VRS X Operator's Handbook

- 01. Startup and Login
- 02. General Settings
- 03. Feeds and Merged Feeds
- 04. Server Settings
- 05. Resources and Visuals
- 06. Users and Security
- 07. Backup, Update, Recovery
- 08. Troubleshooting
- 09. Feeds: Smart Ports and MLAT
- 10. Feed Filter
- 11. Merged Feeds
- 12. Resources: Flags and Silhouettes
- 13. Custom Operator Logo
- 14. Aircraft Colors and Operator Colors
- 15. AircraftDB Editor and Custom Photos
- 16. Coverage
- 17. Aircraft Model and Operator Aliases

01. Startup and Login

Startup and Login

1. First startup

1. Start the VRS X service/application.
2. Open the admin panel:
   - default: `http://localhost:8085/admin`
   - if `AdminPort` was changed: `http://<host>:<admin-port>/admin`
3. Complete activation (if required).
4. Set the administrator password.

2. If the admin panel does not open

The most common issue is a port conflict.

Linux

```
ss -ltnp | rg ':8085|:5000|:8080'
```

Windows (PowerShell)

```
Get-NetTCPConnection -State Listen | Where-Object {$_.LocalPort -in 8080,8085,5000}
```

If the port is already in use:

1. Stop the conflicting application.
2. Open the admin panel.
3. Change `Admin Port` to a free port.
4. Save the configuration and restart VRS X.

3. Minimal post-install workflow

1. `General Settings` - set core radar options.
2. `Feeds` - add at least one data source.
3. `Server Settings` - configure ports and network access.
4. Restart the service.
5. Verify:
   - `http://<host>:<web-port>/AircraftList.json`
   - `http://<host>:<admin-port>/admin`

02. General Settings

General Settings

`General Settings` should contain only global radar behavior and presentation options.

What to configure here

1. Instance name and high-level identity settings.
2. General map behavior and display options.
3. User-facing options rather than infrastructure options.

Recommended configuration order

1. Set general options.
2. Save.
3. Go to `Feeds` and configure data sources.
4. Return to visual fine-tuning only after feed data is stable.

Best practices

1. Apply small changes and save in steps.
2. After major changes, refresh radar and verify `AircraftList.json`.
3. Avoid mixing UI changes and infrastructure changes in one pass.

Current UI behavior notes

Theme modes and map controls

1. Available themes in radar UI: `Classic`, `Futuristic`, `OverFuture`, `Phosphor CRT`.
2. Map action controls (zoom, layers, location, visual panel) use a unified button layout.
3. In `Futuristic` and `OverFuture`, controls should be displayed as standalone buttons without an extra outer frame around the whole control group.

4. `Phosphor CRT` is an immersive retro mode: it uses one fixed CRT-styled map layer, hides modern map/label tuning controls, applies a phosphor treatment to operator logos, draws OpenAIP airspace boundaries and airports as lightweight CRT vectors when an OpenAIP API key is configured, and supports keyboard selection with `ArrowUp` / `ArrowDown`, `ArrowLeft` / `ArrowRight`, `Enter`, and `Escape`.
5. After frontend updates, use a hard refresh to make sure the latest CSS is loaded.

Mobile aircraft detail sheet

1. On mobile viewport, selecting an aircraft opens a bottom sheet in collapsed mode.
2. Tap the sheet handle or swipe up to expand the sheet.
3. Swipe down (when detail scroll is already at top) to collapse.
4. Closing aircraft detail resets sheet state and scroll position.
5. UI is safe-area aware (`viewport-fit=cover`), so bottom spacing should respect device insets on iOS-like devices.

Aircraft list sorting

1. Sortable columns include `Op`, `ICAO`, `Reg`, `Callsign`, `Alt`, `Spd`.
2. Sorting preference is persisted in browser storage.

03. Feeds and Merged Feeds

Feeds and Merged Feeds

This page is a quick navigator for feed topics.

Recommended order

1. Configure feed source and MLAT behavior.
2. Use `Add Feed Set` for multi-port receivers on one IP.
3. Configure merged feeds.
4. Check merged feed coverage if you need to validate reception footprint.
5. Apply feed filter rules last.

Detailed pages:

1. 09-Feeds-Smart-Ports-and-MLAT.md
2. 10-Feed-Filter.md
3. 11-Merged-Feeds.md
4. 16-Coverage.md

Add Feed Set

Use `/admin/feeds` -> `Add Feed Set` when one receiver exposes several standard ports from the same address. It is a shortcut, not a separate feed type: VRS X creates normal feeds underneath with names like `FEED01_ADSB`, `FEED01_MLAT`, and `FEED01_MLATHUB`.

Typical use:

1. enter one `Base Name`,
2. enter one `Address / IP`,
3. select the primary ADS-B port (`30003` or `30005`),
4. optionally add dedicated MLAT (`30105` or `30106`),
5. optionally add MLAT Hub (`31005` or `31006`),
6. optionally attach all created feeds to an existing merged feed.

Important behavior:

1. all generated feeds are TCP active feeds,
2. custom ports, passive mode, and serial receivers still require normal `Add Feed`,
3. endpoints already configured with the same normalized address and port are skipped,
4. when not linking to a merged feed, primary ADS-B stays visible in the radar dropdown while MLAT-only feeds stay hidden by default,
5. when linking to a merged feed, the visibility checkbox applies to created feeds and the selected merged feed receives all new source IDs.

Coverage after merged feeds

`/admin/coverage` works on enabled merged feeds. It visualizes where aircraft were actually observed by the merged feed sources and can be used to validate coverage holes, inactive sources, and archive history. See 16-Coverage.md.

Common issue: selected feed still shows too much

Symptom: selecting a specific feed still shows aircraft from other feeds.

Checklist:

1. Verify `feedId` mapping in output data.
2. Verify frontend sends the correct `feed` parameter.
3. Compare `All feeds` vs single feed for the same bbox.
4. Repeat test on a second feed for the same user.
5. Remember there is a short stability window: aircraft recently seen by selected feed can remain visible briefly to reduce flicker.
6. For merged feeds, filtering remains anchored to selected feed ID (no automatic expansion to all merged sources).

04. Server Settings

Server Settings

Treat `Server Settings` as infrastructure and maintenance.

1. Web Server Port and Admin Port

Web Server Port

- Port used by the public radar (`/AircraftList.json`, map frontend).

Admin Port

- Port used by the admin panel (`/admin`).
- Recommended: use a dedicated admin port, separate from radar.

If `AdminPort = 0` or equals `WebServerPort`, admin endpoints share the same port as public radar, which increases exposure.

2. Network Binding

This setting controls whether the server listens on:

1. localhost only,
2. all interfaces (`0.0.0.0`) for LAN/VPN/public routing.

Default for new installs: all interfaces. This is intentional for headless/server deployments so the activation page and admin panel can be reached from another machine on the network.

A restart is required after changing binding behavior.

3. Radar Visibility

This setting controls radar authentication:

1. `Public` - radar is accessible without login.
2. `Private` - radar requires authentication.

This is independent from `Network Binding`. Setting the radar to `Private` does not force localhost-only binding; use `Network Binding` for that.

4. Database and Imports

Database File Name

- SQLite database file path/name.
- Changing it may require restart.

Data import options

- OpenSky CSV
- ADS-B Exchange JSON/NDJSON
- Airport CSV (OurAirports-compatible)
- BaseStation SQLite upload (`BaseStation.sqb`, `.sqlite`, `.db`, `.sqlite3`)

BaseStation file import note:

1. Use the file selector in Server Settings (no manual path typing required).
2. `Replace` - swaps the whole DB file and creates a `.bak` backup.
3. `Merge` - keeps current DB file and upserts aircraft records by `ModeS`.

ADSBX note:

1. `basic-ac-db.json.gz` includes native `mil` flag support (military classification).
2. There is no dedicated `government` field in this dataset.

Recommended process:

1. Confirm source URL is valid and in expected format.
2. Run import.
3. Check server logs.
4. Verify results on map/list.

AircraftDB Editor and custom photos

Dedicated guide:

1. 15-AircraftDB-Editor-and-Custom-Photos.md

Quick summary:

1. `ModeS` must stay unique (duplicate create/update is blocked),
2. editor provides `Load existing aircraft` path for duplicate ICAO conflicts,
3. custom aircraft photo can be uploaded per ICAO and has priority over PlaneSpotters.

5. Configuration Backup

This section provides:

1. settings export,
2. settings import.

After importing configuration:

1. verify ports,
2. verify binding,
3. restart service if required.

6. Danger Zone

`Clear Application Data` removes critical data (configuration, users, mappings, database).

Use only when:

1. performing full instance reset,
2. current backups exist,
3. irreversible data loss is accepted.

05. Resources and Visuals

Resources and Visuals

1. Navigation

This page is a quick navigator for visual resources and marker customization.

Recommended reading:

1. 12-Resources-Flags-and-Silhouettes.md
2. 13-Custom-Operator-Logo.md
3. 14-Aircraft-Colors-and-Operator-Colors.md
4. 17-Aircraft-Model-and-Operator-Aliases.md

2. Type Icon Mapping (quick note)

Use `Type Icon Mapping` after resources are loaded.

Current UI behavior:

1. mappings are displayed sorted by `ICAO Type Code`,
2. filter matches both `ICAO Type Code` and `Icon Name`.

3. If resources are loaded but still not visible

Checklist:

1. verify CSV keys match ZIP asset naming,
2. clear browser cache,
3. confirm icon mapping is not overriding expected silhouette behavior,
4. verify aircraft records include the expected type/operator metadata,

5. use `Aircraft Models` when aircraft have an ICAO type code but no model name,
6. use `Operator Aliases` when callsign/operator codes should resolve to carrier display names.

4. Theme-related map visuals

1. Map controls (zoom, layers, location, visual settings) use one unified visual style.
2. In `Futuristic` and `OverFuture`, controls should not have an extra outer frame around grouped buttons.
3. If old framing still appears after deployment, clear browser cache and reload the page.

06. Users and Security

Users and Security

1. User management

In `Users`:

1. create operator accounts,
2. reset/change passwords,
3. remove unused accounts.

Avoid using one shared account for multiple operators.

2. Public vs private radar

`Radar Visibility` controls radar access mode:

1. `Public` - no login required for radar view.
2. `Private` - login required.

This does not replace network-level controls (ports, firewall, binding). A private radar can still listen on all interfaces so headless installs are reachable over LAN/VPN.

3. Baseline hardening

1. Use a dedicated `AdminPort`.
2. Restrict admin port access at firewall level (LAN/VPN admin clients only).
3. Use a strong admin password.
4. Rotate password after deployment.
5. Do not expose admin panel publicly unless required.

4. Safe change procedure

1. Apply configuration change.

2. Save.
3. Restart service when required.
4. Verify:
    - radar endpoint works,
    - admin endpoint works,
    - authentication works.

07. Backup, Update, Recovery

Backup, Update, Recovery

1. Minimum backup before changes

Before major changes, create:

1. configuration export (`Configuration Export`),
2. database export (`Database Export`),
3. backup of the instance data directory.

2. Update process (operational model)

Safe minimal flow:

1. stop service,
2. backup,
3. replace binaries,
4. start service,
5. verify endpoints.

Post-update checks:

1. `http://<host>:<web-port>/AircraftList.json`
2. `http://<host>:<admin-port>/admin`
3. hard refresh the radar page to load current static assets,
4. verify at least one desktop and one mobile UI smoke check.

Recommended UI smoke checks:

1. switch theme to `Futuristic` and `OverFuture` and verify map controls have no extra outer frame,
2. on mobile viewport, open aircraft detail sheet and verify expand/collapse gestures,
3. verify aircraft list sorting by `Reg` works and persists after page reload,
4. verify mobile bottom spacing is correct on devices with safe-area insets.

3. Recovery after failed deployment

1. Stop service.
2. Restore previous binaries.
3. Restore configuration and/or database from backup.
4. Start service.
5. Verify feeds and admin access.

4. What to log after deployment

Track:

1. release version,
2. date and host,
3. changed ports/binding settings,
4. endpoint verification results.

08. Troubleshooting

Troubleshooting

1. `/admin` returns `Invalid url`

Most common causes:

1. port conflict,
2. admin is running on a different port than expected,
3. reverse proxy path/port mismatch.

Check:

1. which process listens on admin port,
2. `AdminPort` and `WebServerPort` values in config,
3. `curl -I http://<host>:<admin-port>/admin` after restart.

2. Radar works locally but not on LAN

Check:

1. `Network Binding` is set to all interfaces,
2. host firewall rules,
3. service is listening on `0.0.0.0:<port>`.

Do not use `Radar Visibility` to fix LAN reachability. `Radar Visibility` controls login requirements; `Network Binding` controls which interfaces the server listens on.

Linux example:

```
ss -ltnp | rg ':8080|:5000|:8085'
curl -I http://<lan-ip>:<port>/admin
```

3. `AircraftList.json` becomes empty after some time

Possible causes:

1. feed connection loss,
2. background exception stopping host,
3. overload or unstable upstream data patterns.

Check:

1. service logs (`journalctl -u vrsx`),
2. feed status and message throughput,
3. process uptime after errors.

4. Aircraft positions jump backward or flicker

Common causes:

1. MLAT instability,
2. low-quality feed or high jitter,
3. mixed sources without stable source-priority logic.

Diagnostic flow:

1. compare same feed in VRS X vs reference VRS,
2. test single feed instead of `All feeds`,
3. compare JSON output over identical bbox/time intervals.

5. Resource import says success, but no visual effect

1. Verify `Loaded` counters.
2. Verify CSV keys match asset names.
3. Clear browser cache.
4. Verify icon/type mapping and metadata consistency.

6. Map controls show strange outer frames in `Futuristic` / `OverFuture`

Symptoms:

1. zoom/layer/location control groups have extra rectangular outlines around button stacks.

Recovery steps:

1. perform hard refresh (`Ctrl+Shift+R` or `Cmd+Shift+R`),
2. clear site cache and reload,
3. verify the active theme is really `Futuristic` or `OverFuture`,
4. if issue persists, redeploy frontend static assets and refresh again.

7. Mobile aircraft detail sheet does not collapse/expand as expected

Check:

1. sheet can expand by tapping handle or swiping up,
2. collapse gesture works only when detail content scroll is at top,
3. after closing selected aircraft and reopening, sheet should start collapsed.

8. Bottom sheet is clipped by device home indicator (mobile)

Check:

1. confirm frontend HTML uses viewport meta with `viewport-fit=cover`,
2. clear browser cache so updated CSS safe-area rules are applied,
3. retest on real device (some desktop emulators do not reproduce full safe-area behavior).

9. Marker colors from Aircraft Colors / Operator Colors are not visible

Check:

1. `Color by operator` is enabled in label settings,
2. mapping key format is valid (`ICAO hex` for Aircraft Colors, operator code for Operator Colors),
3. if both mappings exist, remember Aircraft Colors override Operator Colors,
4. emergency or selected marker style can temporarily override normal tint.

10. Custom operator logo upload is rejected or not used

Common causes:

1. invalid proportions (must be `17:4`),
2. image is too small for template (`85x20` minimum for downscale path),
3. existing logo conflict with `overwrite` disabled,
4. invalid operator/type key format.

Check:

1. upload with naming rule `OPERATOR` or `OPERATOR-ICAOTYPE`,
2. if prompted, use explicit downscale to `85x20`,
3. confirm `OperatorFlagsList.csv` and resource status are updated,
4. refresh browser cache and verify with specific aircraft type.

11. AircraftDB Editor shows duplicate ICAO conflict

Symptoms:

1. save returns conflict and message that aircraft already exists.

Expected behavior:

1. app prevents duplicate `ModeS` rows,
2. editor shows `Load existing aircraft` action for the conflicting ICAO.

Resolution:

1. load existing record,

2. apply edits there,
3. save update instead of creating a second record.

12. Custom aircraft photo upload succeeds but detail still shows fallback image

Check:

1. ICAO in editor is valid and matches the aircraft (`6` hex chars),
2. custom photo status in editor shows `custom photo available`,
3. hard refresh detail panel to bypass stale browser cache,
4. if needed, delete and re-upload photo (system keeps only one custom photo variant per ICAO).

09. Feeds: Smart Ports and MLAT

Feeds: Smart Ports and MLAT

1. Feed basics

In `Feeds`, define raw data sources (`Mode-S Raw`, `BaseStation`).

`Mode-S Raw` covers both:

1. Beast family binary (`30005`, `30006`, `30105`, `31005`, `31006`)
2. AVR / raw text (`30002`, `*8D...;`)

For each feed, configure:

1. `Address` and `Port`
2. `Format`
3. `ConnectionType`
4. `Enabled`
5. `Visible in Radar Dropdown`
6. optional `Receiver Location`

After save:

1. verify connection status,
2. verify message counters increase,
3. verify `AircraftList.json` is non-empty.

2. Smart profile ports

Known smart profiles:

1. `30002` -> `[ADS-B] AVR/Raw Text`
2. `30003` -> `[ADS-B] BaseStation/SBS Text`
3. `30005` -> `[ADS-B] Beast/Binary`
4. `30006` -> `[ADS-B] BeastReduce`

5. `30105` -> `[MLAT] Beast/Binary`
6. `30106` -> `[MLAT] BaseStation/SBS Text`
7. `31005` -> `[MLAT Hub] Beast/Binary`
8. `31006` -> `[MLAT Hub] BeastReduce`

When one of these ports is entered, feed editor pre-applies matching format/MLAT defaults.

3. Quick Add: Feed Set

Use `/admin/feeds` -> `Add Feed Set` when one feeder exposes several related ports on the same IP.

The dialog asks for:

1. `Base Name` - auto-filled as the first unused `FEEDxx` family, for example `FEED01`,
2. `Address / IP` - shared by all generated feeds,
3. `Primary Feed` - required ADS-B source, `30003` or `30005`,
4. `Dedicated MLAT Feed` - optional `30105` or `30106`,
5. `MLAT Hub Feed` - optional `31005` or `31006`,
6. `Receiver Location` - copied to every generated feed,
7. optional `Add created feeds to merged feed`.

The UI saves ordinary feeds underneath, for example:

1. `FEED01_ADSB`
2. `FEED01_MLAT`
3. `FEED01_MLATHUB`

`FEED01_MLATHUB` is still stored as an MLAT feed internally (`IsMlatFeed = true`).

This keeps backward compatibility with existing configuration while making multi-port feeders faster to add.

Generated feed behavior:

1. every created feed is a normal TCP active feed,
2. smart port profiles still set parser/MLAT defaults,
3. duplicate endpoints with the same normalized address and port are skipped,
4. if all requested endpoints already exist, nothing is created,
5. if no merged feed is selected, primary ADS-B feeds stay visible in the radar dropdown and MLAT-only feeds stay hidden by default,
6. if a merged feed is selected, all newly created feed IDs are appended to that merged feed and the visibility checkbox is applied to created feeds.

Use normal `Add Feed` instead when the source needs:

1. a custom port outside the smart profile list,
2. passive TCP mode,
3. serial receiver configuration,
4. unusual format settings.

Known `readsb` ports without a feed profile:

1. `30047` -> JSON position stream, not a raw feed input
2. `30152` -> HTTP API JSON, not a raw feed input

4. MLAT options

Feed-level MLAT controls:

1. `MLAT Feed` forces all positions from feed to be treated as MLAT-derived.
2. Use `MLAT Feed` only for dedicated MLAT streams without reliable MLAT marker tagging.
3. For mixed ADS-B + MLAT streams, keep `MLAT Feed` disabled and rely on per-message MLAT detection.
4. `Assume DF18 CF1 = ICAO` is experimental and should be enabled only for known-compatible sources.

5. BeastReduce and MLAT Hub

`BeastReduce` is still Beast on the wire.

1. Use the same `Mode-S Raw` parser as for normal Beast.
2. The difference is reduced rate / lower bandwidth, not a different binary format.

`MLAT Hub` is a consolidated MLAT-results output, not a primary ADS-B receiver feed.

1. `31005` and `31006` are usually advanced, MLAT-only sources.
2. They should not be presented as a normal full-traffic ADS-B feed.
3. Default `MLAT Feed = on` is appropriate for these dedicated MLAT outputs.

6. Quick verification

1. open aircraft detail panel,
2. check `Position Source` (`MLAT position` or `ADS-B position`),
3. compare behavior before and after MLAT option changes.

7. UI visibility note

`Visible in Radar Dropdown` controls whether a feed appears in frontend `All feeds / feed selector`.

10. Feed Filter

Feed Filter

1. What Feed Filter does

`Feed Filter` filters messages before they are propagated to the rest of the system.

2. Main options

1. `Enable Feed Filter` enables/disables filtering globally.
2. `Prohibit MLAT Positions` strips MLAT lat/lon/track from BaseStation and drops MLAT Mode-S frames.
3. `Block Unfilterable Feeds` blocks passthrough/raw feeds that cannot be filtered per message.

3. ICAO filter mode

1. `Blocklist` (`ProhibitIcaos=true`): ICAOs in list/ranges are blocked.
2. `Allowlist` (`ProhibitIcaos=false`): only ICAOs in list/ranges are allowed.
3. Individual ICAO entries must be exactly 6 hex characters.
4. ICAO ranges are inclusive (`start` and `end` included).

4. Apply behavior

1. no service restart required,
2. settings are hot-reloaded,
3. frontend/admin message indicates changes usually become visible within 2-5 minutes.

5. Safe rollout sequence

1. verify raw feed flow first (without filter),
2. enable filter with minimal scope,

3. validate map and list output after each step,
4. only then add additional ICAO rules/ranges.

11. Merged Feeds

Merged Feeds

1. Purpose

`Merged Feeds` build one logical output from multiple source feeds.

Recommended model:

1. keep physical sources in `Feeds`,
2. expose combined output via merged feeds where needed.

2. Merged feed options

Each merged feed has:

1. `Name`
2. `Source Feeds` (`receiverIds`)
3. `Enabled`
4. `Visible in Radar Dropdown`
5. `ICAO Timeout (ms)`
6. `Ignore Aircraft with No Position`

3. Runtime logic

For each ICAO:

1. one source feed is selected as active owner,
2. duplicate frames from other sources are suppressed,
3. if active source is silent longer than `ICAO Timeout`, ownership may switch to another source.

4. ID namespace

Merged feed IDs use dedicated range starting from `10000`, reducing collision risk with regular feed IDs.

5. Workflow notes

1. Feed editor contains shortcut `Add to merged feed` for smart-profile ports.
2. Final source assignment should be verified in `Merged Feeds` panel.
3. Merged feeds can be selected as source in `Rebroadcast Servers`.

12. Resources: Flags and Silhouettes

Resources: Flags and Silhouettes

1. Upload packages

Typical uploads:

1. `OperatorFlags.zip` (or `AirlineLogos.zip`) + optional `OperatorFlagsList.csv`
2. `DVSilhouettes.zip` (or `Silhouettes.zip`) + optional `Silhouettes.csv`

Supported image formats:

1. `.bmp`
2. `.png`

2. Runtime lookup logic

Operator flags:

1. frontend requests `/api/images/operator-flags/{operatorCode}.png?icaoType={type}`,
2. lookup priority is `OPERATOR-ICAOTYPE` first, then `OPERATOR`.

Silhouettes:

1. frontend requests `/api/images/silhouette/{icaoType}.png`,
2. lookup first uses `Silhouettes.csv`,
3. fallback tries direct `DVSilhouettes/{icaoType}.bmp/png` file.

If resource is not found, API returns transparent image instead of broken image icon.

3. CSV behavior on upload

1. separately uploaded CSV has priority over CSV found inside ZIP,
2. for `flags`, if CSV is missing system can regenerate `OperatorFlagsList.csv` from image files,
3. for `silhouettes`, provide `Silhouettes.csv` for deterministic mapping.

4. Verification after upload

1. check `Loaded` counters in admin panel,
2. verify operator flags in list/detail,
3. verify silhouettes in aircraft detail panel,
4. hard refresh browser if old cache is still used.

13. Custom Operator Logo

Custom Operator Logo

Custom operator logos are managed in `/admin/resources` under the custom operator logo section.

1. Naming and priority

Allowed naming patterns:

1. generic logo: `OPERATOR.bmp` or `OPERATOR.png`
2. type-specific logo: `OPERATOR-ICAOTYPE.bmp` or `OPERATOR-ICAOTYPE.png`

Runtime priority:

1. type-specific logo is preferred,
2. generic logo is fallback.

Examples:

1. `DLH.png` applies to all Lufthansa aircraft that resolve to `DLH`,
2. `DLH-A321.png` applies only when the resolved operator is `DLH` and the aircraft ICAO type is `A321`,
3. if both files exist, `DLH-A321.png` wins for `A321` aircraft and `DLH.png` remains the generic fallback.

2. Admin workflow

In `/admin/resources`:

1. enter the operator code,
2. optionally enter the ICAO type for a type-specific logo,
3. upload a `.bmp` or `.png` image,
4. confirm downscale if the image has correct proportions but is larger than `85x20`,
5. enable overwrite only when replacing an existing logo for the same key.

The same page lists uploaded custom logos and allows deleting them. Deleting a custom logo removes the file, synchronizes metadata, and reloads resources immediately.

3. Validation rules

1. file type must be `.bmp` or `.png`,
2. file size limit is `2 MB`,
3. required aspect ratio is `17:4`,
4. template size is `85x20`,
5. if image has valid ratio but larger dimensions, explicit downscale confirmation is required,
6. operator code accepts `A-Z`, `0-9`, `@`, `.`, `_`, `+`, `-` (`2-40` chars),
7. optional ICAO type accepts `A-Z`, `0-9` (`2-12` chars).

4. Overwrite behavior

1. if target logo already exists and `overwrite=false`, upload returns conflict,
2. when saving one extension, counterpart extension for same key is removed to keep deterministic lookup.

5. Storage and metadata

1. files are stored in `Images/AirlineLogos`,
2. upload metadata is tracked in `custom-operator-flags.json`,
3. after upload/delete, `OperatorFlagsList.csv` is synchronized,
4. resource service is reloaded immediately.

6. Troubleshooting

1. If the uploaded image is rejected, check aspect ratio first. Valid examples are `85x20`, `170x40`, and `255x60`.
2. If the logo uploads but is not visible, verify the operator code used by the aircraft. Operator aliases can help callsign-derived codes resolve to expected carrier names, but logo lookup still needs matching operator/type metadata.
3. If an older logo keeps appearing, clear browser cache and verify that the opposite extension was removed for the same key.

14. Aircraft Colors and Operator Colors

Aircraft Colors and Operator Colors

1. Color priority chain

Marker color priority:

1. `Aircraft Colors` by ICAO hex has highest priority,
2. if missing, `Operator Colors` is used,
3. if both missing, default marker color is used.

2. Operator color resolution

`Operator Colors` lookup order:

1. first 3 characters of callsign,
2. `operatorFlagCode` from aircraft database record.

3. Visual modifiers

1. `Color by operator` in label config must be enabled for custom tinting.
2. Emergency squawk rendering forces emergency red.
3. Selected marker style can force yellow selection (unless `preserve` mode is used).

4. Admin/API behavior

1. both mappings are upsert-style (`PUT` creates or updates existing key),
2. mappings are persisted in data directory JSON files,

3. stored mappings are sorted by key on save.

5. Quick diagnostic checklist

1. verify key format (`ICAO hex` for Aircraft Colors, operator code for Operator Colors),
2. verify `Color by operator` is enabled,
3. remember Aircraft Colors overrides Operator Colors,
4. verify emergency/selection state is not masking expected tint.

15. AircraftDB Editor and Custom Photos

AircraftDB Editor and Custom Photos

1. AircraftDB Editor workflow

1. search supports ICAO, registration, owner, manufacturer, and type,
2. search returns up to 50 results,
3. selecting row loads full record by `AircraftID`,
4. create/update/delete operations refresh database lookup cache.

2. Duplicate ModeS behavior

1. `ModeS` is unique (`ICAO hex`),
2. creating/updating with existing `ModeS` returns conflict (`DUPLICATE_MODES`),
3. UI shows conflict warning and offers `Load existing aircraft`,
4. recommended path is to load existing record and update it instead of creating duplicate.

3. Custom aircraft photo upload

Requirements:

1. valid ICAO hex (`[0-9A-F]{6}`),
2. allowed formats: `PNG`, `JPG/JPEG`, `WEBP`, `BMP`,
3. max size: `8 MB`,
4. image signature is validated server-side.

Storage behavior:

1. stored path: `Images/AircraftPhotos/{ICAO}.{ext}`,

2. uploading new custom photo removes previous custom variants for same ICAO.

4. Display priority logic

Aircraft detail photo source order:

1. custom local aircraft photo,
2. PlaneSpotters fallback.

Deleting custom photo restores PlaneSpotters fallback behavior.

16. Coverage

Coverage

`/admin/coverage` is an admin-only coverage explorer for enabled merged feeds.

It answers a practical question: where did this merged feed actually observe aircraft? It does not draw theoretical antenna range from receiver coordinates. The footprint is built from received aircraft positions.

1. What it shows

Coverage is calculated per enabled merged feed and includes:

1. live coverage for the current local day,
2. archived daily snapshots from the local coverage database,
3. a merged footprint polygon,
4. source feed activity for every source inside the merged feed,
5. basic intensity statistics such as active cells and observed cell-hours.

The page is useful for:

1. spotting holes in receiver coverage,
2. finding inactive or weak source feeds,
3. comparing coverage before and after feed changes,
4. validating whether a merged feed is receiving data from expected regions.

2. Live mode

Live mode reads the in-memory coverage collector and refreshes automatically.

Important details:

1. live coverage is for the current local date,
2. the UI refreshes the live snapshot every 15 seconds,
3. only aircraft with latitude and longitude contribute to coverage,
4. source activity is grouped by the direct feed that produced the position,
5. merged feed membership comes from the current configuration.

3. Archive mode

Archive mode reads persisted daily snapshots from `coverage.sqlite`.

Important details:

1. dates are listed per merged feed,
2. the archive uses the server's local timezone,
3. current state is flushed before archive reads so recent activity is not lost,
4. old data is retained for 365 days,
5. snapshots are still tied to the merged feed/source identities from configuration.

4. Protected merged feeds

Coverage follows merged feed access rules.

If a merged feed is password protected:

1. the coverage entry appears locked,
2. the admin must unlock it with the feed password,
3. if the merged feed depends on multiple protected source requirements, more than one password may be required,
4. archive date counts and snapshots are hidden until access is unlocked.

5. Metrics

Common fields:

1. `Sources` - number of source feeds in the merged feed,
2. `Active Sources` - sources that contributed at least one coverage cell for the selected day,
3. `Active Cells` - unique coverage cells observed by the merged feed,
4. `Peak Active Hours / Cell` - longest observed time in one cell,
5. `Total Cell Hours` - sum of observed cell time across all active cells,
6. `Footprint Polygons` - rendered coverage areas after simplification and gap fill.

Source rows show:

1. feed name,
2. receiver location if configured,
3. active cell count,
4. active hours,

5. last seen time.

6. Footprint logic

The raw collector stores positions in map cells and 15-minute time bins. The visible footprint is simplified to a coarser display grid and small linear gaps are closed before polygons are built.

This makes the map stable and readable while still preserving real coverage shape. Small gaps can be filled; large gaps remain visible.

7. Limitations

Coverage means observed traffic, not guaranteed radio range.

Empty or weak coverage can mean:

1. the receiver was down,
2. there was little traffic in that area,
3. positions arrived without latitude/longitude,
4. a source was not part of the selected merged feed,
5. the merged feed or source configuration changed during the day.

Receiver coordinates are displayed as source metadata when available, but they do not create coverage by themselves.

8. Troubleshooting

1. If no merged feeds are listed, create and enable a merged feed first.
2. If a merged feed is locked, unlock it with the feed password.
3. If there are no archive dates, wait for data to be collected and persisted.
4. If live coverage is empty, verify source feeds are connected and aircraft positions include latitude/longitude.
5. If a source has no receiver location, assign one in `/admin/locations`; this improves source context but does not change the observed footprint.

17. Aircraft Model and Operator Aliases

Aircraft Model and Operator Aliases

These tools live in the admin `Visuals` group:

1. `/admin/aircraft-model-mappings` - menu label `Aircraft Models`,
2. `/admin/operator-aliases` - menu label `Operator Aliases`.

They are still present in the admin panel. If they are not visible, expand `Visuals` in the left menu or open the direct routes above.

1. Aircraft Models

`Aircraft Models` maps an ICAO aircraft type code to a readable model name.

Use it when aircraft have a type code such as `A321`, `B738`, or `B77W`, but the aircraft database record does not contain an exact model.

Runtime behavior:

1. an exact model from the aircraft database wins,
2. if the exact model is missing, VRS X looks up the ICAO type code in `Aircraft Models`,
3. if there is no mapping, the model remains empty.

Example:

1. type code: `A321`,
2. model name: `Airbus A321`,
3. aircraft without an exact database model can display `Airbus A321`.

Validation:

1. ICAO type code must be `2-8` characters,
2. allowed type code characters are `A-Z` and `0-9`,

3. model name is required and is stored trimmed,
4. very long model names are shortened to 160 characters.

CSV import:

1. file extension must be `.csv`,
2. maximum upload size is `5 MB`,
3. format is semicolon-separated: `icao_type;model_name`,
4. optional headers such as `icao_type;model_name` are accepted,
5. duplicate ICAO type rows inside the same import are skipped,
6. existing mappings are updated when the imported model name changes.

Mappings are stored in `aircraft-model-mappings.json` in the VRS X data directory.

2. Operator Aliases

`Operator Aliases` maps an operator code to a carrier display name.

Use it when the aircraft list has a callsign/operator code such as `DLH`, `BAW`, or `RYR`, and you want the UI/API to expose a readable name such as `Lufthansa`, `British Airways`, or `Ryanair`.

Runtime behavior:

1. if the aircraft has `OperatorFlagCode`, that code is used first,
2. otherwise VRS X derives a code from the callsign prefix,
3. the derived callsign prefix must be two or three letters followed by a digit, for example `DLH123` -> `DLH`,
4. VRS X looks up that code in `Operator Aliases`,
5. the matched display name is exposed as `operatorName`.

Validation:

1. operator code must be `2-40` characters,
2. allowed characters are `A-Z`, `0-9`, `@`, `.`, `_`, `+`, and `-`,
3. display name is required and is stored trimmed.

CSV import:

1. file extension must be `.csv`,
2. maximum upload size is `5 MB`,
3. format is semicolon-separated: `code;display_name`,
4. optional header `code;display_name` is accepted,
5. existing aliases are updated when the imported display name changes.

Aliases are stored in `operator-aliases.json` in the VRS X data directory.

3. Relationship to colors and logos

These mappings do not replace resource files.

Use:

1. `Operator Aliases` for readable carrier names,
2. `Operator Colors` for marker tinting by operator code,
3. `Custom Operator Logo` for custom `OPERATOR.bmp/png` or `OPERATOR-ICAOTYPE.bmp/png` resources,
4. `Aircraft Models` for filling missing display model names by ICAO type.

If a logo or color does not apply, first verify which operator code the aircraft resolves to. The direct routes above are the fastest way to check and maintain mappings.