tech-bulletins-user-guides>

1.2. Introduction

The BUD system is the NYISO's API platform for managing electricity market bids. The BUD system supports three categories of market bidding operations:

- **Load Bids** — Submission, deletion, and retrieval of load (demand-side) bids, including physical load, virtual load, and virtual supply bids. Also supports retrieval of load unit details (load-detail) and load schedules (load-sch).

- **Transaction Bids** — Submission, deletion, confirmation, and retrieval of bilateral transaction (internal bilateral) bids, including Day-Ahead Market (DAM), Hour-Ahead Market (HAM), and Non-Firm transactions. Also supports retrieval of transaction schedules (tran-sch) and transaction contracts (tran-contract).
- **Generator Bids** — Submission, deletion, and retrieval of generator bids, including unit commitment data, for all generator types (utility, non-utility, quick-start, 30-minute start, curtailable, ESR, BTM, CSR). Also supports retrieval of generator details (gen-detail), DAM/HAM schedules (gen-sch), real-time dispatch schedules (gen-rtd), generator outages/deratings (gen-out), and unit commitment parameters (u-comm), as well as submission of unit commitment data (uc-data).

For information on obtaining a NYISO MIS user account, refer to Section 1 of the Market Participants User's Guide (MPUG). For information on obtaining and linking NAESB-compliant digital certificates, refer to Section 4 of the of the Market Participant User's Guide (MPUG). For full CSV upload/download template specifications, refer to Section 8 of the Market Participant Use's Guide (MPUG). The Market Participant User's Guide is available at on the NYISO website at:

<https://www.nyiso.com/manuals-tech-bulletins-user-guides>

1.3. Using the BUD APIs

1.3.1. General Usage Guidelines

- The BUD API is **stateless** — each request must include full authentication credentials (HTTP Basic Authentication header and a valid NAESB digital certificate).
- All timestamps in request payloads and query parameters must include a **timezone offset** (e.g., -05:00 for EST, -04:00 for EDT). Do not use bare UTC Z suffix.
- All POST submissions return a **structured response** containing accepted, rejected, failed validation, updated, deleted, and error details.
- The nyisoRequestId (UUID) is **auto-generated** by the system for every request for end-to-end request tracing and support correlation. Users do not need to provide this value — it is included automatically in every response.
- **Market windows** (bidding open/close times) are enforced by the system. Submitting a bid outside of the applicable market window will result in a rejection in the response body — not an HTTP-level error.
- **Production Base URL:** <https://updown-api.nyiso.com/updown/api/v1/>

- **Test Environment Base URL:** <https://updown-api-test.nyiso.com/> (when available)

1.3.2. Network Protocols

All BUD API communication uses **HTTPS 1.1 over TLS 1.2** (minimum). Connections that do not present a valid NAESB-compliant digital certificate will be rejected at the transport layer.

1.3.3. Location of BUD API Service Endpoints

Base URL: <https://updown-api.nyiso.com/updown/api/v1/>

All endpoint paths listed in this document are relative to the base URL above. For example, the full URL for the Load Schedules endpoint is:

<https://updown-api.nyiso.com/updown/api/v1/load-sch>

Endpoints are grouped by category:

Load Endpoints: - GET /load-detail - GET /load-sch - POST /load-bid - POST /delete-load-bid

Transaction Endpoints: - GET /tran-sch - GET /tran-contract - POST /tran-bid - POST /confirm-tran-bid - POST /delete-tran-bid

Generator Endpoints: - GET /gen-detail - GET /gen-sch - GET /gen-rtd - GET /gen-out - GET /u-comm - POST /gen-bid - POST /delete-gen-bid - POST /uc-data

CSV Endpoint: - POST /csv

2. Authentication

All BUD API requests require two forms of authentication:

- 1. HTTP Basic Authentication** — Every request must include an Authorization header containing a Base64-encoded username and password.
- 2. NAESB/NYISO Digital Certificate** — Every request must be made over a TLS connection that presents a valid NAESB-compliant digital certificate linked to the same MIS user account.

2.1. How to Obtain Access Credentials

- 1. Step 1:** Register for a NYISO MIS user account. Refer to Section 1 of the Market Participant User's Guide (MPUG) for registration procedures.
 - 2. Step 2:** Obtain a NAESB-compliant digital certificate from an authorized Certificate Authority. Refer to Section 4 of the Market Participant User's Guide (MPUG) for a list of approved providers and certificate requirements.
 - 3. Step 3:** Link the digital certificate to your MIS user account through the MIS certificate management interface. Refer to Section 4 of the Market Participant User's Guide (MPUG) for detailed instructions.
- The Market Participant User's Guide is available at on the NYISO website at:

<https://www.nyiso.com/manuals-tech-bulletins-user-guides>

2.2. HTTP Authorization Header Format

Authorization: Basic <base64(username:password)>

Example: For username EXAMPLE_USER and password MyP@ssw0rd:

Authorization: Basic RVhBTvBMRV9VU0VSOk15UEBzc3cwcmQ=

Important: NYISO recommends **not** including USERID or PASSWORD rows in BUD requests.

Embedding credentials in request payloads is unnecessary and poses a security risk. If these rows are present (e.g., in legacy CSV templates), they are silently ignored and authentication is always taken from the HTTP Authorization header only. This backward-compatibility behavior exists solely to support existing templates formatted per Market Participant User's Guide (MPUG) Section 8.

2.3. NAESB Digital Certificate (TLS Client Certificate)

In addition to the Authorization header, every request must present a valid NAESB-compliant digital certificate over TLS. The certificate file (.pem or .crt) and corresponding private key (.key) must be passed with every request.

curl syntax:

```
--cert /path/to/naesb-cert.pem --key /path/to/naesb-key.pem
```

HTTPIe syntax:

```
--cert /path/to/naesb-cert.pem --cert-key /path/to/naesb-key.pem
```

Note: If your certificate and private key are bundled in a single PKCS#12 (.p12) file, use `--cert /path/to/naesb-cert.p12` (curl will prompt for the passphrase, or use `--cert /path/to/naesb-cert.p12:<passphrase>`). All examples in this guide assume separate PEM files.

For detailed information on obtaining NAESB-compliant digital certificates, approved Certificate Authorities, certificate requirements, and linking certificates to your MIS user account, refer to Section 4 — Digital Certificates of the Market Participant User’s Guide (MPUG).

2.4. Security Roles and Endpoint Access

The BUD system enforces the same MIS user privileges and authorization flags as the existing MIS Upload/Download process. No new security roles are required.

Security notice: Specific security role names, privilege mappings, and endpoint-to-role authorization details are **not** published in this guide. Publishing such information externally would expose the internal authorization model and increase the attack surface for privilege-escalation attempts. The authoritative and complete reference on user account setup, MIS roles, organization roles, privileges, authorization flags, and digital certificate requirements can be found in Section 4 of the NYISO Market Participant User’s Guide, available from the NYISO Web site at the following URL:

<https://www.nyiso.com/manuals-tech-bulletins-user-guides>

2.5. Data Formatting

2.5.1. Formatting Numbers

- **MW values:** Integer or decimal values. Precision varies by field — generator MW values support up to 4 integer digits and 1 fractional digit (e.g., 100.5); load MW values support up to 6 integer digits with no fractional digits (e.g., 300).
- **Dollar values:** Decimal values with up to 2 fractional digits for cost fields (e.g., 999.99), or up to 4 fractional digits for fuel price (e.g., 30.1234).
- **PTID values:** Integer values with up to 6 digits (e.g., 123456).

2.5.2. Formatting Dates and Times

All date/time values in the BUD API use **ISO-8601 format with a timezone offset**:

yyyy-MM-ddTHH:mm-ZZ:ZZ

Examples: - Eastern Standard Time (EST): "2024-11-30T00:00-05:00" - Eastern Daylight Time (EDT): "2024-06-15T00:00-04:00"

25-Hour Day Handling (Fall-Back DST Transition):

On the day when Daylight Saving Time transitions to Standard Time (typically the first Sunday in November), the hour from 01:00 to 02:00 occurs twice. The BUD system handles this by: - The first occurrence of 02:00 uses the EDT offset (-04:00) - The second occurrence of 02:00 (the repeated hour) can be submitted using a bid time of 25:00 in CSV format, or by using the EST offset (-05:00) in JSON format

For example, on November 3, 2024: - First 2:00 AM: "2024-11-03T02:00-04:00" (EDT) - Second 2:00 AM: "2024-11-03T02:00-05:00" (EST)

Request vs. response time conventions. Request payloads and query parameters must include an explicit offset ($\pm$ HH:MM). Response payloads return bidDate values in **UTC** using the bare Z suffix (e.g., "2024-11-30T05:00:00Z"). Response timestamps on the outer envelope (requestTimestamp) use the format yyyy-MM-dd HH:mm:ss z (e.g., 2024-11-30 10:15:30 EST). Client applications that display bid times to users should convert the response UTC value back to Eastern Time using the applicable offset for the bid date.

2.5.3. Response requestType Echo Behavior

The requestType field is **not required** in the request body. The server determines the operation type from the endpoint URL and ignores any requestType value the client may include in submissionParameters. In the **response body**, the server echoes requestType in its lower-case **path form** that matches the endpoint URL (for example, load-bid, delete-load-bid, confirm-tran-bid, uc-data). Client applications that inspect requestType in the response should expect this lower-case hyphenated form.

2.5.4. Worked 25-Hour DST Example (JSON POST/load-bid)

The following /load-bid payload illustrates a complete 25-hour load bid submission for the fall-back DST transition day of **November 3, 2024**. Note the two entries at 02:00: the first uses the EDT offset (-04:00) and the second uses the EST offset (-05:00):

```
{
  "submissionParameters": {
    "doCommit": true,
    "processUnidentifiedBids": true
  }
}
```

```

    },
    "submitLoadBids": [
      { "loadName": "EXAMPLE_VL_NY-CITY", "bidDate": "2024-11-03T00:00-04:00", "priceCap1Mw": 10, "priceCap1MwDollar": 20.0 },
      { "loadName": "EXAMPLE_VL_NY-CITY", "bidDate": "2024-11-03T01:00-04:00", "priceCap1Mw": 10, "priceCap1MwDollar": 20.0 },
      { "loadName": "EXAMPLE_VL_NY-CITY", "bidDate": "2024-11-03T02:00-04:00", "priceCap1Mw": 10, "priceCap1MwDollar": 20.0 },
      { "loadName": "EXAMPLE_VL_NY-CITY", "bidDate": "2024-11-03T02:00-05:00", "priceCap1Mw": 10, "priceCap1MwDollar": 20.0 },
      { "loadName": "EXAMPLE_VL_NY-CITY", "bidDate": "2024-11-03T03:00-05:00", "priceCap1Mw": 10, "priceCap1MwDollar": 20.0 },
      { "loadName": "EXAMPLE_VL_NY-CITY", "bidDate": "2024-11-03T04:00-05:00", "priceCap1Mw": 10, "priceCap1MwDollar": 20.0 },
      { "loadName": "EXAMPLE_VL_NY-CITY", "bidDate": "2024-11-03T05:00-05:00", "priceCap1Mw": 10, "priceCap1MwDollar": 20.0 },
      { "loadName": "EXAMPLE_VL_NY-CITY", "bidDate": "2024-11-03T06:00-05:00", "priceCap1Mw": 10, "priceCap1MwDollar": 20.0 },
      { "loadName": "EXAMPLE_VL_NY-CITY", "bidDate": "2024-11-03T07:00-05:00", "priceCap1Mw": 10, "priceCap1MwDollar": 20.0 },
      { "loadName": "EXAMPLE_VL_NY-CITY", "bidDate": "2024-11-03T08:00-05:00", "priceCap1Mw": 10, "priceCap1MwDollar": 20.0 },
      { "loadName": "EXAMPLE_VL_NY-CITY", "bidDate": "2024-11-03T09:00-05:00", "priceCap1Mw": 10, "priceCap1MwDollar": 20.0 },
      { "loadName": "EXAMPLE_VL_NY-CITY", "bidDate": "2024-11-03T10:00-05:00", "priceCap1Mw": 10, "priceCap1MwDollar": 20.0 },
      { "loadName": "EXAMPLE_VL_NY-CITY", "bidDate": "2024-11-03T11:00-05:00", "priceCap1Mw": 10, "priceCap1MwDollar": 20.0 },
      { "loadName": "EXAMPLE_VL_NY-CITY", "bidDate": "2024-11-03T12:00-05:00", "priceCap1Mw": 10, "priceCap1MwDollar": 20.0 },
      { "loadName": "EXAMPLE_VL_NY-CITY", "bidDate": "2024-11-03T13:00-05:00", "priceCap1Mw": 10, "priceCap1MwDollar": 20.0 },
      { "loadName": "EXAMPLE_VL_NY-CITY", "bidDate": "2024-11-03T14:00-05:00", "priceCap1Mw": 10, "priceCap1MwDollar": 20.0 },
      { "loadName": "EXAMPLE_VL_NY-CITY", "bidDate": "2024-11-03T15:00-05:00", "priceCap1Mw": 10, "priceCap1MwDollar": 20.0 },
      { "loadName": "EXAMPLE_VL_NY-CITY", "bidDate": "2024-11-03T16:00-05:00", "priceCap1Mw": 10, "priceCap1MwDollar": 20.0 },
      { "loadName": "EXAMPLE_VL_NY-CITY", "bidDate": "2024-11-03T17:00-05:00", "priceCap1Mw": 10, "priceCap1MwDollar": 20.0 },
      { "loadName": "EXAMPLE_VL_NY-CITY", "bidDate": "2024-11-03T18:00-05:00", "priceCap1Mw": 10, "priceCap1MwDollar": 20.0 },
      { "loadName": "EXAMPLE_VL_NY-CITY", "bidDate": "2024-11-03T19:00-05:00", "priceCap1Mw": 10, "priceCap1MwDollar": 20.0 },
      { "loadName": "EXAMPLE_VL_NY-CITY", "bidDate": "2024-11-03T20:00-05:00", "priceCap1Mw": 10, "priceCap1MwDollar": 20.0 },
      { "loadName": "EXAMPLE_VL_NY-CITY", "bidDate": "2024-11-03T21:00-05:00", "priceCap1Mw": 10, "priceCap1MwDollar": 20.0 },
      { "loadName": "EXAMPLE_VL_NY-CITY", "bidDate": "2024-11-03T22:00-05:00", "priceCap1Mw": 10, "priceCap1MwDollar": 20.0 },
      { "loadName": "EXAMPLE_VL_NY-CITY", "bidDate": "2024-11-03T23:00-05:00", "priceCap1Mw": 10, "priceCap1MwDollar": 20.0 }
    ]
  }

```

In CSV format, the second 02:00 row is identified using hour 25:00 (see Market Participant User's Guide (MPUG) Section 8 available on the NYISO Website at <https://www.nyiso.com/manuals-tech-bulletins-user->

[guides](#).

EXAMPLE_VL_NY-CITY,11/03/2024 02:00,,,10,20,,,,,,,,

EXAMPLE_VL_NY-CITY,11/03/2024 25:00,,,10,20,,,,,,,,

The 23-hour **spring-forward** day (typically the second Sunday in March) omits the 02:00 hour entirely — do not submit a bid at 02:00 on that day.

2.5.5. JSON Payload Structure

All POST request payloads follow a consistent structure:

Note: JSON is an unordered format — the order of attributes in request and response objects may vary and should not be relied upon. Client applications must access fields by name, not by position.

```
{
  "submissionParameters": {
    "doCommit": true,
    "processUnidentifiedBids": true
  },
  "bids": [
    { /* bid object 1 */ },
    { /* bid object 2 */ }
  ]
}
```

Field	Type	Required	Description
submissionParameters.doCommit	Boolean	Yes	When true, bids that pass validation are committed to the market. When false, only validation is performed (dry run).
submissionParameters.processUnidentifiedBids	Boolean	No	When true, bids for units not directly associated with the submitting user's organization will still be processed. Default behavior varies by endpoint.
bids	Array	Yes	Array of bid objects. The field name varies by operation type (see individual endpoint documentation).

Note (request array naming): All eight POST submission endpoints accept a **canonical** request-body array named bids. Each endpoint also accepts an operation-named alias for that array (e.g., submitLoadBids, deleteLoadBids, submitTransactionBids, etc.) — either name is valid on input; **no other name is accepted**. POST submission responses do **not** echo the request array back; instead they return the

categorized result arrays accepted, rejected, updated, deleted, passedValidation, and failedValidation see [Section 2.6](#) below. Each endpoint section documents both names.

2.6. HTTP Status Codes for Responses

Code	Name	When It Occurs	Response Body
200	OK	Request completed successfully. For GET requests, the response body contains the requested data. For POST requests, the response body contains submission results including accepted, rejected, and validation details.	Structured JSON (see per-endpoint documentation)
400	Bad Request	Invalid query parameters, malformed request body, or field-level validation failures detected before processing.	JSON object with errors array containing descriptive error messages
401	Unauthorized	Missing or invalid credentials in the Authorization header, or the digital certificate is not valid.	Standard server error object(timestamp, status, error, path)
403	Forbidden	The authenticated user does not have the required security role to access the requested endpoint.	Standard server error object (timestamp, status, error, path)
500	InternalServerError	An unexpected system error occurred during processing.	JSON error object with error details

2.6.1. Standard 400 Bad Request Response Shape

For POST endpoints, per-field validation errors are returned in a validationErrors map keyed by "Bid Index: <n> <Unit>: <name>" with an array of error messages per key. For GET endpoints, a simple errors string array is returned.

```
{
  "errors": [
    "Bid date and time cannot be null"
  ],
  "validationErrors": {
    "Bid Index: 0 Load Name: EXAMPLE_VL_NY-CITY": [
      "Bid date and time cannot be null"
    ]
  }
}
```

2.6.2. Standard 401 Unauthorized Response Shape

```
{
  "timestamp": "2026-04-27T18:23:18.966+00:00",
  "status": 401,
  "error": "Unauthorized",
}
```

```
"path": "/updown/api/v1/tran-sch"  
}
```

Note: This is the standard server error response shape — it is **not** a standard BUD API response object. The path value reflects whichever endpoint was called.

Returned when (a) the Authorization header is missing or empty, (b) the username or password is invalid (MIS error UDF-10007), (c) the user's password has expired (UDF-10004), or (d) the presented digital certificate is missing, expired, or not linked to the submitting MIS user account.

2.6.3. Standard 403 Forbidden Response Shape

```
{  
  "timestamp": "2026-04-27T18:25:02.114+00:00",  
  "status": 403,  
  "error": "Forbidden",  
  "path": "/updown/api/v1/gen-bid"  
}
```

Note: Same standard server error shape as 401. The path value reflects whichever endpoint was called. Returned when the authenticated user is valid but lacks the required security role for the requested endpoint.

2.6.4. Standard 500 Internal Server Error Response Shape

```
{  
  "error": "Internal Server Error",  
  "message": "An unexpected error occurred. Please contact NYISO Stakeholder Service  
s and include the nyisoRequestId for tracing.",  
  "status": 500,  
  "nyisoRequestId": "b8c9doe1-f2a3-4567-bcde-789012345678"  
}
```

Every endpoint in this guide defers to the four shapes above for 400/401/403/500 responses. Per-endpoint response sections document only the endpoint-specific success (200) payload and any endpoint-specific validation error detail.

2.7. Interaction Methods Overview

The BUD system supports three methods for interacting with the bidding system:

Option 1: REST JSON API - Modern, programmatic interface using structured JSON payloads - Full per-field validation error reporting - Supports all current and future bid fields - Ideal for automated systems, custom integrations, and CI/CD pipelines - See [Section 3](#) for detailed usage

Option 2: CSV Legacy Endpoint - Backward-compatible with existing MIS Upload/Download CSV templates (Market Participant User's Guide (MPUG) Section 8) - POST CSV content to the /csv endpoint -

Response returned as CSV text in the same format as legacy MIS responses - Ideal for organizations with existing CSV-based automation - See [Section 4](#) for detailed usage

Option 3: Web GUI (BUD UI) - Browser-based drag-and-drop interface - Upload CSV files through a web browser - View results on-screen after upload - Ideal for manual, occasional bid submissions - See [Section 5](#) for detailed usage.

3. Option 1: REST JSON API

3.1. Overview & When to Use

The REST JSON API is the recommended method for programmatic interaction with the BUD system. It provides:

- **Structured JSON responses** with per-field validation error details
- **Full feature parity** with the latest market system capabilities
- **Modern tooling support** — easily consumed by any language or framework that supports HTTP/JSON
- **Granular error handling** — validation errors are returned per-bid with specific field-level messages

Use the REST JSON API when:

- Building automated bid submission systems
- Integrating with energy management systems (EMS)
- Developing custom market participant portals
- Requiring detailed, programmatic error handling

3.2. General Workflow

All BUD JSON endpoints share the same interaction pattern, regardless of the specific operation (submit, delete, confirm, or retrieve):

1. **Construct the request** — Build the JSON request body (POST endpoints) or query-parameter set (GET endpoints) per the contract for the chosen endpoint. See [Section 7](#) for the per-endpoint request schemas, field descriptions, and validation rules.
2. **Authenticate** — Include the Authorization: Basic <credentials> header (Base64-encoded username:password) and present a valid NAESB-compliant TLS client certificate. See [Section 2.3](#) for full authentication details.
3. **Send the request over HTTPS** — POST endpoints use Content-Type: application/json; GET endpoints use a query string (?param=value¶m2=value).
4. **Process the structured JSON response** — POST endpoints return classified outcome arrays (accepted, rejected, failedValidation, deleted); GET endpoints return data arrays appropriate to the request. See [Section 7](#) for the per-endpoint response schemas and examples.

3.3. Repeatability, Retries, and Error Handling

- **Network retries** — All BUD endpoints are safe to retry on connection-level failures (timeout, connection reset, 5xx response) provided the original request did not return an HTTP response body.
- **Application-level results** — Once an HTTP 200 OK response is returned, retrying the same request body may produce duplicate bids (for submission endpoints) or no-op outcomes (for delete endpoints already executed). Safe-retry behavior is the responsibility of the caller — track bid IDs returned from prior submissions to avoid resubmission.
- **Validation errors** — Returned in the failedValidation array of an HTTP 200 OK response. The original request was syntactically valid; only the rejected bids need correction and resubmission.
- **HTTP-level errors** — See [Section 2.6](#) for the full HTTP status code reference and [Section 2.4](#) for authentication-failure scenarios.

3.4. Operations-Specific Reference

For the request body, response schema, validation rules, and worked examples for each individual JSON endpoint (submit, delete, confirm, retrieve — for load, transaction, and generator markets), refer to the corresponding subsection of [Section 7](#):

- Load endpoints [Section 7.1](#)
- Transaction endpoints [Section 7.2](#)
- Generator endpoints [Section 7.3](#)

4. Option 2: CSV Legacy Endpoint

4.1. Overview & When to Use

The CSV endpoint provides a **direct replacement** for the legacy MIS Upload/Download Batch Procedures described in Section 8 of the Market Participant User's Guide (MPUG) available on the NYISO Website at <https://www.nyiso.com/manuals-tech-bulletins-user-guides>. It is intended for Market Participants who already have CSV-based automation in place and want to transition to the BUD system with minimal changes.

Use the CSV endpoint when:

- You have existing automation that generates CSV files in the MIS Upload/Download format
- You want a drop-in replacement with no changes to your CSV file structure
- You prefer text-based CSV responses over JSON

4.2. How the BUD CSV Endpoint Works

1. **POST** your CSV content to <https://updown-api.nyiso.com/updown/api/v1/csv> with Content-Type: text/csv
2. The BUD system accepts the same CSV template format as the legacy MIS Upload/Download Batch Procedures (Market Participant User's Guide (MPUG) Section 8)
3. The BUD system internally routes the CSV to the appropriate processing logic based on the BID_TYPE (for uploads) or QUERY_TYPE (for downloads) header row
4. Response is returned as CSV text in the same format as the legacy MIS response
5. **Authentication** uses HTTP Basic Auth (Authorization header) — the USERID and PASSWORD rows in the CSV are **ignored** by the BUD system

4.3. Supported CSV Upload Operations (BID_TYPE values)

BID_TYPE	Description
LOAD_BID	Submit load bids
DELETE_LOAD_BID	Delete load bids
GEN_BID	Submit generator bids
DELETE_GEN_BID	Delete generator bids
UC_DATA	Submit unit commitment data
TRAN_BID	Submit transaction bids
DELETE_TRAN_BID	Delete transaction bids
CONFIRM_TRAN_BID	Confirm transaction bids

For the complete field definitions, data types, and submission/response column order for each upload template, refer to Section 8 of the Market Participant User's Guide (MPUG) available on the NYISO Website at <https://www.nyiso.com/manuals-tech-bulletins-user-guides>.

4.4. Supported CSV Download Operations (QUERY_TYPE values)

QUERY_TYPE	Description
LOAD_DETAIL	Retrieve load unit details
LOAD_SCH	Retrieve load bid schedules
GEN_DETAIL	Retrieve generator unit details
GEN_SCH	Retrieve generator bids and schedules
GEN_RTD	Retrieve generator real-time dispatch schedules
GEN_OUT	Retrieve generator outage declarations
U_COMM	Retrieve unit commitment parameters
TRAN_SCH	Retrieve transaction bid schedules
TRAN_CONTRACT	Retrieve transaction contracts

For the complete field definitions and response column order for each download template, refer to Section 8 Market Participant User's Guide (MPUG).

4.5. CSV Template Format Overview

CSV upload templates use the following header structure:

```

BID_TYPE=&
USERID=&
PASSWORD=&
DATA_ROWS=&
<data row 1>
<data row 2>
...

```

CSV download templates use:

```

QUERY_TYPE=&
USERID=&
PASSWORD=&

```

Data commit logic: The BUD system performs all-or-nothing initial validation on the submitted data. Business rule validation is performed in 250-row batches.

Note: USERID and PASSWORD rows may be included for backward compatibility with the legacy format but are **not required** and are **ignored** by the BUD system. Authentication is always performed via the HTTP Authorization header.

For complete template format specifications, upload/download process procedures, and field-level data dictionaries, refer to Section 8 of the Market Participant User's Guide (MPUG) available on the NYISO Website at <https://www.nyiso.com/manuals-tech-bulletins-user-guides>.

4.6. Worked Examples

For complete CSV request-body examples for every supported BID_TYPE (upload) and QUERY_TYPE (download), including the 25-hour fall-back DST example, refer to [Section 7.4.1](#) (Request Examples) under the /csv endpoint reference in [Section 7](#) — Complete Endpoint Reference.

5. Option 3: Web GUI (BUD UI)

5.1. Overview & When to Use

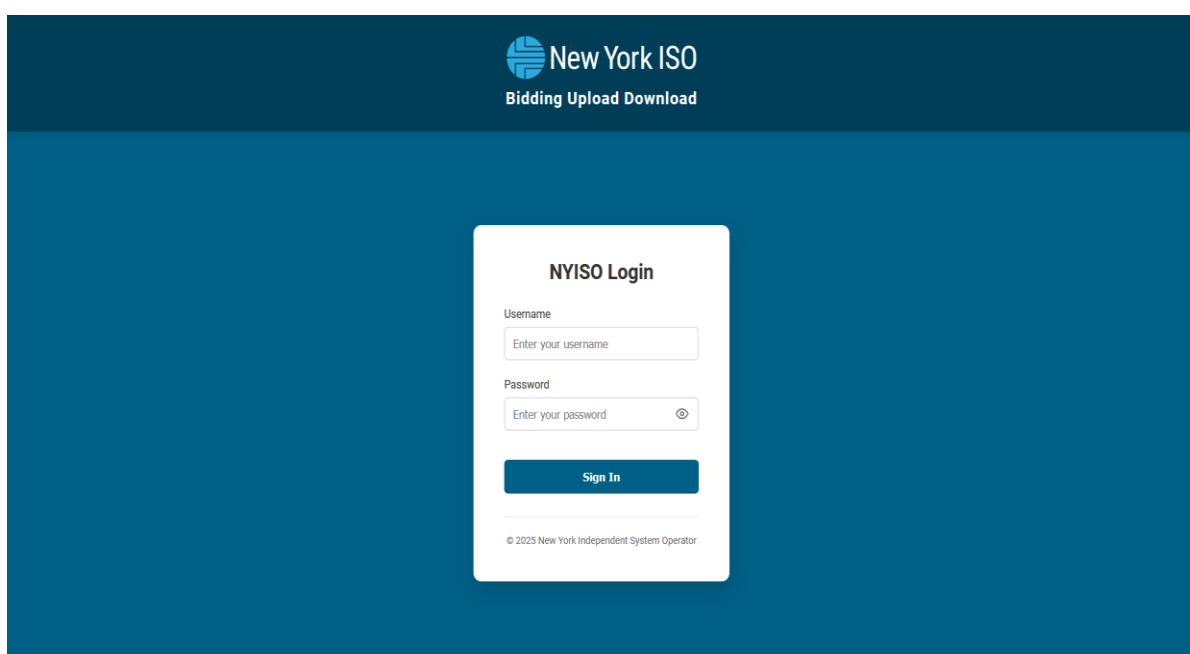
The BUD Web GUI provides a browser-based interface for uploading CSV files. It is designed for manual, occasional bid submissions by Market Participants who prefer a visual interface.

5.2. Accessing the BUD UI

URL: <https://updown-api.nyiso.com/login>

1. Navigate to the BUD UI URL in your web browser
2. Enter your NYISO MIS username and password on the login screen
3. Your browser must present a valid NAESB digital certificate during the TLS handshake

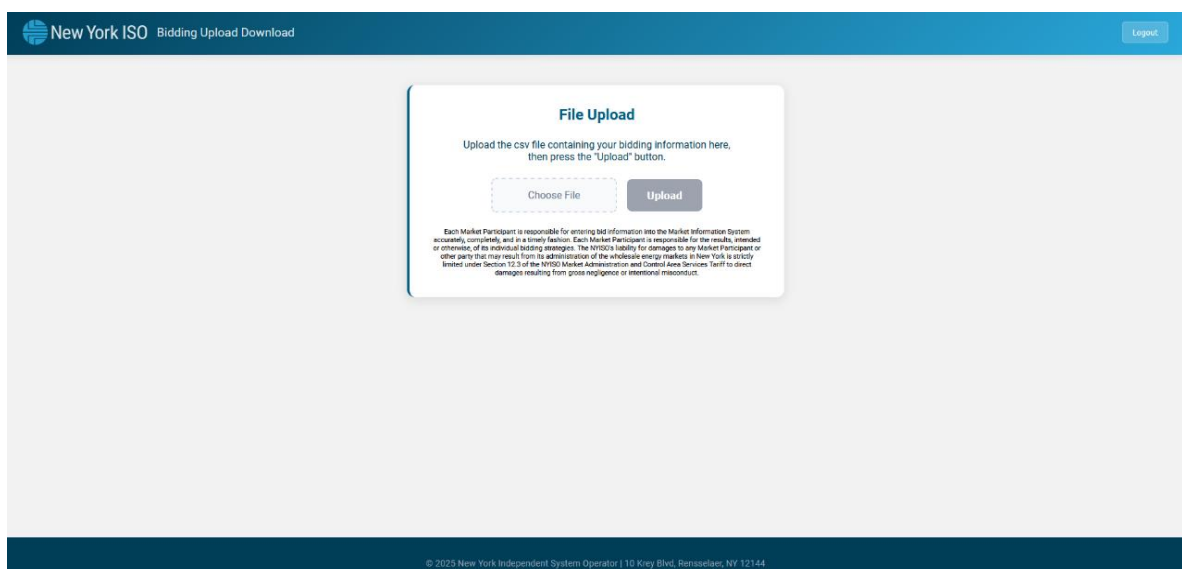
Figure 1: BUD UI Login screen



5.3. Uploading a CSV File

1. **Log in** to the BUD UI with your MIS credentials
2. **Select a file** by either:
 - Dragging and dropping a CSV file onto the upload area, OR
 - Clicking “Choose File” and selecting a CSV file from your file system
3. **Click “Upload”** to submit the file
4. **View results** on the Result page that appears after processing

Figure 2: BUD UI Home Screen showing the drag-and-drop upload area



5.4. Accepted File Types

The BUD UI accepts .csv files only. The CSV file must follow the standard MIS template format (see [Section 4](#) of this guide or Section 8 of the Market Participant User’s Guide (MPUG) available on the NYISO Website at <https://www.nyiso.com/manuals-tech-bulletins-user-guides>).

5.5. Interpreting Results

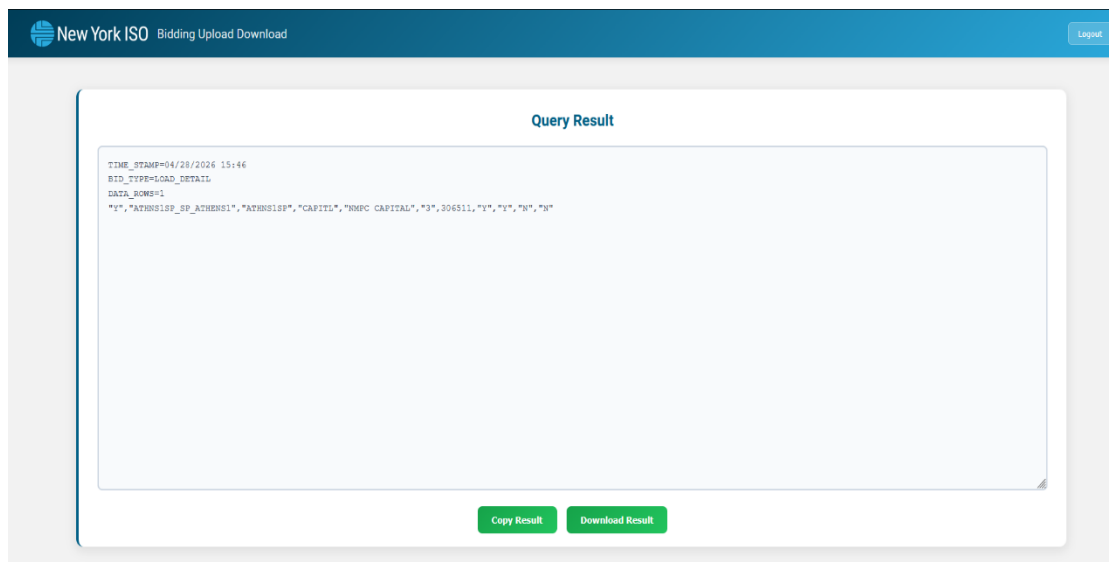
After uploading, the Result page displays:

- The raw response from the BUD system
- Success or failure status for each bid in the file
- Specific error messages for any bids that failed validation or were rejected

Result page actions:

Button	Description
Copy Result	Copies the result text to your clipboard, enabling you to paste it into any application
Download Result	Downloads the result as a CSV file

Figure 3: BUD UI Result page showing the raw BUD response.



Note: The CSV response format from the GUI will not include fields for features added in future releases. For full-featured responses, use the REST JSON API.

6. Comparison: CSV Endpoint vs. JSON API

6.1. Feature Comparison Table

Feature	JSON REST API	CSV Endpoint
Data Format	JSON	CSV (legacy MIS format)
Error Granularity	Per-field validation errors with field names	Row-level error messages
Automation	Full REST client support (any language)	Requires CSV file generation
New Features	All current and future features supported	May not include newest fields
Backward Compatibility	N/A (new interface)	Full legacy MIS Upload/Download compatibility
Response Format	Structured JSON with typed fields	CSV text
GUI Support	N/A	Supported via drag-and-drop upload
Authentication	HTTP Basic Auth + Certificate	HTTP Basic Auth + Certificate
Batch Size	Limited by request body size	Limited by request body size

6.2. Pros and Cons

JSON REST API:

- **✓ Pros:** Granular per-field error handling, structured responses, modern tooling support, full feature parity with latest market capabilities, easy to parse and process programmatically
- **✗ Cons:** Requires JSON payload construction, new integration effort for existing CSV-based systems

CSV Legacy Endpoint:

- **✓ Pros:** Drop-in replacement for the legacy MIS Upload/Download procedures, familiar format for existing users, GUI support for manual uploads, no changes needed to existing CSV generation logic
- **✗ Cons:** Limited error detail compared to JSON, may lag on new features, text-based responses require custom parsing

7. Complete Endpoint Reference

Load Endpoint Table

Category	Method	Path	Full URL	Description
Load	GET	/load-detail	https://updown-api.nyiso.com/updown/api/v1/load-detail	Retrieve load bus details
Load	GET	/load-sch	https://updown-api.nyiso.com/updown/api/v1/load-sch	Retrieve load bids and schedules
Load	POST	/load-bid	https://updown-api.nyiso.com/updown/api/v1/load-bid	Submit load bids
Load	POST	/delete-load-bid	https://updown-api.nyiso.com/updown/api/v1/delete-load-bid	Delete load bids
Transaction	GET	/tran-sch	https://updown-api.nyiso.com/updown/api/v1/tran-sch	Retrieve internal transaction bids and schedules
Transaction	GET	/tran-contract	https://updown-api.nyiso.com/updown/api/v1/tran-contract	Retrieve transaction contracts
Transaction	POST	/tran-bid	https://updown-api.nyiso.com/updown/api/v1/tran-bid	Submit internal transaction bids
Transaction	POST	/confirm-tran-bid	https://updown-api.nyiso.com/updown/api/v1/confirm-tran-bid	Confirm transaction bids
Transaction	POST	/delete-tran-bid	https://updown-api.nyiso.com/updown/api/v1/delete-tran-bid	Delete transaction bids
Generator	GET	/gen-detail	https://updown-api.nyiso.com/updown/api/v1/gen-detail	Retrieve generator details
Generator	GET	/gen-sch	https://updown-api.nyiso.com/updown/api/v1/gen-sch	Retrieve generator bids and schedules
Generator	GET	/gen-rtd	https://updown-api.nyiso.com/updown/api/v1/gen-rtd	Retrieve generator RTD schedules
Generator	GET	/gen-out	https://updown-api.nyiso.com/updown/api/v1/gen-out	Retrieve generator outages/deratings

Category	Method	Path	Full URL	Description
Generator	GET	/u-comm	https://updown-api.nyiso.com/updown/api/v1/u-comm	Retrieve generator commitment parameters
Generator	POST	/gen-bid	https://updown-api.nyiso.com/updown/api/v1/gen-bid	Submit generator bids
Generator	POST	/delete-gen-bid	https://updownapi.nyiso.com/updown/api/v1/delete-gen-bid	Delete generator bids
Generator	POST	/uc-data	https://updown-api.nyiso.com/updown/api/v1/uc-data	Submit generator commitment parameters
CSV	POST	/csv	https://updown-api.nyiso.com/updown/api/v1/csv	Legacy CSV upload/download for all bid types

7.1. Load Endpoints

7.1.1. GET /load-detail – Retrieve Load Details

Request

HTTP Method + Full URL:

GET <https://updown-api.nyiso.com/updown/api/v1/load-detail>

Description: Retrieves detailed information about load units, including load name, PTID, station, voltage class, zone, subzone, and qualification flags (energy bid qualified, price cap qualified, interruptible). Results can be filtered by load name or PTID.

Required Authentication:

Requirement	Value / Format	Description
Authorization Header	Basic <base64(username:password)>	Required. HTTP Basic Authentication credentials.
TLS Client Certificate	NAESB-compliant.pem/.pfx certificate	Required. Valid NAESB digital certificate linked to the same MIS user account (see Section 2.3).

Query Parameters:

Name	Type	Required	Description	Example
loadName	String	No	Name of the load unit. Must contain only alphanumeric characters, underscores, hyphens, and spaces.	EXAMPLE_LOAD_NAME
Ptid	Integer	No	PTID (Point Identifier) of the load, up to 6 digits.	546231

Note: If neither loadName nor ptid is provided, all load details accessible to the authenticated user are returned.

Request Examples

curl:

```
curl -X GET "https://updown-api.nyiso.com/updown/api/v1/load-detail?loadName=EXAMPLE_VL_NY-CITY" \
-H "Authorization: Basic <base64-encoded-credentials>" \
--cert /path/to/naesb-cert.pem --key /path/to/naesb-key.pem
```

HTTPIe:

```
http GET "https://updown-api.nyiso.com/updown/api/v1/load-detail" \
"Authorization:Basic <base64-encoded-credentials>" \
--cert /path/to/naesb-cert.pem --cert-key /path/to/naesb-key.pem \
loadName==EXAMPLE_VL_NY-CITY
```

curl (by PTID):

```
curl -X GET "https://updown-api.nyiso.com/updown/api/v1/load-detail?ptid=546231" \
-H "Authorization: Basic <base64-encoded-credentials>" \
--cert /path/to/naesb-cert.pem --key /path/to/naesb-key.pem
```

Response

200 OK — Success:

Field	Type	Description
requestStartTime	String	ISO-8601 timestamp when request processing began
requestEndTime	String	ISO-8601 timestamp when request processing completed
nyisoRequestId	String	Auto-generated unique request correlation ID (UUID)
Count	Integer	Number of records returned

Field	Type	Description
Message	String	Status message (e.g., "Success")
loadDetails	Array	Array of load detail objects
loadDetails[].ptid	Integer	PTID of the load
loadDetails[].active	Boolean	Whether the load is currently active
loadDetails[].energyBidQualified	Boolean	Whether the load is qualified for energy bids
loadDetails[].priceCapQualified	Boolean	Whether the load is qualified for price cap bids
loadDetails[].interruptible10Minute	Boolean	Whether the load is qualified for 10-minute interruption
loadDetails[].interruptible30Minute	Boolean	Whether the load is qualified for 30-minute interruption
loadDetails[].edcArea	String	Electric distribution company area code
loadDetails[].lse	Object	Load serving entity
loadDetails[].lse.lseName	String	LSE name
loadDetails[].subzone	Object	Subzone information
loadDetails[].subzone.subzoneName	String	Subzone name
loadDetails[].subzone.zone	Object	Zone information
loadDetails[].subzone.zone.zoneName	String	Zone name
loadDetails[].displayName	String	Full display name (e.g., "EXMPLSE_SP_LOADBUS1")

400 Bad Request — Validation Error:

Field	Type	Description
Errors	Array	Array of error message strings describing what failed

401 Unauthorized / 403 Forbidden: See [Section 2.6.2](#) and [Section 2.6.3](#) for the standard server error shape (timestamp, status, error, path).

Request Examples

200 OK — Success:

```
{
  "requestStartTime": "2026-04-27T19:10:36.343404842Z",
  "requestEndTime": "2026-04-27T19:10:36.345389594Z",
  "nyisoRequestId": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
  "count": 1,
  "message": "Success",
}
```

```
"loadDetails": [
  {
    "ptid": 100001,
    "active": true,
    "energyBidQualified": true,
    "priceCapQualified": true,
    "interruptible10Minute": false,
    "interruptible30Minute": false,
    "edcArea": "3",
    "lse": {
      "lseName": "EXMPLSE"
    },
    "subzone": {
      "subzoneName": "EXAMPLE SUBZONE",
      "zone": {
        "zoneName": "EXZONE"
      }
    },
    "displayName": "EXMPLSE_SP_LOADBUS1"
  }
]
```

200 OK – Validation Error (unrecognized load bus):

```
{
  "requestStartTime": "2026-04-28T12:20:57.020226649Z",
  "requestEndTime": "2026-04-28T12:20:57.020228308Z",
  "count": 0,
  "message": "Upload/Download Error",
  "errors": [
    "ERROR: Submitted load bus EXMPLSE_SP_LOADBUS could not be identified. The Bus either does not exist, the user is not an authorized user of the Bus or the Bus is associated with a Load Type that is not authorized to display Load Bus Details. Contact NYISO if you feel you received this message in error."
  ]
}
```

Note: GET endpoints return validation errors as a 200 OK with "message": "Upload/Download Error" and count: 0, not as a 400 Bad Request. The startDateTime query parameter is required for /tran-sch.

7.1.2. GET /load-sch – Retrieve Load Schedules

Request

HTTP Method + Full URL:

GET <https://updown-api.nyiso.com/updown/api/v1/load-sch>

Description: Retrieves load bid schedules for the authenticated user's organization. Results can be

filtered by date/time, number of hours, load name, PTID, bid status, and last modified date.

Required Authentication:

Requirement	Value / Format	Description
Authorization Header	Basic <base64(username:password)>	Required. HTTP Basic Authentication credentials.
TLS Client Certificate	NAESB-compliant .pem/.pfx certificate	Required. Valid NAESB digital certificate linked to the same MIS user account (see Section 2.3).

Query Parameters:

Name	Type	Required	Description	Allowed Values	Example
startDateTime	String	Yes	Start date/time in ISO-8601 format with timezone offset	ISO-8601 with offset	2024-11-30T00:00-05:00
numHours	Integer	No	Number of hours of data to retrieve	1—24 (25 on fall-back DST days)	6
loadName	String	No	Name of the load unit	Alphanumeric + _-	EXAMPLE_VL_NY-CITY
Ptid	Integer	No	PTID of the load (up to 6 digits)	Numeric	546231
Status	String	No	Filter by bid status	WAITING VALIDATION, VALIDATION FAILED, WAITING CONFIRMATION, VALIDATION PASSED, EVALUATING, BID REJECTED, ADVISORY REJECTED, BID ACCEPTED, ADVISORY ACCEPTED, MODIFIED, PRE SCHED QUEUED, PRE SCHED REJECTED, PRE SCHED CONFIRMED, PRE SCH MRKT CLOSED	BID ACCEPTED
modifiedDate	String	No	Filter by last modified date/time	ISO-8601 with offset	2024-11-30T00:00-05:00

Request Examples

curl:

```
curl -X GET "https://updown-api.nyiso.com/updown/api/v1/load-sch?startDateTime=2024-11-30T00:00-05:00&numHours=6" \
-H "Authorization: Basic <base64-encoded-credentials>" \
--cert /path/to/naesb-cert.pem --key /path/to/naesb-key.pem
```

HTTPIe:

```
http GET "https://updown-api.nyiso.com/updown/api/v1/load-sch" \
"Authorization:Basic <base64-encoded-credentials>" \
--cert /path/to/naesb-cert.pem --cert-key /path/to/naesb-key.pem \
startDateTime=="2024-11-30T00:00-05:00" numHours==6
```

Responses

200 OK — Success:

Field	Type	Description
requestStartTime	String	ISO-8601 timestamp when request processing began
RequestEndTime	String	ISO-8601 timestamp when request processing completed
NyisoRequestId	String	Auto-generated request correlation ID (UUID)
Count	Integer	Number of records returned
Message	String	Status message (e.g., "Success")
LoadSchedules	Array	Array of load schedule objects
loadSchedules[].bidDate	String	Bid hour in UTC (e.g., "2026-04-10T05:00Z")
loadSchedules[].displayName	String	Full display name (e.g., "EXMPLSE_VL_LOADBUS2")
loadSchedules[].ptid	Integer	Load PTID
loadSchedules[].bidId	Integer	Unique bid ID assigned by the system
loadSchedules[].forecastMw	Number	Forecasted MW to be consumed
loadSchedules[].fixedBidMw	Number	Fixed-block DAM MW
loadSchedules[].priceCap1Mw	Number	Price-cap MW block 1
loadSchedules[].priceCap1MwDollar	Number	Price-cap \$/MWh for block 1

Field	Type	Description
loadSchedules[].priceCap2Mw	Number	Price-cap MW block 2
loadSchedules[].priceCap2MwDollar	Number	Price-cap \$/MWh for block 2
loadSchedules[].priceCap3Mw	Number	Price-cap MW block 3
loadSchedules[].priceCap3MwDollar	Number	Price-cap \$/MWh for block 3
loadSchedules[].interruptType	String	"10MIN" or "30MIN"
loadSchedules[].interruptFixedMw	Number	Interruptible fixed MW
loadSchedules[].interruptFixedMwDollar	Number	Interruptible fixed \$/MW
loadSchedules[].interruptPriceCapMw	Number	Interruptible price-cap MW
loadSchedules[].interruptPriceCapMwDollar	Number	Interruptible price-cap \$/MW
loadSchedules[].scheduledPriceCapMw	Number	Scheduled energy associated with price-cappedbid(MW)
loadSchedules[].scheduledInterruptFixedMw	Number	Scheduled interruptible fixed MW
loadSchedules[].scheduledInterruptCapMw	Number	Scheduled interruptible price-cap MW
loadSchedules[].bidStatus	String	Bid status (e.g., "VALIDATION_PASSED", "BID_ACCEPTED")
loadSchedules[].passedValidationMessages	Array	Validation messages for passed bids
loadSchedules[].passedValidationMessages[].message	String	Validation message text
loadSchedules[].validationMessages	Array	Validation messages providing bid details
loadSchedules[].validationMessages[].message	String	Validation message text

Note: Only fields with non-null values appear in the response. The actual fields returned depend on the bid type and status.

400 Bad Request: See [Section 2.6.1](#)

401 / 403 / 500: See [Section 2.6.2](#) - [Section 2.6.4](#)

Response Examples

200 OK — Success:

```
{
  "requestStartTime": "2026-04-27T19:41:54.209306600Z",
  "requestEndTime": "2026-04-27T19:41:54.213302300Z",
  "nyisoRequestId": "b2c3d4e5-f6a7-8901-bcde-f12345678901",
  "count": 2,
```

```

"message": "Success",
"loadSchedules": [
  {
    "bidDate": "2026-04-10T05:00Z",
    "displayName": "EXMPLSE_VL_LOADBUS2",
    "ptid": 100002,
    "bidId": 1000000001,
    "priceCap1Mw": 1,
    "priceCap1MwDollar": 1.0,
    "bidStatus": "VALIDATION_PASSED",
    "passedValidationMessages": [
      {
        "message": "Validation Passed Subject to Credit Evaluation."
      }
    ]
  },
  {
    "bidDate": "2026-04-10T06:00Z",
    "displayName": "EXMPLSE_VL_LOADBUS2",
    "ptid": 100002,
    "bidId": 1000000003,
    "priceCap1Mw": 1,
    "priceCap1MwDollar": 1.0,
    "bidStatus": "VALIDATION_PASSED",
    "passedValidationMessages": [
      {
        "message": "Validation Passed Subject to Credit Evaluation."
      }
    ]
  }
]
}

```

7.1.3. POST /load-bid – Submit Load Bids

Request

HTTP Method + Full URL:

POST <https://updown-api.nyiso.com/updown/api/v1/load-bid>

Description: Submits one or more load bids for processing. Each bid represents a single hour of load bidding data for a specific load unit. The system validates all bids before committing. If doCommit is true and all bids pass validation, they are committed to the market. Bids submitted outside the applicable market window will be rejected.

Required Authentication:

Requirement	Value / Format	Description
Authorization Header	Basic <base64(username:password)>	Required. HTTP Basic Authentication credentials.
Content-Type Header	application/json	Required. Request body format.
TLS Client Certificate	NAESB-compliant .pem/.pfx certificate	Required. Valid NAESB digital certificate linked to the same MIS user account (see Section 2.3)

Request Body Schema:

Array name: The canonical request-body array for this endpoint is bids; submitLoadBids is also accepted on input as an alias. Either name works — examples below use submitLoadBids.

Field	Type	Required	Description	Constraints	Example
submissionParameters.doCommit	Boolean	Yes	Commit bids if validation passes	true or false	true
submissionParameters.processUndertifiedBids	Boolean	No	Process bids for units not in user's org	true or false	true
submitLoadBids (alias: bids)	Array	Yes	Array of load bid objects	At least 1 entry	See below
submitLoadBids[].loadName	String	Conditional	Load name (required if ptid not provided)	Alphanumeric + _-	"EXAMPLE_VL_NY-CITY"
submitLoadBids[].ptid	Integer	Conditional	PTID (required if loadName not provided)	Up to 6 digits	546231
submitLoadBids[].bidDate	String	Yes	Bid date/time in ISO-8601 with timezone	Must be top of hour	"2024-11-30T00:00-05:00"
submitLoadBids[].forecastMw	Number	No	Forecasted MW to be consumed	Up to 6 integer digits	300

Field	Type	Required	Description	Constraints	Example
submitLoadBids[].fixedBidMw	Number	No	Fixed MW to procure from DAM	Up to 6 integer digits	300
submitLoadBids[].priceCap1Mw	Number	No	First price cap MW block	Up to 6 integer digits	350
submitLoadBids[].priceCap1MwDollar	Number	No	Dollar per MW for first price cap	Up to 6 integer digits	20.0
submitLoadBids[].priceCap2Mw	Number	No	Second price cap MW block	Up to 6 integer digits	450
submitLoadBids[].priceCap2MwDollar	Number	No	Dollar per MW for second price cap	Up to 6 integer digits	25.0
submitLoadBids[].priceCap3Mw	Number	No	Third price cap MW block	Up to 6 integer digits	500
submitLoadBids[].priceCap3MwDollar	Number	No	Dollar per MW for third price cap	Up to 6 integer digits	30.0
submitLoadBids[].interruptType	String	No	Interruption type	10MIN or 30MIN	"10MIN"
submitLoadBids[].interruptFixedMw	Number	No	Interruptible fixed MW	Up to 6 int, 1 frac	20.0
submitLoadBids[].interruptFixedMwDollar	Number	No	Cost for interruptible fixed MW	Up to 6 integer digits	250.0
submitLoadBids[].interruptPriceCapMw	Number	No	Interruptible price cap MW	Up to 6 int, 1 frac	10.0
submitLoadBids[].interruptPriceCapMwDollar	Number	No	Cost for interruptible price cap MW	Up to 6 integer digits	400.0

Note: Either loadName or ptid (or both) must be provided for each bid. Price cap pairs (MW and Dollar) must both be provided or both be null. Price cap values cannot be negative. The fields above are the

complete set supported for JSON submission; numHours, energyBidMw, fixedMinGenMw, and additional interruptible price-cap pairs are not supported in the current JSON API; submit one row per hour and use the single interruptPriceCapMw / interruptPriceCapMwDollar pair.

Request Examples

curl — Single bid:

```
curl -X POST "https://updown-api.nyiso.com/updown/api/v1/load-bid" \
-H "Authorization: Basic <base64-encoded-credentials>" \
-H "Content-Type: application/json" \
--cert /path/to/naesb-cert.pem --key /path/to/naesb-key.pem \
-d '{
  "submissionParameters": {
    "doCommit": true,
    "processUnidentifiedBids": true
  },
  "submitLoadBids": [{
    "loadName": "EXAMPLE_VL_NY-CITY",
    "bidDate": "2024-11-30T00:00-05:00",
    "priceCap1Mw": 10,
    "priceCap1MwDollar": 10.0
  }]
}'
```

HTTPIe — Single bid:

```
http POST "https://updown-api.nyiso.com/updown/api/v1/load-bid" \
  "Authorization:Basic <base64-encoded-credentials>" \
  "Content-Type:application/json" \
  --cert /path/to/naesb-cert.pem --cert-key /path/to/naesb-key.pem \
  < load-bid-body.json
```

curl — Multi-bid (4 hours):

```
curl -X POST "https://updown-api.nyiso.com/updown/api/v1/load-bid" \
-H "Authorization: Basic <base64-encoded-credentials>" \
-H "Content-Type: application/json" \
--cert /path/to/naesb-cert.pem --key /path/to/naesb-key.pem \
-d '{
  "submissionParameters": {
    "doCommit": true,
    "processUnidentifiedBids": true
  },
  "submitLoadBids": [
    {
      "loadName": "EXAMPLE_VL_NY-CITY",
      "bidDate": "2024-09-30T00:00-04:00",
      "priceCap1Mw": 10,
      "priceCap1MwDollar": 10.0
    },
  ],
}
```

```
{
  "loadName": "EXAMPLE_VL_NY-CITY2",
  "bidDate": "2024-09-30T01:00-04:00",
  "priceCap1Mw": 10,
  "priceCap1MwDollar": 10.0
},
{
  "loadName": "EXAMPLE_VL_NY-CITY3",
  "bidDate": "2024-09-30T02:00-04:00",
  "priceCap1Mw": 10,
  "priceCap1MwDollar": 10.0
},
{
  "loadName": "EXAMPLE_VL_NY-CITY",
  "bidDate": "2024-09-30T03:00-04:00",
  "priceCap1Mw": 10,
  "priceCap1MwDollar": 10.0
}
]
```

Response

200 OK — Submission Result:

Field	Type	Description
requestTimestamp	String	Timestamp of the request (yyyy-MM-dd HH:mm:ss z)
submissionParameters	Object	Echo of the submission parameters
nyisoRequestId	String	Auto-generated unique request correlation ID (UUID) for tracing and support
Errors	Array	Global error messages (empty on success)
validationErrors	Object	Map of bid key -> array of per-field error messages
failedValidation	Array	Array of bid objects that failed validation
passedValidation	Array	Array of bid objects that passed validation
Accepted	Array	Array of bid objects accepted by the market
Rejected	Array	Array of bid objects rejected by the market

400 Bad Request: See [Section 2.6.1](#) for the canonical errors[] + validationErrors{} shape.

validationErrors is keyed by "Bid Index: <n> Load Name: <name>".

401 / 403 / 500: See [Section 2.6.2](#) – [Section 2.6.4](#)

Response Examples

200 OK — Bids accepted and rejected:

```
{
  "requestTimestamp": "2026-04-27 15:51:01 EDT",
  "submissionParameters": {
    "requestType": "load-bid",
    "doCommit": true,
    "processUnidentifiedBids": true,
    "nyisoRequestId": "c3d4e5f6-a7b8-9012-cdef-234567890123",
    "fingerprint": "a1b2c3d4e5f6a7b8c9d0e1f2a3b4c5d6e7f8a9b0",
    "caller": "JSMITH"
  },
  "requestStartTime": "2026-04-27T19:51:01.451228600Z",
  "count": 8,
  "accepted": [
    {
      "bidDate": "2026-05-01T15:00Z",
      "displayName": "EXMPLSE_PA_LOADBUS3",
      "ptid": 100003,
      "bidId": 1000000005,
      "bidStatus": "VALIDATION_PASSED",
      "forecastMw": 170,
      "fixedBidMw": 1
    },
    {
      "bidDate": "2026-05-01T16:00Z",
      "displayName": "EXMPLSE_PA_LOADBUS3",
      "ptid": 100003,
      "bidId": 1000000009,
      "bidStatus": "VALIDATION_PASSED",
      "forecastMw": 170,
      "fixedBidMw": 1
    }
  ],
  "rejected": [
    {
      "bidDate": "2026-05-01T04:00Z",
      "displayName": "EXMPLSE_PA_LOADBUS3",
      "ptid": 100003,
      "bidId": 1000000007,
      "bidStatus": "VALIDATION_PASSED",
      "forecastMw": 170,
      "fixedBidMw": 1,
      "validationMessages": [
        {
          "message": "Bid found with no updates. No action taken."
        }
      ]
    }
  ]
}
```

Note: The accepted[] array above is truncated for brevity — the actual response contained 7 accepted

bids. Only non-null fields appear per bid. The rejected[] array includes validationMessages explaining why each bid was rejected.

7.1.4. POST /delete-load-bid – Delete Load Bids

Request

HTTP Method + Full URL:

POST <https://updown-api.nyiso.com/updown/api/v1/delete-load-bid>

Description: Deletes one or more previously submitted load bids. Bids can be identified either by bidId alone, or by a combination of loadName (or ptid) and bidDate.

Required Authentication:

Requirement	Value / Format	Description
Authorization Header	Basic <base64(username:password)>	Required.
Content-Type Header	application/json	Required.
TLS Client Certificate	NAESB-compliant .pem/.pfx certificate	Required. Valid NAESB digital certificate (see Section 2.3).

Request Body Schema:

Array name: The canonical request-body array for this endpoint is bids; deleteLoadBids is also accepted on input as an alias. Either name works — examples below use deleteLoadBids.

Field	Type	Required	Description	Example
submissionParameters.doCommit	Boolean	Yes	Commit deletion	true
deleteLoadBids (alias: bids)	Array	Yes	Array of bid delete objects	See below
deleteLoadBids[].bidId	Integer	Conditional	Bid ID to delete (sufficient alone)	1174702467
deleteLoadBids[].bidDate	String	Conditional	Bid date (required if no bidId)	"2024-11-30T00:00-05:00"

Field	Type	Required	Description	Example
deleteLoadBids[].loadName	String	Conditional	Load name (required if no bidId)	"EXAMPLE_VS_NY-CITY"
deleteLoadBids[].ptid	Integer	Conditional	PTID (alternative to loadName)	546231

Request Examples

curl — Delete by loadName and bidDate:

```
curl -X POST "https://updown-api.nyiso.com/updown/api/v1/delete-load-bid" \
  -H "Authorization: Basic <base64-encoded-credentials>" \
  -H "Content-Type: application/json" \
  --cert /path/to/naesb-cert.pem --key /path/to/naesb-key.pem \
  -d '{
    "submissionParameters": {
      "doCommit": true,
      "processUnidentifiedBids": false
    },
    "deleteLoadBids": [{
      "bidDate": "2026-02-14T01:00:00-05:00",
      "loadName": "EXAMPLE_VS_NY-CITY"
    }]
  }'
```

HTTPIe:

```
http POST "https://updown-api.nyiso.com/updown/api/v1/delete-load-bid" \
  "Authorization:Basic <base64-encoded-credentials>" \
  "Content-Type:application/json" \
  --cert /path/to/naesb-cert.pem --cert-key /path/to/naesb-key.pem \
  < delete-load-bid-body.json
```

curl — Multi-bid (3 hours for two loads):

```
curl -X POST "https://updown-api.nyiso.com/updown/api/v1/delete-load-bid" \
  -H "Authorization: Basic <base64-encoded-credentials>" \
  -H "Content-Type: application/json" \
  --cert /path/to/naesb-cert.pem --key /path/to/naesb-key.pem \
  -d '{
    "submissionParameters": {
      "doCommit": true,
      "processUnidentifiedBids": false
    },
    "deleteLoadBids": [
      { "bidId": 1174702467 },
      { "loadName": "EXAMPLE_VL_NY-CITY", "bidDate": "2024-01-22T00:00-05:00" },
    ]
  }'
```

```
{ "loadName": "EXAMPLE_VL_NY-CITY", "bidDate": "2024-01-22T01:00-05:00" }  
]  
'
```

Response

Same response structure as POST /load-bid, with deleted array populated on success and rejected array on failure.

400 / 401 / 403 / 500: See [Section 2.6](#)

Response Examples

200 OK — Bid successfully deleted:

```
{  
  "requestTimestamp": "2026-04-27 15:57:17 EDT",  
  "submissionParameters": {  
    "requestType": "delete-load-bid",  
    "doCommit": true,  
    "processUnidentifiedBids": true,  
    "nyisoRequestId": "d4e5f6a7-b8c9-0123-defa-345678901234",  
    "fingerprint": "a1b2c3d4e5f6a7b8c9doe1f2a3b4c5d6e7f8a9bo",  
    "caller": "JSMITH"  
  },  
  "requestStartTime": "2026-04-27T19:57:17.604602300Z",  
  "count": 1,  
  "deleted": [  
    {  
      "bidDate": "2026-05-01T15:00Z",  
      "displayName": "EXMPLSE_PA_LOADBUS3",  
      "ptid": 100003,  
      "bidId": 1000000005,  
      "bidStatus": "VALIDATION_PASSED",  
      "forecastMw": 170,  
      "fixedBidMw": 1  
    }  
  ]  
}
```

200 OK — Bid not found (rejected):

```
{  
  "requestTimestamp": "2026-04-27 15:55:59 EDT",  
  "submissionParameters": {  
    "requestType": "delete-load-bid",  
    "doCommit": true,  
    "processUnidentifiedBids": true,  
    "nyisoRequestId": "e5f6a7b8-c9d0-1234-efab-456789012345",  
    "fingerprint": "b2c3d4e5f6a7b8c9doe1f2a3b4c5d6e7f8a9boc1",  
    "caller": "MJONES"  
  }  
}
```

```

    },
    "errors": [
      "CBM-21134 ERROR: The following bid was not located in the system: Load [] BID ID [1000000005] Date []"
    ],
    "requestStartTime": "2026-04-27T19:55:59.092557900Z",
    "count": 1,
    "rejected": [
      {
        "bidId": 1000000005
      }
    ]
  }

```

Note: When a bid cannot be found, the response still returns HTTP 200 but places the bid in the rejected[] array with only the identifying field(s) provided. The errors array contains the descriptive error message.

7.2. Transaction Endpoints

7.2.1. GET /tran-sch – Retrieve Transaction Schedules

Request

HTTP Method + Full URL:

GET <https://updown-api.nyiso.com/updown/api/v1/tran-sch>

Description: Retrieves transaction (bilateral) bid schedules for the authenticated user's organization. Requires marketType and startDateTime as mandatory parameters.

Required Authentication:

Requirement	Value / Format	Description
Authorization Header	Basic <base64(username:password)>	Required. HTTP Basic Authentication credentials.
TLS Client Certificate	NAESB-compliant .pem/.pfx certificate	Required. Valid NAESB digital certificate linked to the same MIS user account (see Section 2.3).

Query Parameters:

Name	Type	Required	Description	Allowed Values	Example
marketType	String	Yes	Market type	DAM, HAM, NON-FIRM	DAM

Name	Type	Required	Description	Allowed Values	Example
startDateTime	String	Yes	Start date/time (ISO-8601)	ISO-8601 with offset	2024-11-30T00:00:00-05:00
transactionId	Long	No	Filter by transaction contract ID	Numeric	1234
sourceName	String	No	Source generator name	Alphanumeric	EXAMPLE_GEN
sourcePtid	Integer	No	Source PTID	Up to 6 digits	23818
sinkName	String	No	Sink load/generator name	Alphanumeric	EXAMPLE_LOAD
sinkPtid	Integer	No	Sink PTID	Up to 6 digits	110802
Status	String	No	Bid status filter	WAITING VALIDATION, VALIDATION FAILED, WAITING CONFIRMATION, VALIDATION PASSED, EVALUATING, BID REJECTED, ADVISORY REJECTED, BID ACCEPTED, ADVISORY ACCEPTED, MODIFIED, PRE SCHED QUEUED, PRE SCHED REJECTED, PRE SCHED CONFIRMED, PRE SCH MRKT CLOSED	BID ACCEPTED
userReference	String	No	User reference string	Text	S12345
numHours	Integer	No	Number of hours	0—23	6
modifiedDate	String	No	Last modified date/time	ISO-8601 with offset	2024-11-30T00:00-05:00

Request Examples

curl:

```
curl -X GET "https://updown-api.nyiso.com/updown/api/v1/tran-sch?marketType=DAM&startDateTime=2024-11-30T00:00:00-05:00" \
  -H "Authorization: Basic <base64-encoded-credentials>" \
  --cert /path/to/naesb-cert.pem --key /path/to/naesb-key.pem
```

HTTPIe:

```
http GET "https://updown-api.nyiso.com/updown/api/v1/tran-sch" \
  "Authorization:Basic <base64-encoded-credentials>" \
  --cert /path/to/naesb-cert.pem --cert-key /path/to/naesb-key.pem \
  marketType==DAM startDateTime=="2024-11-30T00:00:00-05:00"
```

Response

200 OK — Success:

Field	Type	Description
nyisoRequestId	String	Auto-generated request correlation ID (UUID)
Errors	Array	Empty on success
transactionBids	Array	Array of transaction bid objects
transactionBids[].bidId	Integer	Unique bid ID
transactionBids[].bidDate	String	Bid hour (UTC, Z suffix)
transactionBids[].market	String	DAM, HAM, or NON-FIRM
transactionBids[].bidStatus	String	Bid status (e.g., WAITING_CONFIRMATION, BID_ACCEPTED)
transactionBids[].internalBilateralMw	Number	MW on this transaction
transactionBids[].priceCap	Number	Price-cap \$/MWh (HAM only, if applicable)
transactionBids[].nonFirmPriority	Integer	NERC priority (non-firm only)
transactionBids[].frpConfStatus	Boolean	FRP confirmation
transactionBids[].buyerConfStatus	Boolean	Buyer confirmation
transactionBids[].sellerConfStatus	Boolean	Seller confirmation
transactionBids[].contract	Object	Nested contract object (see below)
transactionBids[].contract.contractId	Integer	Contract ID
transactionBids[].contract.source	Object	Source generator summary (displayName, ptid)
transactionBids[].contract.sink	Object	Sink load summary (displayName, ptid)
transactionBids[].contract.userReference	String	User-supplied reference (e.g., S12345)

400 / 401 / 403 / 500: See [Section 2.6](#)

Response Examples

200 OK — Success:

```
{
  "nyisoRequestId": "c3d4e5f6-a7b8-9012-cdef-345678901234",
  "errors": [],
  "transactionBids": [
    {
      "bidId": 1000000017,
      "bidDate": "2024-11-30T05:00:00Z",
      "contract": {
        "contractId": 900000001,
```

```

    "source": {
      "displayName": "EXAMPLE_GEN",
      "ptid": 23818
    },
    "sink": {
      "displayName": "EXAMPLE_CE_NY-CITY",
      "ptid": 110802
    },
    "userReference": "S12345"
  },
  "bidStatus": "BID_ACCEPTED",
  "market": "DAM",
  "internalBilateralMw": 10.0
}
]
}

```

200 OK – Validation Error (missing required startDateTime parameter):

```

{
  "requestStartTime": "2025-08-15T15:48:32.128968697Z",
  "requestEndTime": "2025-08-15T15:48:32.128971106Z",
  "count": 0,
  "message": "Invalid request parameters",
  "errors": [
    "getTransactionSchedules.startDateTime: UPD-10001 ERROR: The field Date & Time
    can not be null."
  ]
}

```

401 / 403 – User unauthorized to access transaction bidding:

```

{
  "requestStartTime": "2025-09-02T01:06:28.636304556Z",
  "requestEndTime": "2025-09-02T01:06:28.636306257Z",
  "count": 0,
  "message": "Upload/Download Error",
  "errors": [
    "CBM-20000 ERROR: User is unauthorized to access the resource"
  ]
}

```

7.2.2. GET /tran-contract – Retrieve Transaction Contracts

Description: Retrieves bilateral transaction contracts. All parameters are optional filters.

Required Authentication:

Requirement	Value / Format	Description
Authorization Header	Basic <base64(username:password)>	Required. HTTP Basic Authentication credentials.

Requirement	Value / Format	Description
TLS Client Certificate	NAESB-compliant .pem/.pfx certificate	Required. Valid NAESB digital certificate linked to the same MIS user account (see Section 2.3).

Query Parameters:

Name	Type	Required	Description	Example
transactionId	Long	No	Contract ID	90000001
sourceName	String	No	Source name	EXAMPLE_GEN
sourcePtid	Integer	No	Source PTID	23818
sinkName	String	No	Sink name	EXAMPLE_LOAD
sinkPtid	Integer	No	Sink PTID	110802
userReference	String	No	User reference	S12345
numRows	Integer	No	Max rows to return	100
sourceEnergyOrgId	Integer	No	Source energy org ID	546231
sinkEnergyOrgId	Integer	No	Sink energy org ID	546231

Request Examples

curl:

```
curl -X GET "https://updown-api.nyiso.com/updown/api/v1/tran-contract?sourcePtid=23818&sinkPtid=110802" \
  -H "Authorization: Basic <base64-encoded-credentials>" \
  --cert /path/to/naesb-cert.pem --key /path/to/naesb-key.pem
```

HTTPIe:

```
http GET "https://updown-api.nyiso.com/updown/api/v1/tran-contract" \
  "Authorization:Basic <base64-encoded-credentials>" \
  --cert /path/to/naesb-cert.pem --cert-key /path/to/naesb-key.pem \
```

Response

200 OK — Success:

Field	Type	Description
nyisoRequestId	String	Auto-generated request correlation ID
Errors	Array	Empty on success
transactionContracts	Array	Array of contract objects
transactionContracts[].contractId	Integer	Unique contract identifier
transactionContracts[].source.displayName	String	Source generator display name
transactionContracts[].source.ptid	Integer	Source PTID
transactionContracts[].source.generatorType	String	Generator type (e.g., NORMAL_30_MINUTE_START)
transactionContracts[].sink.displayName	String	Sink load display name
transactionContracts[].sink.ptid	Integer	Sink PTID
transactionContracts[].sink.loadType	String	Load type (e.g., REAL_LOAD)
transactionContracts[].userReference	String	User reference
transactionContracts[].sourceEnergyOrgId	Integer	Source energy organization ID
transactionContracts[].sinkEnergyOrgId	Integer	Sink energy organization ID
transactionContracts[].active	Boolean	Whether the contract is currently active

Response Examples

200 OK:

```
{
  "requestStartTime": "2026-01-15T10:00:00.000000000Z",
  "requestEndTime": "2026-01-15T10:00:00.050000000Z",
  "nyisoRequestId": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
  "count": 3,
  "message": "Success",
  "transactionContracts": [
    {
      "contractId": 10000001,
      "source": {
        "displayName": "EXAMPLE_GEN_A",
        "ptid": 20001
      },
      "sink": {
        "ptid": 110001,
        "displayName": "EXAMPLE_LOAD_NORTH"
      },
      "userReference": "REF-001",
      "lastUpdateTime": "2026-01-10T08:30:00Z",
      "lastUpdateUser": {
        "contactName": "Jane Smith"
      }
    }
  ]
}
```

```

    }
  },
  {
    "contractId": 10000002,
    "source": {
      "displayName": "EXAMPLE_GEN_B",
      "ptid": 20002
    },
    "sink": {
      "ptid": 110002,
      "displayName": "EXAMPLE_LOAD_SOUTH"
    },
    "userReference": "REF-002",
    "lastUpdateTime": "2026-01-11T09:15:00Z",
    "lastUpdateUser": {
      "contactName": "John Doe"
    }
  },
  {
    "contractId": 10000003,
    "source": {
      "displayName": "EXAMPLE_GEN_HUB",
      "ptid": 320001
    },
    "sink": {
      "ptid": 100001,
      "displayName": "EXAMPLE_LOAD_HUB"
    },
    "userReference": "REF-HUB-001",
    "sourceEnergyOwnerOrg": {
      "orgId": 9000001
    },
    "sinkEnergyOwnerOrg": {
      "orgId": 9000001
    },
    "lastUpdateTime": "2026-01-12T14:45:00Z",
    "lastUpdateUser": {
      "contactName": "Jane Smith"
    }
  }
]
}

```

400 / 401 / 403 / 500: See [Section 2.6](#)

7.2.3. POST /tran-bid – Submit Transaction Bids

Request

HTTP Method + Full URL:

POST <https://updown-api.nyiso.com/updown/api/v1/tran-bid>

Description: Submits one or more bilateral transaction bids. Each bid represents a single hour of transaction data between a source (generator) and sink (load).

Required Authentication:

Requirement	Value / Format	Description
Authorization Header	Basic <base64(username:password)>	Required. HTTP Basic Authentication credentials.
Content-Type Header	application/json	Required. Request body format.
TLS Client Certificate	NAESB-compliant .pem/.pfx certificate	Required. Valid NAESB digital certificate linked to the same MIS user account (see Section 2.3).

Request Body Schema:

Array name: The canonical request-body array for this endpoint is bids; submitTransactionBids is also accepted on input as an alias.

Field	Type	Required	Description	Example
submissionParameters.doCommit	Boolean	Yes	Commit if valid	true
submissionParameters.processUnidentified Bids	Boolean	No	Process bids for unidentified contracts	true
bids[].bidDate (alias: submitTransactionBids[])	String	Yes	Bid date/time (ISO-8601 with offset)	"2026-05-02T00:00:00-04:00"
bids[].sourceName	String	Conditional	Source generator display name	"EXAMPLE_GEN_A"
bids[].sourcePtid	Integer	Conditional	Source PTID (alternative to sourceName)	100001
bids[].sinkName	String	Conditional	Sink load display name	"EXAMPLE_LOAD_WEST"
bids[].sinkPtid	Integer	Conditional	Sink PTID (alternative to sinkName)	110001

Field	Type	Required	Description	Example
bids[].userReference	String	Yes	Unique user reference for this contract	"S12345"
bids[].market	String	Yes	Market type	"DAM", "HAM", "NON-FIRM"
bids[].bidEnergyMw	Number	Yes	Fixed block energy MW	1000
bids[].nercPriority	Integer	No	NERC curtailment priority (1—7)	7

Request Examples

curl — Single transaction bid (new contract by names):

```
curl -X POST "https://updown-api.nyiso.com/updown/api/v1/tran-bid" \
-H "Authorization: Basic <base64-encoded-credentials>" \
-H "Content-Type: application/json" \
--cert /path/to/naesb-cert.pem --key /path/to/naesb-key.pem \
-d '{
  "submissionParameters": {
    "doCommit": true,
    "processUnidentifiedBids": true
  },
  "bids": [{
    "bidDate": "2026-05-02T00:00:00-04:00",
    "sourceName": "EXAMPLE_GEN_A",
    "sinkName": "EXAMPLE_LOAD_WEST",
    "market": "DAM",
    "nercPriority": 7,
    "userReference": "S12345",
    "bidEnergyMw": 1000
  }]
}'
```

curl — Multi-bid (2 hours, DAM + HAM, by PTID and by name):

```
curl -X POST "https://updown-api.nyiso.com/updown/api/v1/tran-bid" \
-H "Authorization: Basic <base64-encoded-credentials>" \
-H "Content-Type: application/json" \
--cert /path/to/naesb-cert.pem --key /path/to/naesb-key.pem \
-d '{
  "submissionParameters": {
    "doCommit": true,
    "processUnidentifiedBids": true
  },
  "bids": [
    {
      "bidDate": "2026-05-02T00:00:00-04:00",
      "sourceName": "EXAMPLE_GEN_A",
      "sinkName": "EXAMPLE_LOAD_WEST",
      "market": "DAM",
      "nercPriority": 7,
      "userReference": "S12345",
      "bidEnergyMw": 1000
    },
    {
      "bidDate": "2026-05-02T02:00:00-04:00",
      "sourceName": "EXAMPLE_GEN_A",
      "sinkName": "EXAMPLE_LOAD_WEST",
      "market": "HAM",
      "nercPriority": 7,
      "userReference": "S12345",
      "bidEnergyMw": 1000
    }
  ]
}'
```

```
"bids": [
  {
    "bidDate": "2026-05-02T00:00:00-04:00",
    "sourcePtid": 100001,
    "sinkPtid": 110001,
    "market": "DAM",
    "nercPriority": 7,
    "userReference": "S12345",
    "bidEnergyMw": 1000
  },
  {
    "bidDate": "2026-05-02T01:00:00-04:00",
    "sourceName": "EXAMPLE_GEN_A",
    "sinkName": "EXAMPLE_LOAD_WEST",
    "market": "HAM",
    "userReference": "S12346",
    "bidEnergyMw": 800
  }
]
```

HTTPIe:

```
http POST "https://updown-api.nyiso.com/updown/api/v1/tran-bid" \
  "Authorization:Basic <base64-encoded-credentials>" \
  "Content-Type:application/json" \
  --cert /path/to/naesb-cert.pem --cert-key /path/to/naesb-key.pem \
  < tran-bid-body.json
```

Response

Returns a submission response. Successfully submitted bids appear in `accepted[]`; re-bids on an existing contract appear in `updated[]`; rejected bids appear in `rejected[]`.

Field	Type	Description
<code>requestTimestamp</code>	String	Timestamp of the request (yyyy-MM-dd HH:mm:ss z)
<code>submissionParameters</code>	Object	Echo of the submission parameters
<code>nyisoRequestId</code>	String	Auto-generated unique request correlation ID (UUID) for tracing and support
<code>Errors</code>	Array	Global error messages (empty on success)
<code>validationErrors</code>	Object	Map of bid key -> array of per-field error messages
<code>failedValidation</code>	Array	Array of bid objects that failed validation
<code>passedValidation</code>	Array	Array of bid objects that passed validation
<code>Accepted</code>	Array	Array of bid objects accepted by the market

Field	Type	Description
Rejected	Array	Array of bid objects rejected by the market

400 Bad Request: See [Section 2.6.1](#) for the canonical errors[] + validationErrors{} shape.

validationErrors is keyed by "Bid Index: <n> Load Name: <name>".

401 / 403 / 500: See [Section 2.6.2](#) - [Section 2.6.4](#)

Response Examples

200 OK — Success (bid accepted, awaiting confirmation):

```
{
  "requestTimestamp": "2026-05-02 08:53:03 EDT",
  "submissionParameters": {
    "requestType": "tran-bid",
    "doCommit": true,
    "processUnidentifiedBids": true,
    "nyisoRequestId": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
    "fingerprint": "a1b2c3d4e5f6a7b8c9doe1f2a3b4c5d6e7f8a9bo",
    "caller": "JSMITH"
  },
  "requestStartTime": "2026-05-02T12:53:03.620174607Z",
  "count": 1,
  "accepted": [
    {
      "bidDate": "2026-05-02T04:00Z",
      "bidId": 1000000001,
      "contract": {
        "contractId": 900000001,
        "source": {
          "displayName": "EXAMPLE_GEN_A",
          "ptid": 100001
        },
      },
      "sink": {
        "displayName": "EXAMPLE_LOAD_WEST",
        "ptid": 110001
      },
      "userReference": "S12345"
    },
    {
      "bidStatus": "WAITING_CONFIRMATION",
      "frpConfStatus": true,
      "buyerConfStatus": true,
      "sellerConfStatus": false,
      "market": "DAM",
      "internalBilateralMw": 1000,
      "nercPriority": 7
    }
  ]
}
```

```
    ]
  }
```

Note: bidStatus: "WAITING_CONFIRMATION" is the normal initial state for bilateral transaction bids. All parties (frpConfStatus, buyerConfStatus, sellerConfStatus) must confirm before the bid is eligible for market scheduling. See §7.2.4 for the confirm-tran-bid endpoint.

400 Bad Request – Invalid bid date (ISO-8601 violation, e.g., 2025-09-31):

```
{
  "requestStartTime": "2025-08-15T15:48:32.128968697Z",
  "requestEndTime": "2025-08-15T15:48:32.128971106Z",
  "count": 0,
  "message": "Invalid request parameters",
  "errors": [
    "getTransactionSchedules.startDateTime: UPD-10001 ERROR: The field Date & Time
    can not be null."
  ]
}
```

401 / 403 – User unauthorized to access transaction bidding:

```
{
  "requestStartTime": "2025-09-02T01:06:28.636304556Z",
  "requestEndTime": "2025-09-02T01:06:28.636306257Z",
  "count": 0,
  "message": "Upload/Download Error",
  "errors": [
    "CBM-20000 ERROR: User is unauthorized to access the resource"
  ]
}
```

7.2.4. POST /confirm-tran-bid – Confirm Transaction Bids

HTTP Method + Full URL:

POST <https://updown-api.nyiso.com/updown/api/v1/confirm-tran-bid>

Description: Confirms or rejects one or more bilateral transaction bids that are in WAITING_CONFIRMATION status. Bilateral transactions require confirmation from multiple parties (seller, buyer, and/or FRP) before they are eligible for market scheduling.

Required Authentication:

Requirement	Value / Format	Description
Authorization Header	Basic <base64(username:password)>	Required. HTTP Basic Authentication credentials.

Requirement	Value / Format	Description
Content-Type Header	application/json	Required. Request body format.
TLS Client Certificate	NAESB-compliant .pem/.pfx certificate	Required. Valid NAESB digital certificate linked to the same MIS user account (see Section 2.3).

Request Body Schema:

Array name: The canonical request-body array for this endpoint is bids; confirmTransactionBids is also accepted on input as an alias. Either name works.

Field	Type	Required	Description	Example
submissionParameters.doCommit	Boolean	Yes	Commit confirmation	true
bids[].bidId (alias: confirmTransactionBids[])	Integer	Yes	Bid ID to confirm/reject	1000000017
bids[].tranId	Integer	No	Transaction (contract) ID	900000001
bids[].bidDate	String	No	Bid date/time (ISO-8601)	"2026-05-01T00:00:00-04:00"
bids[].confirmFlag	Boolean	Yes	true to confirm, false to reject	true

Request Examples

curl:

```
curl -X POST "https://updown-api.nyiso.com/updown/api/v1/confirm-tran-bid" \
-H "Authorization: Basic <base64-encoded-credentials>" \
-H "Content-Type: application/json" \
--cert /path/to/naesb-cert.pem --key /path/to/naesb-key.pem \
-d '{
  "submissionParameters": {
    "doCommit": true,
    "processUnidentifiedBids": true
  },
  "bids": [{
    "bidId": 1000000017,
    "tranId": 900000001,
    "bidDate": "2026-05-01T00:00:00-04:00",
    "confirmFlag": true
  }
}]'
```

```
}]
}'
```

HTTPIe:

```
http POST "https://updown-api.nyiso.com/updown/api/v1/confirm-tran-bid" \
  "Authorization:Basic <base64-encoded-credentials>" \
  "Content-Type:application/json" \
  --cert /path/to/naesb-cert.pem --cert-key /path/to/naesb-key.pem \
  < confirm-tran-body.json
```

curl — Multi-bid (2 bids, one confirm + one reject):

```
curl -X POST "https://updown-api.nyiso.com/updown/api/v1/confirm-tran-bid" \
  -H "Authorization: Basic <base64-encoded-credentials>" \
  -H "Content-Type: application/json" \
  --cert /path/to/naesb-cert.pem --key /path/to/naesb-key.pem \
  -d '{
    "submissionParameters": {
      "doCommit": true,
      "processUnidentifiedBids": true
    },
    "bids": [
      { "bidId": 1000000021, "confirmFlag": true },
      { "bidId": 1000000019, "confirmFlag": false }
    ]
  }
```

Response

200 OK: Returns submission result with an updated array containing the confirmed/rejected bids. Each entry includes the contract details (source generator and sink load), confirmation statuses (frpConfStatus, buyerConfStatus, sellerConfStatus), and bidStatus.

400 / 401 / 403 / 500: See [Section 2.6](#)

Response Examples

200 OK — Bid confirmation updated (seller rejected, sellerConfStatus: false):

```
{
  "requestTimestamp": "2026-04-28 10:04:15 EDT",
  "submissionParameters": {
    "requestType": "confirm-tran-bid",
```

```

    "doCommit": true,
    "processUnidentifiedBids": true,
    "nyisoRequestId": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
    "fingerprint": "a1b2c3d4e5f6a7b8c9doe1f2a3b4c5d6e7f8a9bo",
    "caller": "JSMITH"
  },
  "updated": [
    {
      "bidId": 1000000017,
      "bidDate": "2026-05-01T04:00:00Z",
      "contract": {
        "contractId": 900000001,
        "source": {
          "displayName": "EXAMPLE_GEN_A",
          "ptid": 100001
        },
        "sink": {
          "displayName": "EXAMPLE_LOAD_WEST",
          "ptid": 110001
        },
        "userReference": "S12345"
      },
      "bidStatus": "WAITING_CONFIRMATION",
      "frpConfStatus": true,
      "buyerConfStatus": true,
      "sellerConfStatus": false,
      "market": "DAM"
    }
  ]
}

```

401 / 403 – Dispatcher/observer role lacks confirm-tran-bid authority:

```

{
  "requestStartTime": "2025-09-04T13:29:37.031694842Z",
  "requestEndTime": "2025-09-04T13:29:37.031696005Z",
  "count": 0,
  "message": "Upload/Download Error",
  "errors": [
    "CBM-20000 ERROR: User is unauthorized to access the resource"
  ]
}

```

7.2.5. POST /delete-tran-bid – Delete Transaction Bids

Request

HTTP Method + Full URL:

POST <https://updown-api.nyiso.com/updown/api/v1/delete-tran-bid>

Description: Deletes one or more previously submitted bilateral transaction bids. Bids can be identified by bidId,

or by a combination of tranId, bidDate, and market.

Required Authentication:

Requirement	Value / Format	Description
Authorization Header	Basic <base64(username:password)>	Required. HTTP Basic Authentication credentials.
Content-Type Header	application/json	Required. Request body format.
TLS Client Certificate	NAESB-compliant .pem/.pfx certificate	Required. Valid NAESB digital certificate linked to the same MIS user account (see Section 2.3).

Request Body Schema:

Field	Type	Required	Description	Example
deleteTransactionBids[].bidId	Integer	Conditional	Bid ID (sufficient alone)	1000000019
deleteTransactionBids[].tranId	Long	Conditional	Transaction/contract ID	900000002
deleteTransactionBids[].market	String	No	Market type	"DAM" or "HAM"
deleteTransactionBids[].bidDate	String	Conditional	Bid date (ISO-8601)	"2025-09-05T06:00:00-04:00"

Request Examples

curl — Delete 3 bids by bidId:

```
curl -X POST "https://updown-api.nyiso.com/updown/api/v1/delete-tran-bid" \
-H "Authorization: Basic <base64-encoded-credentials>" \
-H "Content-Type: application/json" \
--cert /path/to/naesb-cert.pem --key /path/to/naesb-key.pem \
-d '{
  "submissionParameters": {
    "doCommit": true,
    "processUnidentifiedBids": true
  },
  "bids": [
    { "bidId": 1000000001 },
    { "bidId": 1000000003 },
    { "bidId": 1000000005 }
  ]
}'
```

curl — Delete by tranId + bidDate + market (when bidId is unknown):

```
curl -X POST "https://updown-api.nyiso.com/updown/api/v1/delete-tran-bid" \
-H "Authorization: Basic <base64-encoded-credentials>" \
-H "Content-Type: application/json" \
--cert /path/to/naesb-cert.pem --key /path/to/naesb-key.pem \
-d '{
  "submissionParameters": {
    "doCommit": true
  },
  "bids": [
    { "tranId": 90000002, "bidDate": "2026-05-02T00:00:00-04:00", "market": "DAM"
  }
]
}'
```

HTTPIe:

```
http POST "https://updown-api.nyiso.com/updown/api/v1/delete-tran-bid" \
"Authorization:Basic <base64-encoded-credentials>" \
"Content-Type:application/json" \
--cert /path/to/naesb-cert.pem --cert-key /path/to/naesb-key.pem \
< delete-tran-body.json
```

Response

200 OK: Returns submission result with deleted array.

400 / 401 / 403 / 500: See [Section 2.6](#)

Response Examples

200 OK — Success (bid deleted):

```
{
  "requestTimestamp": "2026-05-02 09:52:40 EDT",
  "submissionParameters": {
    "requestType": "delete-tran-bid",
    "doCommit": true,
    "processUnidentifiedBids": true,
    "nyisoRequestId": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
    "fingerprint": "a1b2c3d4e5f6a7b8c9doe1f2a3b4c5d6e7f8a9bo",
    "caller": "JSMITH"
  },
  "requestStartTime": "2026-05-02T13:52:40.317660Z",
  "count": 1,
  "deleted": [
    {
      "bidId": 1000000001,
      "bidDate": "2026-05-02T04:00Z",
```

```

    "contract": {
      "contractId": 900000001,
      "source": {
        "displayName": "EXAMPLE_GEN_A",
        "ptid": 100001
      },
      "sink": {
        "displayName": "EXAMPLE_LOAD_WEST",
        "ptid": 110001
      },
      "userReference": "S12345"
    },
    "bidStatus": "WAITING_CONFIRMATION",
    "market": "DAM"
  }
]
}

```

200 OK — All 3 bids not found (all rejected):

Note: When submitted bidId values do not match any existing bid, the response is still HTTP 200. Each unmatched bid appears in rejected array with the corresponding error in the top-level errors array (one entry per rejected bid). count reflects the number of entries processed — not necessarily deleted.

```

{
  "requestTimestamp": "2026-04-28 13:46:08 EDT",
  "submissionParameters": {
    "requestType": "delete-tran-bid",
    "doCommit": true,
    "processUnidentifiedBids": true,
    "nyisoRequestId": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
    "fingerprint": "a1b2c3d4e5f6a7b8c9doe1f2a3b4c5d6e7f8a9bo",
    "caller": "JSMITH"
  },
  "errors": [
    "CBM-10065 ERROR: Bid not found. Cannot delete. (bidId: 1000000013)",
    "CBM-10065 ERROR: Bid not found. Cannot delete. (bidId: 1000000005)",
    "CBM-10065 ERROR: Bid not found. Cannot delete. (bidId: 1000000007)"
  ],
  "requestStartTime": "2026-04-28T17:46:08.787864300Z",
  "count": 3,
  "rejected": [
    { "bidId": 1000000013, "nercPriority": 7 },
    { "bidId": 1000000005, "nercPriority": 7 },
    { "bidId": 1000000007, "nercPriority": 7 }
  ]
}

```

401 / 403 — User role lacks delete-tran-bid authority:

```
{
  "requestStartTime": "2025-09-04T13:29:37.031694842Z",
  "requestEndTime": "2025-09-04T13:29:37.031696005Z",
  "count": 0,
  "message": "Upload/Download Error",
  "errors": [
    "CBM-10051 ERROR: User is not authorized to delete transactions."
  ]
}
```

7.3. Generator Endpoints

7.3.1. GET /gen-detail – Retrieve Generator Details

Request

HTTP Method + Full URL:

GET <https://updown-api.nyiso.com/updown/api/v1/gen-detail>

Description: Retrieves detailed information about generator units, including generator name, PTID, type, operating limits, zone, and qualifications. Results can be filtered by generator name or PTID.

Required Authentication:

Requirement	Value / Format	Description
Authorization Header	Basic <base64(username:password)>	Required. HTTP Basic Authentication credentials.
TLS Client Certificate	NAESB-compliant .pem/.pfx certificate	Required. Valid NAESB digital certificate linked to the same MIS user account (see Section 2.3).

Query Parameters:

Name	Type	Required	Description	Example
genName	String	No	Generator name	EXAMPLE_GEN
Ptid	Integer	No	Generator PTID (up to 6 digits)	546231

Request Examples

curl:

```
curl -X GET "https://updown-api.nyiso.com/updown/api/v1/gen-detail?genName=EXAMPLE_GEN" \
  -H "Authorization: Basic <base64-encoded-credentials>" \
  --cert /path/to/naesb-cert.pem --key /path/to/naesb-key.pem
```

HTTPIe:

```
http GET "https://updown-api.nyiso.com/updown/api/v1/gen-detail" \
  "Authorization:Basic <base64-encoded-credentials>" \
  --cert /path/to/naesb-cert.pem --cert-key /path/to/naesb-key.pem \
  genName==EXAMPLE_GEN
```

Response

200 OK — Success:

Field	Type	Description
requestStartTime	String	ISO-8601 timestamp when request processing began
requestEndTime	String	ISO-8601 timestamp when request processing completed
nyisoRequestId	String	Auto-generated unique request correlation ID (UUID)
Count	Integer	Number of records returned
Message	String	Status message (e.g., "Success")
generatorDetails	Array	Array of generator detail objects
generatorDetails[].displayName	String	Full display name (e.g., "EXMPLSE_SP_LOADBUS1")
generatorDetails[].ptid	Integer	Generator PTID
generatorDetails[].active	Boolean	Whether the generator is currently active
generatorDetails[].generatorType	String	Generator type (e.g., NON_UTILITY_GENERATOR, NORMAL_30_MINUTE_START, QUICK_START, WIND, ESR, BTM_NG, CSR)
generatorDetails[].physicalMinimumGenerationMw	Number	Physical minimum generation (MW)
generatorDetails[].maxRegulationResponseRate	Number	Max regulation response rate
generatorDetails[].max6SecRegulationResponseRate	Number	Max 6-second regulation response rate
generatorDetails[].emergencyResponseRate	Number	Emergency response rate
generatorDetails[].normalResponseRate1	Number	Normal response rate
generatorDetails[].avrQualified	Boolean	AVR qualified

Field	Type	Description
generatorDetails[].bidDamFixedEnergyQualified	Boolean	DAM fixed energy bid qualified
generatorDetails[].bidHamFixedEnergyQualified	Boolean	HAM fixed energy bid qualified
generatorDetails[].damDispatchableEnergyQualified	Boolean	DAM dispatchable energy qualified
generatorDetails[].hamDispatchableEnergyQualified	Boolean	HAM dispatchable energy qualified
generatorDetails[].dam10MinuteSpinningQualified	Boolean	DAM 10-min spinning reserve qualified
generatorDetails[].dam10MinuteNonSynchQualified	Boolean	DAM 10-min non-synch reserve qualified
generatorDetails[].dam30MinuteSpinningQualified	Boolean	DAM 30-min spinning reserve qualified
generatorDetails[].dam30MinuteNonSynchQualified	Boolean	DAM 30-min non-synch reserve qualified
generatorDetails[].damRegulationControlQualified	Boolean	DAM regulation control qualified
generatorDetails[].ham10MinuteSpinningQualified	Boolean	HAM 10-min spinning reserve qualified
generatorDetails[].ham10MinuteNonSynchQualified	Boolean	HAM 10-min non-synch reserve qualified
generatorDetails[].ham30MinuteSpinningQualified	Boolean	HAM 30-min spinning reserve qualified
generatorDetails[].ham30MinuteNonSynchQualified	Boolean	HAM 30-min non-synch reserve qualified
generatorDetails[].hamRegulationControlQualified	Boolean	HAM regulation control qualified
generatorDetails[].subzone	Object	Subzone information
generatorDetails[].subzone.ptid	Integer	Subzone PTID
generatorDetails[].subzone.subzoneName	String	Subzone name
generatorDetails[].subzone.zone	Object	Zone information
generatorDetails[].subzone.zone.ptid	Integer	Zone PTID
generatorDetails[].subzone.zone.zoneName	String	Zone name
generatorDetails[].limits	Object	Operating limits
generatorDetails[].limits.maxSummerOperatingLimit	Number	Maximum summer operating limit (MW)
generatorDetails[].limits.maxWinterOperatingLimit	Number	Maximum winter operating limit (MW)
generatorDetails[].limits.summerIcapContracts	Number	Summer ICAP contracts (MW)
generatorDetails[].limits.winterIcapContracts	Number	Winter ICAP contracts (MW)
generatorDetails[].contact	Object	Contact info for the generator's responsible party
generatorDetails[].contact.contactName	String	Contact person name
generatorDetails[].contact.contactAddress	String	Full address (single string)

Field	Type	Description
generatorDetails[].contact.addressLine1	String	Address line 1
generatorDetails[].contact.addressLine2	String	Address line 2
generatorDetails[].contact.addressLine3	String	Address line 3
generatorDetails[].contact.primaryPhone	String	Primary phone number
generatorDetails[].contact.secondPhone	String	Secondary phone number
generatorDetails[].contact.faxNumber	String	Fax number
generatorDetails[].contact.primaryEmail	String	Primary email address

400 / 401 / 403 / 500: See [Section 2.6](#)

Response Examples

200 OK — Success:

```
{
  "requestStartTime": "2026-04-28T13:43:38.872468280Z",
  "requestEndTime": "2026-04-28T13:43:39.660559994Z",
  "nyisoRequestId": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
  "count": 1,
  "message": "Success",
  "generatorDetails": [
    {
      "displayName": "EXAMPLE_NUG_FALLS",
      "ptid": 100001,
      "active": true,
      "subzone": {
        "ptid": 200001,
        "subzoneName": "EXAMPLE SUBZONE",
        "zone": {
          "ptid": 300001,
          "zoneName": "EXZONE"
        }
      }
    }
  ],
  "generatorType": "NON_UTILITY_GENERATOR",
  "bidDamFixedEnergyQualified": true,
  "bidHamFixedEnergyQualified": true,
  "dam10MinuteSpinningQualified": false,
  "dam10MinuteNonSynchQualified": false,
  "dam30MinuteSpinningQualified": false,
  "dam30MinuteNonSynchQualified": false,
  "damRegulationControlQualified": false,
  "physicalMinimumGenerationMw": 0,
  "maxRegulationResponseRate": 0.5,
  "max6SecRegulationResponseRate": 0.05,
  "damDispatchableEnergyQualified": true,
  "hamDispatchableEnergyQualified": true,
  "ham10MinuteSpinningQualified": false,
}
```

```

    "ham10MinuteNonSynchQualified": false,
    "ham30MinuteSpinningQualified": false,
    "ham30MinuteNonSynchQualified": false,
    "hamRegulationControlQualified": false,
    "contact": {
      "contactName": "Jane Doe",
      "contactAddress": "Example Energy Corp. 123 Main St. Anytown, NY 10001",
      "addressLine1": "Example Energy Corp.",
      "addressLine2": "123 Main St.",
      "addressLine3": "Anytown, NY 10001",
      "primaryPhone": "555-123-4567",
      "secondPhone": "555-234-5678",
      "faxNumber": "555-345-6789",
      "primaryEmail": "contact@example.com"
    },
    "limits": {
      "maxSummerOperatingLimit": 44,
      "maxWinterOperatingLimit": 44,
      "summerIcapContracts": 0,
      "winterIcapContracts": 0
    },
    "emergencyResponseRate": 0.5,
    "normalResponseRate1": 0.5,
    "avrQualified": false
  }
]
}

```

200 OK — Generator not found or not authorized: > **Note:** Generator lookup errors are returned as HTTP 200 with count: 0 and the error message in the errors[] array (one entry per error).

```

{
  "requestStartTime": "2026-04-28T13:36:55.283474990Z",
  "requestEndTime": "2026-04-28T13:36:55.283476575Z",
  "count": 0,
  "message": "Upload/Download Error",
  "errors": [
    "CBM-40002 ERROR: Submitted Generator could not be identified. The Generator ei
    ther does not exist, the user is not authorized to bid on the Generator or the Generator
    is associated with a Generator Type that is not qualified to bid generation."
  ]
}

```

7.3.2. GET /gen-sch – Retrieve Generator Bids & Schedules

Request

HTTP Method + Full URL:

GET <https://updown-api.nyiso.com/updown/api/v1/gen-sch>

Description: Retrieves generator bids and schedules for the authenticated user's organization.

Requires marketType, startDateTime, and currentAdvisorySchedulesFlag.

Required Authentication:

Requirement	Value / Format	Description
Authorization Header	Basic <base64(username:password)>	Required. HTTP Basic Authentication credentials.
TLS Client Certificate	NAESB-compliant .pem/.pfx certificate	Required. Valid NAESB digital certificate linked to the same MIS user account (see Section 2.3).

Query Parameters:

Name	Type	Required	Description	Allowed Values	Example
marketType	String	Yes	Market type	DAM, HAM	DAM
startDateTime	String	Yes	Start date/time (ISO-8601)	ISO-8601 with offset	2026-05-01T00:00:00-04:00
currentAdvisorySchedulesFlag	String	Yes	Include current advisory schedules	Y, N	Y
genName	String	No	Generator name	Alphanumeric	EXAMPLE_GEN
Ptid	Integer	No	Generator PTID	Up to 6 digits	546231
Status	String	No	Bid status filter	VALIDATION PASSED, VALIDATION FAILED, EVALUATING, BID ACCEPTED, BID REJECTED	BID ACCEPTED
numHours	Integer	No	Number of hours	0—23	6
lastModifiedDateTime	String	No	Last modified date	ISO-8601 with offset	2026-05-01T00:00:00-04:00

Request Examples

curl:

```
curl -X GET "https://updown-api.nyiso.com/updown/api/v1/gen-sch?marketType=DAM&startDateTime=2026-05-01T00:00:00-04:00&currentAdvisorySchedulesFlag=Y" \
-H "Authorization: Basic <base64-encoded-credentials>" \
--cert /path/to/naesb-cert.pem --key /path/to/naesb-key.pem
```

HTTPIe:

```
http GET "https://updown-api.nyiso.com/updown/api/v1/gen-sch" \
"Authorization:Basic <base64-encoded-credentials>" \
--cert /path/to/naesb-cert.pem --cert-key /path/to/naesb-key.pem \
marketType==DAM startDateTime=="2026-05-01T00:00:00-04:00" currentAdvisorySchedulesFlag==Y
```

Response

200 OK — Success:

Field	Type	Description
requestStartTime	String	ISO-8601 timestamp when request processing began
requestEndTime	String	ISO-8601 timestamp when request processing completed
nyisoRequestId	String	Auto-generated unique request correlation ID (UUID)
count	Integer	Number of records returned
message	String	Status message (e.g., "Success")
generatorBidAndSchedules	Array	Array of bid+schedule wrapper objects
generatorBidAndSchedules[].generatorBid	Object	Generator bid details
generatorBidAndSchedules[].generatorBid.generator	Object	Generator identity
generatorBidAndSchedules[].generatorBid.generator.displayName	String	Full generator display name
generatorBidAndSchedules[].generatorBid.generator.ptid	Integer	Generator PTID
generatorBidAndSchedules[].generatorBid.bidDate	String	Bid hour in UTC (Z suffix)
generatorBidAndSchedules[].generatorBid.generatorBidMarket	String	Market type (e.g., DAM or HAM)
generatorBidAndSchedules[].generatorBid.bidStatus	String	Bid status (e.g., VALIDATION_PASSED, BID_ACCEPTED)
generatorBidAndSchedules[].generatorBid.upperOperatingLimit	Number	Upper operating limit (MW)
generatorBidAndSchedules[].generatorBid.emergencyUpperOperatingLimit	Number	Emergency upper operating limit (MW)

Field	Type	Description
generatorBidAndSchedules[].generatorBid.scheduleType	String	Schedule type (e.g., ISO_COMMITTED_FLEXIBLE, SELF_COMMITTED_FLEXIBLE)
generatorBidAndSchedules[].generatorBid.fixedMinimumGenerationMw	Number	Fixed min-gen MW
generatorBidAndSchedules[].generatorBid.fixedMinimumGenerationCost	Number	Fixed min-gen cost
generatorBidAndSchedules[].generatorBid.dispatchCurve	Object	Dispatch curve (dispatchDollar1/dispatchMw1 ... dispatchDollar12/dispatchMw12)
generatorBidAndSchedules[].generatorBid.regulationMw	Number	Regulation MW offered (present when bid includes regulation)
generatorBidAndSchedules[].generatorBid.regulationCost	Number	Regulation cost
generatorBidAndSchedules[].generatorBid.regulationMovementCost	Number	Regulation movement cost
generatorBidAndSchedules[].generatorBid.bidId	Integer	Assigned bid ID
generatorBidAndSchedules[].generatorBid.esrBidFields	Object	ESR-specific fields (e.g., esrLowerOperatingLimit)
generatorBidAndSchedules[].generatorBid.genSchedules	Array	RTD schedule entries for this bid
generatorBidAndSchedules[].generatorBid.genSchedules[].generator	Object	Generator identity (displayName, ptid)
generatorBidAndSchedules[].generatorBid.genSchedules[].market	String	Market identifier (RTD)
generatorBidAndSchedules[].generatorBid.genSchedules[].intervalBeginTimeStamp	String	Schedule interval start time (UTC)
generatorBidAndSchedules[].generatorBid.genSchedules[].intervalEndTimeStamp	String	Schedule interval end time (UTC)
generatorBidAndSchedules[].generatorBid.genSchedules[].schedMw	Number	Scheduled energy MW
generatorBidAndSchedules[].generatorBid.genSchedules[].schedTenMinSpinReserve	Number	Scheduled 10-min spinning reserve MW
generatorBidAndSchedules[].generatorBid.genSchedules[].schedThirtyMinSpinReserve	Number	Scheduled 30-min spinning reserve MW
generatorBidAndSchedules[].generatorBid.genSchedules[].schedRegulationMw	Number	Scheduled regulation MW
generatorBidAndSchedules[].generatorBid.genSchedules[].schedStartupCost	Number	Scheduled startup cost

Field	Type	Description
generatorBidAndSchedules[].generatorBid.genSchedules[].bidId	Integer	Assigned bid ID
generatorBidAndSchedules[].generatorBid.genSchedules[].esrBidFields	Object	ESR-specific fields (e.g., esrLowerOperatingLimit)
generatorBidAndSchedules[].currentAdvisorySchedulesFlag	Boolean	Whether advisory schedules were included for this bid
generatorAdvSchedules	Array	Advisory/historical schedule records (present when currentAdvisorySchedulesFlag=Y)
generatorAdvSchedules[].generator	Object	Generator identity (displayName, ptid)
generatorAdvSchedules[].intervalEndTimeStamp	String	Schedule interval end time (UTC)
generatorAdvSchedules[].schedMw	Number	Scheduled energy MW
generatorAdvSchedules[].schedTenMinSpinReserve	Number	Scheduled 10-min spinning reserve MW
generatorAdvSchedules[].schedThirtyMinSpinReserve	Number	Scheduled 30-min spinning reserve MW
generatorAdvSchedules[].schedRegulationMw	Number	Scheduled regulation MW
generatorAdvSchedules[].schedStartupCost	Number	Scheduled startup cost
generatorAdvSchedules[].isAdvisory	Boolean	Whether this is an advisory schedule
generatorAdvSchedules[].scheduleSourceType	String	Schedule source (e.g., RTD Hist, RTC)

400 / 401 / 403 / 500: See [Section 2.6](#)

Response Examples

200 OK — Success (truncated to 1 of 19 bid records and 2 of 13 advisory schedule records):

```
{
  "requestStartTime": "2026-05-01T13:43:38.872468280Z",
  "requestEndTime": "2026-05-01T13:43:39.660559994Z",
  "nyisoRequestId": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
  "count": 19,
  "message": "Success",
  "generatorBidAndSchedules": [
    {
      "generatorBid": {
        "generator": {
          "displayName": "EXAMPLE_GEN_A",
          "ptid": 100001
        }
      }
    }
  ]
}
```

```

    },
    "bidDate": "2026-05-02T04:00Z",
    "generatorBidMarket": "DAM",
    "bidStatus": "VALIDATION_PASSED",
    "upperOperatingLimit": 250,
    "emergencyUpperOperatingLimit": 250,
    "startupCost": 0,
    "scheduleType": "ISO_COMMITTED_FIXED",
    "fixedMinimumGenerationMw": 20,
    "fixedMinimumGenerationCost": 100,
    "dispatchCurve": {
      "dispatchDollar1": 255,
      "dispatchMw1": 250
    },
    "bidId": 1000000001,
    "esrBidFields": {
      "esrLowerOperatingLimit": 20
    },
    "genSchedules": [
      {
        "generator": {
          "displayName": "EXAMPLE_GEN_A",
          "ptid": 100001
        }
      }
    ],
    "currentAdvisorySchedulesFlag": false
  },
  "generatorAdvSchedules": [
    {
      "generator": {
        "displayName": "EXAMPLE_GEN_A",
        "ptid": 100001
      },
      "intervalEndTimeStamp": "2026-05-01T00:00:00Z",
      "schedTenMinNonSyncReserve": 0,
      "schedThirtyMinNonSyncReserve": 0,
      "schedStartUpCost": 0,
      "isAdvisory": true,
      "scheduleSourceType": "RTD Hist"
    },
    {
      "generator": {
        "displayName": "EXAMPLE_GEN_A",
        "ptid": 100001
      },
      "intervalEndTimeStamp": "2026-05-01T00:15:00Z",
      "schedMw": 0,
      "schedRegulationMw": 0,

```

```

        "isAdvisory": true,
        "scheduleSourceType": "RTC"
    }
]
}

```

400 Bad Request — Missing required currentAdvisorySchedulesFlag parameter:

```

{
  "requestStartTime": "2026-05-01T13:42:03.057002172Z",
  "count": 0,
  "errors": [
    "getGenSch.currentAdvisorySchedulesFlag: UPD-10044 ERROR: The field CURRENT_
    ADVISORY_SCHEDULES_FLAG cannot be null."
  ]
}

```

Response Examples

200 OK — Success (truncated to 2 of 1806 bid records; generatorAdvSchedules similarly truncated):

```

{
  "requestStartTime": "2026-04-28T17:09:48.196632Z",
  "requestEndTime": "2026-04-28T17:09:48.209633800Z",
  "nyisoRequestId": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
  "count": 1806,
  "message": "Success",
  "generatorBidAndSchedules": [
    {
      "generatorBid": {
        "generator": {
          "displayName": "EXAMPLE_GEN_A",
          "ptid": 100001
        },
        "bidDate": "2026-04-15T04:00Z",
        "generatorBidMarket": "HAM",
        "bidStatus": "VALIDATION_PASSED",
        "upperOperatingLimit": 369.9,
        "emergencyUpperOperatingLimit": 369.9,
        "scheduleType": "SELF_COMMITTED_FLEXIBLE",
        "selfCommitmentCurve": {
          "selfCommitMw00": 75,
          "selfCommitMw15": 75,
          "selfCommitMw30": 75,
          "selfCommitMw45": 75
        },
        "fixedMinimumGenerationMw": 75,
        "fixedMinimumGenerationCost": 4455,
        "dispatchCurve": {
          "dispatchDollar1": 255,

```

```

    "dispatchMw1": 250
  },
  "bidId": 1000000001,
  "esrBidFields": {
    "esrLowerOperatingLimit": 75
  },
  "genSchedules": [
    {
      "generator": {
        "displayName": "EXAMPLE_GEN_A",
        "ptid": 100001
      },
      "market": "RTD",
      "intervalBeginTimeStamp": "2026-04-15T04:15:00Z",
      "intervalEndTimeStamp": "2026-04-15T04:30:00Z",
      "schedMw": 75,
      "schedTenMinSpinReserve": 35,
      "schedThirtyMinSpinReserve": 0,
      "schedRegulationMw": 0,
      "schedStartUpCost": 0
    },
    {
      "generator": {
        "displayName": "EXAMPLE_GEN_A",
        "ptid": 100001
      },
      "market": "RTD",
      "intervalBeginTimeStamp": "2026-04-15T04:30:00Z",
      "intervalEndTimeStamp": "2026-04-15T04:45:00Z",
      "schedMw": 75,
      "schedTenMinSpinReserve": 35,
      "schedThirtyMinSpinReserve": 0,
      "schedRegulationMw": 0,
      "schedStartUpCost": 0
    }
  ]
},
"currentAdvisorySchedulesFlag": false
},
{
  "generatorBid": {
    "generator": {
      "displayName": "EXAMPLE_GEN_B",
      "ptid": 100002
    },
    "bidDate": "2026-04-15T04:00Z",
    "generatorBidMarket": "HAM",
    "bidStatus": "VALIDATION_PASSED",
    "upperOperatingLimit": 17,
    "emergencyUpperOperatingLimit": 17,
    "scheduleType": "ISO_COMMITTED_FLEXIBLE",

```

```

    "fixedMinimumGenerationMw": 0,
    "fixedMinimumGenerationCost": 0,
    "dispatchCurve": {
      "dispatchDollar1": 92.91,
      "dispatchMw1": 17
    },
    "bidId": 1000000003,
    "esrBidFields": {
      "esrLowerOperatingLimit": 0
    },
    "genSchedules": [
      {
        "generator": {
          "displayName": "EXAMPLE_GEN_B",
          "ptid": 100002
        },
        "market": "RTD",
        "intervalBeginTimeStamp": "2026-04-15T04:15:00Z",
        "intervalEndTimeStamp": "2026-04-15T04:30:00Z",
        "schedMw": 17.5,
        "schedTenMinSpinReserve": 0,
        "schedThirtyMinSpinReserve": 0,
        "schedRegulationMw": 0,
        "schedStartUpCost": 72.55
      }
    ],
    "currentAdvisorySchedulesFlag": false
  },
  "generatorAdvSchedules": [
    {
      "generator": {
        "displayName": "EXAMPLE_GEN_A",
        "ptid": 100001
      },
      "intervalEndTimeStamp": "2026-03-11T00:00:00Z",
      "schedMw": 75,
      "schedTenMinSpinReserve": 35,
      "schedThirtyMinSpinReserve": 0,
      "schedRegulationMw": 0,
      "schedStartUpCost": 0,
      "isAdvisory": true,
      "scheduleSourceType": "RTD Hist"
    },
    {
      "generator": {
        "displayName": "EXAMPLE_GEN_A",
        "ptid": 100001
      },
      "intervalEndTimeStamp": "2026-03-19T23:30:00Z",

```

```

    "schedMw": 180,
    "schedTenMinSpinReserve": 40,
    "schedThirtyMinSpinReserve": 80,
    "schedRegulationMw": 20,
    "schedStartUpCost": 0,
    "isAdvisory": true,
    "scheduleSourceType": "RTC"
  }
]
}

```

200 OK — Error (missing required currentAdvisorySchedulesFlag parameter):

Note: As with other BUD GET endpoints, parameter validation errors are returned as HTTP 200 with count: 0 and the error in errors[].

```

{
  "requestStartTime": "2026-04-28T13:42:03.057002172Z",
  "count": 0,
  "errors": [
    "getGenSch.currentAdvisorySchedulesFlag: UPD-10044 ERROR: The field CURRENT_
    ADVISORY_SCHEDULES_FLAG cannot be null."
  ]
}

```

7.4. CSV Endpoint

7.4.1. POST /csv – Legacy CSV Upload/Download

The /csv endpoint provides **full backward compatibility** with the legacy MIS Upload/Download Batch Procedures described in Section 8 of the Market Participant User’s Guide (MPUG). It is a single POST endpoint that accepts **both upload and download** CSV templates. The BUD system parses the header rows to determine the operation, routes the request to the appropriate internal processing logic, and returns the response as CSV text.

All legacy Market Participant User’s Guide (MPUG) Section 8 rules, field ordering, data dictionaries, and validation rules apply unchanged. The only differences from the legacy MIS system are:

1. **Authentication** — The BUD system uses HTTP Basic Auth (the Authorization header) instead of the USERID/PASSWORD rows embedded in the CSV. If USERID and PASSWORD rows are present in the CSV, their values are **ignored** by the BUD system. They may be included for backward compatibility with existing CSV templates, but they are **not required**.

2. **Transport** — Requests are sent as POST to <https://updown-api.nyiso.com/updown/api/v1/csv> using HTTPS, rather than the legacy MIS batch upload portal.
3. **DATA_ROWS (uploads)** — Present for template compatibility; the system processes all data rows in the body regardless of this value.

Request

HTTP Method: POST **URL:** <https://updown-api.nyiso.com/updown/api/v1/csv>

Headers:

Header	Value	Required
Content-Type	text/csv, text/plain, or application/octet-stream	Yes
Accept	text/csv or application/json	Optional (defaults to CSV)
Authorization	Basic <base64(username:password)>	Yes
TLS Client Certificate	NAESB-compliant .pem/.pfx certificate	Required. Valid NAESB digital certificate linked to the same MIS user account (see Section 2.3).

Request Body: Raw CSV text conforming to the legacy MPUG Section 8 template format.

The endpoint handles two categories of CSV operations:

Upload Operations (BID_TYPE header): Submit, delete, or confirm bids. The request body begins with a BID_TYPE=<operation>& header row.

BID_TYPE Value	Operation	Internal Route
LOAD_BID	Submit load bids	→ POST /load-bid
DELETE_LOAD_BID	Delete load bids	→ POST /delete-load-bid
GEN_BID	Submit generator bids	→ POST /gen-bid
DELETE_GEN_BID	Delete generator bids	→ POST /delete-gen-bid
UC_DATA	Submit unit commitment data	→ POST /uc-data
TRAN_BID	Submit transaction bids	→ POST /tran-bid
DELETE_TRAN_BID	Delete transaction bids	→ POST /delete-tran-bid
CONFIRM_TRAN_BID	Confirm transaction bids	→ POST /confirm-tran-bid

Upload template header format (each line terminated by & and newline):

```
BID_TYPE=<operation>&
  USERID=<ignored>&
  PASSWORD=<ignored>&
  DATA_ROWS=<number>&
  <comma-separated data row 1>
  <comma-separated data row 2>
  ...
```

Additional upload header rows recognized by the BUD system:

Header Row	Applies To	Description
PROCESS_UNIDENTIFIED_BIDS=Y\ N	DELETE_LOAD_BID, DELETE_GEN_BID, DELETE_TRAN_BID, CONFIRM_TRAN_BID	Controls processing of bids that cannot be matched to an identified unit. Same behavior as legacy MIS.

Download Operations (QUERY_TYPE header): Retrieve bids, schedules, details, and reference data.

The request body begins with a QUERY_TYPE=<operation>& header row.

QUERY_TYPE Value	Operation	Internal Route
LOAD_DETAIL	Retrieve load unit details	→ GET /load-detail
LOAD_SCH	Retrieve load bid schedules	→ GET /load-sch
GEN_DETAIL	Retrieve generator unit details	→ GET /gen-detail
GEN_SCH	Retrieve generator bids and schedules	→ GET /gen-sch
GEN_RTD	Retrieve generator RTD schedules	→ GET /gen-rtd
GEN_OUT	Retrieve generator outage declarations	→ GET /gen-out
U_COMM	Retrieve unit commitment parameters	→ GET /u-comm
TRAN_SCH	Retrieve transaction bid schedules	→ GET /tran-sch
TRAN_CONTRACT	Retrieve transaction contracts	→ GET /tran-contract

Download template header format (each line terminated by & and newline):

```

QUERY_TYPE=<operation>&
  USERID=<ignored>&
  PASSWORD=<ignored>&
  <filter parameter 1>=<value>&
  <filter parameter 2>=<value>&
  ...

```

Download filter parameters by QUERY_TYPE:

QUERY_TYPE	Supported Filter Parameters
LOAD_DETAIL	LOAD_AREA_BUS (name or PTID)
LOAD_SCH	DATE, LOAD_AREA_BUS, STATUS, MODIFIED_DATE, NUM_HOURS
GEN_SCH	MARKET_TYPE, DATE, GENERATOR (name or PTID), STATUS, NUM_HOURS, MODIFIED_DATE, CURRENT_ADVISORY_SCHEDULES_FLAG
GEN_RTD	DATE, GENERATOR (name or PTID), NUM_HOURS, MODIFIED_DATE, CURRENT_ADVISORY_SCHEDULES_FLAG
GEN_DETAIL	GENERATOR (name or PTID)
GEN_OUT	GENERATOR (name or PTID), OUTAGES
U_COMM	GENERATOR (name or PTID)
TRAN_SCH	MARKET_TYPE, DATE, TRANSACTION_ID, SOURCE, SINK, STATUS, USER_REFERENCE, NUM_HOURS, MODIFIED_DATE
TRAN_CONTRACT	TRANSACTION_ID, SOURCE, SINK, USER_REFERENCE, NUM_ROWS, SOURCE_ENERGY_ORG_ID, SINK_ENERGY_ORG_ID

Identifier resolution: For GENERATOR, LOAD_AREA_BUS, SOURCE, and SINK parameters, the BUD system accepts either the unit name (string) or the PTID (integer). If the value parses as an integer, it is treated as a PTID; otherwise it is treated as a name. This matches legacy MIS behavior.

Date formats: Date values use the legacy MM/dd/yyyy HH:mm format. On 25-hour fall-back DST days, the repeated 2 AM hour is expressed as 25:00 (e.g., 11/03/2024 25:00). See Market Participant User's Guide (MPUG) Section 8 for full details.

For complete field-level data dictionaries, submission column ordering, and response column ordering for each template type, refer to Market Participant User's Guide (MPUG) Section 8 (available on the NYISO

Website at the following URL: <https://www.nyiso.com/manuals-tech-bulletins-user-guides>.

Response Examples

Upload — Submit Generator Bids (GEN_BID)

```
curl -X POST "https://updown-api.nyiso.com/updown/api/v1/csv" \
  --cert /path/to/naesb-cert.pem --key /path/to/naesb-key.pem \
  -H "Content-Type: text/csv" \
  -H "Authorization: Basic $(echo -n 'username:password' | base64)" \
  --data-binary @- <<'EOF'
  BID_TYPE=GEN_BID&
  USERID=<your-username>&
  PASSWORD=<your-password>&
  DATA_ROWS=2&
  EXAMPLE_GEN_GT1,05/01/2026 00:00,1,DAM,,45.1,45.1,,,999,3,,,,,0,0,45.1,,,,,,,,,66,,,
  ,,,,,,0,,0,,,
  EXAMPLE_GEN_GT1,05/01/2026 01:00,1,DAM,,45.1,45.1,,,999,3,,,,,0,0,45.1,,,,,,,,,66,,,
  ,,,,,,0,,0,,,
  EOF
```

Upload — Delete Generator Bids (DELETE_GEN_BID)

```
curl -X POST "https://updown-api.nyiso.com/updown/api/v1/csv" \
  --cert /path/to/naesb-cert.pem --key /path/to/naesb-key.pem \
  -H "Content-Type: text/csv" \
  -H "Authorization: Basic $(echo -n 'username:password' | base64)" \
  --data-binary @- <<'EOF'
  BID_TYPE=DELETE_GEN_BID&
  USERID=<your-username>&
  PASSWORD=<your-password>&
  DATA_ROWS=1&
  PROCESS_UNIDENTIFIED_BIDS=N&
  ,EXAMPLE_GEN_GT1,05/01/2026 00:00,DAM
  EOF
```

Upload — Submit Load Bids (LOAD_BID)

```
curl -X POST "https://updown-api.nyiso.com/updown/api/v1/csv" \
  --cert /path/to/naesb-cert.pem --key /path/to/naesb-key.pem \
  -H "Content-Type: text/csv" \
  -H "Authorization: Basic $(echo -n 'username:password' | base64)" \
  --data-binary @- <<'EOF'
  BID_TYPE=LOAD_BID&
  USERID=<your-username>&
  PASSWORD=<your-password>&
  DATA_ROWS=1&
  EXAMPLE_VL_NY-CITY,05/01/2026 00:00,,,100,300,300.8,400,,,,,,
  EOF
```

Upload — Submit Unit Commitment Data (UC_DATA)

```
curl -X POST "https://updown-api.nyiso.com/updown/api/v1/csv" \
  --cert /path/to/naesb-cert.pem --key /path/to/naesb-key.pem \
  -H "Content-Type: text/csv" \
  -H "Authorization: Basic $(echo -n 'username:password' | base64)" \
  --data-binary @- <<'EOF'
  BID_TYPE=UC_DATA&
  USERID=<your-username>&
  PASSWORD=<your-password>&
  DATA_ROWS=1&
  EXAMPLE_GEN_GT,1,2,2,0,1,55,65,78,,1100,12000,13000,14000,,0.12,85,95,96,,450,
  550,650,770,,
  EOF
```

Upload — Submit Transaction Bids (TRAN_BID)

```
curl -X POST "https://updown-api.nyiso.com/updown/api/v1/csv" \
  --cert /path/to/naesb-cert.pem --key /path/to/naesb-key.pem \
  -H "Content-Type: text/csv" \
  -H "Authorization: Basic $(echo -n 'username:password' | base64)" \
  --data-binary @- <<'EOF'
  BID_TYPE=TRAN_BID&
  USERID=<your-username>&
  PASSWORD=<your-password>&
  DATA_ROWS=1&
  05/01/2026 00:00,EXAMPLE_SRC_CT2,EXAMPLE_SNK_CE_NY-CITY,DAM,,,,,,S12345,,1
  O
  EOF
```

Upload — Delete Transaction Bids (DELETE_TRAN_BID)

```
curl -X POST "https://updown-api.nyiso.com/updown/api/v1/csv" \
  --cert /path/to/naesb-cert.pem --key /path/to/naesb-key.pem \
  -H "Content-Type: text/csv" \
  -H "Authorization: Basic $(echo -n 'username:password' | base64)" \
  --data-binary @- <<'EOF'
  BID_TYPE=DELETE_TRAN_BID&
  USERID=<your-username>&
  PASSWORD=<your-password>&
  DATA_ROWS=1&
  PROCESS_UNIDENTIFIED_BIDS=N&
  1000000027,90000003,05/01/2026 10:00,DAM
  EOF
```

Upload — Confirm Transaction Bids (CONFIRM_TRAN_BID)

```
curl -X POST "https://updown-api.nyiso.com/updown/api/v1/csv" \
  --cert /path/to/naesb-cert.pem --key /path/to/naesb-key.pem \
  -H "Content-Type: text/csv" \
  -H "Authorization: Basic $(echo -n 'username:password' | base64)" \
```

```
--data-binary @- <<'EOF'
  BID_TYPE=CONFIRM_TRAN_BID&
  USERID=<your-username>&
  PASSWORD=<your-password>&
  DATA_ROWS=1&
  PROCESS_UNIDENTIFIED_BIDS=Y&
  1000000027,90000003,05/01/2026 00:00,DAM,Y
  EOF
```

Download — Generator Bids & Schedules (GEN_SCH)

```
curl -X POST "https://updown-api.nyiso.com/updown/api/v1/csv" \
  --cert /path/to/naesb-cert.pem --key /path/to/naesb-key.pem \
  -H "Content-Type: text/csv" \
  -H "Authorization: Basic $(echo -n 'username:password' | base64)" \
  --data-binary @- <<'EOF'
  QUERY_TYPE=GEN_SCH&
  USERID=<your-username>&
  PASSWORD=<your-password>&
  MARKET_TYPE=DAM&
  DATE=05/01/2026 00:00&
  GENERATOR=EXAMPLE_GEN_GT1&
  CURRENT_ADVISORY_SCHEDULES_FLAG=Y&
  EOF
```

Download — Load Details (LOAD_DETAIL)

```
curl -X POST "https://updown-api.nyiso.com/updown/api/v1/csv" \
  --cert /path/to/naesb-cert.pem --key /path/to/naesb-key.pem \
  -H "Content-Type: text/csv" \
  -H "Authorization: Basic $(echo -n 'username:password' | base64)" \
  --data-binary @- <<'EOF'
  QUERY_TYPE=LOAD_DETAIL&
  USERID=<your-username>&
  PASSWORD=<your-password>&
  LOAD_AREA_BUS=EXAMPLE_VL_NY-CITY&
  EOF
```

Download — Transaction Contracts (TRAN_CONTRACT)

```
curl -X POST "https://updown-api.nyiso.com/updown/api/v1/csv" \
  --cert /path/to/naesb-cert.pem --key /path/to/naesb-key.pem \
  -H "Content-Type: text/csv" \
  -H "Authorization: Basic $(echo -n 'username:password' | base64)" \
  --data-binary @- <<'EOF'
  QUERY_TYPE=TRAN_CONTRACT&
  USERID=<your-username>&
  PASSWORD=<your-password>&
  EOF
```

Download — Generator RTD Schedules (GEN_RTD)

```
curl -X POST "https://updown-api.nyiso.com/updown/api/v1/csv" \  
  --cert /path/to/naesb-cert.pem --key /path/to/naesb-key.pem \  
  -H "Content-Type: text/csv" \  
  -H "Authorization: Basic $(echo -n 'username:password' | base64)" \  
  --data-binary @- <'EOF'  
QUERY_TYPE=GEN_RTD&  
USERID=<your-username>&  
PASSWORD=<your-password>&  
DATE=05/01/2026 00:00&  
GENERATOR=100001&  
CURRENT_ADVISORY_SCHEDULES_FLAG=N&  
EOF
```

File-based upload (alternative syntax):

```
curl -X POST "https://updown-api.nyiso.com/updown/api/v1/csv" \  
  --cert /path/to/naesb-cert.pem --key /path/to/naesb-key.pem \  
  -H "Content-Type: text/csv" \  
  -H "Authorization: Basic $(echo -n 'username:password' | base64)" \  
  --data-binary @gen-bids.csv
```

Response

The /csv endpoint always returns **HTTP 200 OK** with a CSV-formatted text body, regardless of whether the underlying operation succeeded or encountered validation failures. This matches legacy MIS behavior where success and validation errors are communicated within the CSV body itself.

Upload Response Format:

Upload responses begin with a three-line header:

TIME_STAMP=<MM/dd/yyyy HH:mm>

BID_TYPE=<operation>

DATA_ROWS=<count>

Followed by comma-separated data rows containing the same fields as the Market Participant User's Guide (MPUG) Section 8 response template for the given BID_TYPE, including bid status and validation messages in the trailing columns.

Download Response Format:

Download responses begin with a three-line header:

TIME_STAMP=<MM/dd/yyyy HH:mm>

BID_TYPE=<QUERY_TYPE>

DATA_ROWS=<count>

Followed by comma-separated data rows containing the response fields defined in the MPUG Section 8 download template for the given QUERY_TYPE.

Error Responses:

If the request cannot be parsed or does not match any known template, the BUD system returns HTTP 400 with:

Upload/Download Error --

UDF-10018 ERROR: The template does not match any known template types.

If the request is routed to the appropriate handler but fails processing (e.g., data access error, invalid data), the error is prefixed with Upload/Download Error -- followed by the specific error details.

For complete response field ordering and data type definitions for each BID_TYPE and QUERY_TYPE, refer to the corresponding template in Market Participant User's Guide (MPUG) Section 8 available on the NYISO Website at the following URL: <https://www.nyiso.com/manuals-tech-bulletins-user-guides>

Response Examples

Upload Response — GEN_BID (accepted)

TIME_STAMP=05/01/2026 14:30

BID_TYPE=GEN_BID

DATA_ROWS=2

"EXAMPLE_GEN_GT1",100001,"05/01/2026 00:00","Day-Ahead","",45.1,45.1,,,999,3,,,
,0,0,45.1,,,,,,,,,66,,,,,,,,,0,0,1000000001,"Bid Accepted";""

"EXAMPLE_GEN_GT1",100001,"05/01/2026 01:00","Day-Ahead","",45.1,45.1,,,999,3,,,
,0,0,45.1,,,,,,,,,66,,,,,,,,,0,0,1000000002,"Bid Accepted";""

Upload Response — DELETE_LOAD_BID (deleted)

TIME_STAMP=05/01/2026 14:35

BID_TYPE=DELETE_LOAD_BID

DATA_ROWS=1

"EXAMPLE_VL_NY-CITY",100001,"05/01/2026 00:00",,,,,,,,,,1000000015,,,"Bid Deleted",""

Download Response — LOAD_DETAIL

TIME_STAMP=05/01/2026 14:40

BID_TYPE=LOAD_DETAIL

DATA_ROWS=1

"Y","EXAMPLE_VL_NY-CITY","EXAMPLE_LSE","J","NYC","NYC-EDC",100001,"Y","Y","N","N"

Download Response — U_COMM

TIME_STAMP=05/01/2026 14:45

BID_TYPE=U_COMM

DATA_ROWS=1

"EXAMPLE_GEN_GT",100001,1,2,2,0,1,55,65,78,,,1100,12000,13000,14000,,,0.12,85,95,96,,,450,550,650,770,,,100000001,"EXAMPLE_USER","05/01/2026 10:00"

Error Response — Invalid Template

Upload/Download Error --

UDF-10018 ERROR: The template does not match any known template types.

Appendix A: Field Validation Reference

Note: All existing bidding/upload-download rules from the Market Participant User's Guide apply to the BUD system.

Bid Date/Time:

- Must be in ISO-8601 format with timezone offset
- Must be at the top of the hour (minutes = 00)
- Must be within the allowed bidding window for the applicable market
- Must not be in the past (for submission)

PTID:

- Must be a whole number with up to 6 digits
- Must correspond to an active, registered unit

Unit Name (loadName, generatorName, sourceName, sinkName):

- Must contain only alphanumeric characters, underscores (_), hyphens (-), and spaces
- Must correspond to a registered unit name in the MIS

Price Cap Rules (Load Bids):

- Price cap MW and Dollar must both be provided or both be null
- Price cap values cannot be negative
- Price cap MW values must be in ascending order (Cap1 < Cap2 < Cap3)

Dispatch Curve Rules (Generator Bids):

- At least one dispatch MW/Dollar pair is required for ISO Committed and Self Committed Flexible bids
- Dispatch dollar values must be in ascending order
- Dispatch MW values must not exceed the upper operating limit

Generator Business Rules:

- Upper Operating Limit (UOLn) must not exceed the unit's maximum operating limit
- Emergency Upper Operating Limit (UOLe) must be $\geq$ UOLn
- Fixed Minimum Generation MW must be $\geq$ the unit's physical minimum generation MW
- Schedule Type must be valid for the unit type (, certain types require self-committed schedules)

- Reserve cost fields are only allowed for qualified units

Appendix B: Common Error Messages

The table below summarizes the most common error codes and their messages. Codes with the CBM- prefix originate from the marketplace rules engine; UPD- codes originate from the BUD request-validation layer; UDF- codes originate from legacy CSV-format validation; and BUD- codes are reserved for BUD-specific flow errors.

Error Code	Message	Typical Cause
CBM-20014	Bid date is outside the market window	Bid submitted after the market close for the requested hour
CBM-20056	Upper operating limit (UOL) is required	Generator bid is missing upperOperatingLimit
CBM-20067	User is not authorized to bid for this unit	User's org does not own the load/generator/interface
CBM-20083	Price cap MW and Dollar must both be provided or both be null	Incomplete price-cap MW/\$ pair
CBM-20090	Bid rejected: non-utility generator self-committed bid on 25-hour day	Self-flex bid on fall-back DST day where it is not allowed
CBM-22042	Generator ownership mismatch	displayName / ptid not owned by the authenticated user's org
UPD-10002	Invalid date/time format	bidDate does not match ISO-8601 with offset; minutes must be 00
UPD-10006	Missing required field	A field required by the DTO was null or absent
UPD-10007	Invalid username or password	Basic-auth credentials were not accepted
UPD-10010	Market type must be DAM or HAM	market value is not one of the allowed literals
UDF-10002	DATA_ROWS count does not match the number of data rows	CSV header DATA_ROWS=n disagrees with body row count
BUD-190	Delete bid not found	The bidId supplied does not exist or does not belong to your org
(generic)	Bid date and time cannot be null	The bidDate field is missing
(generic)	PTID must be a number with up to 6 integer digits	Invalid PTID format
(generic)	Price cap MW values must be in ascending order	Price caps not monotonically increasing
(generic)	Bid not found	Attempting to delete a non-existent bid

Note: For the authoritative, up-to-date list of marketplace-rule error codes refer to the NYISO Technical

Bulletin Bidding Upload/Download, available via the NYISO document library. The BUD system returns the full code + message in both the JSON `validationErrors{}` object and the CSV row-level status column.

Appendix C: Glossary

Term	Definition
BUD	Bidding Upload/Download — the NYISO’s API system for managing electricity market bids
BTM	Behind-The-Meter — generation resources located behind the customer meter
CSR	Co-located Storage Resource — an energy storage resource co-located with a generator
DAM	Day-Ahead Market — the forward market for scheduling energy and ancillary services for the next operating day
DST	Daylight Saving Time — the seasonal adjustment of clocks; relevant for 25-hour day handling
ESR	Energy Storage Resource — a resource capable of storing and discharging energy
HAM	Hour-Ahead Market — the market for scheduling energy closer to real-time
MIS	Market Information System — the NYISO’s web-based system for Market Participant interaction
MPUG	Market Participants User’s Guide — the comprehensive NYISO guide for Market Participant operations available at www.nyiso.com
MW	Megawatt — unit of electric power
MWh	Megawatt-hour — unit of electric energy
NAESB	North American Energy Standards Board — the organization that issues digital certificates for energy market authentication
PTID	Point Identifier — a unique numeric identifier for a generation or load resource point
RTD	Real-Time Dispatch — the real-time market dispatch process
TLS	Transport Layer Security — the encryption protocol used for secure HTTPS connections
UC	Unit Commitment — the parameters governing how a generator starts up, runs, and shuts down
UOL	Upper Operating Limit — the maximum MW output a generator can supply
UOLe	Emergency Upper Operating Limit — the maximum MW output under emergency conditions