

📘 Vyoma SaaS License Manager API Documentation

🔹 Base URL

```
https://localhost:5002/api/LicenseValidation
```

All endpoints accept and return JSON.

🟢 0️⃣ Get Authorization Codes (Bootstrap Endpoint)

Endpoint

```
GET /getcodes
```

Full URL

```
https://localhost:5002/api/LicenseValidation/getcodes
```

Description

Provides initial authentication codes required by client applications:

- `ClientId`
- `C2VCode`

Call this **once during setup**. These codes are required for all subsequent license validations.

✅ Success Response

```
{
  "clientId": "CLIENT-ABC",
  "c2vCode": "ENCRYPTED-C2V-CODE"
}
```

❌ Error Response

```
400 Bad Request
```

Returned when no license is available.

📦 Response Model

```
public class AuthCodesForClients
{
    public string ClientId { get; set; }
    public string C2VCode { get; set; }
}
```

🟢 1️⃣ Validate Named License

Endpoint

```
POST /validate-named
```

Full URL

```
https://localhost:5002/api/LicenseValidation/validate-named
```

Description

Validates a named license user. Ensures:

- User is assigned
- Seat limits are enforced
- System time is consistent
- DomainName check passes (if provided)

📩 Request Body

```
{
  "clientId": "CLIENT-ABC",
  "c2vCode": "ENCRYPTED-C2V-CODE",
  "productName": "My Product",
  "sessionId": "GUID-SESSION-ID",
  "userId": 42,
  "userEmail": "user@example.com",
  "domainName": "example.com"
}
```

Notes:

- `clientId`, `c2vCode`, `productName`, `sessionId`, `userId`, `userEmail` → **required**
- `domainName` → **optional**

✅ Success Response

```
{
  "status": 200,
  "message": "Access granted."
}
```

❌ Possible Errors

Status	Meaning
400	Invalid license
401	License expired / user not assigned / seat exceeded
403	System time validation failed

📦 Request Model (C#)

```
public class NamedLicenseRequest
{
    [JsonPropertyName("clientId")]
    public string ClientId { get; set; }

    [JsonPropertyName("c2vCode")]
    public string C2VCode { get; set; }

    [JsonPropertyName("productName")]
    public string ProductName { get; set; }

    [JsonPropertyName("userEmail")]
    public string UserEmail { get; set; }

    public int UserId { get; set; }

    public string SessionId { get; set; }

    [JsonIgnore(Condition = JsonIgnoreCondition.WhenWritingDefault)]
    public string? DomainName { get; set; }
}
```

🟢 2️⃣ Validate Concurrent License

Endpoint

```
POST /validate-concurrent
```

Full URL

```
https://localhost:5002/api/LicenseValidation/validate-concurrent
```

Description

Validates a concurrent license session:

- Enforces real-time seat limits
- Verifies system time
- DomainName optional

📥 Request Body

```
{
  "clientId": "CLIENT-ABC",
  "c2vCode": "ENCRYPTED-C2V-CODE",
  "productName": "My Product",
  "sessionId": "GUID-SESSION-ID",
  "userId": 42,
  "domainName": "example.com"
}
```

Notes:

- `clientId`, `c2vCode`, `productName`, `sessionId`, `userId` → **required**
- `domainName` → **optional**
- No `device` or `requestUrl` included.

✅ Success Response

```
{
  "status": 200,
  "message": "Access granted."
}
```

❌ Possible Errors

Status	Meaning
400	Invalid license
401	Concurrent limit reached / license expired
403	System time validation failed

📦 Request Model (C#)

```
public class ConcurrentLicenseRequest
{
    public string ClientId { get; set; }
    public string C2VCode { get; set; }

    [JsonPropertyName("productName")]
    public string ProductName { get; set; }

    public string SessionId { get; set; }
    public int UserId { get; set; }

    [JsonIgnore(Condition = JsonIgnoreCondition.WhenWritingDefault)]
    public string? DomainName { get; set; }
}
```

🟢 3️⃣ Logout Concurrent Session

Endpoint

```
POST /logout-concurrent
```

Request Body

```
{
  "clientId": "CLIENT-ABC",
  "c2vCode": "ENCRYPTED-C2V-CODE",
  "sessionId": "GUID-SESSION-ID"
}
```

Success Response

```
{
  "message": "Session released."
}
```

🟢 4️⃣ Logout Named Session

Endpoint

```
POST /logout-named
```

Request Body

```
{
  "clientId": "CLIENT-ABC",
  "c2vCode": "ENCRYPTED-C2V-CODE",
  "sessionId": "GUID-SESSION-ID"
}
```

Success Response

```
{
  "message": "Session released."
}
```

🔄 Recommended Integration Flow

Step 1 — Bootstrap

```
GET /getcodes
```

- Store `ClientId` and `C2VCode` for all validations.

Step 2 — On Application Start

- Generate **unique SessionId** (GUID recommended)
- Call either:
- `/validate-concurrent` or
- `/validate-named`

- If `status = 200` → allow application access.

Step 3 — On Logout / Application Exit

- Call `/logout-concurrent` or `/logout-named`

🔐 Important Notes for Integrators

- `sessionId` must be unique per session.
- DomainName can be used to restrict licenses to specific domains.
- Seat limits are enforced in real-time.
- System time is validated; if drift exceeds limits, 403 is returned.
- Application must handle non-200 responses gracefully.
