Data Processing Overview (8/21/2025)

Updated: 8/21/2025

Introduction

Data from the Chesapeake Bay Program's DataHub, NOAA, VECOS, and EOTB were downloaded, harmonized into a common structure, and stored in a partitioned Apache Parquet file format. The Apache Parquet file format was selected for storing and processing integrated water quality datasets. The data span three main sources: cruise-based DataFlow surveys that produce high-resolution, irregularly spaced records; fixed-site sensors sampled every few minutes; and periodic grab samples collected at established monitoring locations. To streamline downstream processing, the data were harmonized into a common structure across a 4D domain of latitude, longitude, depth, and time. Parquet's columnar storage allows efficient, selective access to variables, and its tabular design naturally accommodates irregular time series without workarounds. It also provides strong compression for repeated metadata, schema flexibility for adding new fields, and seamless integration with R tools such as **arrow** and **duckdb**. In addition, Parquet files are accessible from other platforms and tools, including **Python (pandas, pyarrow)**, **MATLAB** (using the built-in parquetread and parquetwrite functions), **Apache Spark**, **Hadoop**, and cloud data warehouses such as **Snowflake**, **BigQuery**, and **Redshift**.

Included Documents

The current dataset was developed incrementally, with subsets assembled as specific analyses required them. Consequently, the processing steps were documented across multiple sources, and relevant portions have been extracted and compiled here.

CBP DataHub Data Processing. This document describes the approach for processing tidal water quality data from traditional partners.

DataFlow Data Processing. This document describes the approach for processing DataFlow from EOTB and VECOS.

Data Transformation Tables. This document presents column mapping and transformations for other high frequency data including ConMon and NOAA's vertical array data.

Chesapeake Monitoring Cooperative Data. This document describes the approach for processing tidal water quality data from non-traditional partners from the CBP DataHub.

Sample Code

The sample code demonstrates how to efficiently explore and query the harmonized monitoring dataset stored in Apache Parquet format. A key feature is the use of **partitions** (by `year`, `cbseg_92`, `source`, and `type`), which makes retrieval of subsets fast and memory-efficient. The examples show how to filter by year, segment, source, or data type, as well as how to restrict which columns are returned to minimize data volume. Safety checks (e.g., `require_filters = TRUE`) help prevent accidental loading of the full dataset, while warnings alert you when a filter does not align with a partitioned column. Users can also retrieve distinct values of `source` or `type` to understand available data options. Together, these features make it straightforward to slice the dataset for targeted analyses, whether pulling one year of DataFlow records, a handful of stations, or integrating across multiple sources. (The main program and helper function are provided below but are also available as scripts.)

Main Program

```
library(arrow)
library(rlang)
library(stringr)
library(tidyverse)

rm(list = ls())                     # Clear global environment
cat("\014")                         # Clear console

# Set file directory
file_db  <- "your/file/location")

source("fun_read_parquet_dataset.R")

# Example 1: Retrieve 2022 data for CB4MH (default arguments included)
df <- read_parquet_dataset(
  data_path = file_db,
  filters = list(year = 2022, cbseg_92 = "CB4MH"),
  select_columns = NULL,
  collect_data = TRUE,
  require_filters = TRUE,
  partition_vars = c("year", "cbseg_92", "source", "type")
)
table(df$type, df$source)

# Example 2: Same as Example 1 (default arguments removed)
df <- read_parquet_dataset(data_path = file_db,
                           filters = list(year = 2022, cbseg_92 = "CB4MH"))
table(df$type, df$source)

# Example 3: Return all CBP DataHub data
df <- read_parquet_dataset(data_path = file_db,
                           filters = list(source = "cbp_datahub"))
table(df$type, df$source)

# Example 4: Retrieve 2021-2022 DataFlow data for RPPMH and RPPOH from vecos,
#            limit which columns are returned
df <- read_parquet_dataset(data_path = file_db,
                           filters = list(year = 2021:2022,
                                          cbseg_92 = c("RPPMH", "RPPOH"),
                                          source   = "va_vecos",
                                          type     = "dataflow"),
                           select_columns = c("cbseg_92", "station", "date_time", "do"))
table(df$station, df$cbseg_92)
```

```r
# Example 5: Retrieve all available station LE3.4 data. (It will work, but you
#            will get a warning indicating that you didn't use a partitioned column.
#            Not using a partitioned column can slow retrieval.)
df <- read_parquet_dataset(data_path = file_db,
                           filters = list(station = "LE3.4")
                           )

# Example 6: An ill-advised retrieval of all data. (Must include the require_filters
#            argument as a matter of safety)
df <- read_parquet_dataset(data_path = file_db, require_filters = FALSE)

# Example 7: Get a list of source and type
ds <- open_dataset(file_db)
ds %>%
  distinct(source) %>%
  collect() %>%
  print(n=Inf)

ds %>%
  distinct(type) %>%
  collect() %>%
  print(n=Inf)
```

Helper Function – read_parquet_dataset (documentation and source code)

Read and Filter a Partitioned Parquet Dataset

Description

Opens a partitioned Parquet dataset using 'arrow::open_dataset()' and applies filtering expressions to minimize in-memory reads. Filters are classified as partition filters or data filters. Optionally, specific columns can be selected, and data can be collected into memory or returned lazily.

Usage

```r
read_parquet_dataset(
  data_path,
  filters = list(),
  select_columns = NULL,
  collect_data = TRUE,
  require_filters = TRUE,
  partition_vars = c("year", "cbseg_92", "source", "type")
)
```

Arguments

`data_path`	Character string. Path to the root of the Parquet dataset.
`filters`	A named list of filtering expressions or atomic vectors. Atomic vectors (e.g., `list(year = 2020:2021)`) are automatically converted into `%in%` expressions.
`select_columns`	Optional character vector of column names to retain. If not `NULL`, only these columns are returned.
`collect_data`	Logical; if TRUE (default), calls `dplyr::collect()` to return an in-memory tibble. If FALSE, returns the filtered Arrow dataset.
`require_filters`	Logical; if TRUE (default), at least one filter must be specified to avoid loading the entire dataset.
`partition_vars`	Character vector of known partition variable names. Used to separate fast (partition) and slow (data-level) filters. Defaults to `c("year", "cbseg_92", "source", "type")`.

Value

A tibble if `collect_data = TRUE`, or a filtered Arrow dataset (class `arrow_dplyr_query`) if FALSE.

```r
read_parquet_dataset <- function(data_path,
                                 filters = list(),
                                 select_columns = NULL,
                                 collect_data = TRUE,
                                 require_filters = TRUE,
                                 partition_vars = c("year", "cbseg_92", "source", "type")) {

  # Open the dataset
  ds <- arrow::open_dataset(data_path)

  # Require at least one filter to prevent loading full dataset accidentally
  if (require_filters && length(filters) == 0) {
    stop("❌ No filters provided. To load the full dataset, set `require_filters = FALSE`.")
  }

  # Separate filters into partition filters and data filters
  partition_filter_exprs <- list()
  data_filter_exprs      <- list()

  for (nm in names(filters)) {
    val <- filters[[nm]]

    # Wrap atomic filters as `%in%`
    if (is.atomic(val)) {
      val <- rlang::expr(!!rlang::sym(nm) %in% !!val)
    }

    # If it's a `%in%` and uses a partition variable, treat as partition filter
    if (rlang::is_call(val, "%in%") &&
        as.character(val[[2]]) == nm &&
        nm %in% partition_vars) {
      partition_filter_exprs[[nm]] <- val
    } else {
      data_filter_exprs[[nm]] <- val
    }
  }

  # Warn user if filters don't align with partitions
  non_partition_vars <- setdiff(names(filters), names(partition_filter_exprs))
  if (length(non_partition_vars) > 0) {
    warning("⚠ Filters on non-partitioned columns (slower read): ",
            paste(non_partition_vars, collapse = ", "),
            "\nℹ Partitioned variables: ",
            paste(partition_vars, collapse = ", "))
  }

  # Apply partition filters
  if (length(partition_filter_exprs) > 0) {
    combined_partition_filter <- Reduce(function(x, y) rlang::expr(!!x & !!y), partition_filter_exprs)
    ds <- ds %>% dplyr::filter(!!combined_partition_filter)
  }

  # Apply data-level filters
  if (length(data_filter_exprs) > 0) {
    combined_data_filter <- Reduce(function(x, y) rlang::expr(!!x & !!y), data_filter_exprs)
    ds <- ds %>% dplyr::filter(!!combined_data_filter)
  }

  # Column selection
  if (!is.null(select_columns)) {
    existing_cols <- names(arrow::schema(ds))
    missing_cols  <- setdiff(select_columns, existing_cols)
    if (length(missing_cols) > 0) {
```

```
      stop(paste("❌ These columns do not exist in the dataset:", paste(missing_cols, collapse = ", ")))
    }
    ds <- ds %>% dplyr::select(dplyr::all_of(select_columns))
  }

  # Collect or return lazy dataset
  if (collect_data) {
    return(ds %>% dplyr::collect())
  } else {
    return(ds)
  }
}
```

CBP DataHub Data Processing (4/21/2025)

Updated: 8/21/2025

Introduction

This document summarizes the method for processing water quality monitoring data from the Chesapeake Bay Program's DataHub. The processing emphasized consistent field naming, datetime handling, parameter downselection, and spatial screening. The final dataset was structured in a wide format for integration with other data sets in future steps.

DataHub data were downloaded as Excel files from the Chesapeake Bay Program's website[1] during June 14, 2023 for data from 1984-2018 and February 21, 2025 for 2019-2023.

Data Transformations and Course-Level Spatial Screening

The data transformation process followed this pipeline:

- **Column Renaming and Standardization.** Incoming Excel files exhibited inconsistencies in column headers due to varying schema conventions. A harmonization routine was applied to align incoming field names to a common structure. Downstream processing used standardized names to ensure reproducibility and consistency across data pulls (see Table 1).
- **Parameter Filtering and Numeric Screening.** Source column names were transformed (Table 2) and the combined dataset was filtered to retain a defined subset of parameters for analysis. These included DO, pH, salinity, specific conductance, water temperature, DO saturation, chlorophyll-a, Secchi depth, light attenuation (Kd), and turbidity (see Table 3). DO values were range-screened, retaining only those between −0.49 and 24.99 mg/L. Records with missing MeasureValue or problematic Problem codes were removed. Exact duplicates were dropped after filtering.
- **Geographic Screening with Tidal Mask.** Latitude and longitude were used to generate spatial features (CRS = 4269) that were screened against a buffered Chesapeake Bay tidal region mask (converted to CRS = 4269 for the analysis). Only observations falling within the spatial mask were retained.
- **Replicate Aggregation and Pivoting to Wide Format**. DataHub columns SampleType, SampleReplicateType, Qualifier, Unit, and Method were reviewed for anomalies. Given the dissolved oxygen focus of future analyses, it was determined to group data by station, event_id, type, date_time, depth_b, depth, and layer. Multiple values of MeasureValue were averaged within these groups. The dataset was then pivoted from long to wide format, with separate columns for each parameter. Analyses that focus on other parameters (e.g., chlorophyll or turbidity) should reevaluate this grouping approach based on parameter-specific considerations.

[1] https://datahub.chesapeakebay.net/FileDownloads

Table 1. Column renaming to address Excel spreadsheet inconsistencies in column headers.

Column Name	Intermediate Column Name
Station	MonitoringLocation
DataSource	Source
Tier	TierLevel
Bias_PC	BiasPC
Precision_PC	PrecisionPC
Event_ID	EventId
Sample_Date	SampleDate
Sample_Time	SampleTime

Table 2. Column mapping and transformations—CBP DataHub.

Source Column	Final Name	Transformation Description
MonitoringLocation	station	Renamed
EventId	event_id	Renamed
Project	type	Recoded: CMON as conmon_cal; DFLO as dataflow_cal; TRIB, MAIN, and TSPECIAL as wqmp
SampleDate†	date_time	Parsed as date; combined with SampleTime
SampleTime†	date_time	Parsed as time (time zone = EST) Fallback to 11:00 for 00:00, 00:01 and NA
TotalDepth	depth_b	Renamed
Depth	depth	Renamed
Layer	layer	Renamed
Parameter	Pivot column names	Values kept: DO, PH, SALINITY, SPCOND, WTEMP, DO_SAT_P, CHLA, SECCHI, KD, TURB_NTU
MeasureValue	Pivot column values	Missing (e.g., NA) values removed
Problem	--	Used for QC screen to remove records‡, column not retained in final output

†SampleDate and SampleTime were combined to generate a full date_time field in EST. Time was set to 11am as a typical midpoint sampling time for those measurements with missing or unlikely sampling times.

‡Records with Problem equal to A, AA, C, D, DD, E, F, FF, GG, JJ, P, QQ, R, TP, U, V, WW, or X were removed.

Table 3. Final Parameters Used in Pivot Table—CBP DataHub.

Parameter	Pivot Column	Description
SECCHI	secchi	Pivoted + renamed to secchi
CHLA	chla	Pivoted + renamed to chla
DO	do	Pivoted + renamed to do; range screen applied ($-0.49 \leq x \leq 24.99$)
PH	ph	Pivoted + renamed to ph
SPCOND	sp_cond	Pivoted + renamed to sp_cond
WTEMP	temp	Pivoted + renamed to temp
KD	kd	Pivoted + renamed to kd
SALINITY	salinity	Pivoted + renamed to salinity
TURB_NTU	turb	Pivoted + renamed to turb
DO_SAT_P	do_sat	Pivoted + renamed to do_sat

Post-Harmonization Data Preparation and Storage

After the core harmonization and pivoting process, the dataset was enriched with spatial metadata and structured for long-term storage and downstream use. As part of the above processing, station-level data were extracted and summarized (station id, latitude, longitude). Chesapeake Bay segment (cbseg_92) assignments were included for each monitoring location.

Each station was also annotated with three additional metadata fields derived from geospatial rasters: depth_b (bottom depth), wb_lat_km (latitudinal waterbody coordinate), and wb_lon_km (longitudinal waterbody coordinate). These were extracted from 10-meter resolution raster surfaces (GeoTIFFs) specific to each Chesapeake Bay Program segment. For each segment, rasters were loaded, and values were extracted at station locations using the extract_nearest_valid() function. This function first performs a direct raster extraction and then resolves missing or NA values using a fallback algorithm based on Euclidean distances within a 500-meter bounding box. This fallback accounts for minor spatial mismatches between input coordinates and raster grid alignment, ensuring extraction near segment boundaries.

To ensure consistency and physical plausibility in bottom depth attribution, the script adjudicates among multiple candidate values for each observation. It first selects the reported bottom depth (depth_b) associated with the measured water quality data when available; if missing, it substitutes the raster-derived bathymetric depth (.bathy_depth_b). This initial candidate value (.depth_candidate) is then compared to three constraints:

1. a selected minimum allowable bottom depth (min_depth_b = 1 m),
2. the reported sample depth plus a selected buffer (depth + 0.5 m),
3. and the event-level maximum sample depth plus a selected buffer (.depth_max + 0.5 m).

The final bottom depth (depth_b) is set to the **maximum** of these values to ensure that it is never shallower than 1 meter and is always at least 0.5 meters deeper than both the individual sample depth and any other sample collected during the same event. This logic prevents unrealistic depth assignments that would violate interpolation assumptions or physical boundaries.

To finalize the harmonized dataset, a schema enforcement step ensures that all required fields are present, properly typed, and consistently ordered. The expected schema is defined in an external JSON configuration file and includes spatial, temporal, physical, and project metadata fields (e.g., station, cbseg_92, date_time, depth_b, wb_lat_km, wb_lon_km, etc.). Variables that are not present in the source data are added with NA values of the correct type, enabling integration across varying data sets. The final dataset is then written in **Parquet format** using the arrow package. The data were partitioned by year, cbseg_92, source, and type to support fast, selective reading during later analysis. Zstandard (ZSTD) compression was applied at level 9 to reduce storage size while maintaining access speed.

DataFlow Data Processing (8/12/2025)

Updated: 8/12/2025

Introduction

This document summarizes the method for processing DataFlow water quality monitoring data

DataFlow data were downloaded from the EOTB website during February 14-19, 2025, and included 63 stations with data from 2001-2023. DataFlow data were also downloaded from the VECOS website[1] during February 8-11, 2025, and included 36 stations with data from 2003-2024. The data from the two data sources were harmonized and stored as described in Attachments 1-3.

[1] Chesapeake Bay National Estuarine Research Reserve in Virginia, Virginia Institute of Marine Science (CBNERR-VA VIMS), 2025. Virginia Estuarine and Coastal Observing System (VECOS). Data accessed from VECOS website: http://vecos.vims.edu; accessed 8-11 February 2025.

Attachment 1 – DataFlow Data Conversion Process

DataFlow data collected from Maryland and Virginia Dataflow monitoring programs were harmonized with a transformation function developed in R. This function standardizes column names, applies variable-specific transformations (e.g., unit conversion, range screening, qualifier-based filtering), and enables optional coordinate transformations. The outputs are appended across multiple files into unified tibbles—one for Maryland (`md_df`) and one for Virginia (`va_df`)—with consistent naming conventions, making them suitable for combined spatial and temporal analyses.

Data transformations and course-level spatial screening

- Column standardization and renaming to a harmonized schema.
- Value screening via numeric bounds and qualifier fields.
- Transformations: unit conversion, conditional replacements, date parsing.
- Row-level filtering based on comment flags (e.g., 'DATA REJECTED').
- Coordinate transformation for VA dataset (WGS84 to NAD83).
- Insertion of default columns (`depth`, `layer`) where needed.
- A course-level spatial screen was applied to ensure the data were located in the Chesapeake Bay tidal region (see Attachment 2).

Table A.1-1. Column mapping and transformations—Maryland DataFlow.

Source Column	**Final Name**[§]	**Transformation Description**
SAMPLE_DATE[†]	date_time	Parsed as date (`%Y-%m-%d`); combined with SAMPLE_TIME
SAMPLE_TIME[†]	date_time	Parsed as time (`%H:%M:%S`, time zone = EST); combined with SAMPLE_DATE
STATIONDESC	station	Renamed and trimmed to character
DEPTH_M	depth_b	Renamed
LATITUDE[‡]	latitude	Renamed
LONGITUDE[‡]	longitude	Renamed
SALINITY_PPT	salinity	Renamed
TEMP_C	temp	Renamed
DO_mg/L	do	Renamed; numeric screen: $-0.49 \leq x \leq 24.99$
DO_%SAT	do_sat	Renamed
PH	ph	Renamed
TURBIDITY_NTU	turb	Renamed
FLUOR	fluor	Renamed
CHL_UG/L	chl	Renamed

[†]SAMPLE_DATE and SAMPLE_TIME were combined to generate a full date_time field in EST.

[§]Additional Columns Added: depth = 0.5; layer = "BS"

[‡]Coordinate Transformation: None (location data provided in NAD83)

Table A.1-2. Column mapping and transformations—Virginia DataFlow.

Source Column	Final Name	Transformation Description
SAMPLE_DATETIME	date_time	Parsed with format `%Y-%m-%d %H:%M:%S`, time zone = EST
STATION	station	Renamed
LATITUDE‡	latitude	Renamed
LONGITUDE‡	longitude	Renamed
SAMPLE_DEPTH_BELOW_SURFACE	depth	Renamed
LAYER	layer	Renamed
DEPTH	depth_b	Renamed; conditional set based on `DEPTH_A`
BOAT_SPEED	speed	Multiplied by 0.514444 (knots to m/s); conditional set† on `BOAT_SPEED_A`
WTEMP	temp	Conditional set† on `WTEMP_A`
SPCOND	sp_cond	Conditional set† on `SPCOND_A`
SALINITY	salinity	Conditional set† on `SALINITY_A`
DO_SAT	do_sat	Conditional set† on `DO_SAT_A`
DO	do	Conditional set† on `DO_A`; numeric screen: $-0.49 \leq x \leq 24.99$
PH	ph	Conditional set† on `PH_A`
TURB_NTU	turb	Conditional set† on `TURB_NTU_A`
FLUOR	fluor	Conditional set† on `FLUOR_A`
TCHL_PRE_CAL	tchl_pre_cal	Conditional set† on `TCHL_PRE_CAL_A`
COMMENTS	--	Used for QC screen to remove records§, column not retained in final output

†Qualifier Codes Triggering Removal: CBF, CDB, CLF, CSW, CTF, CTS, CWD, GBO, GNV, GPC, GPF, GSC, GWL, GWM, NIR, NIS, NND, NNF, NOW, NPF, NQR, PDF, PDP, PSW, ###, CFS, CSF, CBO, QR, SPC, V

§Row Filters Applied: Records excluded if `COMMENTS` contains 'DATA REJECTED' or 'DATA SUSPECT'

‡Coordinate Transformation: EPSG:4326 (WGS84) to EPSG:4269 (NAD83)

Post-Harmonization Data Preparation and Storage

Following the harmonization of Virginia and Maryland Dataflow DataFlow datasets, the cleaned outputs were combined and structured to support efficient spatial analysis and downstream access as outlined below.

- **Dataset Merging and Metadata Annotation**. The Maryland and Virginia datasets were merged, and a source field was added to distinguish between Maryland (md_eotb) and and Virginia (va_vecos) data. A year variable was extracted from the date_time field for time-based filtering and partitioning. A type field was set to 'dataflow' for all data.

- **Spatial Enrichment.** The merged data were converted into an sf object using observation coordinates (longitude, latitude). The nearest Chesapeake Bay segment (cbseg_92) was identified for each record using st_nearest_feature() and appended to the merged data. Each station was also annotated with three additional metadata fields derived from geospatial rasters: depth_b (bottom depth), wb_lat_km (latitudinal waterbody coordinate), and wb_lon_km (longitudinal waterbody coordinate). These were extracted from 10-meter resolution raster surfaces (GeoTIFFs) specific to each Chesapeake Bay Program segment. For each segment, rasters were loaded, and values were extracted at station locations using the extract_nearest_valid() function. This function first performs a direct raster extraction and then resolves missing or NA values using a fallback algorithm based on Euclidean distances within a 500-meter bounding box. This fallback accounts for minor spatial mismatches between input coordinates and raster grid alignment, ensuring extraction near segment boundaries.

To ensure consistency in bottom depth attribution, the script adjudicated among three possible values for each observation: depth_b (as reported in the data), .bathy_depth_b (from the raster extraction), and depth (measured sample depth). The reported bottom depth (depth_b) is used when available; otherwise, it is substituted with the raster-derived depth (.bathy_depth_b), which is censored to a minimum of 1 meter. In both cases, it is compared with the sample depth, and the maximum of the two is retained to ensure the bottom depth is not shallower than the sample depth.

- **Standardization and Cleanup.** To finalize the harmonized dataset, a schema enforcement step ensures that all required fields are present, properly typed, and consistently ordered. The expected schema is defined in an external JSON configuration file and includes spatial, temporal, physical, and project metadata fields (e.g., station, cbseg_92, date_time, depth_b, wb_lat_km, wb_lon_km, etc.). Variables that are not present in the source data are added with NA values of the correct type, enabling integration across varying data sets. Long-form station names were converted to standardized short names via a lookup table (see Attachment 3). Errant depth values of 0.05m were recoded to 0.5m. Preliminary analyses also revealed apparent latitude/longitude discrepancies at the station/cruise level. The below 11 station cruises from the Lower Chester River and two (2) station cruises from the Upper Chester River were manually removed.

Lower Chester River

- April 28, 2005
- June 9, 2005
- August 17, 2005
- April 24, 2006
- June 28, 2006
- July 27, 2006
- August 29, 2006
- September 26, 2006
- October 12, 2006
- September 22, 2005
- May 24, 2006

Upper Chester River

- July 13, 2004
- August 19, 2004

The final dataset is then written in **Parquet format** using the arrow package. The data were partitioned by year, cbseg_92, source, and type to support fast, selective reading during later analysis. Zstandard (ZSTD) compression was applied at level 9 to reduce storage size while maintaining access speed.

Attachment 2 – Boundary Used to Screen Location Data

The Chesapeake Bay 92-segment Tidal Segmentation[2] was used to generate a simplified polygonal boundary suitable for course-level spatial screening tasks. The workflow proceeded as follows:

- **Union of Geometries**: All individual MULTIPOLYGON features in the original sf object were merged into a single unified geometry using st_union().
- **Projection Transformation**: The merged geometry was reprojected from geographic coordinates (EPSG:4326) to a projected coordinate reference system (EPSG:26918 – NAD83 / UTM Zone 18N) to enable accurate distance-based calculations.
- **Simplification**: The geometry was simplified using a 500-meter tolerance to reduce complexity while preserving overall shape and topology.
- **Buffering**: A 500-meter buffer was applied around the simplified polygon to create a screening zone that extends beyond the original boundary.
- **Reprojection to Latitude/Longitude**: The final buffered geometry was transformed back to geographic coordinates (EPSG:4326) for consistency with the source boundary.

The resulting sf object provides a simplified, buffered boundary of the Chesapeake Bay tidal region intended for coarse-level spatial screening.

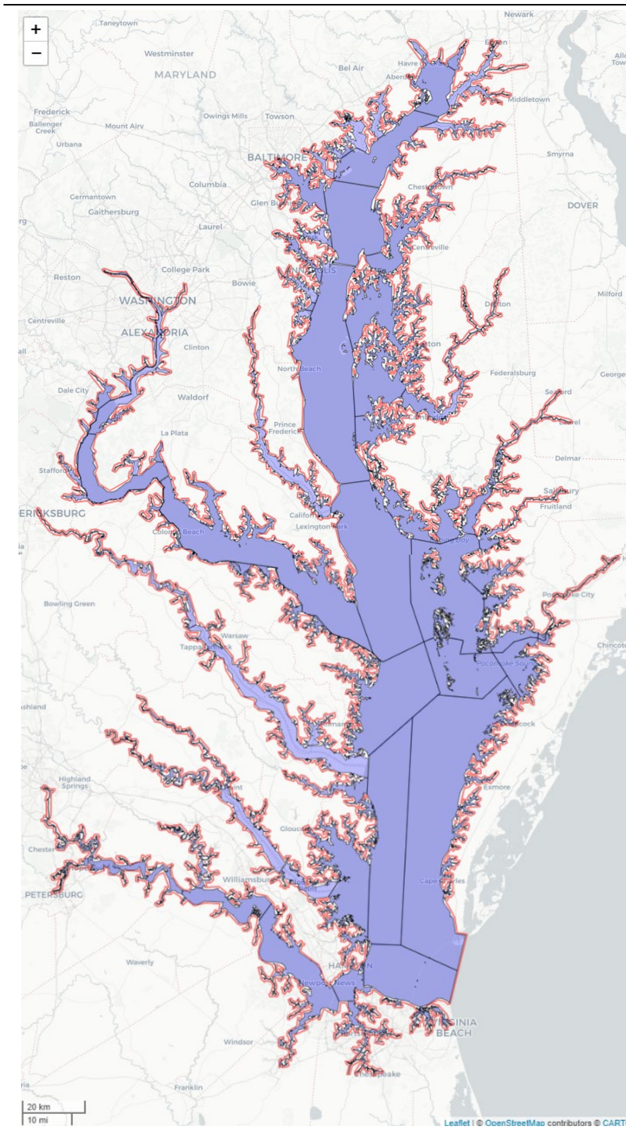

Figure A.2-1. Simplified and buffered spatial mask of the Chesapeake Bay tidal region (red) with Chesapeake Bay 92-segment Tidal Segmentation (blue).

[2] An updated GIS coverage can be obtained from https://data.chesapeakebay.net/datasets/ChesBay::chesapeake-bay-92-segment-tidal-segmentation. However, this analysis relied on a previously acquired version that included additional attributes.

Attachment 3 – Curated Station Names

To unify and streamline the representation of DataFlow data collected across the Chesapeake Bay, station identifiers were standardized using a consistent naming convention. Data collected by Maryland DNR's Dataflow program were prefixed with `md_df`, while those from Virginia's program were labeled with `va_df`. Original station names varied widely in length and format, including both short codes and verbose descriptors. To support graphing, labeling, and programmatic use, names were shortened while preserving some interpretability. The table below summarizes the mapping between original station names and their standardized identifiers used in subsequent analyses.

Table A.3-1. Mapping of original station descriptions and names to standardized identifiers used in analysis.

Station Description or Station Name	Curated Station
Big Annemessex River	md_df_ANNEM
Back River	md_df_BACK
Bohemia River	md_df_BOHEMIA
Bush River	md_df_BUSH
Chesapeake Bay Segment CB2OH East	md_df_CB2OH_E
Chesapeake Bay Segment CB2OH West	md_df_CB2OH_W
Chesapeake Bay Segment CB3MH East	md_df_CB3MH_E
Chesapeake Bay Segment CB3MH West	md_df_CB3MH_W
Chesapeake Bay Segment CB4MH Main	md_df_CB4MH_M
Chesapeake Bay Segment CB4MH North	md_df_CB4MH_N
Chesapeake Bay Segment CB4MH South	md_df_CB4MH_S
Chesapeake Bay Segment CB5MH East	md_df_CB5MH_E
Chesapeake Bay Segment CB5MH East and West	md_df_CB5MH_EW
Chesapeake Bay Segment CB5MH Main	md_df_CB5MH_M
Chesapeake Bay Segment CB5MH West	md_df_CB5MH_W
C&D Canal	md_df_CD_CANAL
Lower Chester River	md_df_CHEST_LOWER
Upper Chester River	md_df_CHEST_UPPER
Choptank River Harris Broad Tred Avon	md_df_CHOP_HBTA
Little Choptank River	md_df_CHOP_LITTLE
Lower Choptank River	md_df_CHOP_LOWER
Mid Choptank River	md_df_CHOP_MID
Upper Choptank River	md_df_CHOP_UPPER
Corsica River	md_df_CORSICA
Eastern Bay	md_df_EASTERN_BAY
Elk River	md_df_ELK
Fishing Bay	md_df_FISHING
Gunpowder River	md_df_GUNPOWDER
Honga River	md_df_HONGA
Magothy River	md_df_MAGOTHY
Manokin River	md_df_MANOKIN
Maryland Coastal Bays	md_df_MD_BAYS
Middle River	md_df_MIDDLE
Miles/Wye Rivers	md_df_MILES_WYE
Lower Nanticoke	md_df_NANT_LOWER
Upper Nanticoke	md_df_NANT_UPPER
Northeast River	md_df_NORTHEAST
Lower Patapsco	md_df_PAT_LOWER
Upper Patapsco	md_df_PAT_UPPER
Patuxent River	md_df_PAX
Lower Pocomoke	md_df_POC_LOWER
Upper Pocomoke	md_df_POC_UPPER
Potomac River Dahlgren	md_df_POT_DAH
Potomac River Lower Oligohaline	md_df_POT_LOH

Station Description or Station Name	Curated Station
Potomac River Lower	md_df_POT_LOW
Potomac River Maryland Tributaries	md_df_POT_MD_TRIBS
Potomac River Nomini	md_df_POT_NOMINI
Potomac River Piney Point	md_df_POT_PP
Potomac River St Marys River	md_df_POT_SM
Potomac River Tidal Fresh	md_df_POT_TF
Potomac River Upper Oligohaline	md_df_POT_UOH
Sassafras River	md_df_SASSAFRAS
Severn River	md_df_SEVERN
South River	md_df_SOUTH_RIVER
North Susquehanna River	md_df_SUSQ_NORTH
South Susquehanna River	md_df_SUSQ_SOUTH
Tangier Sounds North	md_df_TANG_N
Tangier Sounds North and West	md_df_TANG_NW
Tangier Sounds South	md_df_TANG_S
Tangier Sounds South and West	md_df_TANG_SW
Tangier Sounds West	md_df_TANG_W
West/Rhode Rivers	md_df_WEST_RHODE
Wicomico River	md_df_WICOMICO
APPTF	va_df_APPTF
CB5MH	va_df_CB5MH
CB5PH	va_df_CB5PH
CB6PH	va_df_CB6PH
CB7PH	va_df_CB7PH
CB8PH	va_df_CB8PH
CHKOH	va_df_CHKOH
CRRMH	va_df_CRRMH
ELIPH	va_df_ELIPH
JMSMH	va_df_JMSMH
JMSOH	va_df_JMSOH
JMSPH	va_df_JMSPH
JMSTF1	va_df_JMSTF1
JMSTF2	va_df_JMSTF2
LAFMH	va_df_LAFMH
LYNPH	va_df_LYNPH
MOBPH	va_df_MOBPH
MPNOH	va_df_MPNOH
MPNTF	va_df_MPNTF
PIAMH	va_df_PIAMH
PMKOH	va_df_PMKOH
PMKTF	va_df_PMKTF
POCMH	va_df_POCMH
POCOH	va_df_POCOH
POTMH_COA	va_df_POTMH_COA
POTMH_LMA	va_df_POTMH_LMA
POTMH_MAT	va_df_POTMH_MAT
POTMH_NOM	va_df_POTMH_NOM
POTMH_ROS	va_df_POTMH_ROS
POTMH_UMA	va_df_POTMH_UMA
POTMH_YEO	va_df_POTMH_YEO
RPPMH	va_df_RPPMH
RPPOH	va_df_RPPOH
RPPTF	va_df_RPPTF
YRKMH	va_df_YRKMH
YRKPH	va_df_YRKPH

High-Frequency Data Transformations (5/30/2025)

Updated: 5/30/2025

Table 1. Column mapping and transformations—Maryland DNR DataFlow.

Source Column	Final Name[§]	Transformation Description
SAMPLE_DATE[†]	date_time	Parsed as date (`%Y-%m-%d`); combined with SAMPLE_TIME
SAMPLE_TIME[†]	date_time	Parsed as time (`%H:%M:%S`, time zone = EST); combined with SAMPLE_DATE
STATIONDESC	station	Renamed and trimmed to character
DEPTH_M	depth_b	Renamed
LATITUDE[‡]	latitude	Renamed
LONGITUDE[‡]	longitude	Renamed
SALINITY_PPT	salinity	Renamed
TEMP_C	temp	Renamed
DO_mg/L	do	Renamed; numeric screen: -0.49 ≤ x ≤ 24.99
DO_%SAT	do_sat	Renamed
PH	ph	Renamed
TURBIDITY_NTU	turb	Renamed
FLUOR	fluor	Renamed
CHL_UG/L	chl	Renamed

[†]SAMPLE_DATE and SAMPLE_TIME were combined to generate a full date_time field in EST.

[§]Additional Columns Added: depth = 0.5; layer = "BS"

[‡]Coordinate Transformation: None (location data provided in NAD83)

Table 2. Column mapping and transformations—Maryland DNR ConMon.

Source Column	Final Name	Transformation Description
SAMPLE_DATE[†]	date_time	Parsed as date (`%m/%d/%Y`); combined with SAMPLE_TIME
SAMPLE_TIME[†]	date_time	Parsed as time (`%H:%M:%S`, time zone = EST); combined with SAMPLE_DATE
STATION	station	Renamed
LAYER	layer	Renamed
SAMPLEDEPTH_M	depth	Renamed
SALINITY_PPT	salinity	Renamed
TEMP_C	temp	Renamed
DO_mg/L	do	Renamed; numeric screen: -0.49 ≤ x ≤ 24.99
DO_PCTSAT	do_sat	Renamed
PH	ph	Renamed
TURB_NTU	turb	Renamed
CHL_UG/L	chl	Renamed

[†]SAMPLE_DATE and SAMPLE_TIME were combined to generate a full date_time field in EST.

Table 3. Column mapping and transformations—Maryland DNR Profiler.

Source Column	Final Name	Transformation Description
SAMPLE_DATE†	date_time	Parsed as date (`%m/%d/%Y`); combined with SAMPLE_TIME
SAMPLE_TIME†	date_time	Parsed as time (`%H:%M:%S`, time zone = EST); combined with SAMPLE_DATE
STATION	station	Renamed
ROUNDEDDEPTH_M	layer	Renamed; converted to character
ACTUALDEPTH_M	depth	Renamed
SALINITY_PPT	salinity	Renamed
TEMP_C	temp	Renamed
DO_mgL	do	Renamed; numeric screen: -0.49 ≤ x ≤ 24.99
DO_%SAT	do_sat	Renamed
PH	ph	Renamed
TURBIDITY_NTU	turb	Renamed
CHL_UGL	chl	Renamed

†SAMPLE_DATE and SAMPLE_TIME were combined to generate a full date_time field in EST.

Table 4. Column mapping and transformations—NOAA Vertical Array.

Source Column	Final Name	Transformation Description
STATION	station	Renamed
TIME	date_time	Parsed as date_time (`%Y-%m-%dT%H:%M:%SZ`; time zone converted from UTC to EST)
LATITUDE	latitude	Renamed
LONGITUDE	longitude	Renamed
Z	depth	multiply by -1
SEA_WATER_ELECTRICAL_ CONDUCTIVITY	sp_cond	Conditional set† on `*_QC_AGG`; multiply by 1000;
SEA_WATER_TEMPERATURE	temp	Conditional set† on `*_QC_AGG`;
MASS_CONCENTRATION_OF_ CHLOROPHYLL_IN_SEA_WATER	chl	Conditional set† on `*_QC_AGG`;
MASS_CONCENTRATION_OF_ OXYGEN_IN_SEA_WATER -or- MASS_CONCENTRATION_OF_ OXYGEN_IN_SEA_WATER_CORRECTED	do	Conditional set† on `*_QC_AGG`; numeric screen: -0.49 ≤ x ≤ 24.99
SEA_WATER_PRACTICAL_SALINITY	salinity_psu	Conditional set† on `*_QC_AGG`

†Qualifier Codes Triggering Data Retention: 1

Table 5. Column mapping and transformations—Virginia DEQ DataFlow.

Source Column	Final Name	Transformation Description
SAMPLE_DATETIME	date_time	Parsed with format `%Y-%m-%d %H:%M:%S`, time zone = EST
STATION	station	Renamed
LATITUDE[‡]	latitude	Renamed
LONGITUDE[‡]	longitude	Renamed
SAMPLE_DEPTH_BELOW_SURFACE	depth	Renamed
LAYER	layer	Renamed
DEPTH	depth_b	Renamed; conditional set based on `DEPTH_A`
BOAT_SPEED	speed	Multiplied by 0.514444 (knots to m/s); conditional set[†] on `BOAT_SPEED_A`
WTEMP	temp	Conditional set[†] on `WTEMP_A`
SPCOND	sp_cond	Conditional set[†] on `SPCOND_A`
SALINITY	salinity	Conditional set[†] on `SALINITY_A`
DO_SAT	do_sat	Conditional set[†] on `DO_SAT_A`
DO	do	Conditional set[†] on `DO_A`; numeric screen: $-0.49 \leq x \leq 24.99$
PH	ph	Conditional set[†] on `PH_A`
TURB_NTU	turb	Conditional set[†] on `TURB_NTU_A`
FLUOR	fluor	Conditional set[†] on `FLUOR_A`
TCHL_PRE_CAL	tchl_pre_cal	Conditional set[†] on `TCHL_PRE_CAL_A`
COMMENTS	--	Used for QC screen to remove records[§], column not retained in final output

[†]Qualifier Codes Triggering Removal: CBF, CDB, CLF, CSW, CTF, CTS, CWD, GBO, GNV, GPC, GPF, GSC, GWL, GWM, NIR, NIS, NND, NNF, NOW, NPF, NQR, PDF, PDP, PSW, ###, CFS, CSF, CBO, QR, SPC, V

[§]Row Filters Applied: Records excluded if `COMMENTS` contains 'DATA REJECTED' or 'DATA SUSPECT'

[‡]Coordinate Transformation: EPSG:4326 (WGS84) to EPSG:4269 (NAD83)

Table 6. Column mapping and transformations—Virginia DEQ ConMon.

Source Column	Final Name	Transformation Description
SAMPLE_DATETIME	date_time	Parsed as date_time (`%Y-%m-%d %H:%M:%S`, time zone = EST)
STATION	station	Renamed
LAYER	layer	Renamed
LATITUDE‡	latitude	Renamed
LONGITUDE‡	longitude	Renamed
DEPTH	depth	Conditional set† on `DEPTH_A`
PRESSURE_SENSOR_HEIGHT_ABOVE_BOTTOM	.sample_height_from_bottom*	Renamed
WTEMP	temp	Conditional set† on `WTEMP_A`
SPCOND	sp_cond	Conditional set† on `SPCOND_A`
SALINITY	salinity	Conditional set† on `SALINITY_A`
DO_SAT	do_sat	Conditional set† on `DO_SAT_A`
DO	do	Conditional set† on `DO_A`; numeric screen: -0.49 ≤ x ≤ 24.99
PH	ph	Conditional set† on `PH_A`
TURB_NTU	turb	Conditional set† on `TURB_NTU_A`
FLUOR	fluor	Conditional set† on `FLUOR_A`
TCHL_PRE_CAL	tchl_pre_cal	Conditional set† on `TCHL_PRE_CAL_A`
COMMENTS	--	Used for QC screen to remove records§, column not retained in final output

†Qualifier Codes Triggering Removal: CBF, CDB, CLF, CSW, CTF, CTS, CWD, GBO, GNV, GPC, GPF, GSC, GWL, GWM, NIR, NIS, NND, NNF, NOW, NPF, NQR, PDF, PDP, PSW, ###, CFS, CSF, CBO, QR, SPC, V

§Row Filters Applied: Records excluded if `COMMENTS` contains 'DATA REJECTED' or 'DATA SUSPECT'

‡Coordinate Transformation: EPSG:4326 (WGS84) to EPSG:4269 (NAD83)

*Added to depth to compute bottom depth (`depth_b`)

Table 7. Column mapping and transformations—Virginia DEQ Profiler.

Source Column	Final Name	Transformation Description
SAMPLE_DATETIME	date_time	Parsed as date_time (`%m/%d/%Y %I:%M:%S %p`, time zone = EST)
STATION	station	Renamed
LAYER	layer	Renamed
LATITUDE‡	latitude	Renamed
LONGITUDE‡	longitude	Renamed
PROF_TOTAL_DEPTH	Depth_b	Renamed
PROFILE_DEPTH	depth	Conditional set† on `PROFILE_DEPTH_A`
WTEMP	temp	Conditional set† on `WTEMP_ A`
SPCOND	sp_cond	Conditional set† on `SPCOND_ A`
SALINITY	salinity	Conditional set† on `SALINITY_ A`
DO_SAT	do_sat	Conditional set† on `DO_SAT_ A`
DO	do	Conditional set† on `DO_A`; numeric screen: -0.49 ≤ x ≤ 24.99
PH	ph	Conditional set† on `PH_ A`
TURB_NTU	turb	Conditional set† on `TURB_NTU_ A`
FLUOR	fluor	Conditional set† on `FLUOR_ A`
TCHL_PRE_CAL	tchl_pre_cal	Conditional set† on `TCHL_PRE_CAL_ A`
COMMENTS	--	Used for QC screen to remove records§, column not retained in final output

†Qualifier Codes Triggering Removal: CBF, CDB, CLF, CSW, CTF, CTS, CWD, GBO, GNV, GPC, GPF, GSC, GWL, GWM, NIR, NIS, NND, NNF, NOW, NPF, NQR, PDF, PDP, PSW, ###, CFS, CSF, CBO, QR, SPC, V

§Row Filters Applied: Records excluded if `COMMENTS` contains 'DATA REJECTED' or 'DATA SUSPECT'

‡Coordinate Transformation: EPSG:4326 (WGS84) to EPSG:4269 (NAD83)

Chesapeake Monitoring Cooperative Data Integration (6/13/2025)

Updated: 6/13/2025—**INTERNAL DRAFT**

This document provides a cursory overview of incorporating the Chesapeake Monitoring Cooperative data into an overall data file with other related data. Non-traditional/volunteer-based partner data were downloaded from datahub.chesapeakebay.net/FileDownloads for 2017-2023. Additional MDE data for 2024 were provided by the CBP via an emailed spreadsheet.

Data downloaded from the DataHub were screened to only keep Tier 3 data. All further processing steps were identical to those used when processing traditional partner data. A once-off program was written to concatenate the 2024 MDE data to the working data set. (Figure 1 displays site locations.)

Data were harmonized for consistency with previously processed data (e.g., variable naming, assigning of geolocation parameters (bathymetry, waterbody location)).

Data were assigned a source = 'cbp_datahub' and type = 'cmc_t3'. Updated observations counts are below.

```
   source      type         folders        n
 1 cbp_datahub cmc_t3           154    34203
 2 cbp_datahub conmon_cal       507    74656
 3 cbp_datahub dataflow_cal     278    83651
 4 cbp_datahub wqmp            3030   797476
 5 md_eotb     conmon           369 13879294
 6 md_eotb     dataflow         289  4724718
 7 md_eotb     profiler          16   240160
 8 noaa        buoy              17    69669
 9 noaa        vert_array         6   445536
10 va_vecos    conmon           213  7541890
11 va_vecos    dataflow         257  9560881
12 va_vecos    profiler          10   185355
```

```
  type         cbp_datahub  md_eotb    noaa va_vecos
1 cmc_t3             34203        0       0        0
2 conmon_cal         74656        0       0        0
3 dataflow_cal       83651        0       0        0
4 wqmp              797476        0       0        0
5 conmon                 0 13879294       0  7541890
6 dataflow               0  4724718       0  9560881
7 profiler               0   240160       0   185355
8 buoy                   0        0   69669        0
9 vert_array             0        0  445536        0
```

Figure 1—CMC station locations – red sites removed after visual QC.

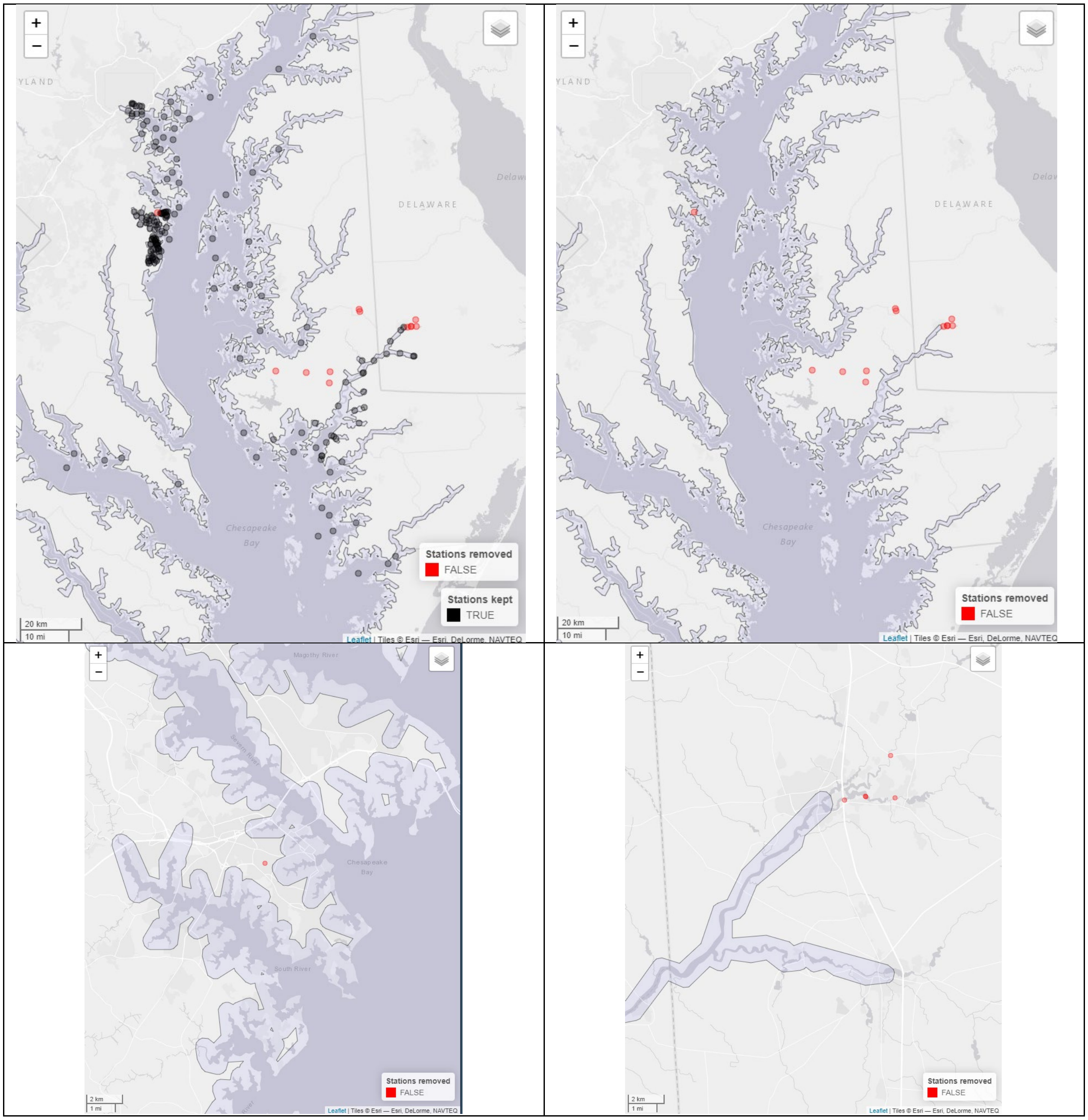

Figure 2-MDE 2024 CMC data sent to CBP- all kept