MIR Reference Implementation

24 September 2022

Oliver Steensen-Bech Haagh
Digital Maritime Consultancy
oliver@dmc.international

What is the MIR Reference Implementation?

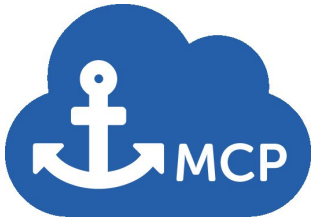
- An implementation of the specifications defined by the MCC IDSec working group
- Serves as a point-of-reference for other implementations of the MIR while still being able to be used in production
- All source code is open source and licensed under the Apache 2.0 license

Components of the MIR Reference Implementation

- MIR PKI Library
  - Provides the cryptographic functionality that is necessary for creating and handling MCP certificates.
- MIR API
  - REST API for managing maritime identities and certificates
- MIR Identity Broker
  - Provides token based authentication based on OpenID Connect (OIDC)

MIR PKI Library

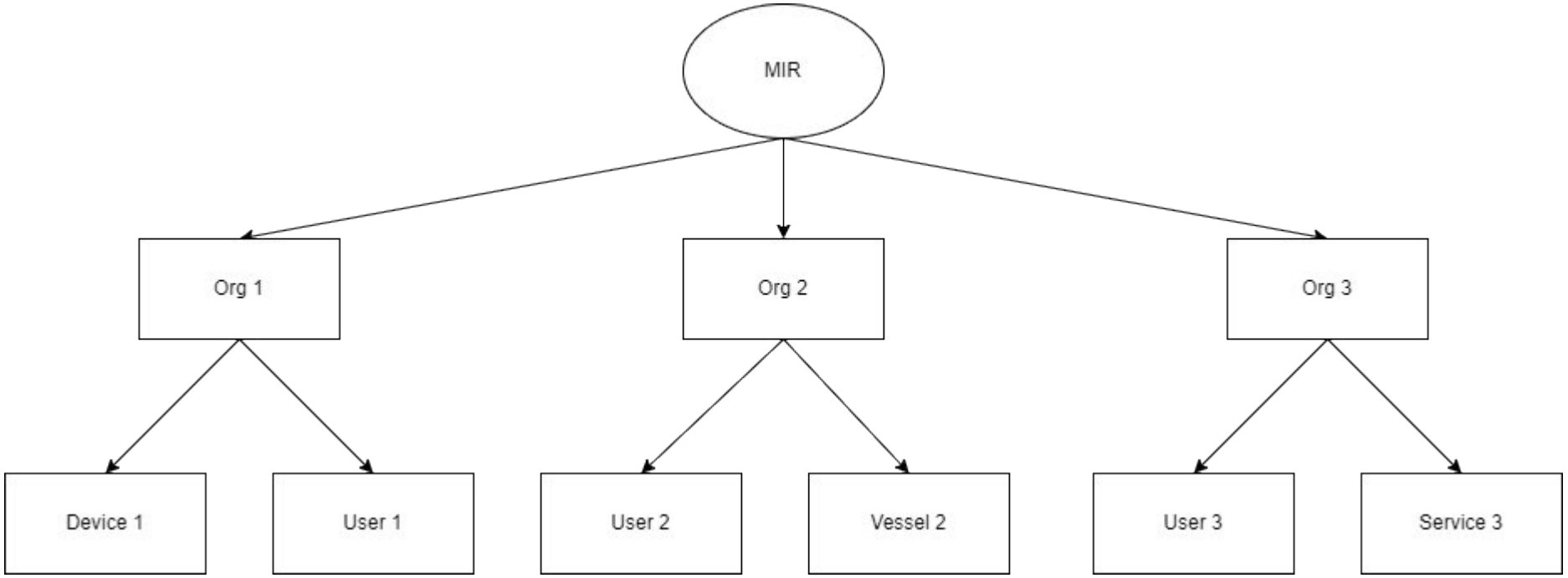
- Library used for handling, creating and signing MCP certificates, certificate revocation lists, OCSP, etc.
- Based on well-known standards such as ITU X.509v3 / RFC 5280
- Written in Java 11
- Uses the Bouncy Castle crypto library for cryptographic operations
- Has optional integration with hardware security modules (HSM) using PKCS#11
  - Based on the SunPKCS11 provider in Java
  - Currently only been tested with a small number of HSMs, including SoftHSMv2 and YubiHSM 2
- Can be used as a standalone CLI tool for generating root and intermediate CA certificates or verifying MCP certificates



MIR API

- REST API for managing maritime identities and certificates
  - Hierarchical identity structure with organizations on top and entities belonging to the organizations beneath them
- Written in Java 17 and is based on the Spring Boot framework
- Uses the MIR PKI Library for cryptographic operations, e.g. issuing and revoking certificates, CRL, OCSP, etc.
- Uses PostgreSQL by default for persistence, but can also use MySQL or MariaDB as long as you provide the JDBC driver yourself
- Entity types:
  - Organization
  - Device
  - MMS
  - Service
  - User
  - Vessel

MIR API – hierarchical structure of entities



Basic data format of an entity

- The name of the entity
  - E.g. John Anon
- A unique MCP MRN of the entity
  - E.g. urn:mrn:mcp:user:mcc:core:john
- An optional secondary MRN
  - An example could be that an AtoN could have an MRN in the IALA MRN namespace
- A list of permissions

Additions to entity data format

- Vessels can have an MMSI number, IMO number, AIS type, etc.
- Organizations and users have an email address
- An image or logo can be added to organizations and vessels
- Services have a version
- A service entity can be linked to a vessel type
  - Useful for when a vessel provides a service that needs an identity

MIR API

- Authorization to the MIR API is done using either OIDC tokens issued by the MIR ID Broker or using MCP certificates that have been previously issued by the MIR for mutual TLS authentication
- Access control for management of identities and certificates is done on the level of individual organizations using role based access control (RBAC)
  - Managing user identities requires the role “ROLE_USER_ADMIN”
  - Managing vessel identities requires the role “ROLE_VESSEL_ADMIN”
  - ...
- Roles can be assigned to individual users by creating permission-to-role mappings:
  - An example could be a mapping that maps from the permission “HR” to the role “ROLE_USER_ADMIN”. Any user that is given this permission will then be able to create and delete users in their own organization

Role based access control (RBAC) in the MIR API

Role	Approve New Org	Edit Own Org	Maintain Org Users	Maintain Org Vessels	Maintain Org Services	Maintain Org Devices	Maintain Org MMSes	Maintain Org Roles	Delete Org
ROLE_SITE_ADMIN	X	X	X	X	X	X	X	X	X
ROLE_ORG_ADMIN		X	X	X	X	X	X	X	
ROLE_ENTITY_ADMIN			X	X	X	X	X		
ROLE_USER_ADMIN			X						
ROLE_VESSEL_ADMIN				X					
ROLE_SERVICE_ADMIN					X				
ROLE_DEVICE_ADMIN						X			
ROLE_MMS_ADMIN							X		
ROLE_APPROVE_ORG	X								
ROLE_USER									

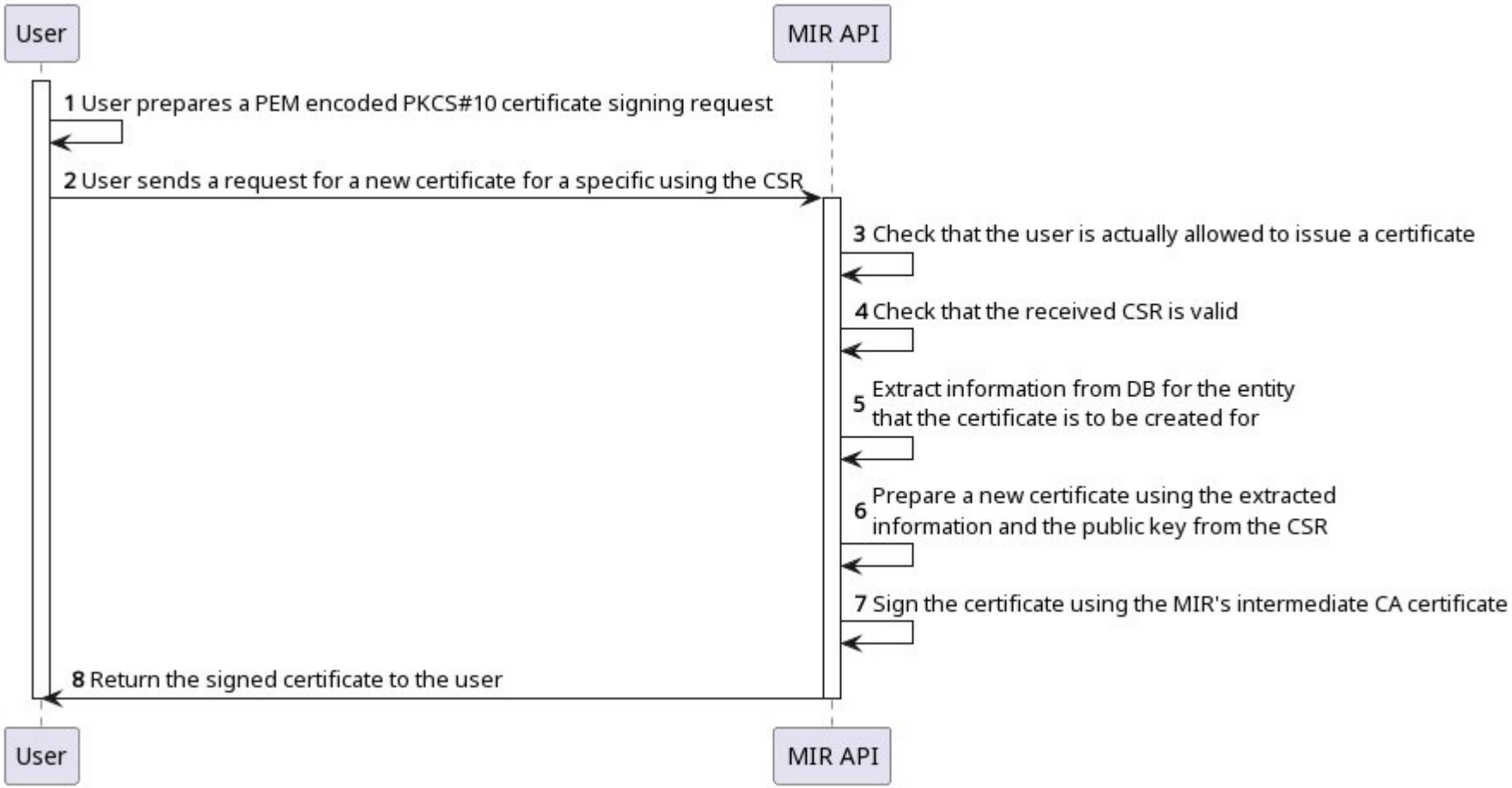
MIR Identity Broker

- Based on Keycloak which is an OpenID Connect (OIDC) server developed by RedHat
- Includes two custom MCP specific plugins:
  - One for synchronization of user data with MIR API
  - One for exchanging MCP client certificates with OIDC tokens
- Uses the MIR PKI Library to be able to parse MCP client certificates
- Can be configured to use external identity providers (ID brokering)

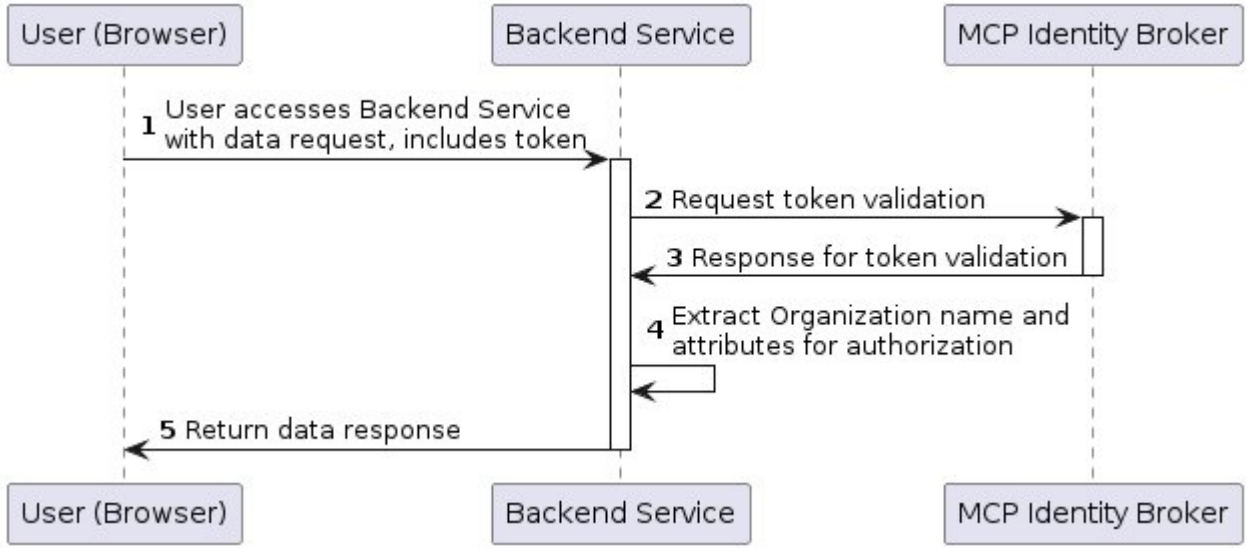
Connection between MIR ID Broker and API

- API uses ID Broker for OIDC token-based user authentication
- User information is synchronized between API and Broker
  - Whenever a new user is created using the MIR API it is also created in the ID Broker
  - Whenever a user logs in to the ID broker using an external ID provider, the user’s information is also registered in the API database

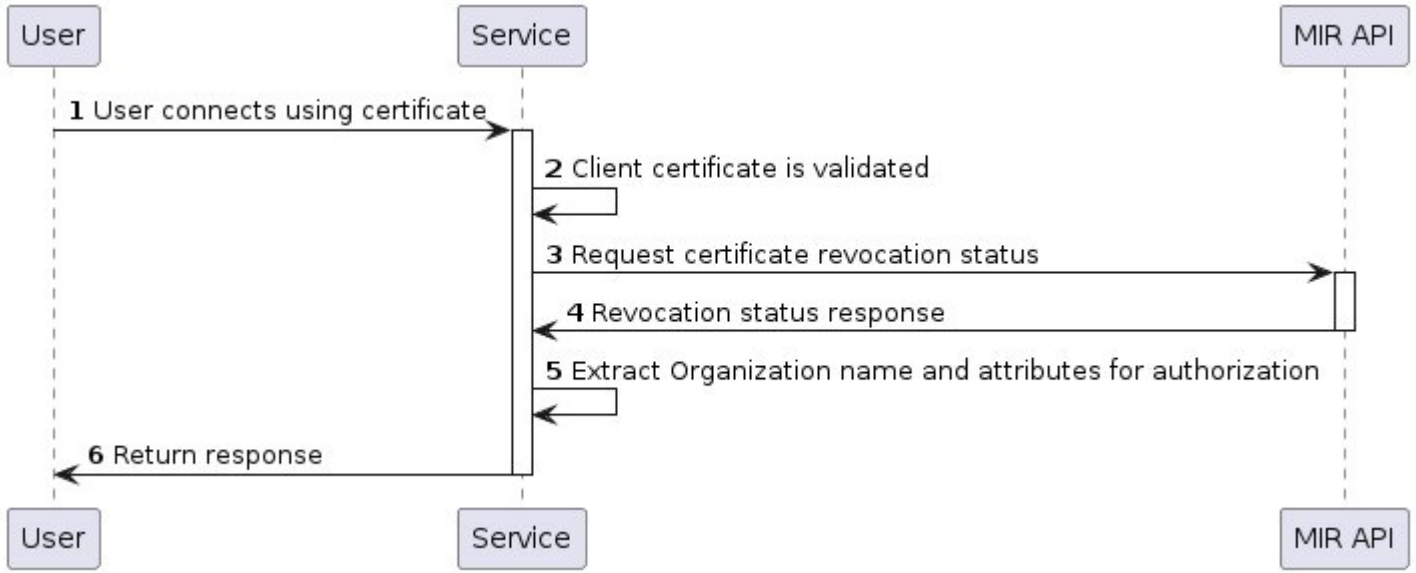
Example: Issuing a new certificate



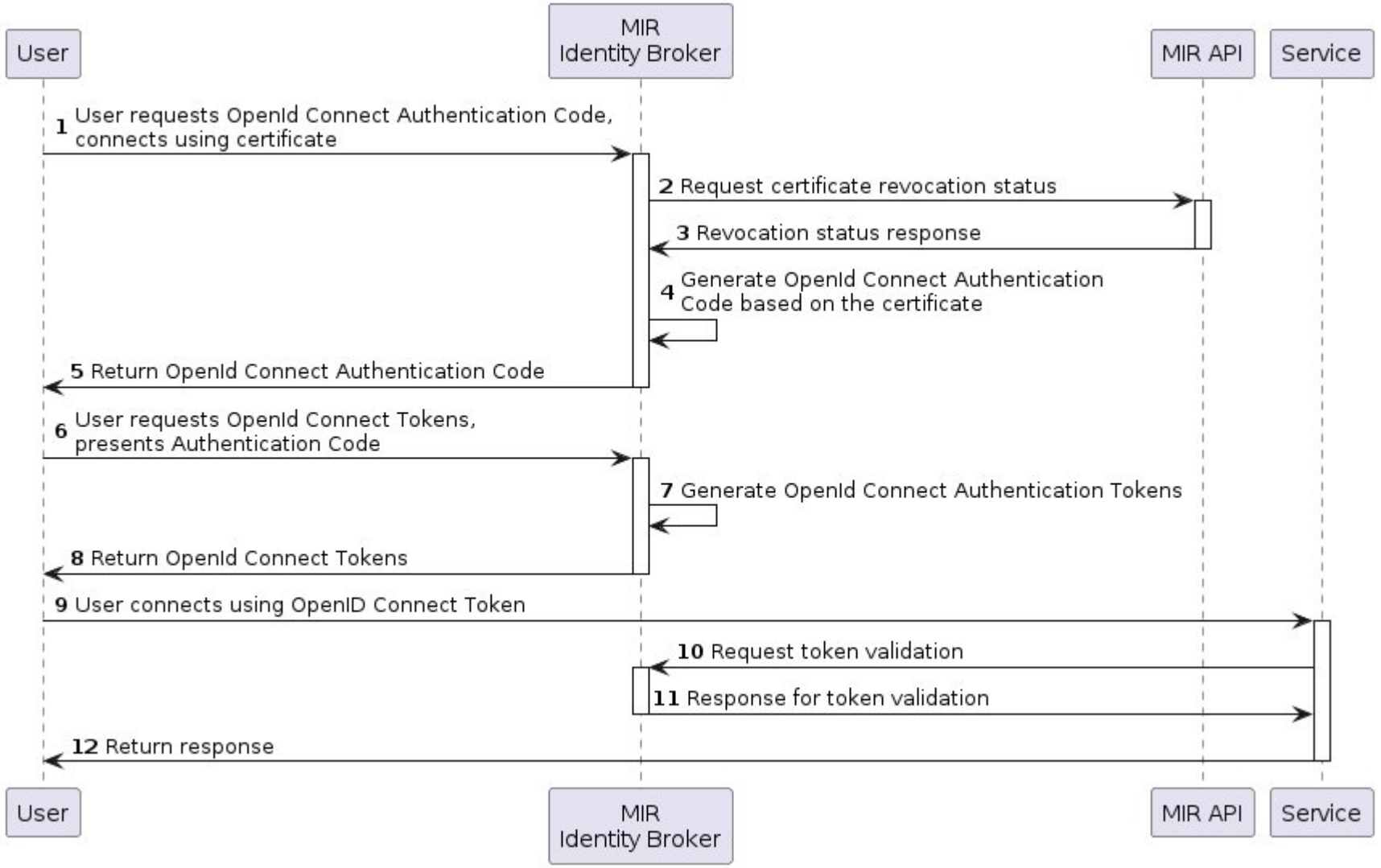
Example: Use OIDC token from ID Broker for authentication to a service



Example: Authenticate to service with MCP Certificate



Example: Get OIDC token using MCP Certificate



Access to the MCC Testbed

1. Go to https://management.maritimeconnectivity.net/login
2. Click “REGISTER”
3. Read the Terms of Use (Please)
4. Fill out the form and click “SUBMIT”

Source Code

- MIR PKI Library:
  - https://github.com/maritimeconnectivity/MCP-PKI
- MIR API:
  - https://github.com/maritimeconnectivity/IdentityRegistry
- MIR ID Broker
  - https://github.com/maritimeconnectivity/MCPKeycloakSpi

Thank you for listening!