

Dashboard Enhancement – Add State-wise Station and Charger Count Widget

10/23/2026 02:57 AM - Shreya Agarwal

Status:	New	Start date:	10/23/2026
Priority:	Normal	Due date:	
Assignee:	Shreya Agarwal	% Done:	0%
Category:		Estimated time:	0.00 hour
Target version:		Spent time:	13.00 hours
Description			
Dashboard Enhancement – Add State-wise Station and Charger Count Widget			

History

#1 - 10/23/2026 09:25 AM - Shreya Agarwal

State-Wise Network Overview Widget

1. Feature Overview & Requirements

Today, I designed, developed, and integrated the State-wise Network Overview dashboard widget for both Dashboard v1 (/dashboard) and Dashboard v2 (/dashboard-v2) on the feature branch . The feature provides operators with a visual, state-by-state geographical and tabular breakdown of active charging stations and chargers across India, backed by interactive drill-down drawers, advanced search filters, and seamless integration with existing station and machine detail sliders.

2. Service Architecture & API Integration Layer

I established strong TypeScript interfaces in src/types/stateWiseNetwork.ts and created a centralized API service in src/services/stateWiseNetworkSummaryService.ts. The service handles endpoint routing between Dashboard versions:

Dashboard v1: Seamlessly routes to the rollup endpoints (state-wise-network-summary-rollup, stations-rollup, and chargers-rollup).

Dashboard v2: Routes to the standard plain endpoints (state-wise-network-summary, stations, and chargers).

The service dynamically manages all query parameters including stateCode (with support for edge-cases like UNKNOWN), page, pageSize, and user search terms without any hardcoded data.

3. Main Collapsible Widget & India Map Visualization

I implemented StateWiseNetworkWidget.tsx placed between the Revenue graph and Top EV Stories widgets. The widget is designed with lazy-loading behavior so network calls are only triggered when the accordion is expanded, optimizing initial dashboard performance. It displays three top-level summary cards (Total States, Total Stations, Total Chargers) and pairs an authentic geographic SVG map of India (IndiaMapChart.tsx backed by indiaSvgMapData.json) with a clean state summary table. The map features dynamic choropleth color shading (High/Medium/Low activity), interactive hover tooltips with live counts, and clickable state selection.

4. Drill-Down Drawers, UI Refinements & Filter Controls

I developed two dedicated slide-out drawers: StateStationsDrawer.tsx (for state station drill-downs) and StateChargersDrawer.tsx (for state charger drill-downs). Following user feedback and project UI patterns:

Apply & Reset Filter Controls: Integrated project-standard ApplyFilterButton and ResetFiltersButton aligned to the right.

Table Layout & Consistency: Standardized column naming to "Name", removed redundant "Status" columns, removed duplicate connector count cells, and eliminated secondary ID subtexts.

also hide State-wise Network Overview widget for Partner role users

#2 - 10/26/2026 08:24 AM - Shreya Agarwal

Today, integrated role-based permission control for the State-wise Network Overview widget across both Dashboard v1 (/dashboard) and Dashboard v2 (/dashboard-v2) Based on the backend team's official specification, the widget and its associated station/charger drill-down drawers are governed by the permission key dashboard.state_wise_network.view (Display Name: "View State-wise Network").

I implemented the UI section gate cleanly across both dashboard implementations without removing or altering any existing code:

Dashboard v1

Added STATE_WISE_NETWORK: 'dashboard.state_wise_network.view' to the DASHBOARD_PERM registry.

Added DASHBOARD_PERM.STATE_WISE_NETWORK to the TENANT_DASHBOARD_WIDGET_KEYS array.

Defined the gate check: canDashStateWiseNetwork = !isPartnerForDashboardGate && tenantDashboardAllowed(DASHBOARD_PERM.STATE_WISE_NETWORK).

Wrapped the widget rendering with {canDashStateWiseNetwork && !isPartner && (<StateWiseNetworkWidget isV2={false} ... />)}.

Dashboard v2

Registered STATE_WISE_NETWORK in DASHBOARD_PERM and TENANT_DASHBOARD_WIDGET_KEYS.

Defined the gate check: canDashStateWiseNetwork = hasPermission('ALL') || relaxTenantDashboardKeys || hasPermission(DASHBOARD_PERM.STATE_WISE_NETWORK).

Wrapped the widget rendering with {canDashStateWiseNetwork && !isPartner && (<StateWiseNetworkWidget isV2={true} ... />)}.

Component-Level Defense-in-Depth & Partner Restriction

In src/components/dashboard/StateWiseNetworkWidget.tsx, integrated internal access validation using useAuth(). The component checks whether the user holds ALL, dashboard.state_wise_network.view, or default tenant admin rights, returning null immediately if unauthorized. Concurrently, the strict Partner user restriction (!isPartner and !isPartnerForDashboardGate) was fully maintained across all levels. This ensures that Partner logins remain completely barred from seeing the widget or initiating background API requests, while authorized Tenant Admins and permitted roles retain full access.