WWF Traffic

Technical Guide

Wildlife Trafficking Incident Analysis — Methodology & Calculation Reference

Version 1.0 · Generated August 04, 2026

1. Overview

The **WWF Traffic** workflow logs in to the **TRAFFIC Wildlife Trade Portal** (wildlifetradeportal.org) on the user's behalf, downloads wildlife trafficking incident export CSVs for a configured time range, filters the resulting incidents to a 30 km buffer of the **Greater Virunga Landscape (GVL)** boundary, and produces an interactive dashboard of trafficking trends by species, incident category, and country.

Unlike the EarthRanger-connected workflows in this fleet, WWF Traffic has no EarthRanger data source. Instead, a lightweight REST client (§2.1) authenticates directly against the portal's API, searches by date range, species, and country, and exports the matching incidents as CSV — the user only supplies portal credentials (or an access token) and a time range, with no manual export/upload step and no browser required. An optional local CSV can still be merged in for supplementary records (§3.8), and the only other network dependency is a one-time download of the GVL boundary file from Dropbox.

> **Note:** This workflow currently has no test-cases.yaml in the repository. CI's test-case validation step will fail until one is added — see the Troubleshooting page.

2. Dependencies & Prerequisites

2.1 Traffic Portal Connection & Download

connect_to_portal builds a WildlifeTradePortalIO REST client — a small requests-based wrapper around the portal's (unofficial, reverse-engineered) JSON API, not a browser. It accepts either an access_token (advanced) or a username/password pair entered on the workflow's own config form — there is no platform-managed named connection for the Traffic Portal yet, so credentials are regular form fields rather than an env-var-backed connection. server, timeout, and verify (whether a supplied token is validated immediately via a lightweight account lookup) round out the advanced fields.

The resulting client is passed to export_search_results along with the workflow's configured time_range, species and countries filters, and a save_to path (ECOSCOPE_WORKFLOWS_RESULTS). It runs the whole search-then-export pipeline in one call: paginate through every incident matching the filters (date_from/date_till sent as plain ISO YYYY-MM-DD), collect their Unique_IDs, then request a server-side export job for those IDs and download each resulting file via a presigned S3 URL.

countries defaults to **Rwanda**, **Uganda**, and **Congo, Democratic Republic of The** — restricting the search itself, not just the local filter in §3.5. This matters for pagination: without a country filter, the search has to walk through every incident worldwide in the requested time range before the true count for these three countries is known, which for a wide/unfiltered time range can mean the search never reaches far enough back to surface older Virunga-region incidents at all. Scoping the search server-side means the full requested history for the region is reachable in far fewer requests.

Export type	Toggle	Downloaded filename pattern
Incident (always included)	—	wtp_export_incident_<id>_<timestamp>.csv
Species/commodity	include_species	wtp_export_species_<id>_<timestamp>.csv
Trade-route/location	include_locations	wtp_export_locations_<id>_<timestamp>.csv

Both toggles default to true in this workflow's spec.yaml — all three files are needed for the merge described in §2.2, so disabling either one will cause the downstream column-presence check in §3.1 to fail. The default justification text is "Research and analysis of wildlife trade incidents in the Virunga region for conservation purposes." (reason).

sub_page_size controls how many rows are requested per underlying API page (client default 100 if left blank) — it is purely a batching size, *not* a cap on the total result set. Every incident matching the filters is always fetched, walking every page regardless of how many that turns out to be. If the portal responds with an HTTP 429 (rate limited), the client automatically waits (honoring a Retry-After header when present, otherwise an exponential backoff) and retries — up to 5 times by default — before raising.

The downloaded file paths feed directly into load_and_merge_csvs.file_paths (§2.2) — there is no separate manual upload step in this workflow's config form.

2.2 Input Data — Three Merged CSV Exports

load_and_merge_csvs takes the list of file paths returned by the download step (§2.1) and merges them sequentially on Report ID. The workflow's retained columns span three functional groups of data that, in the TRAFFIC portal export, come from separate files:

File role	Key columns used
Incident / case file	Category of Incident, Country of Incident, Date of Incident, Outcome, Number of People Arrested/Charged/Fined/Imprisoned
Species / commodity file	Full Scientific Name, Item / Commodity Type, Count, Weight, Unit of Weight, Kingdom, Phylum, Class, Order, Family, Common Name
Trade-route / location file	Role, Order in Trade Route, Country, Latitude, Longitude

Category of Incident, Country of Incident, and Date of Incident appear in more than one file. After the merge, pandas suffixes the duplicated columns (_x, _y, …); map_columns retains only the _x variant of each (i.e. the value from whichever file was loaded first) and renames it to a standard lowercase/snake_case schema (e.g. category_of_incident, country_of_incident, date_of_incident).

2.3 Grouping Strategy

set_groupers is restricted via rjsf-overrides to only **ValueGrouper** fields (temporal and spatial grouping are disabled for this workflow), and further restricted to exactly two grouping columns:

Grouper	Index name	Effect
Species	category	One dashboard view per mapped species group
Country	country	One dashboard view per country

Left blank, the workflow produces a single combined view. The selectable index_name values changed from common_group/country_of_incident to category/country in this revision of spec.yaml — the dropdown labels shown to the user (Species / Country) are unchanged.

2.4 GVL 30 km Buffer Boundary

fetch_and_persist_file downloads gvl_30km_buffer.parquet from Dropbox (overwrite_existing: true, 3 retries) and loads it via load_df. It is reprojected to **EPSG:4326** (reproject_gdf) before being used both for the spatial join (§3.5) and as a static map layer (§4).

2.5 Base Map Tile Layers

Layer	Opacity	Max zoom
ESRI World Topographic Map	100 %	20
ESRI World Imagery	50 %	20

2.6 Colour Palette

set_color_palette defaults to the Matplotlib **tab10** named colormap, but accepts a custom list of hex colours instead. The chosen palette is applied twice via apply_cmap — once to category_of_incident (output column incident_colors) and once to common_group (output column species_colors), both as RGBA tuples at full alpha.

3. Data Ingestion Pipeline

3.1 Column Mapping

After the CSV merge, map_columns retains 23 columns (see §2.2) and renames them to snake_case (raise_if_not_found: true — the run fails immediately if any expected column is missing from the merged data).

3.2 Trade Route De-duplication

A trafficking incident may involve multiple trade-route locations (Origin, Transit, Destination). extract_trade_route retains a single representative row per (report_id, full_scientific_name, commodity_type) combination, using the role column to identify route roles and order_in_trade_route to determine which row to keep when more than one location is present.

3.3 Numeric Conversion & Geometry

convert_column_values_to_numeric coerces count, weight, the four arrest/charge/fine/imprisonment count columns, and latitude/longitude from text to numeric types. df_to_point_gdf then builds point geometries from latitude/longitude in **EPSG:4326**.

3.4 Spatial Join Against the GVL Boundary

spatial_join performs an inner, within-predicate join between the incident points and the reprojected GVL 30 km buffer, dropping any incident that falls outside the buffer.

3.5 Country Restriction

filter_row_values additionally restricts country_of_incident to exactly three values: **Rwanda**, **Uganda**, and **Congo, Democratic Republic of The**. This runs *after* the spatial join, so an incident can pass the 30 km buffer test geometrically but still be dropped here if its country_of_incident text doesn't exactly match one of these three strings.

3.6 Species & Category Mapping

add_mapped_column_value maps full_scientific_name to a simplified common_group label (e.g. Elephantidae/Loxodonta → Elephant, Manidae → Pangolin, Hippopotamus amphibius → Hippo) and category_of_incident to a short incident_category label (Seizure, Poaching, Smuggling, Human-Wildlife Conflict, Enforcement). Both mappings use keep_unmapped: true — a value not covered by the mapping table passes through as its original text rather than being dropped.

3.7 Temporal Index

add_temporal_index keys the dataset to date_of_incident, grouped by the configured grouper. decompose_datetime then extracts year and month components (prefixed incident_, e.g. incident_year) for the time-series aggregations in §6.

3.8 Optional Local Data Merge

load_extra_data (task load_df) optionally loads a user-supplied local CSV of additional incident records — its file_path is left unbound in spec.yaml so it remains a plain config-form field, unlike every other input in this pipeline, which is now sourced automatically from the Traffic Portal download (§2.1). merge_traffic_xlsx then merges it against the temporal-indexed portal dataset from §3.7, and the merged result is re-converted to a point GeoDataFrame before colour mapping (§2.6).

4. Static Map Layers

One static layer — the GVL boundary — is composited onto both the Incidents Map and the Species Map.

4.1 GVL Boundary Layer

Property	Value
Fill	None (outline only)
Line colour	Black (0, 0, 0)
Line width	1.25 px (min 1, max 5)
Opacity	45 %
Legend	“GVL Boundary (30km buffer)”

5. Map Outputs — Methodology

5.1 Incidents Map

create_scatterplot_layer renders each incident as a point coloured by incident_colors (from §2.6), at 2.35 px radius and 75 % opacity with stroked outlines. Combined with the GVL boundary layer, auto-zoomed to the incident extent (max zoom 12 for view-state calculation, 10 for the rendered map), and persisted as HTML.

5.2 Species Map

Identical construction to the Incidents Map, but points are coloured by species_colors instead of incident_colors — same radius, opacity, and stroke settings.

5.3 Filename Suffix Collision

Note: The Incidents Map is persisted with filename_suffix: incidents_by_category — the exact same suffix used by the incidents-by-category bar chart (§6.1). Likewise, the Species Map uses filename_suffix: incidents_by_species, the same suffix as the incidents-by-species bar chart. This is a naming quirk in spec.yaml, not a runtime collision — each persisted file is additionally prefixed with a content hash — but it means the suffix alone cannot distinguish a map file from a chart file.

6. Charts

6.1 Incidents by Species / by Category

Two draw_time_series_bar_chart calls share the same x-axis (date_of_incident, bucketed yearly), y-axis (report_id, counted), and colour column (species_colors — used for *both* charts, even the by-category one). They differ only in the category field: common_group for the species chart, incident_category for the category chart.

7. Summary Metrics

7.1 Per-Year Aggregations

aggregate_by computes, per group and per incident_year: total incident count (no_of_incidents), incident count by incident_year × common_group, total people arrested (sum), and total people imprisoned (sum).

7.2 Conviction Rate — Two Independent Calculations

There are **two separate conviction-rate computations** in this workflow, and only one of them reaches the dashboard:

Path	Scope	Reaches dashboard?
add_conviction_rate	Per incident_year	No — computed but never referenced downstream
conviction_ratio → conviction_rate_pct	All-time single ratio	Yes — feeds the Conviction Rate widget

The dashboard's **Conviction Rate** widget is total_imprisoned ÷ total_arrested × 100, computed with safe_divide (fill_invalid: 0.0) across the entire time range — not a per-year figure.

7.3 Ivory Seizures

filter_row_values restricts commodity_type to Ivory Pieces - Raw, Ivory - Worked, and Tusk, then aggregate_by sums weight per incident_year, feeding the **Total Seizures of Ivory (kg)** widget (all-time sum).

7.4 Most Trafficked Species

aggregate_by counts incidents per common_group across the full time range; get_top_category picks the highest count and formats it as a label (e.g. “Elephant — 214 incidents”) for the **Top Trafficked Species** text widget.

8. Summary Table & CSV Export

subset_columns narrows the dataset to date_of_incident, incident_category, role, and common_group; map_columns renames these to display labels (Date, Incident Category, Role, Species); draw_table renders it with sorting and filtering enabled (download disabled).

Separately, persist_df writes the *full* processed per-group dataset (all mapped/renamed columns, not the narrowed summary table) to CSV with an auto-generated filename.

9. Interactive Dashboard

gather_dashboard assembles the dashboard from twelve widgets:

Widget	Type	Source
Total Number of Incidents	Single value	incidents_per_year (sum)
Incidents This Year vs Last Year	Period comparison	incidents_per_year
Top Trafficked Species	Text	species_totals → get_top_category
Total Suspects Arrested	Single value	total_arrested (sum)
Total Suspects Convicted	Single value	total_imprisoned (sum)
Conviction Rate	Single value	conviction_rate_pct — see §7.2
Total Seizures of Ivory (kg)	Single value	ivory_weight_per_year (sum)
Incidents Map	Map	draw_incidents_map
Species Map	Map	draw_species_map
Incidents by Species Over Time	Plot	ts_species_chart
Incidents by Category Over Time	Plot	ts_category_chart
Summary Table of Incidents	Table	filtered_table_html

Note: Most widgets skip if their input DataFrame is empty (skipif: any_is_empty_df); the Incidents-YoY widget instead uses skipif: never, so it always renders even for groups with no data.

10. Output Files

All files are written to $ECOSCOPE_WORKFLOWS_RESULTS.

File / suffix	Content
<hash>_incidents_by_species.html	Incidents-by-species bar chart *or* Species Map (see §5.3)
<hash>_incidents_by_category.html	Incidents-by-category bar chart *or* Incidents Map (see §5.3)
<hash>_summary_table_of_incidents.html	Filterable/sortable summary table
<group>.csv	Full processed incident dataset for that group

11. Workflow Execution Logic

11.1 Skip Conditions

Two default skip conditions apply to every task (task-instance-defaults): **any_is_empty_df** and **any_dependency_skipped**. Most widget tasks additionally opt into any_is_empty_df explicitly; the Incidents-YoY widget instead uses skipif: never.

11.2 Data Flow Summary

Stage	Tasks
Setup	Workflow details, time range, timezone, groupers, base maps, colour palette
Portal download	Connect to Traffic Portal (credentials or token) → search & export matching incidents (Rwanda/Uganda/Congo DRC by default) for time range
Ingest	Merge downloaded CSVs (+ optional local CSV) → map columns → dedupe trade routes → numeric conversion → point GDF
GVL context	Download GVL buffer → reproject → spatial join → country restriction
Labelling	Species mapping → incident category mapping → temporal index → optional local merge → colormaps
Group split	split_groups by configured grouper (or single combined group)
Metrics	Per-year incidents/arrests/imprisonments, conviction rate, ivory totals, top species
Charts	Species & category time-series bar charts
Maps	GVL boundary layer + incident/species scatterplot layers → Incidents Map, Species Map
Table & CSV	Subset/rename → summary table; full dataset → CSV
Dashboard	gather_dashboard combines all 12 widgets

12. Software Versions

Package	Version	Role
ecoscope-platform	>=2.15.0, <2.16.0	Consolidated core task library and workflow engine
ecoscope-workflows-ext-custom	0.1.0rc14.*	Utility tasks (maps, layers, column mapping)
ecoscope-workflows-ext-ste	0.0.0rc1.*	Spatial operations tasks (spatial join, view state)
ecoscope-workflows-ext-wwf-virunga	0.0.0rc7.*	WWF Virunga domain tasks (colormaps, aggregation, ratios, top-category, portal connect/search/export)
pydeck	0.9.2	Deck.gl map rendering
opentelemetry-sdk	>=1.20.0, <2.0.0	Observability/tracing

This workflow runs on the consolidated ecoscope-platform package scheme. Packages are distributed via the repo.prefix.dev conda channels and pinned to compatible version ranges. The runtime environment is managed by **pixi**.