ADAC REST API

Adac.Api is an ASP.NET Core Minimal API that exposes the capabilities of the `Adac` library over HTTP. It provides endpoints for inspecting, validating, verifying, extracting, and creating ADAC (Archival Digital Asset Container) files — enabling integration with web UIs, automation pipelines, and third-party systems.

Table of Contents

↑ Back to Top

Prerequisites

- .NET 10 SDK or later.

↑ Back to Top

Installation

Running from source

From the repository root:

```
dotnet run --project Adac.Api
```

The API starts at `https://localhost:7200` and `http://localhost:5200`.

Publishing a self-contained binary

To produce a standalone executable that does not require the .NET SDK at runtime:

```
dotnet publish Adac.Api/Adac.Api.csproj -c Release -r win-x64 --self-contained
```

Replace `win-x64` with your target runtime identifier (e.g. `linux-x64`, `osx-arm64`).

Configuration

The API uses the standard ASP.NET Core configuration system. Settings can be overridden using `appsettings.json`, environment variables, or command-line arguments.

Setting	Default	Description
`Logging:LogLevel:Default`	`Information`	Minimum log level for the application.
`Logging:LogLevel:Microsoft.AspNetCore`	`Warning`	Minimum log level for framework messages.
`AllowedHosts`	`*`	Semicolon-separated list of allowed host headers.

To change the listening URLs:

```
dotnet run --project Adac.Api --urls "http://0.0.0.0:8080"
```

Or via the `ASPNETCORE_URLS` environment variable:

```
export ASPNETCORE_URLS="http://0.0.0.0:8080"
dotnet run --project Adac.Api
```

↑ Back to Top

Quick Start

Start the API and create, inspect, and validate a container in under a minute:

```
# 1. Start the server (in a separate terminal)
dotnet run --project Adac.Api

# 2. Create a container from master files on disk
curl -X POST https://localhost:7200/api/containers \
  -H "Content-Type: application/json" \
  -d '{
    "outputPath": "C:\\archives\\photo.adac",
    "masters": ["C:\\scans\\scan_001.tif"],
    "title": "Family portrait, 1923"
  }'

# 3. Inspect its metadata
curl https://localhost:7200/api/containers/info?path=C%3A%5Carchives%5Cphoto.adac

# 4. Validate it
curl -X POST https://localhost:7200/api/containers/validate \
  -H "Content-Type: application/json" \
  -d '{"path": "C:\\archives\\photo.adac"}'

# 5. Verify integrity checksums
curl -X POST "https://localhost:7200/api/containers/verify?path=C%3A%5Carchives%5Cphot
```

↑ Back to Top

OpenAPI / Swagger

An OpenAPI 3.x document is served automatically:

```
GET /openapi/v1.json
```

Point any OpenAPI-compatible tool (Swagger UI, Postman, Insomnia, etc.) at this URL to explore the API interactively.

↑ Back to Top

Base URL

All endpoints are grouped under:

```
/api/containers
```

The examples below assume the server is running at `https://localhost:7200`. Adjust the scheme, host, and port to match your deployment.

↑ Back to Top

Endpoints

`GET /api/containers/info` — Display Container Metadata

Returns the **manifest** (`manifest.json`) and **core metadata** (`metadata/core.json`) from an ADAC container.

Parameter	In	Required	Description
`path`	query	yes	Server-local path to the `.adac` file.

Example request:

```
GET /api/containers/info?path=C%3A%5Carchives%5Cphoto.adac
```

200 OK:

```
{
  "manifest": {
    "adacVersion": "1.0",
    "id": "b3f1a2c4-...",
    "createdOn": "2025-07-15T12:00:00+00:00",
    "createdBy": "adac-api",
    "description": "Scanned photograph",
    "masters": [ ... ]
  },
  "coreMetadata": {
    "id": "b3f1a2c4-...",
    "title": "Family portrait, 1923",
    "format": "TIFF"
  }
}
```

Response field	Type	Description
`manifest`	object	The full deserialized ADAC manifest.
`coreMetadata`	object	The deserialized core metadata record.

404 Not Found — the file does not exist at the given path.

`GET /api/containers/entries` — List Container Entries

Returns every entry path inside the container's ZIP archive.

Parameter	In	Required	Description
`path`	query	yes	Server-local path to the `.adac` file.

Example request:

```
GET /api/containers/entries?path=C%3A%5Carchives%5Cphoto.adac
```

200 OK:

```
{
  "count": 5,
  "entries": [
    "manifest.json",
    "metadata/core.json",
    "master/master_0001.tif",
    "provenance/log.json",
    "provenance/checksums.json"
  ]
}
```

Response field	Type	Description
`count`	integer	Total number of entries.
`entries`	string[]	Container-relative paths for every entry.

404 Not Found — the file does not exist at the given path.

`POST /api/containers/validate` — Validate Container

Validates a container against the ADAC 1.0 specification. Returns structured findings with severity levels, rule codes, and human-readable messages.

Request body (`application/json`):

```
{
  "path": "C:\\archives\\photo.adac",
  "skipChecksums": false
}
```

Field	Type	Required	Default	Description
`path`	string	yes	—	Server-local path to the `.adac` file.
`skipChecksums`	bool	no	`false`	When `true`, skip SHA-256 checksum verification (faster, structure-only).

200 OK:

```
{
  "isValid": false,
  "findings": [
    {
      "severity": "Info",
      "code": "ADAC-010",
      "message": "Container created with ADAC version 1.0.",
      "path": null
    },
    {
      "severity": "Error",
      "code": "ADAC-020",
      "message": "Master directory is empty.",
      "path": "master/"
    }
  ]
}
```

Response field	Type	Description
`isValid`	bool	`true` when no error-level findings exist.
`findings`	array	List of validation findings (see below).
`findings[].severity`	string	`Info`, `Warning`, or `Error`.
`findings[].code`	string	Stable machine-readable rule code (e.g. `ADAC-010`).
`findings[].message`	string	Human-readable description.
`findings[].path`	string?	Container-relative path involved, or `null`.

404 Not Found — the file does not exist at the given path.

`POST /api/containers/verify` — Verify Fixity Checksums

Computes SHA-256 hashes for every file in the container and compares them against the stored checksum manifest (`provenance/checksums.json`). Use this endpoint to confirm that no content has been altered or corrupted.

Parameter	In	Required	Description
`path`	query	yes	Server-local path to the `.adac` file.

Example request:

```
POST /api/containers/verify?path=C%3A%5Carchives%5Cphoto.adac
```

200 OK (all passing):

```
{
  "isValid": true,
  "totalFiles": 5,
  "verifiedFiles": 5,
  "failedFiles": 0,
  "missingFiles": 0,
  "mismatches": []
}
```

200 OK (with failures):

```
{
  "isValid": false,
  "totalFiles": 5,
  "verifiedFiles": 3,
  "failedFiles": 1,
  "missingFiles": 1,
  "mismatches": [
    {
      "path": "master/master_0001.tif",
      "expectedChecksum": "a1b2c3d4...",
      "actualChecksum": "ff00ff00...",
      "isMissing": false
    },
    {
      "path": "metadata/xmp/master_0001.xmp",
      "expectedChecksum": "deadbeef...",
      "actualChecksum": null,
      "isMissing": true
    }
  ]
}
```

Response field	Type	Description
`isValid`	bool	`true` when every file matches its recorded checksum.
`totalFiles`	integer	Number of files in the checksum manifest.
`verifiedFiles`	integer	Files that passed verification.
`failedFiles`	integer	Files with a checksum mismatch.
`missingFiles`	integer	Files listed in the manifest but absent from the container.
`mismatches`	array	Details for each failure (see below).
`mismatches[].path`	string	Container-relative path.
`mismatches[].expectedChecksum`	string	SHA-256 hash from the checksum manifest.
`mismatches[].actualChecksum`	string?	Computed hash, or `null` when the file is missing.
`mismatches[].isMissing`	bool	`true` when the file is absent from the container.

404 Not Found — the file does not exist at the given path.

`GET /api/containers/extract` — Download a Single Entry

Extracts a single entry from the container and returns it as a binary file download.

Parameter	In	Required	Description
`path`	query	yes	Server-local path to the `.adac` file.
`entry`	query	yes	Container-relative path of the entry (e.g., `master/master_0001.tif`).

Example request:

```
GET /api/containers/extract?path=C%3A%5Carchives%5Cphoto.adac&entry=master%2Fmaster_0(
```

200 OK — returns the file bytes with content type `application/octet-stream` and a `Content-Disposition: attachment` header containing the entry's filename.

404 Not Found — the container file does not exist.

> **Tip:** To download the manifest itself, use `entry=manifest.json`. Any entry path returned by the `/entries` endpoint is valid here.

`POST /api/containers` — Create a New Container

Creates a new ADAC container from master files that exist on the server. The API generates a valid manifest, core metadata, checksum manifest, and the required directory structure automatically.

Request body (`application/json`):

```
{
  "outputPath": "C:\\archives\\new-container.adac",
  "masters": [
    "C:\\scans\\page_001.tif",
    "C:\\scans\\page_002.tif"
  ],
  "title": "Two-page letter, 1945",
  "description": "Wartime correspondence",
  "createdBy": "Digitization Lab v2"
}
```

Field	Type	Required	Default	Description
`outputPath`	string	yes	—	File path for the new `.adac` container.
`masters`	string[]	yes	—	Server-local paths to master files. At least one required.
`title`	string	no	`null`	Artifact title (written to core metadata).
`description`	string	no	`null`	Artifact description (written to manifest and core metadata).
`createdBy`	string	no	`"adac-api"`	Creator identifier (written to the manifest).

201 Created:

```
{
  "path": "C:\\archives\\new-container.adac"
}
```

Response field	Type	Description
`path`	string	Absolute path to the newly created container.

The response includes a `Location` header pointing to the `/api/containers/info` endpoint for the new container, for example:

```
Location: /api/containers/info?path=C%3A%5Carchives%5Cnew-container.adac
```

Master files are assigned sequential identifiers (`master_0001`, `master_0002`, ...) and stored in the container's `master/` directory.

400 Bad Request — no masters provided, or a master file does not exist on disk.

↑ Back to Top

Error Responses

All error responses follow a consistent JSON shape:

```
{
  "error": "Human-readable error message."
}
```

Status	Meaning
`400 Bad Request`	Invalid input — missing required fields, empty masters array, or a referenced file does not exist on disk.
`404 Not Found`	The specified `.adac` container file does not exist at the given path.

↑ Back to Top

Usage Examples

cURL

```
# Get container metadata
curl https://localhost:7200/api/containers/info?path=C%3A%5Carchives%5Cphoto.adac

# List entries
curl https://localhost:7200/api/containers/entries?path=C%3A%5Carchives%5Cphoto.adac

# Validate (structure + checksums)
curl -X POST https://localhost:7200/api/containers/validate \
  -H "Content-Type: application/json" \
  -d '{"path": "C:\\archives\\photo.adac"}'

# Validate (structure only — faster)
curl -X POST https://localhost:7200/api/containers/validate \
  -H "Content-Type: application/json" \
  -d '{"path": "C:\\archives\\photo.adac", "skipChecksums": true}'

# Verify fixity checksums
curl -X POST "https://localhost:7200/api/containers/verify?path=C%3A%5Carchives%5Cphot

# Download a specific entry
curl -OJ "https://localhost:7200/api/containers/extract?path=C%3A%5Carchives%5Cphoto.a

# Create a new container
curl -X POST https://localhost:7200/api/containers \
  -H "Content-Type: application/json" \
  -d '{
    "outputPath": "C:\\archives\\new.adac",
    "masters": ["C:\\scans\\page_001.tif", "C:\\scans\\page_002.tif"],
    "title": "Wartime letter",
    "description": "Two-page letter, dated 1945"
  }'
```

PowerShell

```
$base = "https://localhost:7200"

# Get container metadata
Invoke-RestMethod "$base/api/containers/info?path=C:\archives\photo.adac"

# List entries
Invoke-RestMethod "$base/api/containers/entries?path=C:\archives\photo.adac"

# Validate
Invoke-RestMethod -Method Post "$base/api/containers/validate" `
    -ContentType "application/json" `
    -Body '{"path": "C:\\archives\\photo.adac"}'

# Verify fixity
Invoke-RestMethod -Method Post "$base/api/containers/verify?path=C:\archives\photo.ada

# Download an entry to disk
Invoke-WebRequest "$base/api/containers/extract?path=C:\archives\photo.adac&entry=mast
    -OutFile "master_0001.tif"

# Create a container
$body = @{
    outputPath = "C:\archives\new.adac"
    masters    = @("C:\scans\page_001.tif", "C:\scans\page_002.tif")
    title      = "Wartime letter"
} | ConvertTo-Json

Invoke-RestMethod -Method Post "$base/api/containers" `
    -ContentType "application/json" `
    -Body $body
```

C# HttpClient

```
using System.Net.Http.Json;

HttpClient client = new() { BaseAddress = new Uri("https://localhost:7200") };

// Get metadata
var info = await client.GetFromJsonAsync<JsonElement>(
    "/api/containers/info?path=C%3A%5Carchives%5Cphoto.adac");

// List entries
var entries = await client.GetFromJsonAsync<JsonElement>(
    "/api/containers/entries?path=C%3A%5Carchives%5Cphoto.adac");

// Validate
var validateResponse = await client.PostAsJsonAsync(
    "/api/containers/validate",
    new { path = @"C:\archives\photo.adac", skipChecksums = false });

var result = await validateResponse.Content.ReadFromJsonAsync<JsonElement>();

// Create
var createResponse = await client.PostAsJsonAsync(
    "/api/containers",
    new
    {
        outputPath = @"C:\archives\new.adac",
        masters = new[] { @"C:\scans\page_001.tif" },
        title = "Wartime letter",
    });
```

↑ Back to Top

Typical Workflows

Ingest and validate a new scan

```
# 1. Create the container
curl -X POST https://localhost:7200/api/containers \
  -H "Content-Type: application/json" \
  -d '{
    "outputPath": "C:\\archives\\deed.adac",
    "masters": ["C:\\scans\\deed.tif"],
    "title": "Deed of sale, 1887"
  }'

# 2. Validate against the ADAC 1.0 specification
curl -X POST https://localhost:7200/api/containers/validate \
  -H "Content-Type: application/json" \
  -d '{"path": "C:\\archives\\deed.adac"}'

# 3. Later — verify integrity after storage or transfer
curl -X POST "https://localhost:7200/api/containers/verify?path=C%3A%5Carchives%5Cdeed
```

Inspect an existing container

```
# List all entries to see what's inside
curl https://localhost:7200/api/containers/entries?path=C%3A%5Carchives%5Cdeed.adac

# Read full manifest and cataloging metadata
curl https://localhost:7200/api/containers/info?path=C%3A%5Carchives%5Cdeed.adac
```

Download a master file for processing

```
curl -OJ "https://localhost:7200/api/containers/extract?path=C%3A%5Carchives%5Cdeed.ad
```

↑ Back to Top

Project Structure

```
Adac.Api/
├── Adac.Api.csproj                    # ASP.NET Core Web project (net10.0)
├── Program.cs                         # App builder — registers services and maps enc
├── appsettings.json                   # Default configuration
├── Properties/
│   └── launchSettings.json            # Development launch profile
├── Endpoints/
│   └── ContainerEndpoints.cs          # Minimal API route definitions
└── Models/
    └── ContainerRequests.cs           # Request and response DTOs
```

All endpoints resolve ADAC services (`IAdacContainerReader`, `IAdacContainerWriter`, `IAdacValidator`, `IAdacFixityService`) from DI via the `AddAdacCore()` extension method registered in `Program.cs`.

↑ Back to Top

License

↑ Back to Top