readme.md for @idp.global/idp.global

A modern, open-source Identity Provider (IdP) SaaS platform for managing user authentication, registrations, sessions, and organization-based access control.

Built with TypeScript and designed for modern web applications, idp.global provides a complete identity management solution that you can self-host or use as a service.

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.

Features

Authentication & Authorization

- **Multiple Login Methods**: Email/password, email magic links, API tokens
- **JWT-Based Sessions**: Secure token management with automatic refresh
- **Two-Factor Authentication**: Enhanced security with 2FA support
- **Password Reset**: Secure password recovery flow
- **Device Management**: Track and manage authenticated devices

Organization Management

- **Multi-Tenant Architecture**: Support multiple organizations per user
- **Role-Based Access Control (RBAC)**: Fine-grained permissions system
- **Organization Roles**: Admin, member, and custom role support
- **Member Invitations**: Bulk invite and manage team members

- **Ownership Transfer**: Seamlessly transfer organization ownership

□□ Third-Party Integration

- **OpenID Connect (OIDC) Provider**: Full OIDC compliance for third-party apps
  - Discovery endpoint (`/.well-known/openid-configuration`)
  - JWKS endpoint for token verification
  - Authorization code flow with PKCE
  - Token refresh and revocation
- **OAuth 2.0**: Standard OAuth flows for app authorization
- **Supported Scopes**: `openid`, `profile`, `email`, `organizations`, `roles`

□□ Billing Integration

- **Paddle Integration**: Built-in payment processing support
- **Billing Plans**: Flexible subscription management
- **Checkout Flows**: Streamlined payment experiences

□□ Modern Web UI

- **Responsive Design**: Beautiful UI components built with `@design.estate/dees-catalog`
- **Account Management**: User profile, settings, and preferences
- **Organization Dashboard**: Manage members, roles, and apps
- **Admin Panel**: Global administration interface

□□ Real-Time Communication

- **WebSocket Support**: Real-time updates via TypedSocket
- **Typed API Requests**: Type-safe client-server communication
- **Public Key Distribution**: Automatic JWT key rotation notifications

□□ Architecture

idp.global is built as a modular TypeScript monorepo:

```
├── ts/                    # Server-side code (Node.js)
│   └── reception/         # Core identity management logic
├── ts_interfaces/         # Shared TypeScript interfaces (published as
```

```
@idp.global/interfaces)
├── ts_idpclient/          # Browser/Node client library (published as @idp.global/idpclient)
├── ts_idpcli/             # Command-line interface tool
└── ts_web/                # Web frontend (published as @idp.global/web)
```

Core Managers

Manager	Responsibility
`JwtManager`	JWT generation, validation, and key management
`LoginSessionManager`	Session creation and authentication
`UserManager`	User CRUD and profile management
`OrganizationManager`	Organization lifecycle management
`RoleManager`	RBAC and permission management
`OidcManager`	OpenID Connect provider functionality
`AppManager`	OAuth client app registration
`BillingPlanManager`	Subscription and payment handling

□□ Quick Start

□□ Docker Deployment (Recommended)

The easiest way to run idp.global is using Docker:

```
# Pull the latest image
docker pull code.foss.global/idp.global/idp.global

# Run with environment variables
docker run -d \
  -p 2999:2999 \
  -e MONGODB_URL=mongodb://your-mongo:27017/idp \
  -e IDP_BASEURL=https://your-domain.com \
  -e INSTANCE_NAME=idp.global \
  code.foss.global/idp.global/idp.global
```

Environment Variables

Variable	Description	Required
`MONGODB_URL`	MongoDB connection string	□ Yes
`IDP_BASEURL`	Public URL of your idp.global instance	□ Yes
`INSTANCE_NAME`	Name for this IDP instance	No (default: `idp.global`)
`SERVEZONE_PLATFROM_AUTHORIZATION`	ServeZone platform auth token	No

Docker Compose Example

```
version: '3.8'
services:
  idp:
    image: code.foss.global/idp.global/idp.global
    ports:
      - "2999:2999"
    environment:
      MONGODB_URL: mongodb://mongo:27017/idp
      IDP_BASEURL: https://idp.yourdomain.com
      INSTANCE_NAME: my-idp
    depends_on:
      - mongo

  mongo:
    image: mongo:7
    volumes:
      - mongo-data:/data/db

volumes:
  mongo-data:
```

The server listens on port 2999 by default.

□□ Local Development

Prerequisites

- Node.js 20+
- pnpm
- MongoDB (local or remote)
- SMTP server (for email verification in registration flow)

Getting Started

```
# Clone the repository
git clone https://code.foss.global/idp.global/idp.global.git
cd idp.global

# Install dependencies
pnpm install

# Build the project
pnpm build

# Start development server with hot reload
pnpm watch
```

The server runs on **http://localhost:2999** with:

- □□ Auto-restart backend on changes (`ts/`)
- □□ Automatic frontend bundle rebuilding (`ts_web/`)

Environment Setup

Create environment variables for the backend:

```
export MONGODB_URL=mongodb://localhost:27017/idp-dev
export IDP_BASEURL=http://localhost:2999
export INSTANCE_NAME=idp-dev
```

Development Routes

Route	Description

`/`	Welcome/landing page
`/login`	Sign in form
`/register`	New user registration
`/account`	User dashboard (requires auth)

□□ Default Development Credentials

For local development with the test database, use:

Field	Value
Email/Username	`admin@idp.global` or `admin`
Password	`admin`

This account has `isGlobalAdmin: true` for full platform access including the admin panel at `/account/admin`.

> ⚠ **Security Note**: These credentials are for local development only. Never use default credentials in production environments.

□□ Published Packages

This monorepo publishes the following npm packages:

Package	Description
`@idp.global/interfaces`	TypeScript interfaces for API contracts
`@idp.global/idpclient`	Client library for browser and Node.js
`@idp.global/web`	Web UI components

□□ Client Usage

Browser Client

```
import { IdpClient } from '@idp.global/idpclient';

// Initialize the client
const idpClient = new IdpClient('https://idp.global');

// Enable WebSocket connection
await idpClient.enableTypedSocket();

// Check login status
const isLoggedIn = await idpClient.determineLoginStatus();

// Login with email and password
const response = await idpClient.requests.loginWithUserNameAndPassword.fire({
  username: 'user@example.com',
  password: 'securepassword'
});

if (response.refreshToken) {
  await idpClient.refreshJwt(response.refreshToken);
  console.log('□ Login successful!');
}

// Get current user info
const userInfo = await idpClient.whoIs();
console.log('User:', userInfo.user);

// Get user's organizations
const orgs = await idpClient.getRolesAndOrganizations();
console.log('Organizations:', orgs.organizations);
```

Organization Management

```
// Create a new organization
const result = await idpClient.createOrganization('My Company', 'my-company', 'manifest');
console.log('Created:', result.resultingOrganization);

// Invite members
await idpClient.requests.createInvitation.fire({
```

```
  jwt: await idpClient.getJwt(),
  organizationId: 'org-id',
  email: 'newmember@example.com',
  roles: ['member']
});
```

CLI Tool

The `ts_idpcli` module provides a command-line interface:

```
# Login
idp login

# Show current user
idp whoami

# List organizations
idp orgs

# List organization members
idp members --org <org-id>

# Invite a user
idp invite --org <org-id> --email user@example.com
```

□□ OIDC Integration

idp.global implements a full OpenID Connect provider. Third-party applications can use it for SSO:

Discovery Document

```
GET /.well-known/openid-configuration
```

Authorization Flow

```
GET /oauth/authorize?
  client_id=your-client-id&
  redirect_uri=https://yourapp.com/callback&
  response_type=code&
  scope=openid profile email organizations&
  state=random-state&
  code_challenge=PKCE_CHALLENGE&
  code_challenge_method=S256
```

Token Exchange

```
POST /oauth/token
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&
code=AUTHORIZATION_CODE&
redirect_uri=https://yourapp.com/callback&
client_id=your-client-id&
client_secret=your-client-secret&
code_verifier=PKCE_VERIFIER
```

UserInfo

```
GET /oauth/userinfo
Authorization: Bearer ACCESS_TOKEN
```

Response:

```
{
  "sub": "user-id",
  "name": "John Doe",
  "email": "john@example.com",
  "email_verified": true,
  "organizations": [
    { "id": "org-1", "name": "Acme Corp", "slug": "acme", "roles": ["admin"] }
  ],
  "roles": ["user"]
```

```
}
```

□□ Tech Stack

- **Runtime**: Node.js with ES Modules
- **Language**: TypeScript (strict mode)
- **Database**: MongoDB via `@push.rocks/smartdata`
- **Web Server**: `@api.global/typedserver`
- **Real-time**: `@api.global/typedsocket` (WebSocket)
- **JWT**: `@push.rocks/smartjwt` (RS256 signing)
- **Frontend**: `@design.estate/dees-element` (Web Components)
- **Build**: `@git.zone/tsbuild` + `@git.zone/tsbundle`

□□ API Reference

Request Interfaces

All API requests are type-safe. See `ts_interfaces/request/` for the complete API:

- **Authentication**: `IReq_LoginWithEmail`, `IReq_LoginWithApiToken`, `IReq_RefreshJwt`
- **Registration**: `IReq_FirstRegistration`, `IReq_FinishRegistration`
- **User Management**: `IReq_GetUserData`, `IReq_SetUserData`, `IReq_GetUserSessions`
- **Organizations**: `IReq_CreateOrganization`, `IReq_GetOrgMembers`, `IReq_CreateInvitation`
- **Apps & OAuth**: `IReq_GetGlobalApps`, `IReq_CreateGlobalApp`
- **Billing**: `IReq_GetBillingPlan`, `IReq_UpdatePaymentMethod`

Data Models

See `ts_interfaces/data/` for all data structures:

- `IUser` - User profile and credentials
- `IOrganization` - Organization entity
- `IRole` - User roles within organizations
- `IJwt` - JWT token structure
- `IApp` - OAuth application definitions
- `IOidcAccessToken`, `IAuthorizationCode` - OIDC tokens

License and Legal Information

This repository contains open-source code licensed under the MIT License. A copy of the license can be found in the LICENSE file.

Please note: The MIT License does not grant permission to use the trade names, trademarks, service marks, or product names of the project, except as required for reasonable and customary use in describing the origin of the work and reproducing the content of the NOTICE file.

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.

Revision #4
Created 2026-03-28 11:09:35 UTC by foss.global Team
Updated 2026-03-28 12:15:49 UTC by foss.global Team