API & Automation

- REST API Overview
- API Examples

REST API Overview

The CMS exposes a REST API that mirrors all major platform functions, allowing administrators, partners, and developers to automate operations.

API Design Principles

- **Standards-Based**: JSON over HTTPS, aligned with REST conventions.
- **Role-Aware**: All calls are scoped to the authenticated user's role (Platform, Distributor, Partner, Tenant).
- **Secure by Default**: Requires bearer tokens obtained through login or federation flows.
- **Consistent Resources**: Entities in the API (Distributors, Partners, Tenants, Subscriptions, Usage, Billing, Commissions) align directly with portal objects.

Authentication

- **Login Flow**: Obtain a token via the `/api/v1/auth/login` endpoint.
- **Federated Tokens**: Forward Microsoft Entra or Google tokens for verification at `/api/v1/auth/federated`.
- **Token Lifetime**: Tokens are time-bound; refresh before expiry to avoid failures.

Core Endpoints

- **Distributors & Partners**
  - `/api/v1/distributors` – create, list, and manage distributors
  - `/api/v1/partners` – create, list, and manage partners
- **Tenants & Subscriptions**
  - `/api/v1/tenants` – onboard tenants, assign admins
  - `/api/v1/subscriptions` – create and manage subscriptions
- **Plans & Pricing**
  - `/api/v1/plans` – publish and update plans
  - `/api/v1/pricing` – define SKUs, rates, and thresholds
- **Usage & Billing**
  - `/api/v1/usage` – ingest usage data
  - `/api/v1/billing/summary` – generate subscription and tenant billing summaries
- **Commissions**
  - `/api/v1/commissions/summary` – calculate and retrieve commissions for partners and distributors
- **System & Identity**
  - `/api/v1/settings` – update branding, licensing, and identity configs
  - `/api/v1/roles` – assign or update user roles

Typical Automation Scenarios

- **Tenant Onboarding**
  Automate tenant creation, subscription assignment, and role setup via API.
- **Usage Ingestion**
  Post usage records into `/api/v1/usage` on a scheduled job (e.g., nightly processing per region).
- **Billing Automation**
  Trigger invoice generation and retrieve billing summaries at end-of-month.
- **Commission Reconciliation**
  Automate pulling `/commissions/summary` for financial reporting and payouts.
- **Integration with External Systems**
  - Connect billing output to finance/ERP systems.
  - Forward alerts or health checks to monitoring solutions.
  - Sync user identity with corporate directories.

Developer Notes

- Full Swagger documentation is bundled with the CMS at `/swagger`.
- API clients can be generated automatically from the provided `swagger.json`.
- Rate limits apply to protect system performance; batch operations where possible.

API Examples

PowerShell

Below is an example of how to authenticate and work with the CMS Api.

Note: Make sure the user you are using is permitted to perform CMS Authentication.

```
##########################################################################
# Variables
##########################################################################
# Modify for your endpoint
$BaseUrl     = 'https://api.cloudaxis.cloud/api/v1/'
# This user is permitted to do CMS Auth
$EmailAddress = 'sample@api.connection'
$Password     = 'D!dYouRe@llyTh1nkItW0uldB3Th4t3@sy'
##########################################################################
# Functions
##########################################################################
# Get a Token
function Get-CmsBearerToken
{
    [CmdletBinding(DefaultParameterSetName="Initial")]
    param([Parameter(Mandatory=$true, ParameterSetName="Initial")]
          [string]$EmailAddress,
          [Parameter(Mandatory=$true, ParameterSetName="Initial")]
          [string]$Password,
          [Parameter(Mandatory=$true, ParameterSetName="Refresh")]
          [string]$RefreshToken,
          [Parameter(Mandatory=$true)]
          [string]$BaseUrl)
    end
    {
        New-CmsLogMessage -T Info "Retrieving $($PSCmdlet.ParameterSetName) CMS bearer token."

        switch ($PSCmdlet.ParameterSetName)
        {
            'Initial' {
```

```
                $Url = "$($BaseUrl.Trim('/'))/auth/token"
                $Body = @{
                    EmailAddress = $EmailAddress
                    Password = $Password
                } | ConvertTo-Json
            }
            'Refresh' {
                $Url = "$($BaseUrl.Trim('/'))/auth/token-refresh"
                $Body = @{
                    RefreshToken = $RefreshToken
                } | ConvertTo-Json
            }
        }

        $Headers = New-Object "System.Collections.Generic.Dictionary[[String],[String]]"
        $Headers.Add("Content-Type", "application/json")

        try {
            $Response = Invoke-RestMethod $Url -Method 'POST' -Headers $Headers -Body $Body
        } catch {
            New-CmsLogMessage -T Error "Token request failed. $($_.Exception.Message)"
            throw
        }
        return [CMS.Domain.Models.AuthResponse]$Response
    }
}
#############################################################################
function New-CmsApiCall
{
    param([Parameter(Mandatory=$true)]
          [string]$BearerToken,
          [Parameter(Mandatory=$true)]
          [string]$BaseUrl,
          [Parameter(Mandatory=$true)]
          [string]$Uri,
          [psobject]$Body,
          [Parameter(Mandatory=$true)]
          [ValidateSet("GET", "POST", "PUT", "DELETE")]
          [string]$Method,
          [bool]$ExpectNotFound=$false)
```

```
    end
    {
        $Url = ("$($BaseUrl.Trim('/'))/$($Uri.TrimStart('/'))").ToLower()
        New-CmsLogMessage -T Info "Processing Api call - $($Method) $($Url)."

        switch ($Method)
        {
            'PUT' {
                if ($null -eq $Body) {
                    New-CmsLogMessage -T Error "Api call using $($Method) cannot have an empty
Body parameter."
                } else {
                    $Body = $Body | ConvertTo-Json
                }
            }
            'POST' {
                if ($null -eq $Body) {
                    New-CmsLogMessage -T Error "Api call using $($Method) cannot have an empty
Body parameter."
                } else {
                    $Body = $Body | ConvertTo-Json
                }
            }
        }

        $Headers = New-Object "System.Collections.Generic.Dictionary[[String],[String]]"
        $Headers.Add("Authorization", "Bearer $($BearerToken)")
        $Headers.Add("Content-Type", "application/json")

        try {
            $Response = Invoke-RestMethod $Url -Method $Method -Headers $Headers -Body $Body
            return $Response
        } catch {
            if (!$ExpectNotFound) {
                New-CmsLogMessage -T Error "Api call failed. $($_.Exception.Message)"
                return $Response
            }
        }
    }
}
```

```
#############################################################################
# Make Calls
#############################################################################
# Get Token for the Api
$Token = Get-CmsBearerToken -EmailAddress $EmailAddress -Password $Password -BaseUrl
$ApiEndpoint
if (!$Token) {
    Write-Host "No token could be retrieved.";
    exit 1
}
# Get Subscriptions
$ApiResponse = New-CmsApiCall -BearerToken $Token.Token -BaseUrl $ApiEndpoint -Uri
"/subscriptions/" -Method GET
if ($ApiResponse.Code -ne 200) {
    Write-Host "Error retrieving subscriptions $($ApiResponse.Errors).";
    exit 1
} else {
    Write-Verbose "Loaded Subscripions."
    $Subscriptions = $ApiResponse.Data
}
```