



High Level Overview

Jellyfish – Solution Design Overview

1 September 2026

Version 5.01

Owner:	Operations
Contact details:	Website: https://cogitogroup.net/contact/ Email: Security.services@cogitogroup.net
Document status:	RELEASED
© Cogito Group 2026 All rights reserved. No part of this publication may be used, copied, reproduced, modified, distributed, displayed, stored in a retrieval system or transmitted in any form by any means (electronic, mechanical, photocopying, recording or otherwise) without the prior written authorisation of Cogito Group Pty Limited. Reproduction and use of all or portions of this publication is not permitted. No rights or permissions are granted with respect to this work.	

Contents

Table of Figures.....	3
1 Overview.....	4
2 Platform Architecture.....	5
2.1 Microservices-Based Architecture.....	5
2.2 True Multi-Tenancy, Built In, Not Bolted On.....	7
2.3 Deployment Options.....	7
3 From CLM to a Full Credential Platform.....	8
4 Core Capabilities.....	9
4.1 Public Key Infrastructure (CA & RA).....	9
4.2 Key Management as a Service.....	10
4.3 Smartcard & Credential Management, CMS and CAMS.....	11
4.4 SSH, OTP, SSO, HSM Management and Passwords, First Full Credential Management Service.....	12
4.5 CMDB, IPAM, Identity, Monitoring, Alerting & Audit.....	12
4.6 Identity Brokerage & Shared Access Services.....	13
5 Search.....	14
6 Certificate Automation Suite.....	15
6.1 Auto-Enrol Xchange (AEX).....	15
6.2 Simple Certificate Enrolment Protocol (SCEP).....	16
6.3 REST API.....	17
6.4 Enrolment over Certificate Management Protocol (CMP).....	18
6.5 ACME and ACME+.....	18
7 Security, Reliability & Scale.....	20
8 How Jellyfish Compares.....	21
9 Deployment Options & Technical Specifications.....	22
10 Summary & Next Steps.....	23
Appendix: Reference Sources.....	24

Table of Figures

Figure 1: Jellyfish interfaces and platform architecture overview.....	5
Figure 2: Jellyfish interfacing component architecture, in more detail.....	6
Figure 3 - Jellyfish Transition and Automation.....	10
Figure 4: The Jellyfish credential lifecycle.....	11
Figure 5: Auto-Enrol Xchange (AEX), certificate automation flow.....	16
Figure 6: SCEP enrolment via Microsoft Intune.....	17
Figure 7: Jellyfish REST API, integration overview.....	17
Figure 8: CMP, Certificate Management Protocol workflow.....	18
Figure 9: ACME and ACME+, automated certificate issuance flow.....	19

1 Overview

This document provides a solution design overview of Jellyfish, Cogito Group's credential, PKI and identity platform. Jellyfish began as a Certificate Lifecycle Management (CLM) product and has since grown into a Common Criteria certified and NIAP-listed platform covering PKI (CA and RA), key management, smartcard and credential management, SSH, OTP, HSM management and password management. Jellyfish is unique in its ability to unify the full range of credential types under a single, consistent operational model.

The solution is deployed predominantly in an aaS model. It is also sold and can be deployed as software, where a customer has significant internal PKI capabilities and requires this model of deployment.

Three fundamental design decisions define Jellyfish and set it apart from competing platforms:

- A microservices-based architecture, giving the platform the ability to scale individual components independently and to remain highly resilient under load or partial failure.
- True multi-tenancy, built into the platform from day one, rather than retrofitted, which delivers efficiency benefits in cloud deployments and, uniquely among competing products, allows Jellyfish to be run on-premises as an internal “as-a-Service” (aaS) offering to multiple business units, agencies or subsidiaries from a single deployment.
- The web portal communicates to the backend via REST API calls. This means that rather than API calls being an add on to some components, anything you can do within the UI (and some things that have no UI equivalent) can be done via an API call too.

2 Platform Architecture

2.1 Microservices-Based Architecture

Jellyfish is built as a set of discrete, independently deployable microservices, sitting behind a shared web, API and automation front end and connected through to internal and external interfaces, Certificate Authorities, Validation Authorities, data stores and notification channels. The diagram below shows how the internal layer (CAs, HSM-protected Root CA, data store), the external interface layer (VA services, web/API/automation services) and external networks (client workstations, devices and email) all connect through the Jellyfish core.

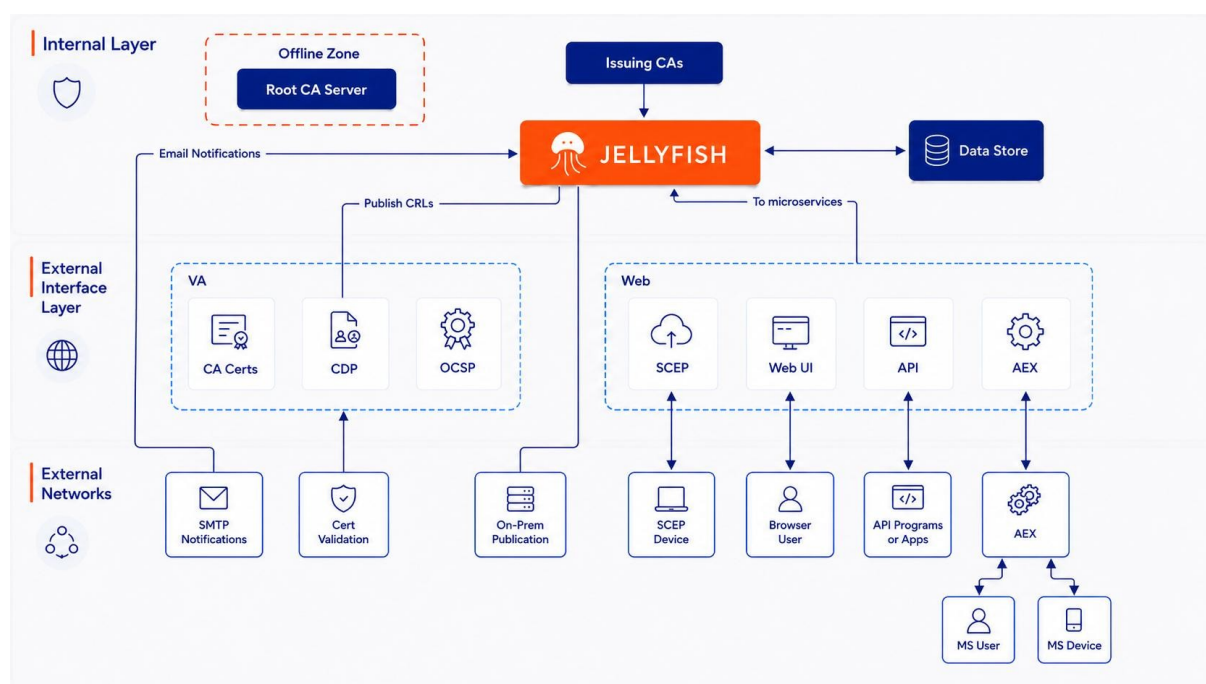


Figure 1: Jellyfish interfaces and platform architecture overview.

For a closer, component-level view, the diagram below breaks the same architecture into some its main microservices and trust zones, Identity & Credentials, Platform Services & Consumer Surface, and the PKI Service Plane, and shows how each service interfaces with the next.

Jellyfish Interfacing Component Architecture

JELLYFISH DESIGN

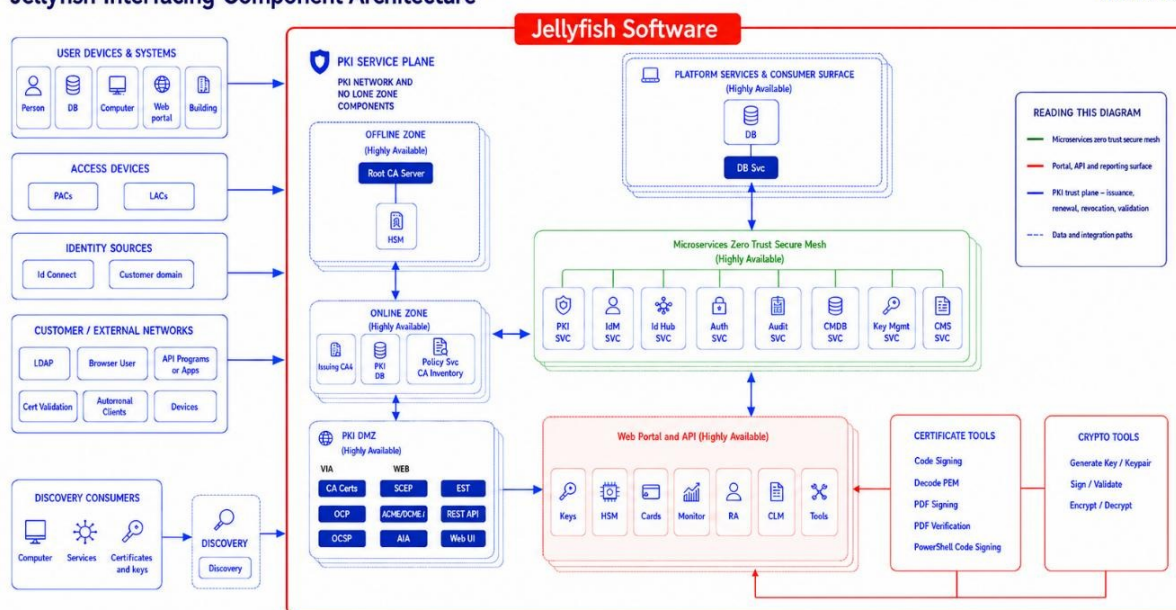


Figure 2: Jellyfish interfacing component architecture, in more detail

Core to the platform is a zero-trust secured microservices mesh. This is comprised of Identity Management (IdM), Identity Hub, Authentication, Audit, CMDB, Key Management, PKI components, Token Management of both Card Management System (CMS) and Hardware Security Module services. Each are independently deployable and scalable as described above. These services are accessed through a shared portal, API and reporting surface that exposes Keys, HSM, Cards, Monitoring, Registration Authority (RA), Certificate Lifecycle Management (CLM) and administrative Tools. A Discovery service continuously scans people, databases, computers, web portals and building systems for unmanaged credentials, working alongside dedicated Certificate Tools (code signing, PDF signing and verification, PowerShell code signing) and Crypto Tools (key/keypair generation, sign/validate, encrypt/decrypt).

Identity & Credentials ties physical and logical identity together: Physical Access Control Systems (PACs) and Logical Access Control Systems (LACs) both draw on the same identity sources, including Cogito's Id Connect and the customer's own domain, so an event in one domain, such as a badge swipe, can be correlated with logical access activity. This can also be extended to Visitor Management Systems (VMS), locker systems, WIFI access points, printers and much more. The benefit of this is that for instance, a user can be prevented from logging onto a computer, where the computer is in a network zone where the user should have entered through a point controlled by a PAC system. In this way, an attacker is unable to access the computer remotely, even if they have a correct (stolen) credential.

The PKI Service Plane anchors trust in an online or offline-zone, soft or hardware backed Root CAs, which signs the Issuing CAs. Issuing CA activity is tracked in a dedicated PKI database via a polling service that maintains CA inventory, and CRLs are published out to a PKI DMZ. The DMZ separates Validation Authority services (CA certificates, CDP, OCSP) from the web-facing enrolment protocols covered in Section 6, SCEP, EST, CMPv2/v3, ACME/ACME+, the REST API, the Web UI and AEX, which in turn serve customer and external networks: browser users, API programs and apps, managed devices, LDAP-based VA consumers and Auto-Enrol clients.

This design has the following direct benefits for clients:

- Scalability, individual services can be scaled horizontally to meet demand (for example, scaling the CA service during a mass certificate renewal event) without needing to scale the entire platform.
- Resilience, because services are decoupled, the failure or maintenance of one component does not bring down the whole platform. This delivers highly available and resilient microservices, health checks, and data protection with fast recovery.
- Fast, elastic scaling, because each microservice is independently deployable, individual services can be scaled out (or back) rapidly in response to demand. For example, during a mass certificate renewal or onboarding event. This can occur without redeploying or resizing the whole platform allowing this scaling out or back to result in now downtime.
- Blue-green upgrades, new versions of a service can be deployed alongside the running version and cut over once validated, so platform upgrades and patching can be carried out with zero downtime and a fast rollback path if an issue is found.
- High availability, inter- and intra-site, the microservices architecture can be configured for high availability within a single site (intra-site, across redundant nodes) and across multiple sites (inter-site), so the platform can continue operating through the loss of a node, a site, or an entire data centre.

2.2 True Multi-Tenancy, Built In, Not Bolted On

Where most competing Certificate Lifecycle Management and PKI platforms were designed as single-tenant systems and have multi-tenancy added afterwards (if at all), Jellyfish was architected from the outset to support multiple tenants natively. This is reflected directly in Cogito's own capability comparison, which lists "true multi-tenant / multi-organisation architecture (shared or isolated)" as a native Jellyfish capability that most competing products cannot confirm.

The practical benefits of this design choice are significant and apply in two very different deployment contexts:

- In the cloud, a single Jellyfish instance can securely serve many customer organisations, each with isolated keys, HSM partitions, databases and policy, reducing infrastructure overhead and cost while preserving strict data separation.
- On-premises, the same multi-tenant model lets an organisation run one internal deployment of Jellyfish as its own internal "as-a-Service" offering, serving multiple business units, departments, subsidiaries or agencies from a single platform instance, rather than standing up separate CLM/PKI infrastructure for each.

This is a genuine architectural differentiator: it is the difference between a platform that can be multi-tenant and one that was multi-tenant by design.

2.3 Deployment Options

Jellyfish is available as a fully managed cloud service, as software installed on a client's own infrastructure or preferred cloud, or as a hybrid combination of both. All deployment models deliver the same capability set, ensuring consistent policy and operations regardless of where the platform runs.

3 From CLM to a Full Credential Platform

Jellyfish began life as a Certificate Lifecycle Management (CLM) product. Over time it has expanded into a considerably broader platform, now certified to Common Criteria and listed on the U.S. National Information Assurance Partnership (NIAP) Product Compliant List. This certification pathway reflects the rigour applied to the platform's certificate authority and underlying security architecture.

From that CLM and PKI foundation, Jellyfish has grown to include:

- Public Key Infrastructure, Certificate Authority (CA) and Registration Authority (RA) components.
- Key Archive and full, general and specific key management capability for both internal and cloud architectures.
- Smartcard Management, split between Card Management Systems (CMS) for the physical/virtual credential lifecycle and Card Application Management (CAMS) for the applications resident on the card.
- SSH key management, One-Time Password (OTP) management, HSM management and password management.
- Configuration Management Database (CMDB), including IP Address Management (IPAM),
- Supporting services such as alerting, monitoring, audit and Identity Management.

Taken together, this breadth of coverage, spanning certificates, keys, smartcards, SSH, OTP, HSMs and passwords within one operational model, is, to Cogito's knowledge, the first time all of these credential types have been unified into a single full-service credential management platform, rather than requiring separate point solutions stitched together.

The benefit of all credential management components being in one place is that relationships or associations can be made between the components allowing for a far better management outcome. The outcome no longer lacks business context, as is the case with other CLM products. An example is that you can report the certificates a user has on their smartcard, by going to the user account, any certificate the user has or the smartcard itself. From there you can see what other smartcards or certificates the user has on or off cards. Another example is you can report on the set of web servers and the certificate expiration on all of them together. This approach also allows far better automation and reporting outcomes that can easily span the different categories of system types, as they are all elements of the one Jellyfish product.

Many vendors in this space have simply brought together Certificate Lifecycle Management and Certificate Authority capabilities and marketed the result as a unified platform. Jellyfish takes a fundamentally different approach: it was designed from the ground up to be a Certificate Authority (CA), Registration Authority (RA), Certificate Lifecycle Management (CLM) platform, Card Management System (CMS), Card Application Management System (CAMS), Configuration Management Database (CMDB), Identity Broker (IdB), Key Manager, HSM manager and Discovery tool, all within a single, natively integrated platform, rather than a set of acquired or bolted-together products.

This ground-up design extends to how Jellyfish interfaces with the wider environment. It can manage both Physical Access Control Systems (PACS) and Logical Access Control Systems (LACS), interface with LDAP and CSV-driven systems, and integrate with building management systems and Wi-Fi access systems, along with many other enterprise systems beyond these.

4 Core Capabilities

4.1 Public Key Infrastructure (CA & RA)

Jellyfish provides its own Certificate Authority and Registration Authority components, supporting certificate discovery, issuance, renewal and revocation across mixed environments, and maintaining continuous trust through OCSP/CRL validation and comprehensive audit trails. The CA is Common Criteria certified and NIAP listed.

Root Certificate Authorities provide the basis of trust within a PKI and are protected accordingly: for higher-assurance deployments, the Root CA is operated as an offline device with its own dedicated Hardware Security Module (HSM), stored in a safe when not in use and forbidden from accessing internal network services, with any data transfer to its subordinate CAs carried out manually rather than over a network connection. This can be delivered with all the Jellyfish components or a much smaller command line version can be used that also includes a Terminal User Interface.

Beneath the Root CA, Jellyfish supports a conventional CA hierarchy of Policy and Issuing Authorities. Policy CAs can be established as intermediate CAs that constrain which policies can be used for certificate issuance beneath them, and are optional depending on the hierarchy required. Issuing CAs are signed by the appropriate Policy CA (or directly by the Root CA where no Policy CA is used) and, as the CAs that actually interact with end entities, are typically the only CAs exposed through the PKI DMZ described in Section 2.1.

Access to CA private keys can be controlled through hardware security modules in combination with hardware-based authentication tokens and passwords, so that credentials issued by one tenant can never be used to access another tenant's keys. This directly reinforces the multi-tenant isolation described in Section 2.2. The services supports the use of Certificate Practice Statements (CPSs), and Certificate Policies (CPs).

Validation Authority services are provided alongside the CAs themselves, in the form of OCSP responders and HTTP/LDAP Certificate Revocation List (CRL) Distribution Points (CDPs). Revocation information is available either as a live OCSP response or from a CRL that is generated and published automatically, so client applications always have access to current revocation status when validating a certificate. These publication points can be stood up automatically as part of standing up the CA.

For specialised government and border-security use cases, Jellyfish's PKI can also support Electronic Machine Readable Travel Document (eMRTD) issuance, including Country Signing Certificate Authority, Document Signing, Active Authentication Key Generation, National Public Key Directory and Document Security Object Validation services, illustrating the platform's ability to support highly specialised certificate ecosystems beyond conventional enterprise PKI.

As a Certificate Lifecycle Management (CLM) platform, Jellyfish is not limited to managing its own certificate authority elements. It supports a wide range of third-party CA types alongside the native Jellyfish CA, including Microsoft Active Directory Certificate Services (AD CS), UniCERT, Entrust Certificate Authority (ECA), EJBCA, Dogtag and OpenSSL-based CAs. Virtually any certificate can be imported directly into the CLM. This lets an organisation bring its existing CA investment under unified lifecycle management rather than needing to migrate to a single vendor's CA to gain CLM capability, including externally trusted CAs and certificates from trusted third parties.

Where an organisation wants to consolidate onto the Jellyfish CA itself, Jellyfish can migrate any of these CA types, AD CS, UniCERT, ECA, EJBCA, Dogtag or OpenSSL, across to a Jellyfish CA while the source CA remains live and issuing, rather than requiring a cutover outage or a parallel-run migration project. This is particularly useful for instance, where a CA may have 20 or more automation points. Jellyfish allows for these CA automation points to be cut over one at a time. This is exceptionally user where, for example CA01 has a discreet Autoenrolment responder in Domain 01, Domain 02, Domain 03, Domain 04, Domain 05 and Domain 06. Domain 01 can be cut over on the Monday, with Domain 02 coming over on Wednesday and so on for instance. On Tuesday, Domain 01 continues to issue certificates from the Jellyfish instance while Domain 2 continues to issue certificates from the legacy CA. All revocation can occur through Jellyfish, as Jellyfish polls in all certificates from the legacy CA01 platform in order to maintain a complete list of holdings.

Jellyfish allows for these CA automation points to be cut over one at a time.

For example, CA01 has a discreet Autoenrolment responder in Domain 01, Domain 02, Domain 03, Domain 04, Domain 05 and Domain 06.

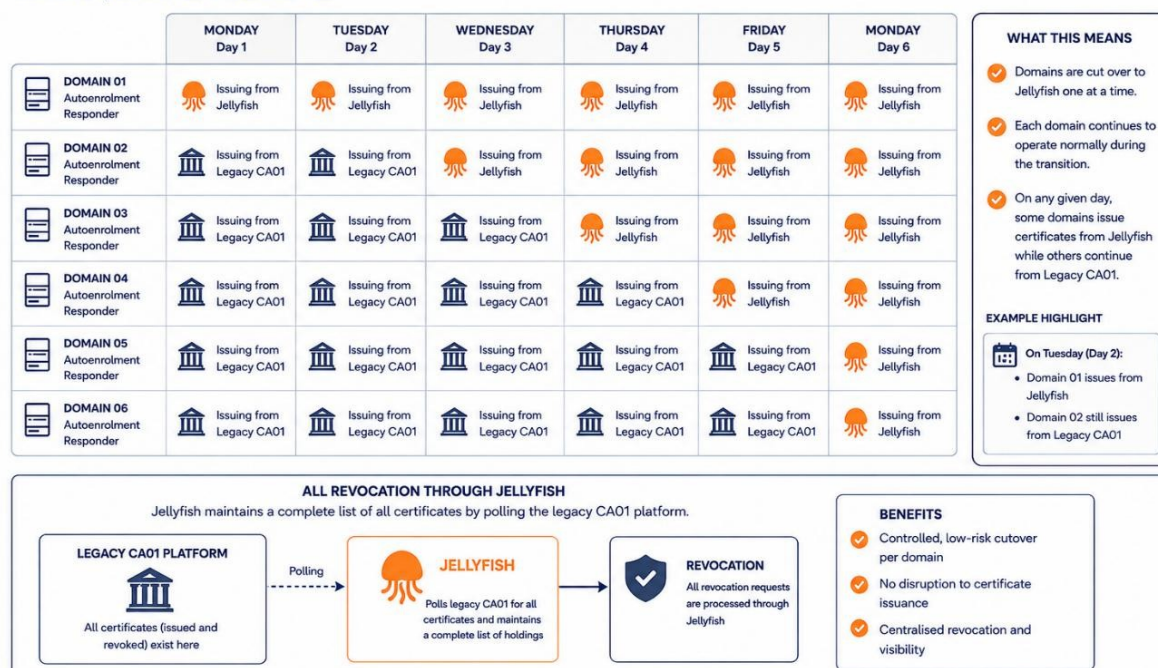


Figure 3 - Jellyfish Transition and Automation

4.2 Key Management as a Service

Key Management as a Service (KMaaS) securely stores and manages keys for both on-premises and cloud-based services. It supports protection of PKI keys, encrypted storage, Transparent Data Encryption (TDE) for databases, API key generation, and Bring Your Own Key (BYOK) models.

Jellyfish generates and manages a broad range of key and key material types, including API keys, symmetric keys (AES, 3DES, etc), asymmetric keys (RSA, ECC, ML-DSA, etc) and CSRs. Jellyfish maintains a secure Key Archive so keys can be recovered and reused across new or existing services rather than being lost if a service is retired. Keys are protected in Hardware Security Modules that are FIPS 140-3 Level 3 accredited.

For cloud integration, Jellyfish supports BYOK key wrapping and export into AWS KMS, Azure Key Vault, Google Cloud Platform CSEK and Salesforce Shield Platform Encryption, along with one-click Credential Sync of certificates and private keys into AWS Certificate Manager and Azure Key Vault.

Because Jellyfish is a full HSM and Key Management platform as well as a CLM, its discovery capability extends well beyond certificates. Jellyfish can discover and inventory certificates and keys wherever they reside, on smartcards, in HSMs, on devices, across third-party and cloud Certificate Authorities, and elsewhere on the network, and do the same for the full range of key types it manages, not just certificate-bound keys. This gives a single, searchable inventory of every credential and key in the environment, rather than a certificate-only view.

4.3 Smartcard & Credential Management, CMS and CAMS

Card Lifecycle Management (CMS) acts as the operational control plane for physical and virtual authenticators, smartcards, USB tokens, hardware security keys, FIDO2 authenticators and PKI-backed certificates, taking each credential through a structured, auditable lifecycle:

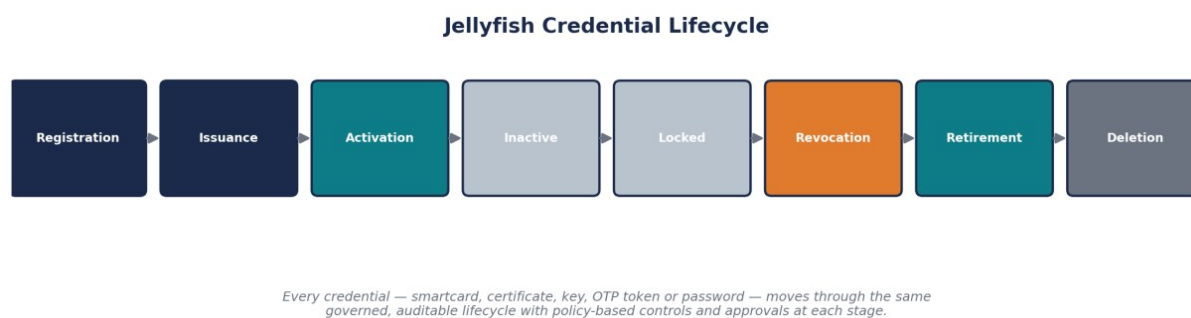


Figure 4: The Jellyfish credential lifecycle

- Registration, secure onboarding and establishment of a root of trust.
- Issuance, policy-based provisioning using templates, supporting individual or batch rollout.
- Activation, admin-led or user self-service enablement.
- Inactive, temporary suspension without permanent teardown.
- Locked, security response to failed PIN attempts, with helpdesk-driven recovery.
- Revocation, the “kill switch” for lost, stolen or compromised credentials, or offboarding.
- Retirement, secure wipe and reuse of hardware to reduce cost and waste.
- Deletion, removal from active inventory, with transaction history retained for audit.

Card Application Management (CAMS) manages the applications resident on a smartcard as a fully integrated part of the same CMS platform, including applet installation, updates and patching, a centralised management console, role-based access control, and full audit and reporting through Cogito's SIEM integration. An application library and customisation model allow application versions to be uploaded, versioned and assigned to specific cards, with updates applied automatically during routine operations rather than requiring a traditional “one-and-done” re-issuance.

Token support throughout CMS and CAMS is standards-based rather than proprietary: Jellyfish supports any cryptographic token exposing the PKCS #11 interface, along with the PIV standard used across

government and enterprise smartcard deployments, so existing hardware investments can be brought under Jellyfish management without replacing compliant tokens already in circulation.

4.4 SSH, OTP, SSO, HSM Management and Passwords, First Full Credential Management Service

Beyond certificates and smartcards, Jellyfish extends the same governed, auditable lifecycle model to SSH keys, One-Time Passwords, Single Sign-On (SSO), Hardware Security Module management and password management. SSO lets users authenticate once against Jellyfish and gain seamless access to connected applications and services without repeated logins, while every access decision remains centrally governed and audited alongside the platform's other credential types. Combined with the PKI, key management and smartcard capabilities above, this makes Jellyfish, as far as Cogito is aware, the first platform to bring the full breadth of enterprise credential types into a single, consistently managed and audited service, rather than requiring separate tools for each credential type with no interaction between those tools.

4.5 CMDB, IPAM, Identity, Monitoring, Alerting & Audit

Jellyfish also provides a full Configuration Management Database (CMDB), IP Address Management (IPAM), Identity Management (IdM), and platform-wide Monitoring, Alerting and Audit. The CMDB tracks and maintains asset details and preserves the relationships between devices, owners, users, locations and certificates, and, uniquely in Cogito's comparison of leading platforms, Jellyfish supports information exchange between physical access control (PAC) and logical access control (LAC) systems, letting an event in one domain (for example, a badge swipe) trigger a control in the other.

Relationships between PACs and LACs are the basis for automated provisioning and deprovisioning: because Jellyfish already understands how identities, devices, credentials and access rights relate to one another, it can automatically provision the applications and accounts a new user, device or service needs, and just as importantly deprovision them the moment a relationship ends, reducing the administrative burden on operations staff and closing the access gaps that manual offboarding processes typically leave open. A good example of this is that disabling a user in a Windows domain can affect building access too or vice versa. Jellyfish finally brings logical and physical access control together as it should be.

Capability	Capability
✓ Certificate Lifecycle Management & PKI (CA + RA)	✓ Key Management (general & specific, internal + cloud) & Key Archive
✓ Smartcard / Credential Management, CMS & CAMS	✓ SSH Key Management
✓ One-Time Password (OTP) Management	✓ HSM Management
✓ Password Management	✓ Identity Management (IdM)
✓ IP Address Management (IPAM)	✓ Configuration Management Database (CMDB)
✓ Alerting	✓ Monitoring
✓ Audit & Reporting	✓ Boolean Search across all credential types

Summary of platform modules, see the full capability comparison in Section 8 for detail.

4.6 Identity Brokerage & Shared Access Services

Identity Brokerage provides a single endpoint for authentication against multiple identity providers. This is particularly useful in service-provider style configurations, where there is a need to authenticate against the identity stores of multiple separate client organisations, or against multiple identity stores belonging to a single client, through one common integration point rather than a separate integration per store.

Built on Identity Brokerage, Jellyfish supports Technology-Enabled Shared Access (TeSA) for organisations sharing a building or facility, allowing shared resources and services to be governed through a single identity and credential model rather than siloed per tenant. Shared services commonly enabled this way include:

- Wi-Fi access
- Shared printing
- Shared lockers
- Shared visitor management
- Shared booking management
- Shared Physical and Electronic Access Control systems (PACs/EACs)

TeSA is built from two components. The Network Access Engine (NAE) integrates with an organisation's existing network authentication, authorisation and accounting infrastructure rather than replacing it: a gateway proxies inbound and outbound authentication traffic, a Radius server handles the authentication requests themselves, and a Jellyfish integration service validates those requests against the credentials Jellyfish manages. In practice, a user presents a certificate to an application or a Wi-Fi access point; the request is proxied to the Radius server, validated against Jellyfish, and access is granted only once both authentication and the user's permissions have been confirmed.

The Shared Accommodation Federated Directory (SAfD) acts as a single identity store so that organisations sharing a facility can jointly leverage third-party platforms, a shared visitor management system, for example, while each organisation retains full control of its own user base. SAfD is populated from each organisation's own directory (via Microsoft Graph API for Azure AD, or a standard LDAPS connection for on-premises Active Directory, either pulled on a schedule or pushed via SCIM on change), translated into a common identity store, and exposed to support staff so they can search users and issue temporary credentials for shared resources. Consistent with the automated provisioning and deprovisioning described in Section 4.5, once an account is provisioned or deprovisioned in the source organisation's directory, that change flows through automatically to every connected shared platform without manual intervention.

A third component can be added to these two being a full Identity Access Management platform to provide wider identity services and governance capabilities.

5 Search

Jellyfish is built around one of the strongest search capabilities of any credential platform, supporting full Boolean search (AND / OR / NOT combinations) across certificates, keys, smartcards, users, devices and CMDB records simultaneously. In Cogito's competitor comparison, “complex Boolean certificate search” and “Boolean search capability” for keys are capabilities Jellyfish supports outright, while at least one named competitor does not support this at all and others report it only for a subset of credential types.

Because search spans the CMDB as well as the credential stores, an administrator can search for a certificate based on any field available on the associated device or user record, or conversely search for a user or device based on any field available in the certificate, collapsing what would otherwise be several separate lookups into a single query.

Key to this capability however is a very easy to use interface. This interface can support saving of such searches and their results, effectively creating a repeatable and very capable reporting tool.

6 Certificate Automation Suite

Certificate outages caused by unmanaged or forgotten certificates are one of the most common causes of unplanned downtime. Jellyfish addresses this with a comprehensive automation suite that verifies and secures certificate requests, monitors and reports on certificates throughout their life, automatically discovers unmanaged certificates, and automates issuance and renewal wherever appropriate, via Microsoft Auto-Enrolment (AEX), SCEP, the REST API, CMP, ACME and ACME+, and ITSM integration (e.g. ServiceNow).

Not every certificate needs to be automated: Jellyfish also supports manual issuance directly through its own interface, where a customer can either submit a certificate signing request or complete a web-based enrolment form to generate a certificate and key pair, with optional approval workflows and automatic expiry notifications. Private keys for end-entity certificates generated this way are never stored by the platform operator. Where automation is preferred but AEX is not applicable, Linux hosts can be automatically enrolled and renewed via a dedicated Auto-Enrolment Proxy, mirroring the Windows AEX capability described below.

6.1 Auto-Enrol Xchange (AEX)

Standard Windows Auto-Enrolment is convenient, but in practice these certificates are frequently the least well managed in an organisation, overlooked relative to manually issued certificates, prone to template misconfiguration and failed renewals, and, in legacy implementations, reliant on DCOM in ways that sit awkwardly with zero-trust network segmentation and modern security practices.

Jellyfish's Auto-Enrol Xchange (AEX) modernises this by implementing Microsoft's MS-XCEP and MS-WSTEP protocols. On the customer intranet, workstations request policy and enrolment from the Jellyfish AEX IIS web server, which collects workstation information from the Domain Controller and receives its GPO deployment from the same source. The AEX web server communicates with the Jellyfish web server in the Jellyfish Cloud, which authenticates the AEX certificate via the Authentication Server, submits the enrolment request to the Auto-Enrolment Services, and collects the customer's enrolment configuration from the Database API Server and Database Server. The PKI API Server then requests certificate issuance and the Certificate Authority issues the certificate back through the chain to the workstation.

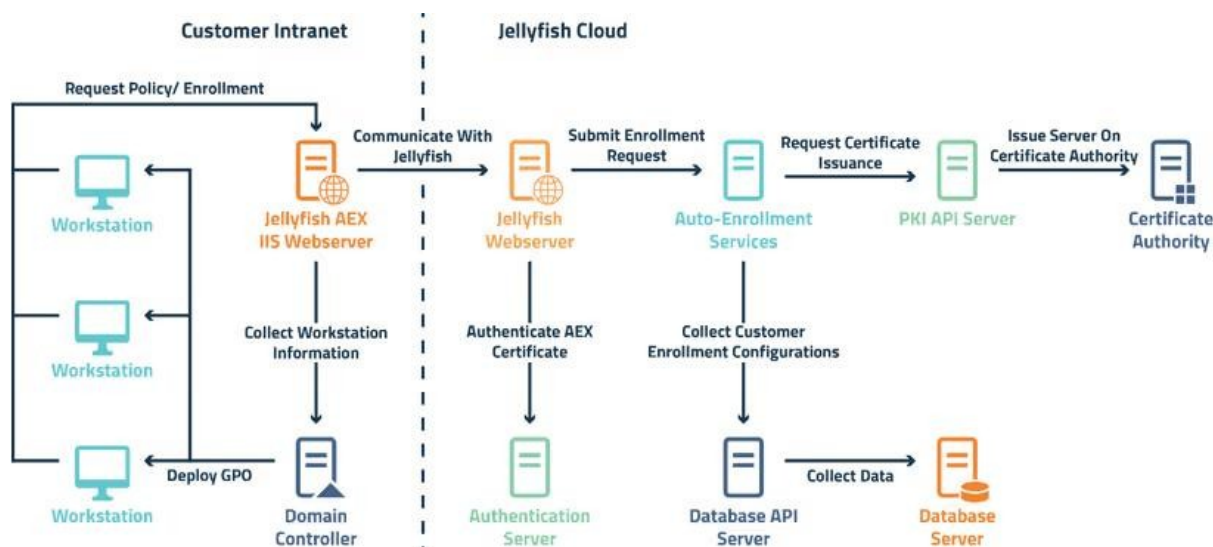


Figure 5: Auto-Enrol Xchange (AEX), certificate automation flow

Certificates issued this way remain fully visible, searchable and revocable through the Jellyfish portal, including dedicated search filters for Auto-Enrolment certificates, closing the visibility gap that standard Auto-Enrolment leaves behind.

6.2 Simple Certificate Enrolment Protocol (SCEP)

Jellyfish's SCEP implementation is fully compliant with RFC 8894 and the Cryptographic Message Syntax it defines. SCEP is the primary automatic enrolment method for devices managed through Microsoft Intune, Autopilot and Company Portal, and is also widely used for Linux, mobile and IoT-class devices such as office phones and printers, supporting use cases from Wi-Fi and VPN authentication to secure email and access to managed company resources.

For Intune-managed devices, Microsoft Intune pushes the SCEP profile, trust certificates and a challenge token to the Microsoft Intune Company Portal on the mobile device. The Company Portal checks Jellyfish's SCEP capabilities, validates the Jellyfish Registration Authority certificate, and requests the device certificate. Jellyfish, in turn, verifies the device's identity and validates the challenge token back with Intune before returning the device certificate.

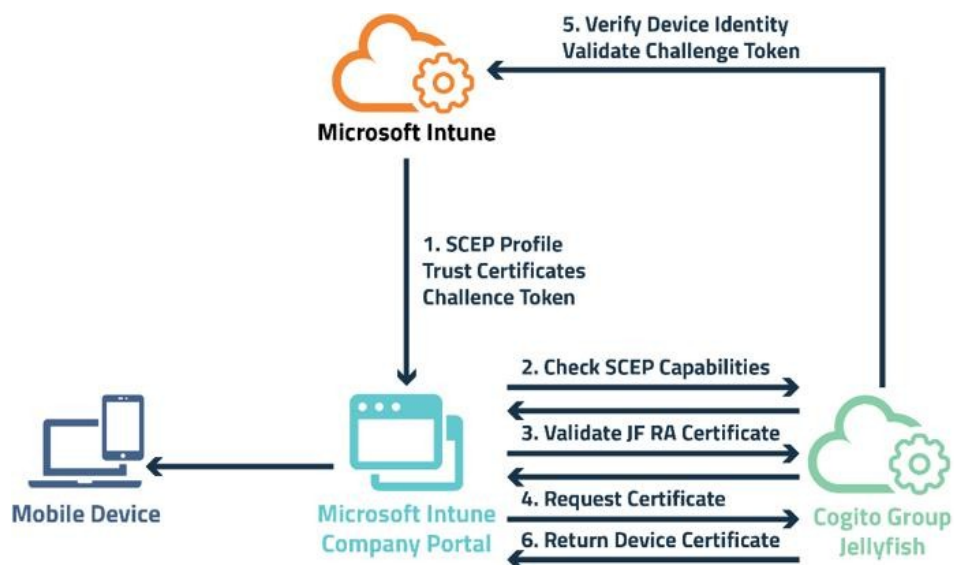


Figure 6: SCEP enrolment via Microsoft Intune

6.3 REST API

The Jellyfish REST API allows any capability available in the web portal to be driven programmatically, the portal itself is built on these same API calls. Requests from workstations, webservices or mobile apps cross into the Jellyfish Cloud as an API request to the Jellyfish web server, which authenticates the API token via the Authentication Server and forwards the request to the appropriate microservice, spanning PKI Services, Identity Services and Cryptography Services.

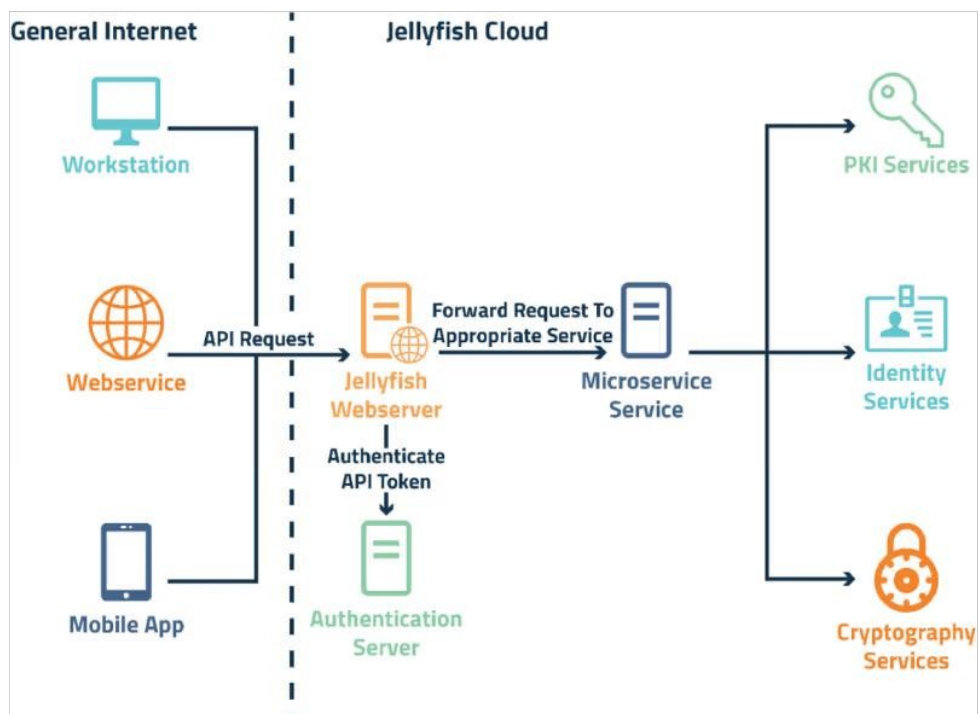


Figure 7: Jellyfish REST API, integration overview

This gives clients a fully documented, supported integration path for connecting Jellyfish into custom applications, ITSM tools and other enterprise systems, rather than depending on portal-only workflows.

6.4 Enrolment over Certificate Management Protocol (CMP)

CMP is a standardised protocol (CMPv2, RFC 4210, moving toward the CMPv3 proposed standard in RFC 9810) for automating certificate enrolment, renewal and revocation, particularly suited to IoT, mobile network and large-scale enterprise PKI deployments. The end entity (client) can issue an Initialization Request (key generation and initialisation), a Certification Request (issue/renew a certificate), a Polling/Response check, a Key Update Request (key update and renewal), or a Revocation Request (revoke and confirm), each exchanged as a secure message with the CA/RA and recorded in the PKI database.

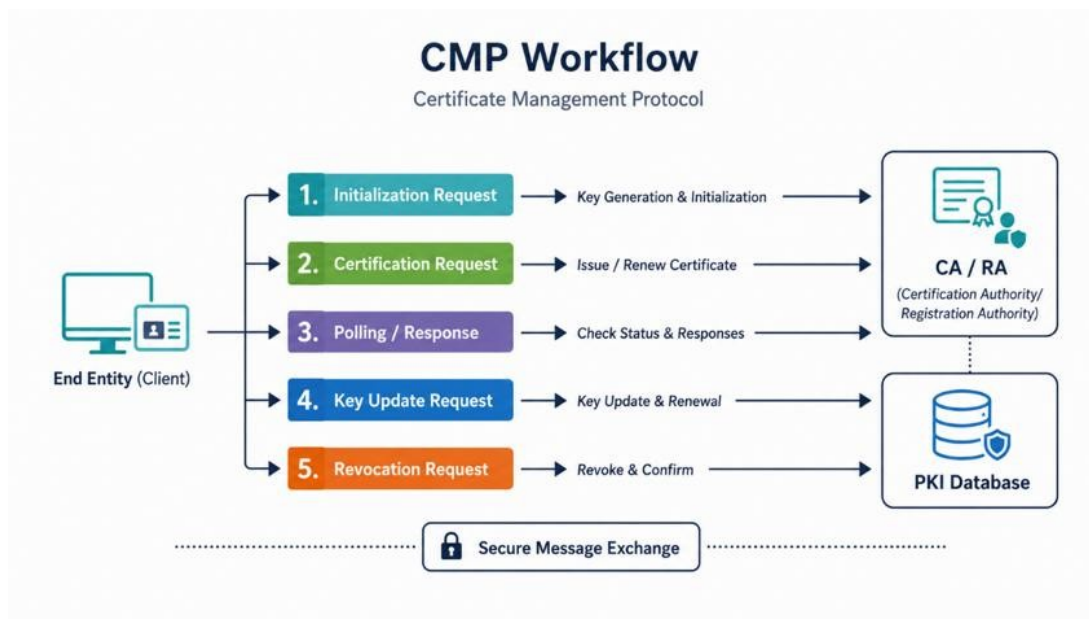


Figure 8: CMP, Certificate Management Protocol workflow.

Jellyfish supports CMPv2/v3, including the newer, more secure cryptographic mechanisms and post-quantum algorithms such as ML-DSA introduced in CMPv3.

6.5 ACME and ACME+

The Automatic Certificate Management Environment (ACME) protocol, standardised in RFC 8555, automates certificate issuance for web servers using signed JSON messages over HTTPS, validated via HTTP or DNS-based domain challenges, the mechanism behind popular tools such as Certbot.

Cogito Group's ACME+ extends the standard protocol beyond web-server certificates using an agent deployed within the customer's network. The customer webserver sends an ACME request to the Jellyfish ACME Agent on the intranet, which forwards a websocket certificate request to the Jellyfish web server in the Jellyfish Cloud. The Jellyfish web server authenticates the ACME agent via the Authentication Server, requests certificate issuance from the PKI API Server, and the Certificate Authority issues the certificate, which is then recorded back through the Database API Server into the Database Server. Configuration of the ACME agent itself is managed directly from the customer's workstation web browser.

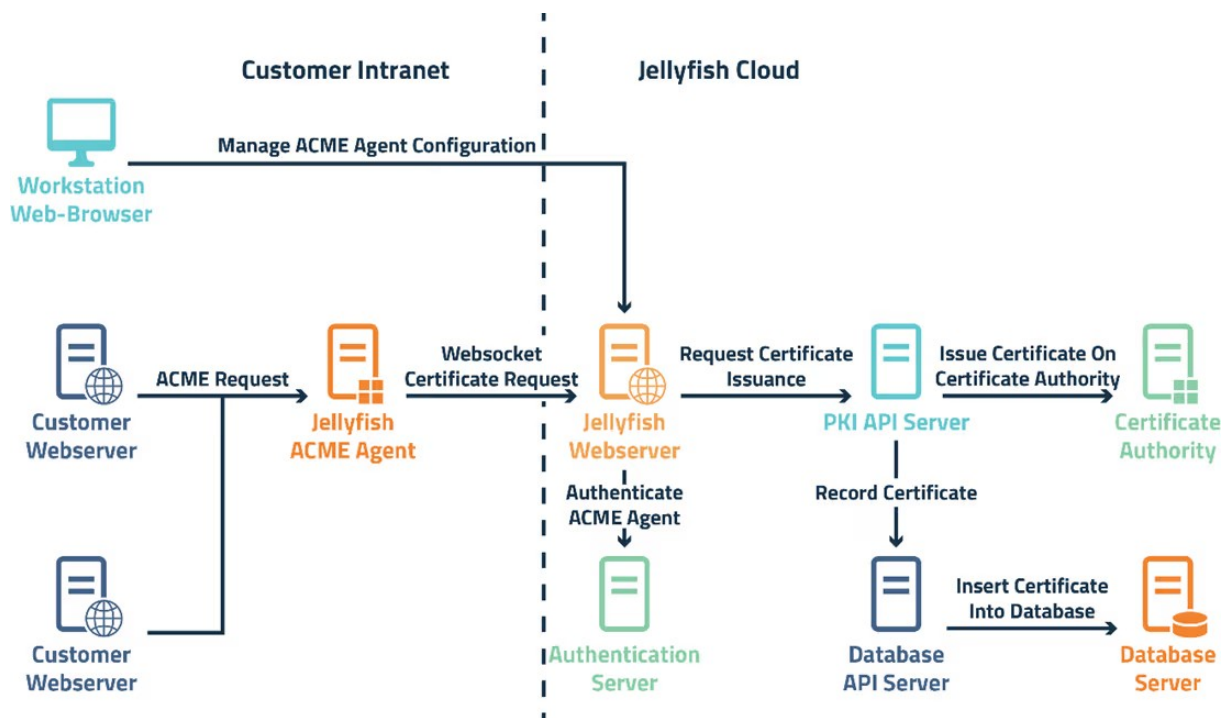


Figure 9: ACME and ACME+, automated certificate issuance flow

This allows automated issuance using familiar tooling even where a DNS name is not available as the identifier, including the ability to semi-automate certificate issuance on disconnected or air-gapped networks, a capability standard ACME cannot provide.

7 Security, Reliability & Scale

The platform's security posture reflects the same design principles as its architecture:

- Zero trust by design, the platform is built to zero trust principles, with least-privilege service permissions and strong cryptographic defaults throughout. Every connection between components, including internal, service-to-service traffic, not just externally-facing connections, is authenticated and encrypted using mutual TLS (mTLS), so no connection inside the platform is implicitly trusted.
- Hardware-backed assurance, private keys are protected in Cogito's own FIPS 140-3 Level 3 accredited HSMs.
- High availability, resilient microservices, health checks and fast recovery, underpinned by the multi-tenant, horizontally scalable architecture described in Section 2.
- Layered, multi-factor authentication, logon supports multiple, stackable authentication factors rather than a single MFA step, such as certificate-based authentication followed by API-based verification, so access can be hardened to match the sensitivity of the action being performed.
- Open integration, REST APIs and webhooks for DevOps and enterprise workflow integration, and direct integration with Certificate Authorities, HSMs, identity directories, PACS/facilities systems and CMDB/network tooling.

8 How Jellyfish Compares

Cogito Group maintains a detailed, 24-category, feature-by-feature comparison of Jellyfish against MyID, Venafi, Keyfactor (with EJBCA) and AppViewX, covering certificate management, CMDB, access control, key management, token management, reporting, supportability and more. A representative sample of differentiating capabilities is shown below; the full comparison is available on request.

Capability	Jellyfish	MyID	Venafi	Keyfactor+EJBCA	AppViewX
Complex Boolean certificate / key search	✓	✗	✓	✓	?
True multi-tenant / multi-org architecture	✓	?	?	?	✓
Common Criteria certified & NIAP listed CA	✓	?	?	?	?
PQC key creation and management	✓	✗	?	✓	✓
Hold Your Own Key (HYOK) for cloud/SaaS	✓	✗	✗	✗	✗
Full CMDB capability	✓	✗	✗	?	?
One Time Password Management	✓	✗	✗	✗	✗
Password Management	✓	✓	✗	?	✗
Relationship building between identities (devices, services, users)	✓	✗	✗	✗	✗
Transfer information between building & ICT systems (LAC↔PAC)	✓	✗	✗	✗	✗
Local (AU/NZ) security-cleared development staff	✓	✗	✗	✗	✗

'?' denotes a capability not confirmed as available for that vendor at the time of writing. Across the full 24-category comparison, Jellyfish is recorded as meeting essentially every listed capability, and is the only platform of the five with confirmed Common Criteria / NIAP certification of its Certificate Authority, native multi-tenant architecture, full Boolean search across certificates and keys, One-Time Password management, and Hold Your Own Key support for cloud/SaaS platforms such as Microsoft 365.

9 Deployment Options & Technical Specifications

Jellyfish is available as a managed cloud service (including via SecureSME for small and medium enterprises), installed on a client's own infrastructure, or as a hybrid of both, with the same capability set delivered consistently across all models. Indicative technical requirements for on-premises or private-cloud deployment scale with organisation size:

Deployment	Server OS	CPU	RAM	Storage
Small	Windows Server (VM)	6 cores	8 GB	500 GB
Medium	Windows Server / CentOS (VM)	24 cores	64 GB	4 TB
Large	Sized to requirements (virtual or physical)	Sized to requirements	Sized to requirements	Sized to requirements

10 Summary & Next Steps

Jellyfish combines a resilient, independently scalable microservices architecture with genuine multi-tenancy designed in from the outset, delivering the efficiency of a cloud service and the option to run as an internal as-a-Service platform on-premises. From its origins in Certificate Lifecycle Management, it has grown into a Common Criteria certified, NIAP-listed platform spanning PKI, key management, smartcard and credential management, SSH, OTP, HSM management and passwords, backed by enterprise-grade Boolean search and a full suite of automated enrolment options, AEX, SCEP, REST API, CMP, ACME and ACME+.

We would welcome the opportunity to walk through a live demonstration of the platform, tailored to your environment, and to discuss which deployment model, cloud, on-premises or hybrid, best fits your organisation's requirements.

Appendix: Reference Sources

This document draws on the following Cogito Group and Jellyfish Training Academy sources, together with Cogito's internal architecture and workflow diagrams:

- cogitogroup.net/about-jellyfish
- cogitogroup.net/as-a-service
- training.cogitogroup.net/documentation/key-management-as-a-service
- training.cogitogroup.net/documentation/card-lifecycle-management
- training.cogitogroup.net/documentation/card-application-management
- training.cogitogroup.net/documentation/jellyfish-cert-management-platform-comparison
- training.cogitogroup.net/documentation/autoenrol-exchange-aex
- training.cogitogroup.net/documentation/simple-certificate-enrolment-protocol-scep
- training.cogitogroup.net/documentation/certificate-automation
- training.cogitogroup.net/documentation/enrolment-over-cmp
- training.cogitogroup.net/documentation/acme-facts
- training.cogitogroup.net/documentation/how-jellyfish-works
- Cogito Group internal architecture diagrams, Jellyfish interfaces, AEX, SCEP/Intune, REST API, CMP and ACME workflow diagrams (supplied directly by Cogito Group).

The credential lifecycle diagram (Figure 2) is an original illustration produced for this design overview to represent the lifecycle described in Cogito's Card Lifecycle Management documentation.