External Certificate Verification API

Comprehensive documentation for verifying single-use certificates via third-party systems

Quick Navigation

Overview

The External Certificate Verification API provides a dedicated endpoint for external systems to verify single-use certificates (SUC) by their certificate ID. The system records the IP address of the requester for audit and tracking purposes.

Base URL

```
/api/v1/external
```

Authentication

This API uses **Keycloak** for authentication. All requests must include a valid JWT token in the Authorization header.

Obtaining a JWT Token

cURL

```
curl -X POST \
  'https://{url}/auth/realms/cfhub/protocol/openid-connect/token' \
  -H 'Content-Type: application/x-www-form-urlencoded' \
  -d 'client_id=external-verifier&client_secret=your-client-secret&grant_type=password&username=your-username
```

Example Response

JSON

```
{
  "access_token": "eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA...",
  "expires_in": 300,
  "refresh_expires_in": 1800,
  "refresh_token": "eyJhbGciOiJIUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA...",
  "token_type": "Bearer",
```

```
  "not-before-policy": 0,
  "session_state": "a856fb91-eabc-4158-9f35-e0927613d9c6"
}
```


Required Keycloak Roles

Role	Description
`external-verifier`	Basic role required for certificate verification

Endpoints

Verify Certificate

POST `/verify/{certificateId}`

Verifies a Single Use Certificate by its unique ID.

Path Parameters

Parameter	Type	Description	Required
`certificateId`	String	Unique identifier of the certificate to verify	Required

Headers

Header	Description	Required
`Authorization`	Bearer JWT token from Keycloak authentication	Required
`Accept`	Should be set to `application/json`	Required
`Content-Type`	Should be set to `application/json`	Required
`X-Forwarded-For`	IP address of the client (automatically captured by proxies)	Optional
`X-Real-IP`	Alternative header for client IP (if X-Forwarded-For is not available)	Optional

Example Request

cURL

```
curl -X POST \
  'https://dev.cfhub.net/api/v1/external/verify/SUC-2025-08-12345AB' \
  -H 'Authorization: Bearer eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA...' \
  -H 'Accept: application/json' \
  -H 'Content-Type: application/json'
```

Response Examples

200 OK - Successful Verification

JSON

```
{
  "status": "SUCCESS",
  "message": "Certificate verification completed",
  "data": {
    "certificateId": "SUC-2025-08-12345AB",
    "cargoId": "CRG-2025-08-54321",
    "status": "ACTIVE",
    "isValid": true,
    "isExpired": false,
    "customsEntryNumber": "CE12345678",
    "issuedAt": "2025-08-10T10:15:30Z",
    "expiresAt": "2025-09-10T23:59:59Z",
    "verifiedAt": "2025-08-26T15:30:45Z",
    "verificationCount": 3,
    "agentName": "John Doe",
    "agentLicense": "AG-12345-XY",
    "issuingAuthority": "Customs Authority",
    "blNumber": "BL987654321",
    "idfNumber": "IDF-2025-08-001",
    "message": null
  },
  "timestamp": "2025-08-26T15:30:45.123Z"
}
```

200 OK - Invalid Certificate

JSON

```
{
  "status": "SUCCESS",
  "message": "Certificate verification completed",
  "data": {
    "certificateId": "SUC-2025-08-12345AB",
    "status": "REVOKED",
    "isValid": false,
    "isExpired": false,
    "verifiedAt": "2025-08-26T15:33:22Z",
    "verificationCount": 4,
    "message": "Certificate has been revoked"
  },
  "timestamp": "2025-08-26T15:33:22.789Z"
}
```

401 Unauthorized

JSON

```
{
  "status": "ERROR",
  "message": "Authentication failed: Invalid or expired token",
  "data": null,
  "timestamp": "2025-08-26T15:31:30.456Z"
}
```

403 Forbidden

JSON

```
{
  "status": "ERROR",
  "message": "Authorization failed: Insufficient permissions to verify certificates",
  "data": null,
  "timestamp": "2025-08-26T15:31:45.789Z"
}
```

404 Not Found

JSON

```
{
  "status": "ERROR",
  "message": "Certificate verification failed: Certificate with ID SUC-2025-08-INVALID not found",
  "data": null,
  "timestamp": "2025-08-26T15:32:10.456Z"
}
```

Response Field Descriptions

Field	Type	Description
`certificateId`	String	Unique identifier for the certificate
`cargoId`	String	Identifier for the associated cargo/shipment
`status`	String	Current status of the certificate (ACTIVE, USED, EXPIRED, REVOKED, etc.)
`isValid`	Boolean	Indicates if the certificate is valid for use
`isExpired`	Boolean	Indicates if the certificate has expired
`customsEntryNumber`	String	Associated customs entry number
`issuedAt`	ISO DateTime	Date and time when the certificate was issued
`expiresAt`	ISO DateTime	Date and time when the certificate expires
`verifiedAt`	ISO DateTime	Date and time of the current verification
`verificationCount`	Number	How many times this certificate has been verified
`agentName`	String	Name of the agent who issued the certificate
`agentLicense`	String	License number of the issuing agent
`issuingAuthority`	String	Name of the authority that issued the certificate
`blNumber`	String	Bill of Lading number associated with the shipment

`idfNumber`	String	Import Declaration Form number associated with the shipment
`message`	String	Additional information about verification result (only present in certain cases)

Security Considerations

JWT Authentication: All requests must include a valid JWT token from Keycloak in the Authorization header.

Token Expiration: JWT tokens have a limited validity period. Applications should handle token refresh when needed.

IP Address Tracking: Each verification request records the IP address of the requester for security and audit purposes.

Rate Limiting: Excessive verification requests from the same IP address may be subject to rate limiting.

Audit Trail: All verification attempts are logged, including successful and failed attempts.

Role-Based Access: The API uses Keycloak roles to determine access permissions. External verifiers must have the appropriate role assigned.

Implementation Examples

JavaScript (fetch API)

JavaScript

```
async function verifyCertificate(certificateId, jwtToken) {
  const response = await fetch(`https://dev.cfhub.net/api/v1/external/verify/${certificateId}`, {
    method: 'POST',
    headers: {
      'Authorization': `Bearer ${jwtToken}`,
      'Accept': 'application/json',
      'Content-Type': 'application/json'
    }
  });

  return await response.json();
}

// Usage
verifyCertificate('SUC-2025-08-12345AB', 'eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA...')
  .then(result => console.log(result))
  .catch(error => console.error('Verification failed:', error));
```

Python (requests)

Python

```
import requests
```

```python
def verify_certificate(certificate_id, jwt_token):
    headers = {
        'Authorization': f'Bearer {jwt_token}',
        'Accept': 'application/json',
        'Content-Type': 'application/json'
    }

    response = requests.post(
        f'https://dev.cfhub.net/api/v1/external/verify/{certificate_id}',
        headers=headers
    )

    return response.json()

# Usage
jwt_token = 'eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA...'
result = verify_certificate('SUC-2025-08-12345AB', jwt_token)
print(result)
```

Implementation Notes

- The API is designed to be stateless, with each request containing all necessary information for verification.
- Certificate verification results include detailed information about the certificate when verification is successful.
- When a certificate is invalid, expired, or revoked, the response will include a relevant message explaining the reason.
- JWT tokens should be securely stored and transmitted using HTTPS.
- Applications should implement token refresh logic to handle token expiration gracefully.