AQDx Standard Format Guidance

Version 3.0 - January 2025

Colorado Department of Public Health and Environment (CDPHE)

Table of contents

1. Air Quality Data Exchange (AQDx) Standard Format Guidance

Version 3.0 - February 2026

Download the Full Specification (PDF)

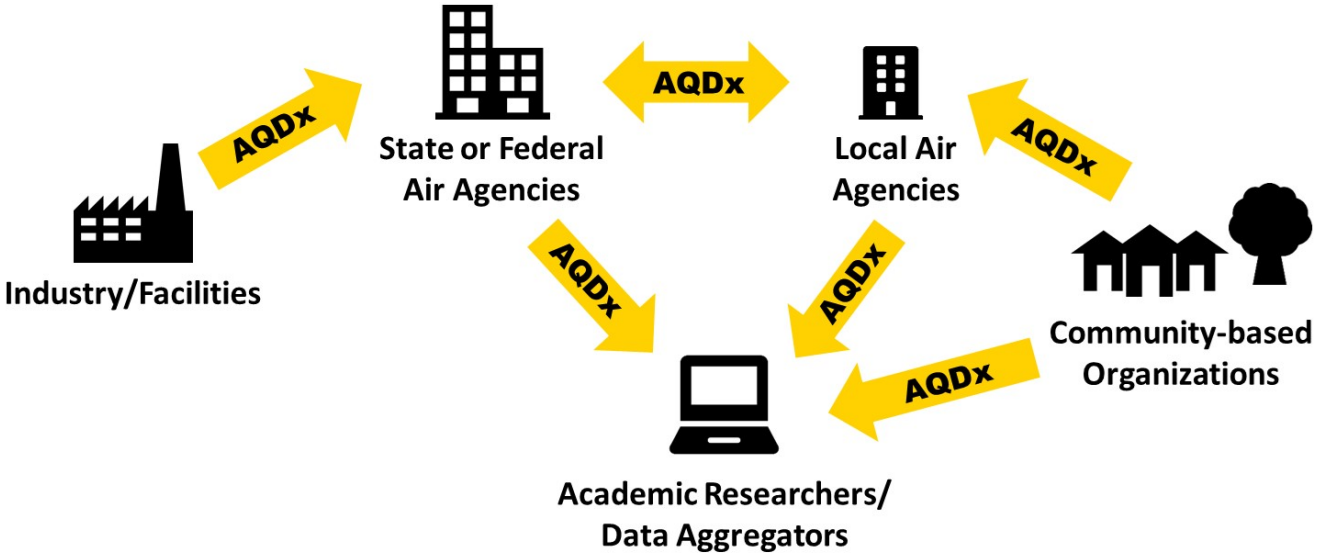

Figure 1. Data exchange using AQDx between different organizations and agencies.

1.1 Introduction

Air monitoring is fundamentally changing. With new technology emerging, a wide range of organizations and individuals can now measure air pollution. Additionally, more organizations want to collect and aggregate these new datasets. With all these advances, the potential exists for organizations to integrate and harness these data to create transformative and systemic change. Many challenges exist for organizations collecting air quality data such as exchanging, integrating, harmonizing, and using these new data sources. There are too many data formats and ways to name parameters, little consistency in time formatting, various ways to indicate data quality, and more. This results in data that are more expensive to manage, harder to exchange, and not fully utilized. Universal methods are urgently needed for describing and exchanging data between organizations and their data management systems. Establishing standard parameter names, conventions for time reporting, and data quality levels will make it easier for organizations to collect and exchange data with many other organizations. Figure 1 shows how data exchange between organizations/systems needs a common format that describes the data, its collection, and its quality.

1.1.1 Purpose and intended use

The Air Quality Data eXchange (AQDx) format is intended for:

- Enabling the exchange of air quality and weather data between organizations
- Exchanging of both historical and real-time data
- Establishing standardized naming conventions for parameters, units, etc.
- Documenting data so it is self-describing with location, device, organization, and quality levels
- Enabling open and public exchange of air quality data

1.2 What is AQDx?

AQDx is not a piece of software, a database, or a specific file format. **It is a schema.**

By adhering to this schema, data becomes self-describing. Any person or software program that understands AQDx can instantly read, map, and visualize air quality data without needing to ask for the experimental specifics of an individual dataset.

1.2.1 The Two Components of the Standard

AQDx splits information into two distinct files to handle different types of information efficiently. To ensure clarity across all documentation, we use the following controlled vocabulary:

1. **AQDx data (The Measurements)** Strict, structured files that hold the actual numbers, dates, and codes. Each record contains a single air quality measurement paired with a timestamp and location.
   - **AQDx datasheet:** Measurements formatted as a static table (e.g., CSV, Parquet).
   - **AQDx datastream:** Measurements formatted as a continuous flow (e.g., JSON).
2. **AQDx metadata (The Context)** The experiment-level details that accompany the data. This is a structured YAML document (typically generated from a 3-tab spreadsheet) that captures the "who, where, and how" such as project ownership, site configurations, and instrument QA/QC procedures.

1.2.2 What AQDx Specifies

- **Identity:** Who collected the data (`data_steward_name`) and with what device (`device_id`).
- **Time:** When it happened, using strict ISO 8601 formatting to eliminate timezone confusion.
- **Location:** Where it happened (`latitude` , `longitude`).
- **Measurement:** What was found (`parameter_value`) and the unit used (`unit_code`).
- **Quality:** How reliable the data is (`validity_code` , `review_level_code`).

1.2.3 What AQDx Does Not Specify

- **Storage Technology:** You can store AQDx data in SQL, NoSQL, data lakes, or simple spreadsheets. The schema is technology-agnostic.
- **File Size:** Whether you have ten rows of data or ten billion, the rules for naming your columns remain the same.

1.3 Implementations

While the core AQDx schema is format-agnostic, we provide strict specifications for how to apply it to common file formats used in the real world:

- **Tabular (CSV, Excel, Parquet):** For historical data, bulk uploads, and archives.
- **JSON:** For real-time streaming, APIs, and web applications.

The AQDx format was built upon the successful AirNow Air Quality Comma Separated Values (AQCSV) format. It is developed and maintained by the Colorado Department of Public Health and Environment (CDPHE) with input from the U.S. Environmental Protection Agency (EPA) and partners in the academic and community science sectors.

2. Quick Start Guide

Ready to format your data? A valid AQDx data package consists of exactly **two** files that work together. This guide provides the minimum requirements to get started.

2.1 The Two Components

A complete AQDx submission consists of two files:

1. **The Data File** (CSV, Parquet, or JSON) containing the actual measurements.
2. **The Metadata File** (YAML) containing context about who you are, where the site is, and how the data was collected.

> These two files are linked together by a unique **`dataset_id`**. You must generate this ID and place it in *every row* of your data file and at the *top* of your metadata file.

2.2 Step 1: Create the Data File

Most users will use the **Tabular (CSV)** format. Your file must adhere to these strict rules:

2.2.1 1. Headers are Mandatory

Your file **must** include every single column header listed in the Field Dictionary, even if you don't have data for that field.

- *Example:* If you don't measure `elevation`, you must still have an `elevation` column header, and leave the cells below it empty.

2.2.2 2. Exact Naming

Column headers must match the standard **exactly** (case-sensitive).

- ✅ `device_id`
- ❌ `Device ID`, `deviceID`, `Serial_Number`

2.2.3 3. Strict Data Types

Values must match the defined Data Types:

- **Dates:** Must be ISO 8601 with time zone offsets (e.g., `2024-05-23T14:00:00-07:00`).
- **Numbers:** No commas or units in the cell (e.g., `1500`, not `1,500` or `1500ppb`).
- **Missing Data:** Leave the cell empty. Do not use `-999`, `NA`, or `Null` strings.

2.3 Step 2: Create the Metadata File

The metadata provides the context that makes your numbers meaningful.

1. **Download the Template:** AQDx_metadata_form_v3.yaml.
2. **Edit in Text Editor:** Open the file in a text editor (e.g., VS Code, Notepad++, or standard Notepad).
3. **Fill Required Fields:**
   - **Data Steward:** Contact info and the `dataset_id` (must match your CSV).
   - **Site Info:** Location name and coordinates.
   - **Instrument Info:** Details on the hardware (nested under the Site).
4. **Save:** Save as a `.yaml` file alongside your data.

3. Data Field Definitions

3.1 Field Dictionary

This page defines the standard vocabulary for the AQDx format. Regardless of whether you are using a tabular file (CSV/Excel) or a JSON stream, these field names and data types serve as the single source of truth.

3.1.1 1. Time & Measurement

These fields define *what* was measured, *when* it was measured, and *how much* was found.

Field Name	Data Type	Value Required	Description
datetime	ISO 8601 String (29)	Yes	The date and time of the measurement (start of the sampling period).
parameter_code	String (5)	Yes	The 5-digit AQS code identifying the pollutant or variable.
parameter_value	Decimal (12,5)	No	The actual measured value.
unit_code	String (3)	Yes	The 3-digit AQS code identifying the unit of measure.
method_code	String (3)	No	The 3-digit code for the measurement method.
duration	Decimal (12,3)	Yes	The duration of the sample in seconds.
aggregation_code	Integer (1)	Yes	Indicates the mathematical or physical method used to represent the data over time.

datetime

Format: ISO 8601 String (29) **Example:** `2008-10-08T12:00:43-06:00`

The date and time of the data value. It must follow the "Date and time with the offset" ISO 8601 format `YYYY-MM-DDThh:mm:ssTZD`, where `TZD` is the Time Zone Designator (offset from UTC). See also: https://en.wikipedia.org/wiki/ISO_8601

- **Precision:** For data reporting faster than 1Hz (less than one second), report seconds with a decimal (ss.sss). The maximum allowed precision is milliseconds, translating to a maximum allowed string length of 29 characters.
- **Timing:** The timestamp corresponds to the **beginning** of the averaging or sampling period.
- **Time Zone:** Must include the offset (e.g., `-06:00` for CST, `+00:00` for UTC). Do not use "Z" for UTC.
- see more details in Data Types & Conventions

parameter_code

Format: String (5) **Example:** `44201` (Ozone)

A 5-digit numerical code that identifies the parameter being measured. These codes are based on the EPA's Air Quality System (AQS) parameter library.

- **Common Codes:**
- `44201` : Ozone (O3)
- `88101` : PM2.5 - Local Conditions
- `61101` : Wind Speed
- **Note:** Only list one parameter code per data record.
- **Note:** Parameter code **cannot be blank**, if a parameter code or method_code is missing for your project, please open a github issue to have a code or codes added.
- View Parameter Codes

parameter_value

Format: Decimal (12,5) **Example:** `35.5`

The actual data value of the specified parameter.

- **Precision:** Round data to the 5th decimal place if the measured value has larger precision than 5 decimal places.
- **Formatting:** Do not use commas (e.g., use `1500`, not `1,500`).
- **Can be blank:** Leave the field blank if the measurement could not be taken or is missing.
- **Zero vs. Null:** distinct meanings must be preserved.
- `0.0` : A valid measurement indicating zero concentration.
- **Blank / Null**: The absence of a measurement (e.g., power failure, maintenance, sensor error).
- **Validation Rules:**
- If `parameter_value` is blank, the `validity_code` must be **9** (Invalid/Missing) for processed data or **0** (Not Validated) for raw data.
- If `parameter_value` is blank, it is recommended to provide a `qualifier_code` (e.g., `AM` for Miscellaneous Void) to explain the missing data.

unit_code

Format: String (3) **Example:** `008` (ppb)

A 3-digit code associated with the units of the measurement.

- **Common Codes:**
- `008` : Parts per billion (ppb)
- `001` : Micrograms/cubic meter (µg/m³) at 25°C
- `105` : Micrograms/cubic meter (µg/m³) at Local Conditions
- `017` : Degrees Centigrade (°C)
- View Unit Codes

method_code

Format: String (3) **Example:** `170` (Met One BAM-1020)

A 3-digit code associated with the reference method used to perform an EPA-designated FRM or FEM measurement.

- `Value Required` Conditional
- Regulatory Instruments (FRM/FEM): **Required.** If the instrument is an EPA Federal Reference Method (FRM) or Federal Equivalent Method (FEM), you **must** provide the specific 3-digit code defined by the EPA (e.g., `170` for BAM-1020).
- Low-Cost Sensors / Non-Regulatory: Leave Blank. If the device is a low-cost sensor or has not been EPA-designated, leave this field blank.
- View Method Codes

duration

Format: Decimal (12,3) **Example:** `3600` or `1.500`

The duration of the sampling period or mathematical aggregation window (see `aggregation_code` below) in seconds.

General Guidelines

- **Integers Preferred:** For standard intervals, use whole numbers without decimal padding (e.g., use `3600`, not `3600.000`).
- **Precision:** Fractional seconds are allowed up to milliseconds (3 digits after the decimal point) if high-precision timing is required.
- **Variable Duration:** If a sensor's physical sampling duration varies slightly row-to-row (e.g., fluctuating between 90 and 92 seconds), you should use a consistent, approximated nominal duration (e.g., `90`) for the entire dataset to reduce computational burden and make the data easier to query and compare.
- **Instantaneous / Unknown:** Use `0` to explicitly flag a measurement where the duration is near-instantaneous, highly inconsistent (sub-minute), or completely unknown. This specifically designates the data as coming from a low-cost sensor rather than a precision regulatory or research grade monitor.

For long-term aggregations (like months or years), **standard generalized timeframes are recommended** to maintain consistency across leap years and varying month lengths, unless the exact physical duration of a specific period is required.

Common Duration Values:

- `0` = Instantaneous / Unknown (Low-Cost Sensor flag)
- `60` = 1 Minute
- `3600` = 1 Hour
- `86400` = 1 Day (24 Hours)
- `604800` = 1 Week (7 Days)
- `2592000` = 1 Month (Standardized 30-Day Period)
- `31536000` = 1 Year (Standardized 365-Day Period)

aggregation_code

Format: Integer (1) **Example:** `1` (Mean)

Indicates the mathematical or physical method used to represent the data over the specified `duration`.

- `0`: **None / Native Resolution.** The data is reported at the instrument's native sampling frequency.
- `1`: **Mean (Average).** The mathematical average of measurements over the specified `duration`.
- `2`: **Time-Integrated (Physical).** A single physical sample accumulated over the `duration` (e.g., a 24-hour PM filter or VOC canister).
- `3`: **Maximum.** The highest single value recorded within the `duration`.
- `4`: **Median (50th Percentile).** The middle value of the measurements within the `duration`.
- `5`: **Rolling / Moving Average.** A mathematical average calculated over a moving look-back window (e.g., an 8-hour rolling ozone average).
- `6`: **Spatial Aggregation.** Data grouped by a geographic boundary rather than strictly by time (e.g., binning mobile data into 50-meter road segments).
- Specific details of the method used must be documented in the accompanying AQDx metadata form.
- For `duration`, report the total integration time (sum of durations) of all observations included in the spatial bin.
- `7`: **Other.** Any aggregation method not listed above, including specific statistical percentiles (e.g., 90th, 98th). Specific details of the method used must be documented in the accompanying AQDx metadata form.

3.1.2 2. Location

These fields define *where* the measurement was taken.

Field Name	Data Type	Value Required	Description
latitude	Decimal (9,5)	Conditional	Latitude in WGS84 decimal degrees.
longitude	Decimal (9,5)	Conditional	Longitude in WGS84 decimal degrees.
elevation	Decimal (8,2)	No	Elevation of the device in meters.

latitude

Format: Decimal (9,5) **Example:** `39.7392`

Latitude in decimal degrees (WGS84).

- **Positive:** North of the Equator.
- **Negative:** South of the Equator.
- **Precision:** Report to the 5th decimal point (~1 meter precision).
- **Conditional:** `latitude` is required except in the following circumstances:
- **Mobile Monitoring Exception:** `latitude` may be left blank due to a temporary loss of GPS fix on a mobile platform, but you must include the `IG` (GPS invalid) code in the `qualifier_codes` field.

longitude

Format: Decimal (9,5) **Example:** `-104.9903`

Longitude in decimal degrees (WGS84).

- **Positive:** East of the Prime Meridian.
- **Negative:** West of the Prime Meridian (e.g., USA).
- **Precision:** Report to the 5th decimal point.
- **Conditional:** `longitude` is required except in the following circumstances:
- **Mobile Monitoring Exception:** `longitude` may be left blank due to a temporary loss of GPS fix on a mobile platform, but you must include the `IG` (GPS invalid) code in the `qualifier_codes` field.

elevation

Format: Decimal (8,2) **Example:** `1609.3`

Elevation of the device in meters above mean sea level (MSL). Can be left blank.

3.1.3 3. Device & Organization

These fields define *who* collected the data and *with what* hardware.

Field Name	Data Type	Value Required	Description
data_steward_name	String (64)	Yes	The organization responsible for the data.
device_id	String (64)	Yes	An internal identifier used by the data steward.
measurement_technology_code	String (14)	Yes	categorizes the physical measurement technology of an instrument.
instrument_classification	Integer (1)	Yes	Regulatory standing or operational tier of the instrument.
dataset_id	String (128)	Yes	Unique identifier to connect dataset to metadata form.

data_steward_name

Format: String (64) **Example:** `CityOfDenver` or `city_of_denver`

Name of the party responsible for data oversight.

- **Formatting:** Use PascalCase or snake_case to separate words.
- **Forbidden:** Do not use commas, spaces, or periods.

device_id

Format: String (64) **Example:** `A123-Sensor-01`

An internal identifier used by the data steward to uniquely distinguish this specific instrument within the dataset. Its primary purpose is to link the measurements in the data file to the instrument's details in the accompanying metadata form.

- **Internal Identification:** This is a localized text field to differentiate measurements. It is not intended to be a globally searchable, standardized hardware ID.
- **Recommended Convention:** We recommend using a combination of `[device model]-[ID#]-[sensor type or operating principle]`. The `ID#` can be an internal project number, a device serial number, or a device MAC address.
- **Example:** `atmotube_pro_01_ls` (where "ls" stands for light scattering)
- **Example:** `nodeA_macaddress_pms5003`
- **Other Valid Formats:** A simple hardware serial number, MAC address, or custom project ID (e.g., `Monitor_1`) are also acceptable.
- **Allowed Characters:** Spaces and hyphens.
- **Forbidden:** Do not use commas or periods.

measurement_technology_code

Format: String (14) **Examples:** `DA-00-SC`, `ICsu-GCca-MSpt`, `RS-00-OP`

A structured, hierarchical code that chronologically categorizes the physical journey of a sample from acquisition to the final signal.

- **Acquisition:** How the sample is acquired (e.g., in-situ, canister, remote sensing, etc.)
- **Conditioning:** The most significant treatment step applied to the sample (e.g., gas chromatography, de-humidification, thermal desorption, etc.)
- **Detection:** The actual method of detection (e.g., mass spectrometry, PID sensor)

Code Structure: `[Acquisition]-[Conditioning]-[Detection]` Each of the three steps requires a 2-character broad uppercase code (`XX`). You can optionally append two lowercase characters (`xx`) to designate a specific hardware subtype (e.g., `ICsu` for Integrated Canister, Summa). The blocks must be separated by hyphens.

Key Rules:

- **System Boundary:** The code describes the *end-to-end* measurement system. For integrated or passive methods, it includes both the field acquisition AND the downstream laboratory analysis. Deep analytical nuances belong in the accompanying **AQDx Metadata Form (YAML)**.
- **The "00" Bright Line:** Use `00` for the Conditioning block ONLY if no intentional physical or chemical transformation occurred before the detector. If the system intentionally changes humidity, removes interferents, or selects a size fraction, it is *not* `00`.
- **Conditioning Priority:** If multiple conditioning steps exist, encode the one that most constrains what physically reaches the detector (e.g., a size cut or preconcentration) and document the rest in the YAML metadata form.

*Note: You must use approved vocabulary. Please refer to the **Measurement Technology Codes Lookup Table** in the code lookup tables to find the exact tokens permitted for your setup.*

instrument_classification

Format: Integer (1) **Example:** `3`

Indicates the objective regulatory standing or operational tier of the instrument generating the data.

Allowed Values:

- `1` = **FRM / FEM (Regulatory Designated):** The instrument is operating under a formal, active designation from a recognized environmental authority (e.g., the US EPA) for the specific parameter being reported. *Note: If this code is used, the exact FRM/FEM designation should ideally be recorded in the* `method_code` *field if applicable.*
- `2` = **Research-Grade Analytical Monitor:** High-fidelity instruments or methods that do not hold a formal regulatory designation but are widely accepted for rigorous scientific study. This includes advanced non-designated continuous monitors, as well as physical samples collected in the field and transported to a laboratory for discrete analytical analysis (e.g., GC/MS on canisters, XRF on filter tape).
- `3` = **Consumer-Grade Monitor:** Continuous monitors or indicative devices that actively measure ambient air but lack formal regulatory designation or research-grade analytical rigor. These devices are highly valuable for spatial mapping, identifying local trends, and supplemental public awareness.

dataset_id

Format: String (128) **Example:** `CDPHE_DowntownStation_20260213` or `123e4567-e89b-12d3-a456-426614174000`

A unique identifier that explicitly links this specific row of data to its corresponding AQDx dataset-level metadata file (e.g., `AQDx_metadata_form_v3.yaml`). This exact string must be present on every row of the tabular data file and must perfectly match the `dataset_id` field defined at the top of the accompanying metadata file.

- **Forbidden:** Do not use spaces, commas, or special characters other than hyphens (`-`), underscores (`_`), and periods (`.`).

To ensure global uniqueness across the AQDx ecosystem without relying on a central registry, data creators must generate this ID using one of the following three approved methods.

- **Method 1: Semantic Namespace (Recommended).** Create a self-documenting, human-readable string by combining your organization's metadata fields with high-resolution temporal or spatial identifiers.
- *Formula:* `[data_steward_name]_[project_or_device_id]_[YYYYMMDD]`
- *Single Sensor Example:* `CleanAirVision_A123-Sensor-01_20260213`
- *note*: if you are submitting a dataset with multiple sensors, use project notation below.
- *Network/Project Example:* `CDPHE_WinterInversion_20260213`
- **Method 2: UUID v4.** Generate a standard Universally Unique Identifier. This is ideal for automated, programmatic data pipelines.
- *Tooling:* Specialists can generate these natively in Python (`import uuid; uuid.uuid4()`) or R (`uuid::UUIDgenerate()`).
- *Web Generator:* If generating manually, use a trusted standard generator such as https://www.uuidgenerator.net/version4.
- *Example:* `123e4567-e89b-12d3-a456-426614174000` **(do not use this uuid!)**
- **Method 3: External DOI / URI.** If the dataset is published to an academic or government repository (e.g., Zenodo, Dataverse), use its assigned permanent identifier (DOI) or record number.
- *Example:* `10.5281/zenodo.1234567`

3.1.4 4. Quality Control (QC)

These fields describe the quality and processing level of the data.

Field Name	Data Type	Value Required	Description
validity_code	Integer (1)	Yes	The assessed validity of the individual measurement.
calibration_code	Integer (1)	Yes	Indicates whether the data has been corrected or calibrated against a known standard.
review_level_code	Integer (1)	Yes	Indicates the level of human review the dataset has undergone.
detection_limit	Decimal (12,5)	No	Detection limit for the method used to measure `parameter_value`.
qualifier_codes	String (254)	No	Space-separated codes explaining why data was flagged or describing specific events.

validity_code

Format: Integer (1) **Example:** `0` (Valid)

The assessed validity of the individual measurement. Validation extends beyond simple statistical outlier detection; it evaluates physical limits, hardware faults, "sticking" (unchanging) values, sensor degradation, and data completeness.

- `0`: **Validation not performed.** Raw data directly from the device. No QC checks have been applied to verify if a blank value is a true outage or a transmission error.
- **Note:** Use this code for gaps in raw, real-time streams (`parameter_value` is blank) where no post-processing has occurred
- `1`: **Valid.** Data passed all QC checks and is considered accurate for analysis.
- `3`: **Estimated.** Data is considered valid, but the value was mathematically derived or interpolated rather than directly measured at this exact timestamp.
- `5`: **Suspect.** Data is physically possible but exhibits anomalous behavior (e.g., unexplained spikes, deviation from neighboring sensors, or operation during extreme weather). There is insufficient evidence to invalidate it entirely, but it should be used with caution.
- `8`: **QA/QC data.** Legitimate measurements taken during quality control procedures, such as zero/span checks, flow audits, or calibration events. While these values are "valid" representations of the instrument's response to a reference standard, they do not represent ambient air quality and **must be excluded** from environmental statistics (e.g., daily averages, AQI calculations).
- `9`: **Invalid or Missing.** Data that should not be used for analysis. Includes missing values (e.g., power failures, maintenance gaps, lost data), instrument malfunctions, failed range checks, or data failing completeness criteria (e.g., insufficient uptime for an hourly average).
- If `parameter_value` is blank in a processed dataset, this code must be used.

calibration_code

Format: Integer (1) **Example:** `2` (Formally Verified)

Indicates the level of rigor and documentation of any post-processing corrections or calibrations applied by the Data Steward *after* the data was output by the instrument. This field tracks human-applied adjustments to the `parameter_value`, distinct from any internal processing performed by the sensor firmware.

- `0`: **None / Factory Default.** The data is reported exactly as output by the device using the manufacturer's default factory calibration. No post-collection mathematical adjustments have been made.
- `1`: **Ad-Hoc / Project-Specific.** The data was mathematically adjusted using a custom, localized, or project-specific method. While the method may be highly effective for the specific project, it has not been formally vetted through a standardized regulatory or peer-review process.
- `2`: **Formally Verified (Math/Model).** The data was corrected using a robust, widely accepted methodology. To qualify for this code, the method must be explicitly documented in a Quality Assurance Project Plan (QAPP), derived from or available in a peer-reviewed scientific publication, or accepted for use by a government environmental agency (e.g., the US EPA's extended U.S.-wide correction for PurpleAir sensor data).
- `3`: **Physical Reference Standard.** The instrument was directly calibrated against a certified physical reference standard or secondary standard (e.g., physically adjusted using National Institute of Standards and Technology (NIST) traceable zero-air and span gas checks).

review_level_code

Format: Integer (1) **Example:** `1` (Internal Review)

Indicates the level of human review the dataset has undergone.

- `0`: **Raw.** Direct from device, no human review.
- `1`: **Internal.** Reviewed by the data creator/project team.
- `2`: **External.** Audited by an independent third party.
- `3`: **Certified.** Legally certified for regulatory use (requires FRM/FEM).

detection_limit

Format: Decimal (12,5) **Example:** `0.50000`

- The detection limit for the measurement, expressed in the same units as `parameter_value` (i.e., the record's `unit_code`).
- This field should be left blank/omitted when a detection limit is unknown or not applicable.
- Please note the method used to determine the detection limit in the metadata form included with the submission.

qualifier_codes

Format: String (254) **Example:** `IM`

Space-separated codes explaining why data was flagged or describing specific events.

- **Examples:**
- `IM` (Prescribed Fire)
- `LJ` (High Winds)
- `AA AG BG ND` (Multiple qualifier codes in one measurement)
- ` ` (Can be blank)
- View Qualifier Codes

3.2 Data Types & Conventions

This page defines the specific data formats used across the AQDx standard. Strict adherence to these types ensures data can be parsed correctly by any system, regardless of whether it is transmitted via CSV, Excel, or JSON.

3.2.1 Core Data Types

Data Type	Definition	Tabular CSV Example	JSON Example
String (n)	Alphanumeric or numeric only text with a maximum length of *n* characters.	`...,cdphe_apcd_atops,...` `...,008,...`	`{"data_steward_name":"cdphe_apcd_atops"}` `{"unit_code":"008"}`
Integer (n)	Whole number with a maximum length of *n* digits. No decimals.	`...,1,...`	`{"aggregation_code":1}`
Decimal (p,s)	A fixed-point number with precision *p* and scale *s*. Example: (5,3)	`...,45.231,...`	`{"parameter_value":45.231}`

String (n)

Format: `String (n)` **Definition:** A text value with a maximum length of *n* characters.

Numeric string: A "numeric string" is a String that consists only of digits `0–9` but MUST still be represented as a String because leading zeros are meaningful.

- Example: unit codes like `008` must be transmitted/stored as `"008"` (a String) and not as `8` (an Integer), otherwise the code changes meaning.

Allowed

- Any value whose character length is `<= n`.
- Digit-only values (numeric strings) such as `008`, `000`, `01234`, when the field definition requires a String (common for codes).

Not allowed

- Values longer than *n* characters.
- Automatically converting digit-only code strings into numbers (e.g., turning `008` into `8`).
- Using Strings to store true numeric measurements when the Field Dictionary specifies `Decimal (p,s)` (e.g., `value="45.2"` is not allowed when `value` is Decimal).

Integer (n)

Format: `Integer (n)` **Definition:** Zero or a positive whole number with up to *n* digits. Integers do not have a decimal point.

Note on Codes: Fields defined as `Integer (1)` (like `validity_code`) are categorical flags. Even though they look like numbers (0, 1, 2), they must be treated as exact integers. `1.0` is not a valid status code; `1` is.

Allowed

- `0` through the maximum value representable with *n* digits (e.g., `Integer (1)` allows `0..9`).
- JSON numbers without decimals (e.g., `{"aggregation_code":1}`).

Not allowed

- Decimal points (e.g., `1.0`, `3.14`).
- Thousands separators or commas (e.g., `1,000`).
- Using Integer when the Field Dictionary defines a code as `String (n)` (for example, codes that may have leading zeros).

Decimal (p, s)

Format: `Decimal (p, s)` **Definition:** A fixed-point number where:

- **p (precision):** The total number of digits allowed (left + right of the decimal).
- **s (scale):** The number of digits allowed to the right of the decimal.

This type is used for all measured quantities (e.g., `parameter_value`, `latitude`, `duration`).

CHECKING YOUR DATA AGAINST DECIMAL (P, S)

The notation `Decimal (p, s)` defines the exact space available for your number. To ensure your data fits, perform these two checks:

1. The "Left Side" Check (Magnitude) Ensure your number isn't too large. The maximum number of digits allowed to the *left* of the decimal point is `p - s`.

- **Example:** For `Decimal (8, 3)`, you have `8 - 3 = 5` allowed digits to the left.
- *Fits:* `12345.123` (5 digits left).
- *Too Large:* `123456.123` (6 digits left).

2. The "Right Side" Check (Precision) Ensure your number isn't too precise. The `s` value defines the maximum number of digits allowed to the *right* of the decimal point.

- **Rule:** If your instrument reports more decimal places than `s`, you must round the value.
- **Example:** For `Decimal (12, 5)`, if you have `45.123456`:
- *Action:* Round to the 5th decimal place.
- *Result:* Submit `45.12346`.

Common AQDx Scenarios:

- **Coordinates (`Decimal (9, 5)`):**
- *Left Side:* `9 - 5 = 4` digits allowed. This comfortably fits Longitude (e.g., `-180` uses 3 digits).
- *Right Side:* Round long GPS decimals (e.g., `-105.1234567`) to 5 places (`-105.12346`).
- **Measurements (`Decimal (12, 5)`):**
- *Left Side:* `12 - 5 = 7` digits allowed. This fits values up to `9,999,999`.
- *Right Side:* Round high-precision sensor readings to 5 decimal places.

SYNTAX & FORMATTING RULES

- **No Commas:** Never use thousands separators (e.g., use `1500.0`, not `1,500.0`).
- **No Scientific Notation:** Use plain fixed-point notation (e.g., use `0.00015`, not `1.5e-4`).
- **Optional Signs:** A negative sign (`-`) is allowed and does *not* count toward precision (`p`).
- **Optional Decimal Point:** Whole numbers are valid (e.g., `85` is accepted as `85.0`).

3.2.2 Specific Guidance

Date & Time (`datetime`)

All timestamps are stored as Strings but must follow the **ISO 8601** extended format.

- **Format:** `YYYY-MM-DDThh:mm:ssTZD`
- **Time Zone:** You must include a Time Zone Designator (TZD).
- **UTC:** Ends in `+00:00`. "Z" Notation is not allowed.
- **Offset:** `+hh:mm` or `-hh:mm` (e.g., `-07:00` for MST).
- Note: Unlike AQS, local standard time is not a requirement (daylight time may be used), but you must use the correct TZD
- **Precision:** Seconds are required. Decimals for partial seconds (`ss.sss`) are allowed up to milliseconds but not required.
- **24-Hour Clock:** Use `14:00`, not `2:00 PM`.

Examples:

- ✅ `2024-05-23T14:30:00-07:00` (Local time with offset)
- ✅ `2024-05-23T14:30:00.343-07:00` (Local time with offset - up to milliseconds are allowed)
- ❌ `2024-05-23T21:30:00Z` (UTC using "Z" notation is not allowed - use `2024-05-23T21:30:00+00:00` instead)
- ❌ `2024-05-23 14:30:00` (Missing "T" and Time Zone)

Required vs. Optional Fields ("Value Required")

1. REQUIRED FIELDS (`Value Required : Yes`)

These fields define the core identity of the record (e.g., `datetime`, `device_id`, `dataset_id`).

- **Tabular (CSV):** The cell **must** contain a value. It cannot be empty between commas.
- **JSON:** The key **must** exist, and the value **cannot** be `null`.

2. OPTIONAL FIELDS (`Value Required : No`)

These fields provide extra context that may not always be available (e.g., `method_code` for non FRM/FEM, or `elevation`).

- **Tabular (CSV):** The column header **must still exist**. Do not delete the column. Leave the cell completely empty between the commas.
- *Correct:* `44201,,45.2`
- *Incorrect:* `44201,NA,45.2` (See "Forbidden Placeholders" below)
- **JSON:** You may omit the key entirely, or send the key with a `null` value.
- *Preferred:* `{"parameter": "44201", "value": 45.2}` (Key omitted)
- *Allowed:* `{"method_code": null}`

FORBIDDEN PLACEHOLDERS (NULL VALUES)

Air quality researchers often use specific codes to indicate missing data. **Do not use these in AQDx.** The only valid representation of a missing value is an empty cell (CSV) or `null` (JSON).

Format	❌ Invalid (Do Not Use)	✅ Valid
Text	`"NA"`, `"N/A"`, `"null"`, `"Missing"`	(Empty)
Numeric	`-999`, `-9999`, `NaN`	(Empty)
Empty Strings	`""`, `" "`	(Empty)

Quotation Marks

Strict adherence to quotation rules ensures compatibility across parsers.

- **Double Quotes (`"`)**
- **JSON: Required.** All keys and string values *must* be wrapped in standard double quotes (e.g., `{"parameter": "44201"}`).
- Note that numeric values do not need quotes `{"parameter_value": 0.00232}`
- **CSV: Allowed.** Use standard double quotes to enclose fields if necessary.
- **Single Quotes (`'`)**
- **Not Allowed.** Do not use single quotes to wrap strings or keys in either format.
- **Smart / Curly Quotes (`“ ”`)**
- **Not Allowed.** These characters (often auto-formatted by text editors like Word) will cause parsing errors. Always use standard "straight" Double Quotes.

4. Data File Formats

4.1 Tabular Data Files (CSV, Excel, Parquet)

The Tabular format is the standard method for exchanging historical datasets, batch uploads, and archival data. In this format, every record is a row, and every field is a column.

AQDx supports the following file types:

- **CSV** (`.csv`): Comma-Separated Values (Universal standard).
- **Excel** (`.xlsx`): Microsoft Excel Open XML Spreadsheet.
- **Parquet** (`.parquet`): Columnar storage format for large datasets.

4.1.1 General Requirements

Regardless of the file type, all tabular files must adhere to these structure rules:

1. **Header Row:** The first row of the file must contain the field names.
2. **All Columns Required:** You must include **every column** listed in the Field Dictionary as a header, even if you do not have data for that field (e.g., `elevation` or `qualifier_codes`). If a column is missing, the file will be rejected.
3. **Exact Naming:** Column headers must **exactly match** the field names defined in the Field Dictionary (e.g., use `device_id` , not `Device ID`).
4. **Column Order:** The exact order of the columns does not matter for machine parsing, provided all required headers are present and spelled correctly. However, for human readability, it is **strongly recommended** to group columns in the order shown in the field dictionary.
5. **One Record Per Row:** Each subsequent row represents one unique measurement (a specific parameter, at a specific time, from a specific device).
6. **Location Columns:** You must use separate columns for `latitude` and `longitude` .

4.1.2 Format-Specific Rules

1. CSV Files (`.csv`)

- **Delimiter:** Fields must be separated by a comma (`,`).
- **Encoding:** UTF-8 encoding is required.
- **No Internal Commas:** Data fields like `data_steward_name` must not contain commas, as this breaks the parsing structure.
- **Handling Empty Fields:** If an optional field is blank (e.g., `method_code`), keep the comma placeholders.
- *Correct:* `...,44201,,45.2,...`
- *Incorrect:* `...,44201,45.2,...` (Skipping the column entirely shifts all subsequent data).
- **File Compression:** CSV files may be gzip compressed (e.g. `*.csv.gz`) to save storage space.

2. Excel Files (`.xlsx`)

- **First Sheet Only:** Data must be located on the **first sheet** (Worksheet 1) of the workbook. Data on subsequent sheets will be ignored.
- **No Metadata Headers:** Do not include "title rows" or merged cells above the header row. Row 1 must be the column headers.
- **Formulas:** Avoid formulas. Values must be stored as raw text or numbers.

3. Parquet Files (`.parquet`)

- **Schema Enforcement:** Ensure the column data types in the Parquet schema match the AQDx Data Types (e.g., `parameter_value` should be stored as a Float/Double, not a String).
- **Efficiency:** This format is recommended for datasets exceeding 1 million rows.

4.1.3 Example file (`.csv`)

```
datetime,parameter_code,parameter_value,unit_code,method_code,duration,aggregation_code,latitude,longitude,elevation,data_steward_name,device_id,measurement_tec
hnology_code,instrument_classification,dataset_id,validity_code,calibration_code,review_level_code,detection_limit,qualifier_codes
2024-05-23T14:00:00-07:00,88101,12.50000,105,170,3600,1,39.75500,-105.01000,1580.0,CityOfDenver,MetOne,B2-Station,CF-SSvs-BA,
1,CityOfDenver_B2_20240523,1,2,1,0.50000,
2024-05-23T15:00:00-07:00,88101,,105,170,3600,1,39.75500,-105.01000,1580.0,CityOfDenver,MetOne,B2-Station,CF-SSvs-BA,1,CityOfDenver_B2_20240523,9,2,1,0.50000,AB
```

datetime	parameter_code	parameter_value	unit_code	method_code	duration
2024-05-23T14:00:00-07:00	88101	12.50000	105	170	3600
2024-05-23T15:00:00-07:00	88101		105	170	3600

Note on Missing Data in CSV: Notice how missing or optional values are handled in the example above. For instance, the parameter_value is missing in the second row, and qualifier_codes is not used in the first row.

4.2 JSON Data Format

The JSON (JavaScript Object Notation) format is the standard AQDx method for **real-time data transmission**, API responses, and streaming data. Unlike the Tabular format, which is optimized for human readability and bulk history, JSON is optimized for machine-to-machine communication in real time.

AQDx supports two common JSON structures:

1. **Single Record / Stream (NDJSON):** A continuous stream of individual objects, separated by newlines. Ideal for live feeds.
2. **Batch Array:** A list of record objects wrapped in brackets `[...]`. Ideal for API responses containing a short history.

4.2.1 General Requirements

To ensure your JSON data can be parsed by AQDx systems, you must follow these rules:

1. **Field Names:** JSON keys must **exactly match** the field names defined in the Field Dictionary.
   - *Correct:* `"device_id": "A1"`
   - *Incorrect:* `"DeviceID": "A1"` or `"deviceId": "A1"`
2. **Data Types:** You must respect the strict data types (String vs. Number).
   - **Strings:** Must be wrapped in double quotes (e.g., `"unit_code": "008"`).
   - **Numbers:** Must **not** be wrapped in quotes (e.g., `"parameter_value": 45.2`, `"validity_code": 1`).
3. **Required Fields:** All fields marked **"Value Required : Yes"** in the Field Dictionary must be present in every JSON object.
4. **File Termination:** All multi-line JSON files must end with a single newline character (`\n`).

4.2.2 Handling Optional & Missing Values

Unlike CSV files where you must leave an empty space between commas, JSON allows flexibility for optional fields (like `elevation` or `method_code`).

If you do not have data for an optional field, you have two valid choices:

1. **Omit the Key (Preferred):** Simply do not include the field in the JSON object. This reduces file size.
2. **Send Null:** Include the key with a `null` value.

Example:

```
// Option 1: Key Omitted (Preferred)
{
  "parameter_code": "44201",
  "parameter_value": 45.2
}

// Option 2: Explicit Null
{
  "parameter_code": "44201",
  "parameter_value": 45.2,
  "method_code": null
}
```

4.2.3 Structure 1: Newline Delimited JSON (Streaming)

For real-time applications, use Newline Delimited JSON (NDJSON). Each line represents one complete measurement record. There are no commas between lines and no enclosing brackets.

Format:

```
{record_1}\n
{record_2}\n
{record_3}\n
```

Example stream


```
{"datetime": "2024-05-23T14:00:00-07:00", "parameter_code": "88101", "parameter_value": 12.5, "unit_code": "105", "method_code": "170", "duration": 3600,
"aggregation_code": 1, "latitude": 39.755, "longitude": -105.010, "elevation": 1580.0, "data_steward_name": "CityOfDenver", "device_id": "B2-Station",
"measurement_technology_code": "CF-SSvs-BA", "instrument_classification": 1, "dataset_id": "CityOfDenver_B2_20240523", "validity_code": 1, "calibration_code":
2, "review_level_code": 1, "detection_limit": 0.5}
{"datetime": "2024-05-23T15:00:00-07:00", "parameter_code": "88101", "parameter_value": null, "unit_code": "105", "method_code": "170", "duration": 3600,
"aggregation_code": 1, "latitude": 39.755, "longitude": -105.010, "elevation": 1580.0, "data_steward_name": "CityOfDenver", "device_id": "B2-Station",
"measurement_technology_code": "CF-SSvs-BA", "instrument_classification": 1, "dataset_id": "CityOfDenver_B2_20240523", "validity_code": 9, "calibration_code":
2, "review_level_code": 1, "qualifier_codes": "AM"}
```


4.2.4 Structure 2: Standard JSON File (Batch)

For archival files or API responses, wrap the records in a standard JSON Array [...]. Note that this example has been indented and formatted for visualization purposes.

Example file


```
[
  {
    "datetime": "2024-05-23T14:00:00-07:00",
    "parameter_code": "88101",
    "parameter_value": 12.5,
    "unit_code": "105",
    "method_code": "170",
    "duration": 3600,
    "aggregation_code": 1,
    "latitude": 39.755,
    "longitude": -105.01,
    "elevation": 1580.0,
    "data_steward_name": "CityOfDenver",
    "device_id": "B2-Station",
    "measurement_technology_code": "CF-SSvs-BA",
    "instrument_classification": 1,
    "dataset_id": "CityOfDenver_B2_20240523",
    "validity_code": 1,
    "calibration_code": 2,
    "review_level_code": 1,
    "detection_limit": 0.5
  },
  {
    "datetime": "2024-05-23T15:00:00-07:00",
    "parameter_code": "88101",
    "parameter_value": null,
    "unit_code": "105",
    "method_code": "170",
    "duration": 3600,
    "aggregation_code": 1,
    "latitude": 39.755,
    "longitude": -105.01,
    "elevation": 1580.0,
    "data_steward_name": "CityOfDenver",
    "device_id": "B2-Station",
    "measurement_technology_code": "CF-SSvs-BA",
    "instrument_classification": 1,
    "dataset_id": "CityOfDenver_B2_20240523",
    "validity_code": 9,
    "calibration_code": 2,
    "review_level_code": 1,
    "qualifier_codes": "AM"
  }
]
```

5. Metadata Guidance

5.1 AQDx v3 Metadata Reference Guide

Download: AQDx_metadata_form_v3.yaml (Template) note: right click "Save Link As..." to download the yaml file.

This document details the fields and definitions required for the AQDx Version 3 Metadata YAML form. Metadata provides essential context for air quality datasets, including project ownership, site locations, instrument specifications, and quality assurance procedures.

To streamline data submission, the metadata is organized into three distinct sections: Overview, Sites, and Instruments. This standardized structure avoids repetitive data entry by allowing submitters to define a monitoring site once, and then link multiple individual instruments to that location.

*(Note: Data submitters may fill out this information using either a standard three-tab spreadsheet (e.g. Microsoft Excel, Google Sheets) or editing the YAML file in a text editor. Please see the accompanying guide on **Using the AQDx Excel Template and Conversion Script** for step-by-step instructions on data entry.)*

5.1.1 Metadata Organization

The schema relies on unique text identifiers to link data across the different sections:

- **Sites:** Each physical monitoring location is defined once and identified by a unique `site_name`. This section holds all geographic and location-based data.
- **Instruments:** Each instrument is identified by a combination of its `device_id` and the specific `parameter_code` it measures. This allows you to define different detection limits and calibrations for different pollutants measured by the *same* physical sensor package.
- **The Link:** Every instrument entry must include a `site_name` that exactly matches a location defined in the Sites list.

5.1.2 Data Quality Documentation

Data quality information is divided into two levels to accurately reflect different monitoring practices:

1. **Dataset-Level Quality:** Broad procedures that apply to the entire project (e.g., QAPP links, general data review workflows, project-wide automated QC).
2. **Instrument/Parameter-Level Quality:** Specific performance metrics (e.g., precision, bias, detection limits, and correction methods) that apply uniquely to a specific parameter measured by a specific device.

5.1.3 1. Dataset Header

These root-level fields define the entire submission package and ensure the metadata maps correctly to the accompanying row-level data file.

Field	Type	Description
dataset_id	String	**Required.** Unique identifier for the dataset. Must exactly match the `dataset_id` in your tabular data file.
aqdx_metadata_version	String	**Required.** Version of the metadata schema used (e.g., "3.0").
aqdx_data_version	String	**Required.** Version of the attached tabular data format (e.g., "3.0").

5.1.4 2. Overview & Data Steward

This section provides context about the organization responsible for the data.

Field	Type	Description
data_steward_name	String	**Required.** Max 64 chars. Must exactly match `data_steward_name` in the tabular data.
contact_name	String	**Required.** First and Last name. Max 64 chars.
contact_email	String	**Required.** Max 64 chars.
contact_phone	String	*Optional.* Phone number.
organization_type	Integer	**Required.** `1`-Gov, `2`-NGO, `3`-Community, `4`-Academic, `5`-Industry, `6`-Consulting, `7`-Education, `8`-Other.
organization_name_full	String	**Required.** Full name (e.g., Colorado Department of Public Health and Environment). Max 128 chars.
address	String	*Optional.* Physical address of the organization. Max 128 chars.
last_update_date	Integer	**Required.** Date the metadata was last modified. Format: `YYYYMMDD`.
is_regulatory_data	Integer	**Required.** Is this dataset regulatory? `1` (Yes) or `0` (No).
data_abstract	String	*Optional.* Brief description of the dataset/project. Max 500 chars.

5.1.5 3. Dataset-Level Quality

General quality assurance procedures applied to the dataset as a whole.

Field	Type	Description
automated_qc_applied	Boolean	**Required.** `true` or `false`. Should align with 'auto_qc' in the tabular data.
automated_qc_methods	String	*Optional.* Comma-separated exact options: *bounds check, sticking values, comparison to other signals, low values check, high values check, completeness, comparison to co-located duplicate sensor or sample, automated bump checks, automated zero checks, other algorithmic checks*.
automated_qc_description	String	*Optional.* Free text summary of automated QC.
data_review_undergone	Boolean	**Required.** `true` or `false`. Should align with 'review_level_code' in the tabular data.
data_review_methods	String	*Optional.* Comma-separated options: *qualitative review, quantitative review, external party reviewing - [add here], based on established methods associated with an official Program - [add name here]*.
data_review_description	String	*Optional.* Free text summary of the review process.
official_monitoring_programs	String	*Optional.* E.g., US EPA NATTS Program.
other_processing_desc	String	*Optional.* Any other processing users should be aware of.
useful_links	String	*Optional.* Comma-separated URLs (e.g., project webpages, QAPPs, publications, procedures).

5.1.6 4. Sites

Defines the physical locations of the monitoring stations. Provide one entry per site.

Special Case: Handling Mobile and Wearable Data

AQDx fully supports mobile platforms (e.g., regulatory vans) and wearable sensors. Because the precise, second-by-second GPS coordinates for mobile data are recorded in your tabular data file, the `Sites` metadata entry is used to define the **General Study Area** or the **Mobile Platform** itself.

If your data is mobile or wearable, fill out the Sites section using these guidelines:

- **site_name:** Name the route, the bounding area, or the platform (e.g., "Denver Mobile Van 1", "I-25 Route", "Wearable Subject A").
- **latitude / longitude:** Enter the centroid (center point) of your study area, the starting point of your route, or the "home base" of the instrument.
- **surroundings_type:** Choose the code that best represents the overall area, or select `10` (Mixed) if the route covers diverse environments.

(Note: When you set the `monitoring_approach` on the Instruments tab to `4` (Mobile) or `5` (Wearable), data users and databases will automatically know to look at your tabular data for the exact timestamps and moving coordinates.)

Field	Type	Description
site_name	String	**Required.** Unique identifier for the location. Max 64 chars.
latitude / longitude	Decimal	**Required.** Coordinates in decimal degrees.
original_gis_datum	String	**Required.** Expected coordinate system: `WGS84`. Use this field to indicate if the data was converted to WGS84.
address	String	*Optional.* Physical address of the site.
state_code / county_code	Integer	**Required.** AQS state and county codes.
site_owner	String	**Required.** Person or organization owning the site. Max 128 chars.
site_photos_url	String	*Optional.* Link to site imagery. Max 200 chars.
surroundings_type	Integer	**Required.** `1`-Urban, `2`-Rural, `3`-Suburban, `4`-Industrial, `5`-Residential, `6`-Ag, `7`-Natural, `8`-Rec, `9`-Water, `10`-Mixed, `11`-Other.
nearby_sources	String	*Optional.* Free text description of nearby pollution sources.
reg_aqs_id	Integer	**Required if regulatory.** 9-digit AQS monitoring location code.
reg_monitoring_scale	Integer	**Required if regulatory.** `1`-Micro, `2`-Middle, `3`-Neighborhood, `4`-Urban, `5`-Regional, `6`-National, `7`-Global.
reg_site_type	Integer	**Required if regulatory.** Objective category code.
reg_groundcover	Integer	**Required if regulatory.** Dominant groundcover code.

5.1.7 5. Instruments & Parameter-Level Quality

Defines the specific devices, configurations, and chemical/physical parameters measured. **Provide one entry per unique combination of `device_id` and `parameter_code`.**

Field	Type	Description
device_id	String	**Required.** The device identifier. Matches tabular data.
parameter_code	String	**Required.** 5-digit AQS parameter code. Matches tabular data.
site_name	String	**Required.** Must exactly match a `site_name` defined in the Sites section.
manufacturer_name	String	**Required.** Manufacturer of the instrument.
device_model	String	**Required.** Model name assigned by `manufacturer_name`.
firmware_version	String	*Optional.* Instrument firmware version.
method_code	String	**Required.** 3-digit method code. Matches tabular data.
monitor_start_date	Integer	**Required.** Format: `YYYYMMDD`.
probe_height_m	Decimal	**Required.** Height above ground in meters.
monitoring_approach	Integer	**Required.** `1`-Stationary Cont, `2`-Stationary Integrated, `3`-Intermittent, `4`-Mobile, `5`-Wearable.
monitoring_objective	Integer	**Required.** `1`-Ambient, `2`-Near-source, `3`-Fenceline, `4`-Community, `5`-Personal, `6`-Indoor, `7`-Other.
expanded_objective	String	**Required.** Free text expanded objective.
sampling_frequency_sec	Decimal	**Required.** Frequency in seconds (e.g., 60, 3600).
residence_time_sec	Decimal	*Optional.* Residence time for reactive parameters.
airflow_arc_degrees	Integer	**Required.** Unrestricted airflow (0–360).
instrument_photos_url	String	*Optional.* URL to instrument installation photos.
dist_obstructions_m	Decimal	**Required.** Distance from obstructions not on roof (meters).
dist_roof_obstructions_m	Decimal	*Optional.* Distance from obstructions on roof (meters).

Instrument/Parameter-Specific Data Quality

These fields apply *only* to the specific device and parameter defined in the row.

Field	Type	Description
corrections_applied	Boolean	**Required.** `true` or `false`. Matches tabular 'corr_code'.
corrections_methods	String	*Optional.* Comma-separated options: *manual correction, automated correction, global correction model applied, unit/device-specific corrections applied, corrections derived from [frequency] co-locations, published model used [add name/description], corrections based on periodic calibration via standards, correction after re-processing/re-analysis*.
corrections_description	String	*Optional.* Free text including frequency.
detection_limit_methods	String	*Optional.* Comma-separated options: *method detection limit, instrument detection limit, determined empirically, taken from manufacturer specifications, method used - [add name/description].*
detection_limit_desc	String	*Optional.* Free text limit description.
precision_quantified	Boolean	*Optional.* `true` or `false`.
precision_desc	String	*Optional.* Metrics (e.g., standard deviation, CV) and dataset used.
bias_linearity_quantified	Boolean	*Optional.* `true` or `false`.
bias_linearity_desc	String	*Optional.* Metrics (e.g., simple linear regression) and dataset used.
accuracy_error_quantified	Boolean	*Optional.* `true` or `false`.
accuracy_error_desc	String	*Optional.* Metrics (e.g., RMSE) and dataset used.
maintenance_procedures_desc	String	*Optional.* Frequency and date last performed for this device.

Regulatory Fields

Complete these fields only if `is_regulatory_data` *is set to* `1`*. Otherwise, leave null or blank.*

Field	Type	Description
reg_monitor_type	String/Int	*Optional.* AQS monitor type.
reg_method_type	String/Int	*Optional.* AQS method type.
reg_network_affiliation	String	*Optional.* Network affiliation. Max 64 chars.
reg_collecting_agency	String	*Optional.* Collecting agency name. Max 64 chars.
reg_agency_code	Integer	*Optional.* Agency code.
reg_analysis_method	String	*Optional.* Analysis method used. Max 64 chars.
reg_analytical_lab	String	*Optional.* Analytical lab name. Max 64 chars.
reg_probe_material	String	*Optional.* Material of the probe. Max 64 chars.

6. Code Lookup Tables

6.1

6.2

6.3 Measurement Technology Codes

This table contains the approved, active `measurement_technology_code` combinations for the AQDx standard.

These 8-to-14 character codes chronologically categorize the physical journey of a sample from acquisition to the final analytical signal. Rather than naming specific brand-name instruments, this taxonomy breaks the measurement down into three functional, hyphen-delimited blocks:

`[Acquisition] - [Conditioning] - [Detection]`

Need a new code? If your specific hardware configuration is not listed below, please submit a "New Code Request" via the project's GitHub Issue Tracker. Do not invent custom codes.

6.4

6.5

6.6 Supplemental Codes

These supplemental codes are maintained by CDPHE Air Toxics and Ozone Precursors program to supplement available EPA codes as needed.

6.6.1 Qualifier Codes

Qualifier Code	Qualifier Description	Qualifier Type	Qualifier Type Code	Additional Notes
UD	Raw or uncorrected data, potentially requiring correction	Quality Assurance Qualifier	QA	May be appropriate for use with air sensor data
CD	Corrected data (e.g., via application of a calibration model)	Quality Assurance Qualifier	QA	May be appropriate for use with air sensor data
QW	Wind speed and/or direction data questionable	Quality Assurance Qualifier	QA	May be especially relevant to mobile monitoring data
QG	GPS data questionable	Quality Assurance Qualifier	QA	May be especially relevant to mobile monitoring data
CG	Corrected or modified GPS data	Quality Assurance Qualifier	QA	May be especially relevant to mobile monitoring data
IG	GPS data invalid	Quality Assurance Qualifier	QA	May be especially relevant to mobile monitoring data
CO	Co-located field sample	Quality Assurance Qualifier	QA	For physically co-located samples (different from duplicates)
ZI	Zone: indoor. Indoor air sample	Quality Assurance Qualifier	QA	For measurements taken indoors

6.6.2 Parameter Codes

Parameter Code	Parameter	Parameter Abbreviation	Parameter Alternate Name	CAS Number	Standard Units	Roun Trunc
75101	O2			7782-44-7		R
75102	Flow Rate					R
75103	Exit Temp					R
75104	Opacity					R
75105	CO Emission Factor Heat Rate					R
75106	NOx Emission Factor Heat Rate					R
75107	Fuel Gas					R
75108	Combustion Zone Net Heating Value					R
75109	Heat Content					R

Parameter Code	Parameter	Parameter Abbreviation	Parameter Alternate Name	CAS Number	Standard Units	Roun Trunc
75110	Heat Rate					R
75111	H2S Flare Gas					R

6.6.3 Unit Codes

Unit Code	Units	Additional Notes
301	Standard cubic foot per minute at 1 ATM and 68°F	May be relevant to continuous emissions monitoring data
302	Standard cubic foot per hour at 1 ATM and 68°F	May be relevant to continuous emissions monitoring data
303	Standard cubic foot per minute at 1 ATM and 60°F	May be relevant to continuous emissions monitoring data
304	Standard cubic foot per hour at 1 ATM and 60°F	May be relevant to continuous emissions monitoring data
305	Actual cubic foot per hour	May be relevant to continuous emissions monitoring data
306	ppm at 0% O2	May be relevant to continuous emissions monitoring data
307	lbs/min	May be relevant to continuous emissions monitoring data
308	lbs/hour	May be relevant to continuous emissions monitoring data
309	lbs/MMBTU	May be relevant to continuous emissions monitoring data
310	lbs/ton coke burned	May be relevant to continuous emissions monitoring data
311	tons/year	May be relevant to continuous emissions monitoring data
312	% volume wet	May be relevant to continuous emissions monitoring data
313	% volume dry	May be relevant to continuous emissions monitoring data
314	BTUH/scf	May be relevant to continuous emissions monitoring data
315	BTUL/scf	May be relevant to continuous emissions monitoring data
316	MMBTU/hr	May be relevant to continuous emissions monitoring data
317	lbs/day	May be relevant to continuous emissions monitoring data
318	kg/hr	May be relevant to monitoring providing emissions estimates

6.6.4 Method Codes

Example Parameters	Method Code	Recording Mode	Collection Description	Analysis Description	Round Truncate Indicator	Addi Note
Carbon monoxide, Nitrogen dioxide, Ozone		Continuous	Sensor	Electrochemical	R	May relev quali
Carbon dioxide		Continuous	Sensor	Nondispersive Infrared	R	May relev quali
Volatile organic compounds	Continuous	Sensor	Photoionization detector	R	May be relevant to air quality sensors	
Xylene(s), benzene		Continuous	Semi-Continuous Analyzer	GC/PID	R	May relev quali
Carbon monoxide, Nitrogen dioxide, Ozone		Continuous	Sensor	Electrochemical	R	May relev quali
Carbon monoxide, Nitrogen dioxide, Ozone		Continuous	Sensor	Electrochemical	R	May relev quali
Carbon monoxide, Nitrogen dioxide, Ozone		Continuous	Sensor	Electrochemical	R	May relev quali
Carbon monoxide, Nitrogen dioxide, Ozone		Continuous	Sensor	Electrochemical	R	May relev quali

7. Appendices

7.1 Glossary & Acronyms

This glossary defines common terms and acronyms used throughout the AQDx standard, particularly those derived from US EPA regulatory programs.

7.1.1 Acronyms

Acronym	Definition	Context
AQDx	**Air Quality Data Exchange**	The standard data format defined in this documentation.
AQCSV	**Air Quality Comma Separated Values**	The precursor format to AQDx, originally developed for the EPA's AirNow program.
AQS	**Air Quality System**	The EPA's repository of ambient air quality data. AQDx uses AQS codes for parameters and methods.
ARM	**Approved Regional Method**	A method for measuring PM2.5 that has been approved for use within a specific geographic area.
CASTNET	**Clean Air Status and Trends Network**	A long-term monitoring network designed to assess trends in air quality and deposition.
COATTS	**Colorado Air Toxics Trends Station**	Colorado specific implementation of NATTS sites specified by HB22-1244
CSN	**Chemical Speciation Network**	A network that monitors the chemical composition of PM2.5.
EPA	**Environmental Protection Agency**	The US federal agency responsible for protecting human health and the environment.
FEM	**Federal Equivalent Method**	A monitoring method designated by the EPA as equivalent to a Federal Reference Method (FRM) for regulatory compliance.
FRM	**Federal Reference Method**	The "gold standard" method for measuring a pollutant, as defined in the Code of Federal Regulations (CFR).
IMPROVE	**Interagency Monitoring of Protected Visual Environments**	A monitoring network established to protect visibility in Class I areas (national parks and wilderness areas).
LC	**Local Conditions**	Used in parameter codes to denote local condition rather than (e.g. parameter_code `105`)
NAAQS	**National Ambient Air Quality Standards**	Limits on atmospheric concentration of six common pollutants (O_3, PM, CO, SO_2, NO_2, Pb) set by the EPA.
NATTS	**National Air Toxics Trends Stations**	A network designed to provide long-term monitoring data for certain air toxics.
NCore	**National Core Network**	A multi-pollutant network that integrates several advanced measurement systems.
PAMS	**Photochemical Assessment Monitoring Stations**	Stations that monitor ozone precursors (volatile organic compounds, oxides of nitrogen) in serious ozone nonattainment areas.
QAPP	**Quality Assurance Project Plan**	A formal document describing in detail the necessary quality assurance, quality control, and other technical activities for a project.
QC	**Quality Control**	Operational techniques and activities used to fulfill requirements for quality (e.g., sticking checks, zero checks).
SLAMS	**State or Local Air Monitoring Stations**	The primary network of monitoring stations used by state and local agencies for NAAQS compliance.
SPM	**Special Purpose Monitor**	A monitor used for special studies or short-term objectives, often not strictly for regulatory compliance.
STN	**Speciation Trends Network**	A component of the Chemical Speciation Network (CSN).
UTC	**Coordinated Universal Time**	

Acronym	Definition	Context
		The primary time standard by which the world regulates clocks and time. AQDx requires time zone offsets relative to UTC.
WGS84	**World Geodetic System 1984**	The standard coordinate system for Earth, used by GPS. AQDx requires latitude/longitude in WGS84.

7.1.2 Terminology

Data Steward

The organization or individual responsible for the oversight, collection, processing, and distribution of the dataset. The Data Steward is the primary contact for any questions regarding the data.

Method Code

A 3-digit integer from the AQS library that identifies the specific technology or analysis method used (e.g., `170` for a specific BAM-1020 setup). For low-cost sensors without a regulatory designation, this is often left blank or set to a generic code.

Parameter Code

A 5-digit integer from the AQS library that identifies the pollutant or variable being measured (e.g., `44201` for Ozone, `88101` for PM2.5).

Regulatory Data

Data collected with the specific intent of comparing it against National Ambient Air Quality Standards (NAAQS). This data must meet strict quality assurance requirements, typically using FRM or FEM instruments.

ISO 8601

The international standard for date and time formatting. AQDx uses the extended format with time zone offsets: `YYYY-MM-DDThh:mm:ssTZD` (e.g., `2024-01-01T14:30:00-07:00`).

7.2 Version Changelog

This document tracks major changes to the AQDx standard, including updates to the metadata schema, file formats, and documentation structure.

7.2.1 Version 3.0 (January 2025)

Focus: Machine Readability, Version Control, and Developer Experience.

1. Metadata Restructuring (Major Change)

The metadata submission process has been completely overhauled to support modern data engineering workflows.

- **Format Change:** Switched from **Excel (`.xlsx`)** to **YAML (`.yaml`)**.
- *Why?* YAML allows for nested structures (Instruments nested under Sites), is human-readable, and allows metadata to be tracked in version control systems like Git.
- **Schema Updates:**
- **Nesting:** Instruments are no longer flat rows; they are now children of specific Site objects.
- **Field Standardization:** Field names in the metadata now exactly match the field names in the data files (e.g., `data_steward_name` is consistent across both).
- **Data Quality:** The "Quality Questionnaire" has been replaced by a structured `data_quality` object with boolean flags (`automated_qc`, `corrections`) and explicit description fields.

2. Documentation Architecture

- **Migration to MkDocs:** Documentation has moved from static **PDFs** to a live **ReadTheDocs** website generated from Markdown.
- **Searchability:** Users can now search for specific field definitions or error codes instantly across the entire standard.
- **Versioning:** Documentation is now versioned alongside the schema, allowing users to view guidance for previous versions if needed.

3. Data Format Enhancements

- **Parquet Support:** Added Apache Parquet (`.parquet`) as a supported format for high-volume, archival datasets.

7.2.2 Version 2.0 (July 2024)