Kulturpool APIs

Die Kulturpool APIs ermöglichen den programmatischen Zugriff auf digitale Kultur- und Kunstsammlungen österreichischer Institutionen. Die interaktive OpenAPI-Dokumentation ist verfügbar unter: api.kulturpool.at/docs

Einstieg: Kulturpool APIs

> Die Kulturpool APIs ermöglichen den programmatischen Zugriff auf digitale Kultur- und Kunstsammlungen österreichischer Institutionen. Sie bietet strukturierte Endpunkte für Institutionen, redaktionelle Inhalte und Assets mit umfangreichen Metadaten und mehrsprachiger Unterstützung.

APIs für Digitalisate

Suche

Die Suche API ermöglicht Volltext- und Facettensuche über die gesamte Kulturpool-Sammlung.

Objekt

Die Objekte API bietet direkten Zugriff auf die detaillierten Metadaten und Mediendateien einzelner Kulturgüter im Kulturpool.

OAI-PMH

OAI-PMH ist ein Standard-Protokoll für das Harvesting von Metadaten aus digitalen Archiven und Repositorien, das eine standardisierte Schnittstelle für die Übertragung von Metadaten zwischen verschiedenen Systemen bietet.

APIs für Redaktionelle Inhalte

Institutionen

Die Institutionen API bietet Zugriff auf Informationen über alle teilnehmenden Museen, Bibliotheken und Kultureinrichtungen im Kulturpool. Sie liefert strukturierte Daten mit Standortinformationen, Kontaktdaten und mehrsprachigen Beschreibungen.

Inhalte

Die Inhalte API ermöglicht den Zugriff auf redaktionelle Inhalte des Kulturpools. Sie liefert strukturierte Content-Seiten mit verschiedenen Block-Typen, SEO-Metadaten und mehrsprachigen Übersetzungen für die Website-

Darstellung.

Assets

Die Assets API ermöglicht den Zugriff auf Bilder und andere Medien-Dateien mit dynamischen Transformationsparametern.

Authentifizierung

Die APIs sind öffentlich zugänglich und erfordert keine Authentifizierung. Alle Endpunkte sind öffentlich verfügbar.

Rate Limiting

Aktuell sind keine expliziten Rate Limits implementiert. Die API nutzt Caching zur Performance-Optimierung.

API-Dokumentation

Die interaktive OpenAPI-Dokumentation ist verfügbar unter:

- **Swagger UI**: `https://api.kulturpool.at/docs`
- **ReDoc**: `https://api.kulturpool.at/redoc`

Suche API

> Der `/search` Endpunkt ermöglicht es, die gesamte Kulturpool-Sammlung über die integrierte Typesense-Suchmaschine zu durchsuchen. Die API leitet Anfragen automatisch an den Typesense-Server weiter und fügt den erforderlichen API-Schlüssel hinzu, sodass Nutzer ohne Authentifizierung suchen können.

Nähere Informationen zur Typesense-Suchmaschine finden Sie in der **Typesense-Dokumentation**.

Basis-URL

https://api.kulturpool.at/search

Parameter

Pflichtparameter

- **q** (string): Der Suchbegriff
  - Beispiele: `"Wien"`, `"Porträt"`, `"Albertina"`

Optionale Parameter

- **query_by** (string): Kommagetrennte Liste der zu durchsuchenden Felder
  - Standard: `"title,description,creator,subject,dataProvider,publisher,contributor,coverage,spatial,temporal,dcType,medium,isPartOf,identifier,alternative"`
- **filter_by** (string): Filter für Facetten
  - Beispiele: `"dataProvider:=Albertina"`, `"edmType:=IMAGE"`, `"dateMin:>1900"`
- **sort_by** (string): Sortierung der Ergebnisse
  - Beispiele: `"titleSort:asc"`, `"dateMin:desc"`
- **page** (integer): Seitennummer (ab 1, Standard: 1)
- **per_page** (integer): Anzahl Ergebnisse pro Seite (1-250, Standard: 20)
- **facet_by** (string): Kommagetrennte Liste der Facetten

    - Standard: `"dataProvider,creator,edmType,dcType,subject,medium,publisher,isPartOf,edmRightsName,edmRightsReusePolicy,dateMin,contributor,intermediateProvider"`
- **max_facet_values** (integer): Maximale Anzahl Facetten-Werte (1-100, Standard: 50)
- **highlight_full_fields** (string): Felder für vollständiges Highlighting
    - Standard: `"title,description,creator,subject"`
- **use_cache** (boolean): Typesense-Cache verwenden (Standard: true)

Verfügbare Suchfelder

Hauptfelder für Volltextsuche

- **title**: Titel des Objekts
- **description**: Beschreibung/Inhalt
- **creator**: Schöpfer:in/Künstler:in
- **subject**: Themen und Schlagwörter
- **dataProvider**: Institution (Datenlieferant)
- **publisher**: Veröffentlicht durch
- **contributor**: Mitwirkende:r Personen/Organisationen

Weitere durchsuchbare Felder

- **coverage, spatial, temporal**: Räumlicher und zeitlicher Bezug
- **dcType**: Dublin Core Objekttyp
- **medium**: Material/Technik/Medium
- **isPartOf**: Ist ein Teil von (Sammlungszugehörigkeit)
- **identifier**: Identifikatoren
- **alternative**: Alternativtitel

Verfügbare Facetten

Institutionen und Anbieter

- **dataProvider**: Hauptinstitution (z.B. "Albertina", "Belvedere", "Österreichische Nationalbibliothek")
- **intermediateProvider**: Zwischenanbieter
- **publisher**: Veröffentlicht durch

Objekttypen

- **edmType**: Europeana Data Model Typ
  - `IMAGE`: Bilder
  - `VIDEO`: Videos
  - `SOUND`: Audiodateien
  - `TEXT`: Textdokumente
  - `3D`: 3D-Objekte
- **dcType**: Dublin Core Typ (z.B. "Fotografie", "Gemälde", "Skulptur")

Inhaltliche Kategorien

- **subject**: Themen und Schlagwörter
- **medium**: Material/Technik/Medium (z.B. "Öl auf Leinwand", "Bromsilbergelatinepapier")
- **isPartOf**: Ist ein Teil von (Sammlung/Serie)

Rechtliche Informationen

- **edmRightsName**: Nutzungsrecht
  - `"Public Domain"`: Gemeinfrei
  - `"CC BY"`: Creative Commons Attribution
  - `"CC BY-SA"`: Creative Commons Attribution-ShareAlike
  - `"CC BY-NC"`: Creative Commons Attribution-NonCommercial
- **edmRightsReusePolicy**: Kann ich es weiterverwenden? (`OPEN`, `RESTRICTED`, `PERMISSION`)

Zeitangaben

- **dateMin**: Früheste Datierung (numerisch, Unix-Timestamp)
- **dateMax**: Späteste Datierung (numerisch, Unix-Timestamp)

Personen

- **creator**: Schöpfer:in/Künstler:in
- **contributor**: Mitwirkende:r Personen

Filterbeispiele

```
# Nur Objekte der Albertina
filter_by=dataProvider:=Albertina

# Nur Bilder
filter_by=edmType:=IMAGE

# Nur gemeinfreie Objekte
filter_by=edmRightsName:="Public Domain"

# Nach 1900 entstanden
filter_by=dateMin:>=-2208988800

# Kombinierte Filter (UND-Verknüpfung)
filter_by=dataProvider:=Albertina && edmType:=IMAGE

# ODER-Verknüpfung
filter_by=dataProvider:=[Albertina,Belvedere]

# Enthält-Filter
filter_by=creator:*Mozart*
```

Sortieroptionen

- **titleSort:asc/desc**: Alphabetisch nach Titel
- **dataProvider:asc/desc**: Nach Institution
- **dateMin:asc/desc**: Chronologisch (aufsteigend = älteste zuerst)
- **dateMax:asc/desc**: Chronologisch (absteigend = neueste zuerst)

Beispielanfragen

Einfache Suche

```
GET /search/?q=Wien
```

Erweiterte Suche mit Filtern

```
GET /search/?q=Porträt&filter_by=dataProvider:=Albertina&sort_by=dateMin:asc
```

Suche mit Facetten

```
GET /search/?q=Landschaft&facet_by=creator,medium,dateMin&max_facet_values=20
```

Paginierte Suche

```
GET /search/?q=Fotografie&page=2&per_page=50
```

Antwortformat

Die API gibt eine JSON-Antwort im Typesense-Format zurück:

Struktur der Antwort


```
{
  "found": 1234,
  "hits": [
    {
      "document": {
        "id": "67f4fc8fbb263def5311bac4",
        "title": ["Wien. - Praterleben. Aussenseiter."],
        "creator": ["Emil Mayer", "Brüder Kohn Wien (Postkartenverlag)"],
        "dataProvider": "Albertina",
        "edmType": "IMAGE",
        "dcType": ["Fotografie"],
        "medium": ["Bromsilbergelatinepapier (Postkarte, Rotationsverfahren)"],
        "isShownAt": "https://sammlungenonline.albertina.at/...",
        "isShownBy": "https://sammlungenonline.albertina.at/...",
        "previewImage": "https://static.kulturpool.at/images/...",
        "edmRights": "http://creativecommons.org/publicdomain/mark/1.0/",
```

```
        "edmRightsName": "Public Domain",
        "edmRightsReusePolicy": "OPEN",
        "dateMin": -1893456000,
        "dateMax": -1862006400,
        // ... weitere Felder
      },
      "highlight": {
        "title": [
          {
            "matched_tokens": ["Wien"],
            "snippet": "<mark>Wien</mark>. - Praterleben. Aussenseiter.",
            "value": "<mark>Wien</mark>. - Praterleben. Aussenseiter."
          }
        ]
      },
      "text_match": 578730123365187700
    }
  ],
  "facet_counts": [
    {
      "field_name": "dataProvider",
      "counts": [
        {"count": 456, "highlighted": "Albertina", "value": "Albertina"},
        {"count": 234, "highlighted": "Belvedere", "value": "Belvedere"}
      ]
    }
  ],
  "search_time_ms": 12,
  "page": 1
}
```


Wichtige Felder in den Objektdaten

Anzeige und URLs

- **title**: Titel des Objekts (Array)
- **description**: Beschreibung (Array)
- **isShownAt**: URL zur Detailseite der Institution
- **isShownBy**: URL zum Vollbild
- **object**: URL zur mittleren Auflösung

- **previewImage**: URL zum optimierten Vorschaubild
- **iiifManifest**: IIIF-Manifest URL (falls verfügbar)

Verknüpfung zu Objektdetails

- **kp_id**: Die Kulturpool-ID, die zum Abrufen der vollständigen Objektinformationen über die Objekt API verwendet werden kann.

Metadaten

- **creator**: Schöpfer:in/Künstler:in (Array)
- **contributor**: Mitwirkende:r (Array)
- **publisher**: Veröffentlicht durch (Array)
- **dataProvider**: Hauptinstitution (String)
- **intermediateProvider**: Zwischenanbieter (Array)

Zeitangaben

- **dateMin/dateMax**: Numerische Zeitstempel
- **created**: Datum (Erstellung) - Textuelle Datierung (Array)
- **date**: Weitere Datumsangaben (Array)
- **temporal**: Zeitlicher Bezug (Array)

Klassifikation

- **edmType**: Haupttyp (IMAGE, VIDEO, SOUND, TEXT, 3D)
- **dcType**: Spezifischer Objekttyp (Array)
- **medium**: Material/Technik/Medium (Array)
- **subject**: Themen/Schlagwörter (Array)

Rechteinformationen

- **edmRights**: Rights-URI
- **edmRightsName**: Nutzungsrecht (lesbarer Status)
- **edmRightsReusePolicy**: Kann ich es weiterverwenden? (OPEN, RESTRICTED, PERMISSION)
- **dcRights**: Zusätzliche Rechteangaben (Array)

Sammlungskontext

- **isPartOf**: Ist ein Teil von (Sammlungszugehörigkeit) (Array)
- **identifier**: Identifikatoren/Objektnummern (Array)
- **isReferencedBy**: Verweise (Array)

Räumliche Angaben

- **spatial**: Räumlicher Bezug/Ortsangaben (Array)
- **coverage**: Bezug/Abdeckung (Array)

Technische Daten

- **language**: Sprache (Array)
- **alternative**: Alternativtitel (Array)

Tipps für effektive Suchen

1. **Kombination von Suchbegriffen**: Nutzen Sie mehrere Begriffe für präzisere Ergebnisse
2. **Facetten nutzen**: Verwenden Sie Facetten zur Eingrenzung großer Ergebnismengen
3. **Wildcards**: `*` am Ende von Begriffen für Teilworttreffer
4. **Institutionen**: Filtern Sie nach spezifischen Institutionen für fokussierte Suchen
5. **Zeiträume**: Nutzen Sie dateMin/dateMax-Filter für chronologische Eingrenzung
6. **Nutzungsrecht**: Filtern Sie nach Nutzungsrechten bei Verwendung der Objekte

Die Typesense-Suchmaschine bietet fuzzy matching und kann auch bei Tippfehlern relevante Ergebnisse liefern.

Objekt API

> Die Objekte API bietet direkten Zugriff auf die detaillierten Metadaten und Mediendateien einzelner Kulturgüter im Kulturpool. Sie liefert strukturierte Daten im Europeana Data Model (EDM) als JSON-LD sowie eine aufbereitete Liste aller zugehörigen Medien.

Basis-URL

https://api.kulturpool.at/object

Endpunkte

Einzelnes Objekt abrufen

```
GET /object/
```

Ruft detaillierte Informationen zu einem spezifischen Objekt anhand seiner Kulturpool-ID ab.

Parameter

Parameter	Typ	Beschreibung	Beispiel
`id`	string	Die Kulturpool-ID als eindeutige UUID des Originaldatensatzes	`b1eaac0f-69a1-4174-8ac6-90a61615d0ae`

Beispiel-URL

```
https://api.kulturpool.at/object/?id=b1eaac0f-69a1-4174-8ac6-90a61615d0ae
```

Antwort

Die Antwort der API besteht aus zwei Hauptteilen: `metadata` und `media`.

- Das **`metadata`**-Objekt enthält die reichhaltigen beschreibenden Daten des Kulturguts. Es wird im Format **JSON-LD** (eine RDF-Serialisierung) ausgeliefert und folgt dem **Europeana Data Model (EDM)** . Diese Struktur zeigt die Daten so, wie sie im Kulturpool verarbeitet und an die Europeana

weitergegeben werden, und ermöglicht so eine maximale Interoperabilität.
- Das **`media`**-Objekt ist ein JSON-Array, das alle mit dem Objekt verknüpften externen Mediendateien auflistet. Jeder Eintrag im Array enthält eine `url`, einen `type` und den `content_type` (MIME-Typ) der Datei. Dies ermöglicht eine einfache und schnelle Integration der visuellen oder auditiven Repräsentationen des Objekts in eigene Anwendungen.


```
{
  "metadata": {
    "@context": "https://api.kulturpool.at/ns/v1/edm.json",
    "id": "https://id.kulturpool.at/b1eaac0f-69a1-4174-8ac6-90a61615d0ae/aggregation",
    "type": "Aggregation",
    "provider": "Kulturpool",
    "dataProvider": "Wienbibliothek im Rathaus",
    "edmRights": "http://creativecommons.org/publicdomain/mark/1.0/",
    "isShownAt": "https://www.digital.wienbibliothek.at/id/475632",
    "isShownBy": {
      "id": "https://www.digital.wienbibliothek.at/download/webcache/1000/475634",
      "type": "WebResource",
      "hasService": {
        "id": "https://www.digital.wienbibliothek.at/i3f/v20/475634",
        "type": "Service",
        "conformsTo": [{ "id": "http://iiif.io/api/image" }],
        "implements": { "id": "http://iiif.io/api/image/2/level2.json" }
      },
      "isReferencedBy": [
        { "id": "https://www.digital.wienbibliothek.at/i3f/v20/475632/manifest" }
      ]
    },
    "hasView": [
      {
        "id": "https://www.digital.wienbibliothek.at/download/pdf/475632",
        "type": "WebResource"
      }
    ],
    "aggregatedCHO": {
      "id": "https://id.kulturpool.at/b1eaac0f-69a1-4174-8ac6-90a61615d0ae/cho",
      "type": "ProvidedCHO",
      "title": [
        "Neueste Nachrichten. Alpenländisches Morgenblatt mit Handels-Zeitung ... Nr. 207/ 2.
```

```
Jahrgang/ 28. Juli 1915"
      ],
      "dcType": ["Plakat", "book", "poster"],
      "issued": ["1915"],
      "edmType": "TEXT",
      "spatial": ["Salzburg <Stadt>"],
      "language": ["ger"],
      "publisher": ["Neueste Nachrichten"],
      "identifier": ["urn:nbn:at:AT-WBR-22887"],
      "extent": ["1 Bogen, 48 x 32 cm"],
      "currentLocation": " ; P-226563 ; PS-A2-0003"
    }
  },
  "media": [
    {
      "url": "https://www.digital.wienbibliothek.at/i3f/v20/475632/manifest",
      "type": "iiif",
      "content_type": "application/json"
    },
    {
      "url": "https://www.digital.wienbibliothek.at/download/pdf/475632",
      "type": "pdf",
      "content_type": "application/pdf"
    }
  ]
}
```


Datenstruktur

Hauptobjekt

Feld	Typ	Beschreibung
`metadata`	object	Ein JSON-LD-Objekt mit den Metadaten gemäß dem Europeana Data Model.
`media`	Media[]	Ein Array von Media-Objekten, die alle zugehörigen Dateien repräsentieren.

Media

Feld	Typ	Beschreibung
`url`	string	Die direkte URL zur Mediendatei oder zum IIIF-Manifest.
`type`	string	Der Typ der Mediendatei. Mögliche Werte: `iiif`, `image`, `video`, `audio`, `3d`, `pdf`, `embed`, `other`.
`content_type`	string	Der MIME-Typ der Ressource (z.B. `application/json`, `image/jpeg`).

Verbindung zur Suche API

In den Ergebnissen der Suche API finden Sie für jedes Objekt das Feld `kp_id`. Der Wert dieses Feldes ist die Kulturpool-ID, die zum Abrufen der detaillierten Informationen über die Objekte API verwendet werden kann.

Fehlerbehandlung

404 - Objekt nicht gefunden

Wird zurückgegeben, wenn keine passende Kulturpool-ID gefunden wurde.

```
{
  "detail": "Error fetching data from upstream API"
}
```

500 - Server-Fehler

Wird bei unerwarteten Fehlern auf dem Server zurückgegeben.

```
{
  "detail": "Error fetching data from upstream API"
}
```

Institutionen API

> Die Institutionen API bietet Zugriff auf Informationen über alle teilnehmenden Museen, Bibliotheken und Kultureinrichtungen im Kulturpool. Sie liefert strukturierte Daten mit Standortinformationen, Kontaktdaten und mehrsprachigen Beschreibungen.

Basis-URL

https://api.kulturpool.at/institutions

Endpunkte

Alle Institutionen abrufen

```
GET /institutions
```

Ruft eine vollständige Liste aller registrierten Institutionen ab.

Antwort


```
{
  "data": [
    {
      "id": 42,
      "name": "Kulturinstitution Musterhausen",
      "intermediate_provider": null,
      "location": {
        "type": "MultiPoint",
        "coordinates": [[10.12345, 52.67890]]
      },
      "web_collection_url": "https://www.kultur-musterhausen.example",
      "website_url": "https://www.kultur-musterhausen.example",
```

```
      "favicon": {
        "id": "12345678-1234-5678-9abc-123456789abc"
      },
      "hero_image": {
        "id": "87654321-8765-4321-9def-987654321fed",
        "title": "Hauptgebäude der Kulturinstitution Musterhausen",
        "author": "Max Mustermann",
        "license": "CC BY-SA",
        "license_url": "https://creativecommons.org/licenses/by-sa/4.0/deed.de"
      },
      "logo": {
        "id": "11111111-2222-3333-4444-555555555555"
      },
      "translations": [
        {
          "place": "Musterhausen",
          "title_official": "Kulturinstitution Musterhausen (GmbH)",
          "title": "Kulturinstitution Musterhausen",
          "slug": "kulturinstitution-musterhausen",
          "summary": "Die Kulturinstitution Musterhausen sammelt und bewahrt bedeutende Werke
der regionalen Kunst und Kultur.",
          "content": "<p>Die Kulturinstitution wurde 1950 gegründet und ist seitdem ein
wichtiger Ort für Kunst und Kultur in der Region.</p>",
          "languages_code": "de"
        }
      ]
    }
  ]
}
```


Einzelne Institution abrufen

```
GET /institutions/{institution_id}
```

Ruft detaillierte Informationen zu einer spezifischen Institution ab.

Parameter

Parameter	Typ	Beschreibung

`institution_id`	integer	Die eindeutige ID der Institution (Pfad-Parameter)

Beispiel

```
curl -X GET "https://api.kulturpool.at/institutions/42"
```

Antwort


```
{
  "data": {
    "id": 42,
    "name": "Kulturinstitution Musterhausen",
    "intermediate_provider": null,
    "location": {
      "type": "MultiPoint",
      "coordinates": [[10.12345, 52.67890]]
    },
    "web_collection_url": "https://www.kultur-musterhausen.example",
    "website_url": "https://www.kultur-musterhausen.example",
    "favicon": {
      "id": "12345678-1234-5678-9abc-123456789abc"
    },
    "hero_image": {
      "id": "87654321-8765-4321-9def-987654321fed",
      "title": "Hauptgebäude der Kulturinstitution Musterhausen",
      "author": "Max Mustermann",
      "license": "CC BY-SA",
      "license_url": "https://creativecommons.org/licenses/by-sa/4.0/deed.de"
    },
    "logo": {
      "id": "11111111-2222-3333-4444-555555555555"
    },
    "translations": [
      {
        "place": "Musterhausen",
        "title_official": "Kulturinstitution Musterhausen (GmbH)",
        "title": "Kulturinstitution Musterhausen",
        "slug": "kulturinstitution-musterhausen",
        "summary": "Die Kulturinstitution Musterhausen sammelt und bewahrt bedeutende Werke
```

```
der regionalen Kunst und Kultur.",
        "content": "<p>Die Kulturinstitution wurde 1950 gegründet und ist seitdem ein
wichtiger Ort für Kunst und Kultur in der Region.</p>",
        "languages_code": "de"
      }
    ]
  }
}
```


Datenstruktur

Institution

Feld	Typ	Beschreibung
`id`	integer	Eindeutige ID der Institution
`name`	string	Offizieller Name der Institution
`intermediate_provider`	string/null	Zwischenanbieter (falls vorhanden)
`location`	Location	Geografische Koordinaten
`web_collection_url`	string	URL zur digitalen Sammlung
`website_url`	string	Offizielle Website
`favicon`	File	Favicon der Institution
`hero_image`	HeroImage	Hauptbild mit Metadaten
`logo`	File	Logo der Institution
`translations`	Translation[]	Mehrsprachige Beschreibungen

Location

Feld	Typ	Beschreibung
`type`	string	GeoJSON-Typ (meist "MultiPoint")
`coordinates`	number[][]	Array von Koordinaten-Paaren [Längengrad, Breitengrad]

HeroImage

Feld	Typ	Beschreibung
`id`	string	Asset-ID des Bildes
`title`	string	Bildtitel
`author`	string	Fotograf/Künstler
`license`	string	Lizenztyp (z.B. "CC BY-SA")
`license_url`	string	URL zur vollständigen Lizenz

Translation

Feld	Typ	Beschreibung
`place`	string	Ortsname
`title_official`	string	Offizieller Titel mit Rechtsform
`title`	string	Anzeigename
`slug`	string	URL-freundlicher Bezeichner
`summary`	string	Kurzbeschreibung
`content`	string	Ausführliche Beschreibung (HTML)
`languages_code`	string	Sprachcode (ISO 639-1)

File

Feld	Typ	Beschreibung
`id`	string	UUID des Assets im CMS

Beispiel-Code

JavaScript/Fetch

```
// Alle Institutionen abrufen
const response = await fetch('https://api.kulturpool.at/institutions');
const { data: institutions } = await response.json();

// Einzelne Institution abrufen
const institutionResponse = await fetch('https://api.kulturpool.at/institutions/42');
```

```
const { data: institution } = await institutionResponse.json();

console.log(`Institution: ${institution.name}`);
console.log(`Ort: ${institution.translations[0]?.place}`);
```

Python/requests

```
import requests

# Alle Institutionen abrufen
response = requests.get('https://api.kulturpool.at/institutions')
institutions = response.json()['data']

# Einzelne Institution abrufen
institution_response = requests.get('https://api.kulturpool.at/institutions/42')
institution = institution_response.json()['data']

print(f"Institution: {institution['name']}")
print(f"Ort: {institution['translations'][0]['place'] if institution['translations'] else
'N/A'}")
```

cURL

```
# Alle Institutionen abrufen
curl -X GET "https://api.kulturpool.at/institutions" \
  -H "Accept: application/json"

# Einzelne Institution abrufen
curl -X GET "https://api.kulturpool.at/institutions/42" \
  -H "Accept: application/json"
```

Anwendungsfälle

Institutionen-Verzeichnis erstellen

```
async function createInstitutionDirectory() {
  const response = await fetch('https://api.kulturpool.at/institutions');
  const { data: institutions } = await response.json();

  const directory = institutions.map(institution => ({
    id: institution.id,
    name: institution.name,
    place: institution.translations[0]?.place,
    website: institution.website_url,
    collection: institution.web_collection_url
  }));

  return directory;
}
```

Karte mit Institutionen

```
async function getInstitutionLocations() {
  const response = await fetch('https://api.kulturpool.at/institutions');
  const { data: institutions } = await response.json();

  return institutions
    .filter(institution => institution.location)
    .map(institution => ({
      id: institution.id,
      name: institution.name,
      coordinates: institution.location.coordinates[0], // [lng, lat]
      title: institution.translations[0]?.title
    }));
}
```

Institution mit Assets

```
async function getInstitutionWithImages(institutionId) {
  const response = await fetch(`https://api.kulturpool.at/institutions/${institutionId}`);
  const { data: institution } = await response.json();
```

```
  const enriched = {
    ...institution,
    faviconUrl: institution.favicon ?

`https://api.kulturpool.at/assets/${institution.favicon.id}?format=png&width=32&height=32` :
null,
    logoUrl: institution.logo ?

`https://api.kulturpool.at/assets/${institution.logo.id}?format=png&width=200&height=100&fit=c
ontain` : null,
    heroImageUrl: institution.hero_image ?

`https://api.kulturpool.at/assets/${institution.hero_image.id}?format=webp&width=1200&height=4
00&fit=cover` : null
  };

  return enriched;
}
```

Fehlerbehandlung

404 - Institution nicht gefunden

```
{
  "detail": "Error fetching data from upstream API"
}
```

500 - Server-Fehler

```
{
  "detail": "Error fetching data from upstream API"
}
```

Inhalte API

> Die Content API ermöglicht den Zugriff auf redaktionelle Inhalte des Kulturpools. Sie liefert strukturierte Content-Seiten mit verschiedenen Block-Typen, SEO-Metadaten und mehrsprachigen Übersetzungen für die Website-Darstellung.

Basis-URL

https://api.kulturpool.at/content

Endpunkte

Alle Inhalte abrufen

```
GET /content
```

Ruft alle verfügbaren redaktionellen Inhalte vom CMS ab.

Antwort


```
{
  "data": [
    {
      "id": 1,
      "status": "published",
      "sort": 1,
      "date_published": "2024-01-15T09:00:00Z",
      "color_text": "#333333",
      "color_background": "#ffffff",
      "seo": {
        "id": 1,
        "title": "Willkommen im Kulturpool - Digitale Kunst entdecken",
```

```
        "meta_description": "Entdecken Sie die Vielfalt österreichischer Kultur im digitalen
Raum",
        "canonical_url": "https://kulturpool.at/willkommen"
      },
      "category": {
        "id": 1,
        "sort": 1,
        "translations": []
      },
      "translations": [
        {
          "id": 1,
          "languages_code": "de",
          "title": "Willkommen im Kulturpool",
          "slug": "willkommen",
          "summary": "Entdecken Sie die Vielfalt österreichischer Kultur",
          "items": [
            {
              "id": 1,
              "collection": "block_text",
              "item": {
                "id": 1,
                "type": "hero",
                "heading": "Willkommen im Kulturpool",
                "body": "<p>Hier finden Sie digitalisierte Kunst und Kultur aus
österreichischen Institutionen.</p>",
                "heading_tag": "h1",
                "is_enabled": true
              }
            }
          ]
        }
      ]
    }
  ]
}
```


Einzelnen Content-Eintrag abrufen

```
GET /content/{content_id}
```

Ruft einen spezifischen Content-Eintrag anhand seiner ID ab.

Parameter

Parameter	Typ	Beschreibung
`content_id`	integer	Die eindeutige ID des Content-Eintrags (Pfad-Parameter)

Beispiel

```
curl -X GET "https://api.kulturpool.at/content/1"
```

Antwort


```
{
  "data": {
    "id": 1,
    "status": "published",
    "sort": 1,
    "date_published": "2024-01-15T09:00:00Z",
    "color_text": "#333333",
    "color_background": "#ffffff",
    "seo": {
      "id": 1,
      "title": "Willkommen im Kulturpool - Digitale Kunst entdecken",
      "meta_description": "Entdecken Sie die Vielfalt österreichischer Kultur im digitalen
Raum",
      "canonical_url": "https://kulturpool.at/willkommen"
    },
    "category": {
      "id": 1,
      "sort": 1,
      "translations": []
    },
    "translations": [
      {
        "id": 1,
        "languages_code": "de",
        "title": "Willkommen im Kulturpool",
```

```
        "slug": "willkommen",
        "summary": "Entdecken Sie die Vielfalt österreichischer Kultur",
        "items": [
          {
            "id": 1,
            "collection": "block_text",
            "item": {
              "id": 1,
              "type": "hero",
              "heading": "Willkommen im Kulturpool",
              "body": "<p>Hier finden Sie digitalisierte Kunst und Kultur aus österreichischen
Institutionen.</p>",
              "heading_tag": "h1",
              "is_enabled": true
            }
          }
        ]
      }
    ]
  }
}
```


Datenstruktur

Content

Feld	Typ	Beschreibung
`id`	integer	Eindeutige ID des Content-Eintrags
`status`	string	Veröffentlichungsstatus ("published", "draft")
`sort`	integer	Sortierreihenfolge
`date_published`	string	Veröffentlichungsdatum (ISO 8601)
`color_text`	string	Textfarbe (Hex-Code)
`color_background`	string	Hintergrundfarbe (Hex-Code)
`seo`	SEO	SEO-Metadaten

Feld	Typ	Beschreibung
`category`	ContentCategory	Zugeordnete Kategorie
`translations`	ContentTranslation[]	Mehrsprachige Versionen

SEO

Feld	Typ	Beschreibung
`id`	integer	SEO-Eintrag ID
`title`	string	Meta-Titel für Suchmaschinen
`meta_description`	string	Meta-Beschreibung
`canonical_url`	string	Kanonische URL
`no_index`	boolean	Suchmaschinenindexierung verhindern
`no_follow`	boolean	Link-Verfolgung verhindern
`og_image`	File	Open Graph Bild

ContentTranslation

Feld	Typ	Beschreibung
`id`	integer	Übersetzungs-ID
`languages_code`	string	Sprachcode (ISO 639-1)
`title`	string	Seitentitel
`slug`	string	URL-Slug
`summary`	string	Kurzzusammenfassung
`items`	ContentTranslationsItem[]	Content-Blöcke

ContentTranslationsItem

Feld	Typ	Beschreibung
`id`	integer	Item-ID
`collection`	string	Block-Typ ("block_text", "block_slider", etc.)
`item`	ContentItem	Der eigentliche Content-Block
`sort`	integer	Sortierreihenfolge

Content-Block-Typen

BlockText

Textblöcke für Überschriften, Fließtext und Hero-Bereiche.

Feld	Typ	Beschreibung
`id`	integer	Block-ID
`type`	string	Block-Untertyp ("hero", "text", "quote")
`heading`	string	Überschrift
`body`	string	Haupttext (HTML)
`summary`	string	Zusammenfassung
`heading_tag`	string	HTML-Tag für Überschrift ("h1", "h2", etc.)
`heading_size`	string	Überschriftsgröße
`is_enabled`	boolean	Block ist aktiv

BlockSlider

Bildergalerien und Slider-Komponenten.

Feld	Typ	Beschreibung
`id`	integer	Block-ID
`type`	string	Slider-Typ
`aspect_ratio`	string	Seitenverhältnis (z.B. "16:9")
`row_mode`	boolean	Reihen-Modus aktiviert
`pick_items`	integer	Anzahl anzuzeigender Items
`randomize`	boolean	Zufällige Reihenfolge
`items`	object	Slider-Inhalte

BlockLinks

Linksammlungen und Navigationsblöcke.

Feld	Typ	Beschreibung
`id`	integer	Block-ID

Feld	Typ	Beschreibung
`heading`	string	Block-Überschrift
`type`	string	Link-Block-Typ
`summary`	string	Beschreibung
`item`	object	Link-Daten

BlockEmbed

Eingebettete Inhalte wie Videos oder externe Widgets.

Feld	Typ	Beschreibung
`id`	integer	Block-ID
`type`	string	Embed-Typ
`link`	string	URL des eingebetteten Inhalts
`code`	string	Embed-Code
`aspect_ratio`	string	Seitenverhältnis
`max_width`	integer	Maximale Breite in Pixeln
`file`	File	Eingebettete Datei

BlockList

Listen-Darstellungen für Sammlungen und Aufzählungen.

Feld	Typ	Beschreibung
`id`	integer	Block-ID
`type`	string	Listen-Typ
`heading`	string	Listen-Überschrift
`summary`	string	Listen-Beschreibung
`collection`	string	Quell-Collection
`mode`	string	Darstellungsmodus
`pick_items`	integer	Anzahl Items
`show_filter`	boolean	Filter anzeigen
`row_mode`	boolean	Reihen-Layout
`randomize`	boolean	Zufällige Sortierung
`items`	object[]	Listen-Items

Beispiel-Code

JavaScript/Fetch

```
// Alle Inhalte abrufen
const response = await fetch('https://api.kulturpool.at/content');
const { data: contentPages } = await response.json();

// Einzelne Seite abrufen
const pageResponse = await fetch('https://api.kulturpool.at/content/1');
const { data: page } = await pageResponse.json();

// Deutsche Übersetzung extrahieren
const germanTranslation = page.translations.find(t => t.languages_code === 'de');
console.log(`Titel: ${germanTranslation.title}`);
console.log(`Slug: ${germanTranslation.slug}`);
```

Python/requests

```
import requests

# Alle Inhalte abrufen
response = requests.get('https://api.kulturpool.at/content')
content_pages = response.json()['data']

# Einzelne Seite abrufen
page_response = requests.get('https://api.kulturpool.at/content/1')
page = page_response.json()['data']

# Deutsche Übersetzung finden
german_translation = next(
    (t for t in page['translations'] if t['languages_code'] == 'de'),
    None
)

if german_translation:
```

```
    print(f"Titel: {german_translation['title']}")
    print(f"Slug: {german_translation['slug']}")
```

Content-Blöcke verarbeiten

```
function processContentBlocks(translation) {
  return translation.items.map(item => {
    const { collection, item: block } = item;

    switch (collection) {
      case 'block_text':
        return {
          type: 'text',
          heading: block.heading,
          body: block.body,
          tag: block.heading_tag || 'h2'
        };

      case 'block_slider':
        return {
          type: 'slider',
          aspectRatio: block.aspect_ratio,
          items: block.items || []
        };

      case 'block_embed':
        return {
          type: 'embed',
          url: block.link,
          code: block.code,
          maxWidth: block.max_width
        };

      default:
        return { type: 'unknown', data: block };
    }
  });
}
```

Anwendungsfälle

Dynamische Website-Seiten

```
async function renderContentPage(slug) {
  // Alle Seiten abrufen und nach Slug filtern
  const response = await fetch('https://api.kulturpool.at/content');
  const { data: pages } = await response.json();

  const page = pages.find(page =>
    page.translations.some(t => t.slug === slug)
  );

  if (!page) {
    throw new Error(`Seite mit Slug "${slug}" nicht gefunden`);
  }

  const translation = page.translations.find(t => t.languages_code === 'de');

  return {
    title: translation.title,
    seoTitle: page.seo?.title || translation.title,
    metaDescription: page.seo?.meta_description || translation.summary,
    canonicalUrl: page.seo?.canonical_url,
    blocks: processContentBlocks(translation),
    theme: {
      textColor: page.color_text,
      backgroundColor: page.color_background
    }
  };
}
```

SEO-Metadaten extrahieren

```
function extractSEOMetadata(page, language = 'de') {
  const translation = page.translations.find(t => t.languages_code === language);

  return {
    title: page.seo?.title || translation?.title,
    description: page.seo?.meta_description || translation?.summary,
```

```
    canonical: page.seo?.canonical_url,
    noIndex: page.seo?.no_index || false,
    noFollow: page.seo?.no_follow || false,
    ogImage: page.seo?.og_image?.id ?

`https://api.kulturpool.at/assets/${page.seo.og_image.id}?format=jpg&width=1200&height=630&fit
=cover` : null
  };
}
```

Content-Management für Headless CMS

```
class ContentManager {
  constructor(baseUrl = 'https://api.kulturpool.at') {
    this.baseUrl = baseUrl;
    this.cache = new Map();
  }

  async getPage(id) {
    if (this.cache.has(id)) {
      return this.cache.get(id);
    }

    const response = await fetch(`${this.baseUrl}/content/${id}`);
    const { data: page } = await response.json();

    this.cache.set(id, page);
    return page;
  }

  async getPageBySlug(slug, language = 'de') {
    const response = await fetch(`${this.baseUrl}/content`);
    const { data: pages } = await response.json();

    return pages.find(page =>
      page.translations.some(t =>
        t.slug === slug && t.languages_code === language
      )
    );
  }
```

```
  getBlocks(page, language = 'de') {
    const translation = page.translations.find(t => t.languages_code === language);
    return translation?.items || [];
  }

  getActiveBlocks(page, language = 'de') {
    return this.getBlocks(page, language)
      .filter(item => item.item.is_enabled !== false)
      .sort((a, b) => (a.sort || 0) - (b.sort || 0));
  }
}
```

Mehrsprachige Navigation

```
async function buildNavigation() {
  const response = await fetch('https://api.kulturpool.at/content');
  const { data: pages } = await response.json();

  const navigation = {};

  pages.forEach(page => {
    page.translations.forEach(translation => {
      const { languages_code, title, slug } = translation;

      if (!navigation[languages_code]) {
        navigation[languages_code] = [];
      }

      navigation[languages_code].push({
        title,
        slug,
        url: `/${languages_code}/${slug}`,
        sort: page.sort || 999
      });
    });
  });

  // Sortieren nach sort-Wert
```

```
  Object.keys(navigation).forEach(lang => {
    navigation[lang].sort((a, b) => a.sort - b.sort);
  });

  return navigation;
}
```

Fehlerbehandlung

404 - Content nicht gefunden

```
{
  "detail": "Content not found"
}
```

500 - Validierungsfehler

```
{
  "detail": "Response validation error for content 1: ..."
}
```

Assets API

> Die Assets API ermöglicht den Zugriff auf Bilder und andere Medien-Dateien mit dynamischen Transformationsparametern. Sie bietet optimierte Bilddelivery mit Format-Konvertierung, Größenanpassung und verschiedenen Anpassungsoptionen.

Basis-URL

https://api.kulturpool.at/assets

Endpunkt

Asset abrufen

```
GET /assets/{asset_id}
```

Ruft ein Asset vom CMS ab und wendet die angegebenen Transformationsparameter an.

Parameter

Parameter	Typ	Beschreibung	Beispiel
`asset_id`	string	Asset-ID (UUID) aus dem CMS	`8e1cf6a9-d1d6-4090-9a36-303f1f57a016`
`width`	integer	Breite in Pixeln	`1500`
`height`	integer	Höhe in Pixeln	`700`
`quality`	integer	Bildqualität (1-100)	`80`
`withoutEnlargement`	boolean	Automatische Vergrößerung deaktivieren	`true`
`format`	string	Ausgabeformat	`webp`, `jpg`, `png`, `avif`, `tiff`, `auto`
`fit`	string	Anpassungsmodus	`cover`, `contain`, `inside`, `outside`

Beispiele

```
# Einfacher Asset-Abruf
curl "https://api.kulturpool.at/assets/8e1cf6a9-d1d6-4090-9a36-303f1f57a016"

# Optimiertes Thumbnail
curl "https://api.kulturpool.at/assets/8e1cf6a9-d1d6-4090-9a36-
303f1f57a016?format=webp&width=300&height=200&fit=cover&quality=85"

# Hero-Image für moderne Browser
curl "https://api.kulturpool.at/assets/8e1cf6a9-d1d6-4090-9a36-
303f1f57a016?format=avif&width=1920&height=800&fit=cover&withoutEnlargement=true"
```

Parameter im Detail

Format-Optionen

Format	Beschreibung	Verwendung
`auto`	Automatische Format-Auswahl (Standard)	Versucht WebP/AVIF für moderne Browser, sonst JPEG
`webp`	WebP-Format	Gute Kompression, breite Browser-Unterstützung
`avif`	AVIF-Format	Beste Kompression, neuere Browser
`jpg`	JPEG-Format	Universelle Kompatibilität
`png`	PNG-Format	Für Transparenz und verlustfreie Kompression
`tiff`	TIFF-Format	Hochqualitative Archivierung

Anpassungsmodi (fit)

Modus	Beschreibung	Verwendung
`cover`	Bild vollständig in Dimensionen einpassen (Standard)	Hero-Images, Thumbnails
`contain`	Bild mit Letterboxing in Dimensionen einpassen	Logos, vollständige Bilder
`inside`	So groß wie möglich innerhalb der Dimensionen	Responsive Bilder
`outside`	So klein wie möglich innerhalb/außerhalb der Dimensionen	Spezielle Layouts

Qualitätseinstellungen

Qualität	Beschreibung	Dateigröße	Verwendung
`95-100`	Höchste Qualität	Sehr groß	Druckvorbereitung
`85-95`	Hohe Qualität	Groß	Hero-Images
`75-85`	Gute Qualität	Mittel	Standard-Bilder
`60-75`	Moderate Qualität	Klein	Thumbnails
`40-60`	Niedrige Qualität	Sehr klein	Vorschaubilder

Antwort

Die API gibt eine **HTTP 302 Redirect** zum optimierten Asset im CMS zurück:

```
HTTP/1.1 302 Found
Location: https://cms.kulturpool.at/assets/8e1cf6a9-d1d6-4090-9a36-
303f1f57a016/?format=webp&width=1500&height=700&fit=cover
Content-Type: application/json
```

Beispiel-Code

Responsive Bilder in HTML

```
<picture>
  <source
    srcset="https://api.kulturpool.at/assets/8e1cf6a9-d1d6-4090-9a36-
303f1f57a016?format=avif&width=1920&height=800&fit=cover"
    type="image/avif"
    media="(min-width: 1200px)">
  <source
    srcset="https://api.kulturpool.at/assets/8e1cf6a9-d1d6-4090-9a36-
303f1f57a016?format=webp&width=1200&height=600&fit=cover"
    type="image/webp"
    media="(min-width: 768px)">
```

```
  <source
    srcset="https://api.kulturpool.at/assets/8e1cf6a9-d1d6-4090-9a36-
303f1f57a016?format=webp&width=800&height=400&fit=cover"
    type="image/webp">
  <img
    src="https://api.kulturpool.at/assets/8e1cf6a9-d1d6-4090-9a36-
303f1f57a016?format=jpg&width=800&height=400&fit=cover"
    alt="Beschreibung"
    loading="lazy">
</picture>
```

JavaScript Asset-URL Generator

```
class AssetUrlBuilder {
  constructor(baseUrl = 'https://api.kulturpool.at/assets') {
    this.baseUrl = baseUrl;
  }

  build(assetId, options = {}) {
    const {
      width,
      height,
      quality = 85,
      format = 'auto',
      fit = 'cover',
      withoutEnlargement = false
    } = options;

    const params = new URLSearchParams();

    if (width) params.set('width', width);
    if (height) params.set('height', height);
    if (quality !== 85) params.set('quality', quality);
    if (format !== 'auto') params.set('format', format);
    if (fit !== 'cover') params.set('fit', fit);
    if (withoutEnlargement) params.set('withoutEnlargement', 'true');

    const queryString = params.toString();
```

```
    return `${this.baseUrl}/${assetId}${queryString ? `?${queryString}` : ''}`;
  }

  // Vordefinierte Größen
  thumbnail(assetId, size = 150) {
    return this.build(assetId, {
      width: size,
      height: size,
      format: 'webp',
      quality: 80
    });
  }

  hero(assetId, width = 1920, height = 800) {
    return this.build(assetId, {
      width,
      height,
      format: 'webp',
      quality: 85,
      fit: 'cover'
    });
  }

  logo(assetId, maxWidth = 200) {
    return this.build(assetId, {
      width: maxWidth,
      format: 'png',
      fit: 'contain',
      withoutEnlargement: true
    });
  }
}

// Verwendung
const assetBuilder = new AssetUrlBuilder();
const thumbnailUrl = assetBuilder.thumbnail('8e1cf6a9-d1d6-4090-9a36-303f1f57a016');
const heroUrl = assetBuilder.hero('8e1cf6a9-d1d6-4090-9a36-303f1f57a016', 1500, 600);
```

Python Asset-Manager

```python
from urllib.parse import urlencode

class AssetManager:
    def __init__(self, base_url='https://api.kulturpool.at/assets'):
        self.base_url = base_url

    def build_url(self, asset_id, **kwargs):
        """Asset-URL mit Parametern erstellen"""
        params = {k: v for k, v in kwargs.items() if v is not None}

        # Boolean-Werte in Strings konvertieren
        for key, value in params.items():
            if isinstance(value, bool):
                params[key] = str(value).lower()

        url = f"{self.base_url}/{asset_id}"
        if params:
            url += f"?{urlencode(params)}"

        return url

    def get_responsive_urls(self, asset_id, format_type='webp'):
        """Responsive Bild-URLs für verschiedene Bildschirmgrößen"""
        return {
            'mobile': self.build_url(asset_id, width=480, height=320, format=format_type,
fit='cover'),
            'tablet': self.build_url(asset_id, width=768, height=512, format=format_type,
fit='cover'),
            'desktop': self.build_url(asset_id, width=1200, height=800, format=format_type,
fit='cover'),
            'large': self.build_url(asset_id, width=1920, height=1080, format=format_type,
fit='cover')
        }

    def get_optimized_thumbnail(self, asset_id, size=300):
        """Optimiertes Thumbnail"""
        return self.build_url(
            asset_id,
```

```
            width=size,
            height=size,
            format='webp',
            quality=80,
            fit='cover'
        )

# Verwendung
asset_manager = AssetManager()
asset_id = '8e1cf6a9-d1d6-4090-9a36-303f1f57a016'

# Responsive URLs
responsive_urls = asset_manager.get_responsive_urls(asset_id)
print(f"Mobile: {responsive_urls['mobile']}")

# Thumbnail
thumbnail_url = asset_manager.get_optimized_thumbnail(asset_id, 150)
print(f"Thumbnail: {thumbnail_url}")
```

Anwendungsfälle

Content Management System Integration

```
// Asset-URLs für Institution Hero-Images generieren
async function getInstitutionWithOptimizedImages(institutionId) {
  const response = await fetch(`https://api.kulturpool.at/institutions/${institutionId}`);
  const { data: institution } = await response.json();

  const assetBuilder = new AssetUrlBuilder();

  return {
    ...institution,
    optimizedImages: {
      favicon: institution.favicon?.id ?
        assetBuilder.build(institution.favicon.id, {
          width: 32,
```

```
          height: 32,
          format: 'png'
        }) : null,
      logo: institution.logo?.id ?
        assetBuilder.logo(institution.logo.id, 200) : null,
      hero: institution.hero_image?.id ?
        assetBuilder.hero(institution.hero_image.id) : null,
      heroMobile: institution.hero_image?.id ?
        assetBuilder.build(institution.hero_image.id, {
          width: 768,
          height: 400,
          format: 'webp',
          fit: 'cover'
        }) : null
    }
  };
}
```

Bildergalerie mit Lazy Loading

```
class ImageGallery {
  constructor(containerId) {
    this.container = document.getElementById(containerId);
    this.assetBuilder = new AssetUrlBuilder();
    this.observer = new IntersectionObserver(this.handleIntersection.bind(this));
  }

  addImage(assetId, alt = '') {
    const imageContainer = document.createElement('div');
    imageContainer.className = 'gallery-item';
    imageContainer.dataset.assetId = assetId;
    imageContainer.dataset.alt = alt;

    // Placeholder
    const placeholder = document.createElement('div');
    placeholder.className = 'image-placeholder';
    placeholder.style.backgroundColor = '#f0f0f0';
    placeholder.style.aspectRatio = '16/9';
```

```
  imageContainer.appendChild(placeholder);
  this.container.appendChild(imageContainer);

  // Für Lazy Loading beobachten
  this.observer.observe(imageContainer);
}

handleIntersection(entries) {
  entries.forEach(entry => {
    if (entry.isIntersecting) {
      this.loadImage(entry.target);
      this.observer.unobserve(entry.target);
    }
  });
}

loadImage(container) {
  const assetId = container.dataset.assetId;
  const alt = container.dataset.alt;

  const picture = document.createElement('picture');

  // WebP für moderne Browser
  const sourceWebP = document.createElement('source');
  sourceWebP.srcset = this.assetBuilder.build(assetId, {
    width: 800,
    height: 450,
    format: 'webp',
    fit: 'cover'
  });
  sourceWebP.type = 'image/webp';

  // JPEG Fallback
  const img = document.createElement('img');
  img.src = this.assetBuilder.build(assetId, {
    width: 800,
    height: 450,
    format: 'jpg',
```

```
      fit: 'cover'
    });
    img.alt = alt;
    img.loading = 'lazy';

    picture.appendChild(sourceWebP);
    picture.appendChild(img);

    container.innerHTML = '';
    container.appendChild(picture);
  }
}
```

CDN-Integration und Caching

```
class CachedAssetManager {
  constructor() {
    this.assetBuilder = new AssetUrlBuilder();
    this.cache = new Map();
    this.cacheTimeout = 5 * 60 * 1000; // 5 Minuten
  }

  async preloadAsset(assetId, options = {}) {
    const url = this.assetBuilder.build(assetId, options);
    const cacheKey = `${assetId}_${JSON.stringify(options)}`;

    if (this.cache.has(cacheKey)) {
      const cached = this.cache.get(cacheKey);
      if (Date.now() - cached.timestamp < this.cacheTimeout) {
        return cached.blob;
      }
    }

    try {
      const response = await fetch(url);
      const blob = await response.blob();

      this.cache.set(cacheKey, {
```

```
        blob,
        timestamp: Date.now()
      });

      return blob;
    } catch (error) {
      console.error(`Fehler beim Laden von Asset ${assetId}:`, error);
      return null;
    }
  }

  createObjectUrl(assetId, options = {}) {
    const cacheKey = `${assetId}_${JSON.stringify(options)}`;
    const cached = this.cache.get(cacheKey);

    if (cached?.blob) {
      return URL.createObjectURL(cached.blob);
    }

    // Fallback: Direkte URL zurückgeben
    return this.assetBuilder.build(assetId, options);
  }

  // Cache aufräumen
  clearExpiredCache() {
    const now = Date.now();
    for (const [key, value] of this.cache.entries()) {
      if (now - value.timestamp >= this.cacheTimeout) {
        this.cache.delete(key);
      }
    }
  }
}
```

SEO-optimierte Asset-Integration

```
function generateSEOOptimizedAssetUrls(assetId, title = '') {
  const assetBuilder = new AssetUrlBuilder();

  return {
    // Open Graph Image
    ogImage: assetBuilder.build(assetId, {
      width: 1200,
      height: 630,
      format: 'jpg',
      fit: 'cover',
      quality: 85
    }),

    // Twitter Card Image
    twitterImage: assetBuilder.build(assetId, {
      width: 1024,
      height: 512,
      format: 'jpg',
      fit: 'cover',
      quality: 85
    }),

    // JSON-LD Structured Data Image
    structuredDataImage: assetBuilder.build(assetId, {
      width: 1200,
      height: 800,
      format: 'jpg',
      fit: 'cover',
      quality: 90
    }),

    // Apple Touch Icon
    appleTouchIcon: assetBuilder.build(assetId, {
      width: 180,
      height: 180,
      format: 'png',
      fit: 'cover'
    })
```

```
  };
}
```

Performance-Optimierung

Bildformate nach Browser-Unterstützung

```
function getBestSupportedFormat() {
  // Feature Detection für moderne Bildformate
  const canvas = document.createElement('canvas');
  const ctx = canvas.getContext('2d');

  // AVIF Support
  if (canvas.toDataURL('image/avif').indexOf('data:image/avif') === 0) {
    return 'avif';
  }

  // WebP Support
  if (canvas.toDataURL('image/webp').indexOf('data:image/webp') === 0) {
    return 'webp';
  }

  return 'jpg'; // Fallback
}

// Automatische Format-Auswahl
const optimalFormat = getBestSupportedFormat();
const optimizedUrl = assetBuilder.build(assetId, {
  width: 800,
  height: 600,
  format: optimalFormat
});
```

Batch-Preloading für bessere Performance

```
async function preloadAssetBatch(assetIds, options = {}) {
  const promises = assetIds.map(async (assetId) => {
    const link = document.createElement('link');
    link.rel = 'preload';
    link.as = 'image';
    link.href = assetBuilder.build(assetId, options);

    document.head.appendChild(link);

    return new Promise((resolve) => {
      link.onload = resolve;
      link.onerror = resolve; // Auch bei Fehlern fortfahren
    });
  });

  await Promise.allSettled(promises);
}

// Verwendung für kritische Bilder
const criticalAssets = ['asset-1', 'asset-2', 'asset-3'];
await preloadAssetBatch(criticalAssets, {
  width: 1200,
  height: 600,
  format: 'webp'
});
```

Fehlerbehandlung

404 - Asset nicht gefunden

```
{
  "detail": "Asset mit ID 'invalid-id' wurde nicht gefunden"
}
```

500 - CMS-Fehler

```
{
  "detail": "Fehler beim Abrufen des Assets vom CMS"
```

```
}
```

Fehlerbehandlung im Frontend

```
function createAssetWithFallback(assetId, options = {}, fallbackSrc = '/placeholder.jpg') {
  const img = new Image();
  const primaryUrl = assetBuilder.build(assetId, options);

  img.onerror = () => {
    img.src = fallbackSrc;
  };

  img.src = primaryUrl;
  return img;
}
```

OAI-PMH

> Der `/oai` Endpunkt stellt einen OAI-PMH (Open Archives Initiative Protocol for Metadata Harvesting) Server zur Verfügung, der einen Großteil der Digitalisate aus dem Kulturpool bereitstellt.

OAI-PMH ist ein Standard-Protokoll für das Harvesting von Metadaten aus digitalen Archiven und Repositorien, das eine standardisierte Schnittstelle für die Übertragung von Metadaten zwischen verschiedenen Systemen bietet.

Basis-URL

https://api.kulturpool.at/oai

OAI-PMH Verben

Das OAI-PMH Protokoll unterstützt sechs Standard-Verben (Befehle):

Identify

Gibt grundlegende Informationen über das Repository zurück.

Parameter:

- **verb** (string): `Identify` (Pflicht)

Beispiel:

```
GET /oai?verb=Identify
```

ListMetadataFormats

Listet alle verfügbaren Metadatenformate auf.

Parameter:

- **verb** (string): `ListMetadataFormats` (Pflicht)
- **identifier** (string): Spezifischer Datensatz (optional)

Beispiel:

```
GET /oai?verb=ListMetadataFormats
```

ListSets

Listet alle verfügbaren Sets (Sammlungen) auf.

Parameter:

- **verb** (string): `ListSets` (Pflicht)
- **resumptionToken** (string): Token für paginierte Ergebnisse (optional)

Beispiel:

```
GET /oai?verb=ListSets
```

ListIdentifiers

Listet die Identifikatoren aller verfügbaren Datensätze auf.

Parameter:

- **verb** (string): `ListIdentifiers` (Pflicht)
- **metadataPrefix** (string): Format der Metadaten (Pflicht)
- **from** (string): Datum ab wann (optional, Format: YYYY-MM-DD)
- **until** (string): Datum bis wann (optional, Format: YYYY-MM-DD)
- **set** (string): Spezifisches Set/Sammlung (optional)
- **resumptionToken** (string): Token für paginierte Ergebnisse (optional)

Beispiel:

```
GET /oai?verb=ListIdentifiers&metadataPrefix=edm&set=kulturpool-europeana:wien-bibliothek
```

ListRecords

Ruft die Metadaten aller verfügbaren Datensätze ab.

Parameter:

- **verb** (string): `ListRecords` (Pflicht)
- **metadataPrefix** (string): Format der Metadaten (Pflicht)
- **from** (string): Datum ab wann (optional, Format: YYYY-MM-DD)
- **until** (string): Datum bis wann (optional, Format: YYYY-MM-DD)
- **set** (string): Spezifisches Set/Sammlung (optional)
- **resumptionToken** (string): Token für paginierte Ergebnisse (optional)

Beispiel:

```
GET /oai?verb=ListRecords&metadataPrefix=edm&set=kulturpool-europeana:wien-bibliothek
```

GetRecord

Ruft einen einzelnen Datensatz ab.

Parameter:

- **verb** (string): `GetRecord` (Pflicht)
- **identifier** (string): Eindeutige Kennung des gewünschten Datensatzes (Pflicht)
- **metadataPrefix** (string): Format der Metadaten (Pflicht)

Beispiel:

```
GET /oai?verb=GetRecord&metadataPrefix=edm&identifier=https://id.kulturpool.at/b1eaac0f-69a1-4174-8ac6-90a61615d0ae
```

Verfügbare Metadatenformate

EDM (Europeana Data Model)

- **metadataPrefix**: `edm`
- **Schema**: Europeana Data Model
- **Namespace**: http://www.europeana.eu/schemas/edm/

Beispielanfragen

Repository-Informationen abrufen

```
GET /oai?verb=Identify
```

Alle verfügbaren Sets anzeigen

```
GET /oai?verb=ListSets
```

Identifikatoren aus der Wienbibliothek

```
GET /oai?verb=ListIdentifiers&metadataPrefix=edm&set=kulturpool-europeana:wien-bibliothek
```

Vollständige Datensätze aus der Wienbibliothek

```
GET /oai?verb=ListRecords&metadataPrefix=edm&set=kulturpool-europeana:wien-bibliothek
```

Spezifischen Datensatz abrufen

```
GET /oai?verb=GetRecord&metadataPrefix=edm&identifier=https://id.kulturpool.at/b1eaac0f-69a1-4174-8ac6-90a61615d0ae
```

Datensätze aus einem bestimmten Zeitraum

```
GET /oai?verb=ListRecords&metadataPrefix=edm&from=2024-01-01&until=2024-12-31
```

Antwortformat

Alle OAI-PMH Antworten sind im XML-Format strukturiert:

Erfolgreiche Antwort (GetRecord Beispiel)

```
<?xml version="1.0" encoding="UTF-8"?>
<OAI-PMH xmlns="http://www.openarchives.org/OAI/2.0/"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/
         http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd">
  <responseDate>2025-08-05T10:30:00Z</responseDate>
  <request verb="GetRecord" metadataPrefix="edm"
           identifier="https://id.kulturpool.at/b1eaac0f-69a1-4174-8ac6-90a61615d0ae">
    https://api.kulturpool.at/oai
  </request>
  <GetRecord>
    <record>
      <header>
        <identifier>https://id.kulturpool.at/b1eaac0f-69a1-4174-8ac6-90a61615d0ae</identifier>
        <datestamp>2024-12-15T09:22:13Z</datestamp>
        <setSpec>kulturpool-europeana:wien-bibliothek</setSpec>
      </header>
      <metadata>
        <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                 xmlns:edm="http://www.europeana.eu/schemas/edm/"
                 xmlns:dc="http://purl.org/dc/elements/1.1/"
                 xmlns:dcterms="http://purl.org/dc/terms/"
                 xmlns:ore="http://www.openarchives.org/ore/terms/">
          <!-- EDM/RDF Metadaten hier -->
          <edm:ProvidedCHO rdf:about="https://id.kulturpool.at/b1eaac0f-69a1-4174-8ac6-
90a61615d0ae">
            <dc:title>Beispiel Titel</dc:title>
            <dc:creator>Beispiel Künstler</dc:creator>
            <dc:description>Beispiel Beschreibung</dc:description>
            <!-- weitere Metadatenfelder -->
          </edm:ProvidedCHO>
        </rdf:RDF>
      </metadata>
    </record>
  </GetRecord>
</OAI-PMH>
```

Fehlerantwort

```
<?xml version="1.0" encoding="UTF-8"?>
<OAI-PMH xmlns="http://www.openarchives.org/OAI/2.0/">
  <responseDate>2025-08-05T10:30:00Z</responseDate>
  <request>https://api.kulturpool.at/oai</request>
  <error code="badVerb">Illegal OAI verb</error>
</OAI-PMH>
```

Fehlerbehandlung

Das OAI-PMH Protokoll definiert standardisierte Fehlercodes:

- **badArgument**: Ungültiges Argument
- **badResumptionToken**: Ungültiger oder abgelaufener Resumption Token
- **badVerb**: Ungültiges Verb
- **cannotDisseminateFormat**: Nicht unterstütztes Metadatenformat
- **idDoesNotExist**: Datensatz existiert nicht
- **noMetadataFormats**: Keine Metadatenformate verfügbar
- **noRecordsMatch**: Keine Datensätze entsprechen den Kriterien
- **noSetHierarchy**: Repository unterstützt keine Sets

Paginierung

Bei großen Datenmengen verwendet OAI-PMH Resumption Tokens für die Paginierung:

```
<resumptionToken completeListSize="5000" cursor="100">
  token123456789
</resumptionToken>
```

Um die nächste Seite abzurufen:

```
GET /oai?verb=ListRecords&resumptionToken=token123456789
```

Datensets zum Herunterladen

> Der Kulturpool stellt täglich aktualisierte Komplett-Exporte aller Objektdaten zum Download bereit. Diese Datensets ermöglichen die lokale Verarbeitung, Analyse und Wiederverwendung des gesamten Datenbestands für Forschung, eigene Anwendungen oder Archivierungszwecke.

Download-URL

Die tagesaktuellen Datensets sind als komprimiertes TAR-Archiv unter der folgenden, stabilen URL verfügbar:

https://s3.kpool.at/nightly/full_records.tar.gz

Datenformat

1. Archiv (`.tar.gz`)

Der Download ist eine einzelne, mit `gzip` komprimierte TAR-Archivdatei. Sie müssen diese Datei nach dem Herunterladen entpacken, um auf die eigentlichen Datensätze zugreifen zu können.

2. Datendateien (`.jsonl`)

Das entpackte Archiv enthält mehrere Dateien mit der Endung `.jsonl` – für jede datenliefernde Institution eine. Das JSONL-Format (JSON Lines) bedeutet, dass jede Datei aus mehreren Zeilen besteht, wobei **jede Zeile ein vollständiges, eigenständiges JSON-Objekt** darstellt.

3. Datensatz (JSON-LD / EDM)

Jedes JSON-Objekt (also jede Zeile in einer `.jsonl`-Datei) repräsentiert ein einzelnes Kulturgut. Der Datensatz ist im **JSON-LD**-Format strukturiert und folgt dem **Europeana Data Model (EDM)**. Die Struktur entspricht exakt dem `metadata`-Objekt, das auch von der Objekte API ausgeliefert wird.

Struktur eines Datensatzes

Jede Zeile in den `.jsonl`-Dateien ist ein JSON-Objekt, das wie folgt aussehen kann:


```
{
  "@context": {
    "ore": "[http://www.openarchives.org/ore/terms/](http://www.openarchives.org/ore/terms/)",
    "edm": "[http://www.europeana.eu/schemas/edm/](http://www.europeana.eu/schemas/edm/)",
    "dc": "[http://purl.org/dc/elements/1.1/](http://purl.org/dc/elements/1.1/)",
    "...": "..."
  },
  "id": "#belvedere_5420_AGG",
  "type": "Aggregation",
  "aggregatedCHO": {
    "id": "#belvedere_5420",
    "type": "ProvidedCHO",
    "dc:title": [
      {
        "lang": "de",
        "value": "Der Kuss"
      }
    ],
    "dc:creator": [
      {
        "value": "Gustav Klimt"
      }
    ]
  },
  "provider": {
    "id": "[https://www.belvedere.at/](https://www.belvedere.at/)",
    "type": "Agent",
    "dc:title": [
      {
        "lang": "de",
        "value": "Belvedere"
      }
    ]
  },
```

```
  "edm:dataProvider": {
    "id": "[https://www.belvedere.at/](https://www.belvedere.at/)",
    "type": "Agent",
    "dc:title": [
      {
        "lang": "de",
        "value": "Belvedere"
      }
    ]
  },
  "edm:isShownAt": {
    "id": "[https://digital.belvedere.at/objects/5420/der-kuss-
liebespaar](https://digital.belvedere.at/objects/5420/der-kuss-liebespaar)",
    "type": "WebResource"
  },
  "edm:isShownBy": {
    "id":
"[https://digital.belvedere.at/resources/images/xl/5420.jpg](https://digital.belvedere.at/reso
urces/images/xl/5420.jpg)",
    "type": "WebResource"
  },
  "edm:rights": {
    "id":
"[http://creativecommons.org/publicdomain/mark/1.0/](http://creativecommons.org/publicdomain/m
ark/1.0/)",
    "type": "RightStatement"
  }
}
```

Beispiel zur Verarbeitung (Python)

Das folgende Python-Skript zeigt, wie Sie das Archiv herunterladen, entpacken und die einzelnen JSONL-Dateien Zeile für Zeile verarbeiten können.

```
import requests
import tarfile
import json
import io
```

```
# URL zum Datenset
DATASET_URL =
"[https://s3.kpool.at/nightly/full_records.tar.gz](https://s3.kpool.at/nightly/full_records.ta
r.gz)"

print(f"Lade Datenset von {DATASET_URL} herunter...")
response = requests.get(DATASET_URL, stream=True)
response.raise_for_status()

# Erstelle ein file-like object aus dem heruntergeladenen Inhalt
file_like_object = io.BytesIO(response.content)

print("Entpacke Archiv im Speicher...")
with tarfile.open(fileobj=file_like_object, mode="r:gz") as tar:
    # Iteriere durch jede Datei im Archiv
    for member in tar.getmembers():
        if member.isfile() and member.name.endswith('.jsonl'):
            print(f"\n--- Verarbeite Datei: {member.name} ---")

            # Extrahiere die Datei in den Speicher
            file_content = tar.extractfile(member).read().decode('utf-8')

            # Verarbeite jede Zeile der JSONL-Datei
            for line in file_content.strip().split('\n'):
                try:
                    # Lade die Zeile als JSON-Objekt
                    record = json.loads(line)

                    # Greife auf Daten zu (Beispiel: Titel extrahieren)
                    # Achtung: Die Struktur kann variieren
                    if 'aggregatedCHO' in record and 'dc:title' in record['aggregatedCHO']:
                        # Finde einen Titel, idealerweise auf Deutsch
                        title_de = next((t['value'] for t in
record['aggregatedCHO']['dc:title'] if t.get('lang') == 'de'), None)
                        title_any = record['aggregatedCHO']['dc:title'][0]['value'] if
record['aggregatedCHO']['dc:title'] else "Kein Titel"

                        print(f"Gefundener Titel: {title_de or title_any}")
```

```
                except json.JSONDecodeError:
                    print(f"Fehler beim Parsen einer Zeile in {member.name}")
                except Exception as e:
                    print(f"Ein unerwarteter Fehler ist aufgetreten: {e}")

print("\nVerarbeitung abgeschlossen.")
```