**Zero cross twenty-nine architecture**

**Elements :**

* Tool user: Initiates diagnostic service via native tool.
* Native Windows App Tool: frontend tool installed on tester's laptop used to interact with vehicle
* Certificate Auto tool: Manages certificate-based authentication (likely generates CSR, etc.).
* Cloud MAM(mobile accessibility management): Acts as the middleware managing sessions and authentication logic.
* IAM(Amazon cognido, OIDC provider): Handles identity verification and provides access tokens.
* Identity Provider: Authenticates user credentials (can be LDAP, SSO, etc.).
* OCSP Responder: Validates the certificate (Online Certificate Status Protocol).
* OMS/CTAS (License Management): Checks if user has valid access license to perform diagnostic services.
* Logging/Database: Logs access and service usage; used for auditing and monitoring.
* ECM/ECU (Electronic Control Module): The in-vehicle target component where diagnostics are performed.

**Communication Protocols Used:**

* **HTTPS (TLS/SSL)**: Secure communication (used throughout for certificates and cloud components).
* **OCSP**: For certificate status verification.
* **OIDC (OpenID Connect)**: For identity verification and token management.
* **Custom APIs**: Likely RESTful APIs used between App Tool and MAM.

**Step by step flow explanation:**

* Pre-setup:
* User installs Windows App Tool and configures the system (once setup).
* Certificates are either installed manually or generated using the Certificate Auto Tool.
* Login Flow:
* User logs into Native App Tool:
* Tool calls the IAM (Cognito OIDC Provider) for OIDC authentication.
* IAM communicates with Identity Provider (LDAP, SSO etc.) to authenticate the user.
* If successful, IAM provides an ID Token and Access Token back to the App Tool.
* Token Verification & Session Start:
* App Tool sends the token to Cloud MAM.
* Cloud MAM:
* Validates the token with IAM.
* Extracts user roles/permissions.
* Contacts OMS/CTAS to verify if the user has a valid license.
* Uses OCSP Responder to verify the certificate status (revoked/valid).
* Logs the request in Logging/Database.
* ECU Communication:
* Once all checks pass, Cloud MAM authorizes the session.
* App Tool initiates communication with the ECU using a secure channel (likely TLS).
* ECU verifies the presented certificate (via OCSP or built-in logic).
* Diagnostic Service Execution:
* User performs service operations (e.g., read DTCs, reset counters).
* Each request goes via App Tool → Cloud MAM → ECU.
* Logs are updated in Database for traceability.
* Session Termination:
* Session is closed.
* Tokens are invalidated.
* Logging is completed.
* Audit trails are recorded.

1. Cloud MAM:

Cloud MAM stands for Cloud-based Mobile Access Management or Module Access Management, depending on context. In your case—related to ZeroCross 29 Remote Diagnostics and vehicle ECUs—Cloud MAM is:

A centralized authorization and access control service that ensures only valid, authenticated, and licensed users/tools can interact with sensitive vehicle modules (like ECUs) through secure cloud communication.

Key Responsibilities of Cloud MAM in This Architecture

1. Authentication & Authorization

* 1. Receives tokens from IAM (e.g., Cognito OIDC).
  2. Validates the identity and roles of the user/tool.
  3. Grants access to diagnostic/flash/remote services based on token and role.
  4. Performs OCSP checks to validate the certificate presented by the diagnostic tool.
  5. Ensures that expired or revoked certificates are denied access.

1. License Validation
   1. Contacts License Management System (OMS/CTAS) to verify:
      1. Is the user/tool licensed to perform diagnostics?
      2. Are there usage limits or quotas?
      3. What types of ECUs or services are allowed?
2. Audit Logging
   1. Records each user action/session to a logging database for:
      1. Security auditing
      2. Compliance (e.g., ISO 21434, GDPR)
      3. Diagnostic traceability
3. Command Proxying
   1. Acts as a secure bridge between the Native App Tool and the ECU:
      1. Routes diagnostic commands (DTC read/write, flash) securely.
      2. May modify or verify payloads before allowing access to the ECU.
      3. Ensures session management and ECU availability checks.

2. **certificate authority and OCSP responder**

A Certificate Authority (CA) is a trusted organization or entity that:

1. Issues digital certificates to users, applications, or devices.
2. Verifies the identity of whoever is requesting a certificate.
3. Signs the certificate using its private key, which others can verify using the CA's public key.

Purpose of a CA:

1. Ensures authentication: Confirms who the certificate really belongs to.
2. Supports secure communication: Enables encrypted TLS/SSL channels (like HTTPS).
3. Adds trust to public key infrastructure (PKI).

Example:

You want to secure a diagnostic tool:

1. You generate a Certificate Signing Request (CSR) from the tool.
2. The CA verifies your identity (via organization, email, etc.).
3. It issues a certificate, binding your identity with a public key.
4. Now, other systems (like Cloud MAM) trust your tool if it has a valid cert.

3.**Identity provider IAM(cognito ,OIDC provider)**

1. Identity Provider :
2. An Identity Provider is a trusted service that manages user identities and authentication.
3. Verifying who the user is (authentication).
4. Issuing tokens (such as access tokens, ID tokens) after successful login.
5. Supporting Single Sign-On (SSO).
6. Managing user profiles, credentials, multi-factor auth, etc.
7. **IAM: Identity and Access Management:**
   1. IAM defines who (identity) can access what (resources).
   2. It includes permissions, roles, policies, and authorization logic

**4. OIDC: OpenID Connect:**

OIDC is a modern identity protocol built on top of OAuth 2.0.

OIDC allows:

1. Secure authentication using ID tokens.
2. Identity verification using standard claims (e.g., name, email).
3. Federated login (e.g., login via Google, Facebook, enterprise).

**5. Amazon Cognito:**

Amazon Cognito is AWS’s managed Identity Provider that:

1. Supports user pools (user directory).
2. Handles authentication (login, password, MFA, etc.).
3. Issues OAuth2 / OIDC-compliant tokens.
4. Integrates with other AWS services like IAM, API Gateway, etc.

**5. existing licence management like OMS/CTAS(Operations Management System/Centralized Tool Authorization System):**

The OMS/CTAS license management ensures:

1. Only licensed users or tools can:
   1. Connect to vehicle ECUs remotely.
   2. Perform operations like read/write DTCs, firmware flashing, or calibration.
2. Tools or users have valid access rights, such as:
   1. Which ECU types or vehicle brands/models they can access.
   2. Which operations they’re allowed to perform (diagnostic, programming, etc.).
3. Usage is tracked and limited:
   1. Number of allowed sessions.
   2. License expiry date.
   3. Region or time-based restrictions.

**6. logging and database**

Logging is the process of recording detailed information about events that occur in the system.

Purpose of Logging in this architecture:

1. Audit Trails
   1. Tracks who did what, when, from where, and on which ECU/tool.
   2. Useful for security audits and traceability.
2. Diagnostics & Debugging
   1. Logs help trace errors or bugs in the remote diagnostic workflow.
3. Security Compliance
   1. Logs show that proper access control was enforced (e.g., rejected unauthorized access).
4. Performance & Usage Monitoring
   1. How often tools are used, which ECUs are most accessed, etc.

Database:

1. The database stores both persistent structured data and log records, such as:
2. Who logged in, session tokens, session expiry, etc.
3. Each command sent to ECU via Cloud MAM (e.g., “read DTC”, “firmware write”).
4. Whether the tool had valid permission (validated via OMS/CTAS).
5. Records of certificate status and validation timestamps.
6. Data Flow Example:
7. User opens App Tool → requests operation.
8. Auth via Cognito → Cloud MAM handles request.
9. MAM checks with OMS/CTAS for license.
10. Certificate check via OCSP Responder.
11. Logging records:
    1. Timestamp
    2. User ID
    3. ECU accessed
    4. Operation performed
    5. License status

**Command's:**

1. UpdateOCSPresponse: The App Tool is requesting an OCSP response update — it wants fresh certificate status
2. Nonce: A nonce (number used once) is a random value added to the OCSP request.

Working:

* + The App Tool sends an OCSP nonce request to the OCSP responder:
  + It includes:
    - Certificate serial number
    - Nonce (random ID)
  + 📥 The OCSP responder checks certificate status with the CA.
  + 📤 It replies with an OCSP response:
  + Certificate status: good / revoked / unknown
  + Includes same nonce to prove it's fresh and not cached.

1. MAM\_initDiagAccess: initializes MAM cloud instance of user so MAM is prepared to connect to module with that tool user is logged in on Checks for valid issued cert in DB, goes and gets one if needed, obtains license NOTE-This logic does not currently have "Device binding", but may be the place to add if we want that funcitonality.
2. MAM\_initECM: gets updated certs from MAM (if needed) & tells MAM what ECM will be connecting
3. MAM\_getEph Key: gets an ephpubkey from MAM that is unique and associated with a specific connection instance
4. MAM\_getbidirVerifyCert: gets informaiton needed to send verifyCertificateBidirectional request to ECM, the tool certificate, and a challenge
5. MAM\_putbidirVerifyCert: takes response from ECM and performs all cryptographic operations (verify cert, verify proof of ownership) and returns a tool proof of ownership generated from ECM's challenge so that Tool can request proof OfOwnsership
6. MAM getSessionKey: requests the generated session key so diagnostics can commence (ECM has been unlocked) Should only work if associated instance is showing authentication steps have occured