

# Amacathera XML Sitemap Implementation Guide

## Overview

This guide explains how to implement a Google-compliant XML sitemap for your Amacathera Next.js application.

## Why You Need a Sitemap

According to Google's guidelines, sitemaps help search engines:

- Discover all pages on your website
- Understand your site structure
- Crawl pages more efficiently
- Index new pages faster
- Track when content was last updated

## Implementation Steps

### Step 1: Create the Sitemap Route Handler

1. Create a new file: `app/api/sitemap/route.ts`
2. Copy the provided route handler code
3. Update the `BASE_URL` environment variable in your `.env.local`:

```
env

NEXT_PUBLIC_BASE_URL=https://amacathera.com
```

### Step 2: Place robots.txt File

1. Create or update: `public/robots.txt`
2. Copy the provided robots.txt content
3. This file tells search engines where to find your sitemap

### Step 3: Generate Static Sitemap (Optional)

For a static sitemap file at `public/sitemap.xml`:

1. Create the file and manually add your URLs, or
2. Use a build script to generate it dynamically

## Step 4: Submit to Google Search Console

1. Go to Google Search Console
2. Navigate to your property
3. Select "Sitemaps" in the left menu
4. Enter the sitemap URL: `https://amacathera.com/api/sitemap` or `https://amacathera.com/sitemap.xml`
5. Click "Submit"

## Sitemap Structure

### Static Pages (Manually Configured)

- Homepage: `/`
- About: `/about`
- Contact: `/contact-us`
- Products: `/AMT`, `/hydrogel-platform`, `/pipeline`
- Company: `/our-team`, `/career`
- Content: `/news`, `/publications`

### Dynamic Pages (Auto-Generated from Database)

- News articles: `/news/[id]`
- Publications: `/publications/[id]`
- Team members: `/our-team/[id]` (can be added)

## XML Tags Explained

Each URL entry contains:

- `<loc>` - The complete URL (required)
- `<lastmod>` - Last modification date (YYYY-MM-DD format)
- `<changefreq>` - Suggested crawl frequency:
  - `always`, `hourly`, `daily`, `weekly`, `monthly`, `yearly`, `never`
  - Note: Google ignores this value
- `<priority>` - Relative priority (0.0 to 1.0)
  - Note: Google ignores this value

## Best Practices Implemented

- ✓ **UTF-8 Encoding** - Proper file encoding

- ✓ **Absolute URLs** - Full URLs with domain
- ✓ **Canonical URLs** - Single version of each page
- ✓ **Last Modified Dates** - Accurate update tracking
- ✓ **Proper Location** - Accessible at root level
- ✓ **robots.txt Reference** - Search engines can find it easily
- ✓ **Size Limits** - Under 50MB and 50,000 URLs
- ✓ **Mobile Friendly** - Works for all versions

## Monitoring Your Sitemap

### In Google Search Console

1. Check "Sitemaps" report to see:
  - When Google last crawled it
  - How many URLs were indexed
  - Any processing errors
  - Coverage statistics
2. Monitor over time:
  - Index coverage rate
  - Errors or warnings
  - Crawl efficiency

### In Your Application

Add logging to track:

```
typescript

console.log('Sitemap accessed:', new Date().toISOString());
console.log('Total URLs:', allEntries.length);
```

## Environment Setup

### Required Environment Variables

```
env

NEXT_PUBLIC_BASE_URL=https://amacathera.com
MONGODB_URI=your_mongodb_connection_string
```

### Next.js Configuration (next.config.js)

No special configuration needed, but ensure:

- Static files are served from `public/`
- API routes are enabled (default)
- Build completes without errors

## Troubleshooting

### Sitemap Not Found

- Check if `app/api/sitemap/route.ts` exists
- Verify URL is accessible: `https://amacathera.com/api/sitemap`
- Check server logs for errors

### Invalid XML Format

- Ensure UTF-8 encoding
- Validate URLs are properly escaped
- Check for special characters (&, <, >, ", ')

### Google Not Crawling Sitemap

- Submit via Search Console
- Check robots.txt syntax
- Wait 24-48 hours for Google to recrawl
- Verify no robots.txt blocking

## Advanced Options

### Sitemap Index (For Large Sites)

If you exceed 50,000 URLs or 50MB:

```
xml

<?xml version="1.0" encoding="UTF-8"?>
<sitemapindex xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
  <sitemap>
    <loc>https://amacathera.com/sitemap-1.xml</loc>
  </sitemap>
  <sitemap>
    <loc>https://amacathera.com/sitemap-2.xml</loc>
  </sitemap>
</sitemapindex>
```

## Dynamic Generation with Cron Jobs

Update sitemap daily:

```
typescript

// Using Vercel Cron (requires Vercel Pro)
// pages/api/cron/update-sitemap.ts
export const config = {
  runtime: 'nodejs',
};

export default async function handler(req) {
  if (req.headers.authorization !== `Bearer ${process.env.CRON_SECRET}`) {
    return { status: 401 };
  }
  // Generate and cache sitemap
}
```

## Performance Tips

### 1. Cache Aggressively

- Set 1-hour cache for sitemap
- Use `stale-while-revalidate` for longer caching

### 2. Generate on Demand

- Dynamic route handler fetches fresh data
- Database queries are optimized with `.lean()`

### 3. Monitor Database Performance

- Index `_id` and `updatedAt` fields
- Consider pagination for very large datasets

## References

- [Google Sitemap Guidelines](#)
- [Sitemaps Protocol](#)
- [robots.txt Specification](#)

## Next Steps

1. ☒ Implement the route handler
2. ☒ Add robots.txt file

3. ☒ Set environment variables
4. ☒ Test sitemap URL in browser
5. ☒ Submit to Google Search Console
6. ☒ Monitor coverage and errors
7. ☒ Update dynamic content regularly