@serve.zone/catalog

a catalog of PaaS centric components for serve.zone

- readme.md for @serve.zone/catalog
- changelog.md for @serve.zone/catalog

readme.md for @serve.zone/catalog

The complete UI component library for **serve.zone** — a full-featured admin and management interface for onebox server management, built as a collection of reusable web components.

Install

```
pnpm install @serve.zone/catalog
```

Issue Reporting and Security

For reporting bugs, issues, or security vulnerabilities, please visit community.foss.global/. This is the central community hub for all issue reporting. Developers who sign and comply with our contribution agreement and go through identification can also get a code.foss.global/ account to submit Pull Requests directly.

□□ What It Does

`@serve.zone/catalog` provides **30+ production-ready web components** covering every aspect of server management:

- □□ **Dashboard** — Real-time cluster overview, resource usage, traffic metrics, quick actions
- □□ **Services** — Docker container management, deployment, logs, live stats, backups, and an integrated IDE workspace
- □□ **Network** — Reverse proxy configuration, DNS record management, domain & SSL certificate monitoring
- □□ **Registries** — Container registry management (onebox + external registries like Docker Hub, GHCR, ECR)
- □□ **Auth** — Login view, API token management (global + CI tokens)

- ⚙ **Settings** — Appearance, Cloudflare integration, SSL/TLS config, network settings, account management
- □□ **Platform Services** — MongoDB, MinIO, ClickHouse, Redis, Caddy monitoring and control

Every component supports **light and dark themes** out of the box and communicates via standard `CustomEvent` dispatching.

□□ Usage

Import

```
// Import everything
import * as szCatalog from '@serve.zone/catalog';

// Or import specific components
import { SzDashboardView, SzLoginView, SzServiceDetailView } from '@serve.zone/catalog';
```

Components auto-register as custom elements when imported. Use them directly in your HTML:

```
<sz-dashboard-view .data="${dashboardData}"></sz-dashboard-view>
<sz-login-view @login="${handleLogin}"></sz-login-view>
<sz-service-detail-view .service="${serviceData}" .logs="${logEntries}"></sz-service-detail-view>
```

Full Application Shell

For a complete app experience, use the demo app shell which wires up sidebar navigation, app bar menus, and all views via `dees-appui`:

```
import '@serve.zone/catalog';
// Then use <sz-demo-app-shell> for a fully configured application
```

□□ Component Reference

Dashboard

Component	Tag	Description
`SzDashboardView`	`<sz-dashboard-view>`	Main dashboard orchestrating all grid sections — cluster, services, network, infrastructure
`SzStatCard`	`<sz-stat-card>`	Single statistic card with label, value, icon, and color variant
`SzResourceUsageCard`	`<sz-resource-usage-card>`	CPU/memory progress bars, network I/O, top memory consumers
`SzTrafficCard`	`<sz-traffic-card>`	HTTP traffic metrics — requests, errors, response time, status distribution
`SzQuickActionsCard`	`<sz-quick-actions-card>`	Configurable action button grid

Dashboard Grids

Component	Tag	Description
`SzStatusGridCluster`	`<sz-status-grid-cluster>`	4-column stat card grid — total/running/stopped services, Docker status
`SzStatusGridServices`	`<sz-status-grid-services>`	Resource usage + platform services side by side
`SzStatusGridNetwork`	`<sz-status-grid-network>`	Traffic, reverse proxy, and certificates in a responsive grid
`SzStatusGridInfra`	`<sz-status-grid-infra>`	DNS/SSL status + quick actions

Services

Component	Tag	Description
`SzServicesListView`	`<sz-services-list-view>`	Table of deployed services with status badges and action buttons
`SzServiceDetailView`	`<sz-service-detail-view>`	Full service detail page — info, logs, live stats, actions, backups, and integrated workspace/IDE mode
`SzServiceCreateView`	`<sz-service-create-view>`	Service deployment form — image, ports, env vars, volumes, resource limits

Component	Tag	Description
`SzServicesBackupsView`	`<sz-services-backups-view>`	Backup schedule and backup history management

Platform Services

Component	Tag	Description
`SzPlatformServicesCard`	`<sz-platform-services-card>`	Lists infrastructure services (MongoDB, MinIO, etc.) with status indicators
`SzPlatformServiceDetailView`	`<sz-platform-service-detail-view>`	Detailed platform service view — connection info, config, metrics, logs

Network

Component	Tag	Description
`SzNetworkProxyView`	`<sz-network-proxy-view>`	Reverse proxy management — traffic targets table and access log viewer
`SzNetworkDnsView`	`<sz-network-dns-view>`	DNS record management with Cloudflare sync
`SzNetworkDomainsView`	`<sz-network-domains-view>`	Domain list with certificate status and provider info
`SzDomainDetailView`	`<sz-domain-detail-view>`	Domain detail — SSL certificate info, proxy routes, DNS records
`SzReverseProxyCard`	`<sz-reverse-proxy-card>`	Compact proxy status card (HTTP/HTTPS ports, route count)
`SzDnsSslCard`	`<sz-dns-ssl-card>`	Cloudflare DNS and ACME config status
`SzCertificatesCard`	`<sz-certificates-card>`	Certificate status counts — valid, expiring, expired

Registries

Component	Tag	Description
`SzRegistryAdvertisement`	`<sz-registry-advertisement>`	Onebox registry info card with docker quick-start commands
`SzRegistryExternalView`	`<sz-registry-external-view>`	External registry management (Docker Hub, GHCR, GCR, ECR)

Auth & Settings

Component	Tag	Description
`SzLoginView`	`<sz-login-view>`	Login page with serve.zone branding, credentials form, error display
`SzTokensView`	`<sz-tokens-view>`	API token management — global and CI tokens with copy/regenerate/delete
`SzSettingsView`	`<sz-settings-view>`	Full settings panel — appearance, Cloudflare, SSL/TLS, network, account

□□ Architecture

Component Pattern

All components follow a consistent pattern:

```
import { DeesElement, customElement, html, css, cssManager, property } from
'@design.estate/dees-element';

@customElement('sz-my-component')
export class SzMyComponent extends DeesElement {
  // TC39 standard decorators with accessor keyword
  @property({ type: String })
  public accessor label: string = '';

  public static styles = [
    cssManager.defaultStyles,
    css`
      /* Light/dark theme support */
      :host { color: ${cssManager.bdTheme('#18181b', '#fafafa')}; }
    `,
  ];

  // Events via CustomEvent (bubbles + composed for shadow DOM)
  private handleClick() {
    this.dispatchEvent(new CustomEvent('action', {
```

```
      detail: { id: this.id },
      bubbles: true,
      composed: true,
    }));
  }
}
```

Project Structure

```
@serve.zone/catalog/
├── html/                    # WccTools dev server entry point
│   ├── index.html           # HTML shell
│   └── index.ts             # WccTools config (pages + elements sections)
├── ts_web/
│   ├── index.ts             # Barrel export
│   ├── elements/            # All web components
│   │   ├── index.ts         # Element barrel export
│   │   ├── sz-*.ts          # Individual components
│   │   └── sz-demo-view-*.ts  # Demo orchestration wrappers
│   └── pages/               # Page-level components
│       ├── sz-demo-app-shell.ts  # Full app shell (dees-appui)
│       └── ...
└── dist_ts_web/             # Compiled output (npm entry point)
```

TypeScript Interfaces

The library exports comprehensive TypeScript interfaces for all data structures:

```
// Dashboard
import type { IDashboardData, IResourceUsage, ITrafficData, IClusterStats } from
'@serve.zone/catalog';

// Services
import type { IServiceDetail, IServiceStats, ILogEntry, IServiceBackup } from
'@serve.zone/catalog';

// Network
```

```
import type { IDomainDetail, ICertificateDetail, IDnsRecord, ITrafficTarget } from
'@serve.zone/catalog';

// Settings & Auth
import type { ISettings, IToken, IExternalRegistry } from '@serve.zone/catalog';
```

□□ Development

```
# Install dependencies
pnpm install

# Start dev server (wcctools dashboard with live reload)
pnpm run watch

# Production build
pnpm run build

# Run tests
pnpm test
```

The **wcctools dev server** provides an interactive dashboard where every component is rendered with demo data. Components are organized by group (Dashboard, Services, Network, etc.) in the sidebar. Demo view wrappers (`sz-demo-view-*`) are filtered out from the element list — they serve as full-page orchestration demos accessible through the Pages section.

License and Legal Information

Trademarks

This project is owned and maintained by Task Venture Capital GmbH. The names and logos associated with Task Venture Capital GmbH and any related products or services are trademarks of Task Venture Capital GmbH or third parties, and are not included within the scope of the MIT license granted herein.

Use of these trademarks must comply with Task Venture Capital GmbH's Trademark Guidelines or the guidelines of the respective third-party owners, and any usage must be approved in writing. Third-party trademarks used herein are the property of their respective owners and used only in a descriptive manner, e.g. for an implementation of an API or similar.

Company Information

Task Venture Capital GmbH Registered at District Court Bremen HRB 35230 HB, Germany

For any legal inquiries or further information, please contact us via email at hello@task.vc.

By using this repository, you acknowledge that you have read this section, agree to comply with its terms, and understand that the licensing of the code does not imply endorsement by Task Venture Capital GmbH of any derivative works.

changelog.md for @serve.zone/catalog

2026-03-19 - 2.9.0 - feat(app-store-view)

add app details action to store cards

- adds a new "View Details" button alongside the existing deploy action
- dispatches a bubbling "view-app" event with the selected app payload for parent handlers
- updates card action layout and button styling to support multiple actions

2026-03-18 - 2.8.0 - feat(elements)

add app store view component for browsing and deploying app templates

- introduces a new sz-app-store-view element with app card rendering, search/filter empty states, and deploy action events
- exports the new app store view from the elements index for public consumption

2026-03-17 - 2.7.0 - feat(sz-service-detail-view)

replace the custom logs panel with dees-chart-log in the service detail view

- Removes the bespoke log streaming UI styles and markup in favor of a shared log chart component.
- Maps service log entries to structured timestamp, level, and message data for the new component.

- Enables auto-scrolling, metrics display, and a higher log entry limit in the embedded log viewer.

2026-03-16 - 2.6.2 - fix(platform-service-detail-view)

wrap service logs chart in a full-width container to preserve layout

- Places the logs component inside a container spanning all grid columns.
- Keeps the service logs view aligned correctly within the detail page layout.

2026-03-16 - 2.6.1 - fix(platform-service-detail-view)

replace custom service log markup with dees-chart-log in the platform service detail view

- Removes bespoke log container styles and rendering logic in favor of the shared dees-chart-log component
- Normalizes log timestamps to ISO format before passing entries to the chart component
- Enables log auto-scroll, entry limits, and metrics display for service logs

2026-03-16 - 2.6.0 - feat(service-create-view)

add platform service toggles for MongoDB, S3, and ClickHouse provisioning

- Adds a new Platform Services section to the service creation view with dedicated toggles for managed infrastructure dependencies.
- Includes MongoDB, S3-compatible storage, and ClickHouse selections in the emitted create-service configuration payload.
- Resets selected platform services after form submission to keep create flow state consistent.

2026-02-23 - 2.5.0 - feat(sz-config-section)

add header action buttons to sz-config-section allowing configurable actions/events

- Introduce IConfigSectionAction interface (label, icon, event, detail).
- Add actions property to SzConfigSection and render header action buttons in the component template.
- Add styles for .header-action and hover state to match design system.
- Dispatch CustomEvent when an action is clicked, using action.event (defaults to 'action') and action.detail.
- Update demo (sz-demo-view-config) to include a sample 'View Routes' action showing usage.

2026-02-23 - 2.4.0 - feat(elements)

add configuration overview and section components with demo view and index exports

- Adds new sz-config-section component (IConfigField interface, rich renderers for boolean, pills, badge, code, link and 'Not configured' handling).
- Adds new sz-config-overview wrapper component with heading/info banner and slot styling.
- Adds demo view sz-demo-view-config that supplies example configuration groups and fields for System, Proxy, Email, DNS, TLS, Cache, RADIUS and Remote Ingress.
- Exports new components from ts_web/elements/index.ts so they are available to the element registry.

2026-02-22 - 2.3.0 - feat(routes)

add route UI components and demo view with list/card and app-shell integration

- Add new route UI components: sz-route-card, sz-route-list-view, and sz-demo-view-routes under ts_web/elements
- Export new components from ts_web/elements/index.ts and register demo view in the demo app shell menu (ts_web/pages/sz-demo-app-shell.ts)
- sz-route-card introduces route types/interfaces (IRouteConfig, IRouteAction, IRouteMatch, IRouteTls, IRouteSecurity) and rich rendering (ports/domains formatting, feature icons,

security and headers display)
- sz-route-list-view provides demo data, search, filtering (by action and enabled state), results count, grid rendering of sz-route-card, and emits a 'route-click' event
- Demo view integrates with app UI secondary menu (actions and statistics) and wires up route-click handling for interactivity

2026-02-21 - 2.2.0 - feat(demo-mta)

add MTA / Email demo views and components and integrate into demo app shell

- Add sz-mta-list-view component with search, direction/status filters, and a results table
- Add sz-demo-view-mta demo page with sample emails and detailed SMTP log data; handles list -> detail navigation
- Export new MTA elements from ts_web/elements/index.ts and register sz-demo-view-mta
- Integrate MTA view into sz-demo-app-shell navigation and include it in the Infrastructure section

2026-02-20 - 2.1.0 - feat(catalog)

add comprehensive README documenting package purpose, components, usage, development workflow, and legal information

- Adds new readme.md at repository root for @serve.zone/catalog describing 30+ UI components, tags, and example usage.
- Includes installation instructions, import examples, demo app shell info, development scripts (pnpm install/watch/build/test) and project structure overview.
- Documents TypeScript interfaces exported, component architecture patterns, and license/trademark/company contact details.

2026-02-20 - 2.0.1 - fix(catalog)

no changes detected

- No files changed in the diff
- package.json version remains 2.0.0

2026-02-20 - 2.0.0 - BREAKING CHANGE(elements)

rename Onebox registry component to Registry Advertisement (sz-registry-onebox-view → sz-registry-advertisement) and update exports, demos and docs

- Replaced ts_web/elements/sz-registry-onebox-view.ts with ts_web/elements/sz-registry-advertisement.ts
- Updated export in ts_web/elements/index.ts to export the new component
- Updated demo (ts_web/elements/sz-demo-view-registries.ts) to use the new tag and updated headings/labels
- Updated readme.hints.md to reference sz-registry-advertisement instead of sz-registry-onebox-view
- Breaking change: custom element tag name changed; consumers must update any usages/imports

2026-02-20 - 1.1.0 - feat(elements)

add demoGroups metadata, filter demo view wrappers in wcctools, and update demo payloads

- Add public static demoGroups to many web components to enable sidebar categorization in the wcctools dashboard (groups like Dashboard, Dashboard Grids, Network, Services, Platform, Auth & Settings).
- Filter out demo-view wrapper components from the wcctools elements list (html/index.ts) so full-page demo orchestration wrappers are not shown as regular elements.
- Adjust demo/example payload shapes for sz-status-grid-network and sz-status-grid-services (renamed and flattened traffic/status and resource usage properties for the demo fixtures).
- Add readme.hints.md with project structure, web component patterns, demo groups and build/watch instructions.

2026-02-20 - 1.0.1 - fix(deps)

bump dependencies and devDependencies, update watch script, add npmextra preset, and remove Playwright artifacts

- package.json: bump @design.estate/dees-catalog ^3.29.1 → ^3.43.0, @design.estate/dees-domtools ^2.3.6 → ^2.3.8, @design.estate/dees-element ^2.1.3 → ^2.1.6, @design.estate/dees-wcctools ^3.4.0 → ^3.8.0
- devDependencies: bump @git.zone/tsbuild ^4.0.2 → ^4.1.2, @git.zone/tsbundle ^2.6.3 → ^2.8.3, @git.zone/tstest ^3.1.3 → ^3.1.8, @git.zone/tswatch ^2.3.13 → ^3.1.0, @types/node ^25.0.3 → ^25.3.0
- scripts: change watch from "tswatch element" to "tswatch" (simplify dev workflow)
- npmextra.json: add preset for @git.zone/tswatch (preset: "element")
- repo cleanup: expand .gitignore and remove large Playwright screenshots/artifacts from .playwright-mcp

2026-01-03 - 1.0.0 - maintenance

General maintenance update; the commit message provides no detailed information.

- Commit message: "update" (no further details).
- No specific user-facing or functional changes are documented.