Viewport-Aware AircraftList.json

Viewport-Aware AircraftList.json

The radar only asks for what the screen can use

VRS X does not treat `AircraftList.json` as a blind "send every aircraft every time" endpoint. The radar frontend sends the current map viewport to the backend, and the backend shapes the response around that exact view.

That means a world view, a Europe view, and a close zoom over Frankfurt do not have to cost the browser the same amount of work.

What the frontend sends

The map reports its current bounds and zoom, then `useAircraftList` includes them in every `AircraftList.json` request:

1. `fNBnd` - north latitude bound,
2. `fSBnd` - south latitude bound,
3. `fEBnd` - east longitude bound,
4. `fWBnd` - west longitude bound,
5. `zoom` - current map zoom,
6. `vw` and `vh` - viewport width and height,
7. `selected` - selected aircraft id, when one is pinned,
8. `maxItems` - render budget derived from zoom,
9. `renderMode=cluster` - tells the backend to return clusters when needed.

The current radar poll interval is one second, but each poll is viewport-aware instead of globally naive.

Zoom-aware render budget

The frontend asks for smaller render budgets when the map is zoomed out:

1. zoom `<= 2` -> about `70` drawn objects,
2. zoom `<= 4` -> about `100` drawn objects,
3. zoom `<= 5` -> about `140` drawn objects,
4. zoom `<= 6` -> about `260` drawn objects,
5. closer zooms -> about `1200` drawn objects.

The backend also clamps requested budgets and applies its own low-zoom caps before deciding whether it can return individual aircraft directly or should return clusters.

What the counters mean

A status line such as:

`12 Shown / 4557 Visible / 4948 Total · 70 Drawn`

is the feature in action.

1. `Total` is the full aircraft snapshot currently known by the server.
2. `Visible` is the aircraft count relevant to the current map viewport after bbox filtering.
3. `Shown` is the number of individual aircraft markers returned in `acList`.
4. `Drawn` is the actual number of map objects rendered: individual aircraft plus cluster objects.

So the radar can honestly report almost five thousand tracked aircraft and more than four thousand aircraft in the current viewport, while drawing only about seventy objects on the map. That is the whole point: preserve situational awareness without asking the browser to draw thousands of overlapping markers.

The left aircraft list stays complete

The map and the aircraft list use different data paths.

The map uses `AircraftList.json` with bbox, zoom and render budget. The left aircraft list uses `/api/aircraft/sidebar`, which is not limited by the map bounding box. It supports search, sorting and paged loading, so the list remains a full aircraft list for the selected feed or filter even when the map is heavily clustered.

This means operators can keep a wide, efficient map view while still browsing or searching the complete aircraft list on the left.

What the backend does

`AircraftController.GetAircraftList` receives the viewport bounds and filters the aircraft snapshot before mapping it to JSON.

The backend:

1. excludes aircraft outside the requested viewport,
2. ignores aircraft without latitude/longitude when a viewport is active,
3. keeps selected aircraft visible even if it would otherwise fall outside the current viewport result,
4. handles anti-meridian bounds correctly, so crossing `180/-180` does not break the map,
5. returns individual aircraft directly when the visible count fits the render budget,
6. returns clusters when the visible count is too large,
7. keeps selected aircraft and emergency squawks as priority objects.

Why this is cool

The response contains both global and viewport-level truth:

1. `totalAc` tells you how many aircraft exist in the full snapshot,
2. `visibleAc` tells you how many aircraft are relevant to the current viewport,
3. `acList` contains the aircraft that should be drawn individually,
4. `clusters` represent dense groups without forcing the browser to render every marker.

This is why the UI can say things like "Shown / Visible / Total" while still drawing far fewer map objects than the raw aircraft count.

The practical win

This design keeps VRS X responsive with large feeds because filtering and clustering happen before the browser receives the render workload. The client gets enough information to stay honest about traffic volume, but not so much that the map becomes a CPU benchmark.

It is one of the quiet features that makes the radar feel fast: the server understands the map, the map asks better questions, and the aircraft list remains useful even when the map is aggressively optimized.

Revision #2
Created 2026-06-24 20:50:16 UTC by Codex Codex
Updated 2026-06-24 20:52:21 UTC by Codex Codex