

# Enhanced Content Loaders - Complete Package

**Version:** 2.0.0

**Status:** Production Ready

**Target:** RainmeterManager/RenderProcess/Content

---

## Package Contents

### Core Loaders (4 Files)

#### 1. **FileContentLoader.cs** (35 KB)

- 50+ file format support
- Syntax highlighting for 20+ languages
- Thumbnail generation
- Metadata extraction
- Content analysis

#### 2. **WebContentLoader.cs** (28 KB)

- 50+ curated web sources
- 13 categories (Space, Weather, News, etc.)
- Smart caching system
- Rate limiting
- API key management

#### 3. **MediaContentLoader.cs** (18 KB)

- Video/audio support (20+ formats)
- Playlist management
- Metadata extraction (requires FFmpeg)
- Thumbnail generation

#### 4. **APIContentLoader.cs** (45 KB)

- REST API support with authentication
- DynamicEnvironmentLoader (15+ environments)
- Response caching

- Retry logic with exponential backoff
- Time-based, weather-reactive, animated backgrounds

## Documentation

- **Enhanced Content Loaders - Complete Documentation.pdf**
  - Full API reference
  - Configuration examples
  - 50+ curated sources list
  - Performance metrics
  - Troubleshooting guide

## Deployment

- **Deploy-ContentLoaders.ps1** (PowerShell script)
  - Automated deployment
  - Backup existing files
  - Verification checks
  - Dry-run mode



## Quick Start

### 1. Deployment

#### Option A: Automated (Recommended)

```
powershell

# Dry run first (see what would happen)
.\Deploy-ContentLoaders.ps1 -DryRun

# Deploy with backup
.\Deploy-ContentLoaders.ps1 -Backup

# Force overwrite without prompts
.\Deploy-ContentLoaders.ps1 -Force
```

#### Option B: Manual

```
powershell
```

```
# Copy files to target directory
```

```
Copy-Item FileContentLoader.cs .\RenderProcess\Content\
```

```
Copy-Item WebContentLoader.cs .\RenderProcess\Content\
```

```
Copy-Item MediaContentLoader.cs .\RenderProcess\Content\
```

```
Copy-Item APIContentLoader.cs .\RenderProcess\Content\
```

## 2. Project Integration

Add to `RenderProcess/Program.cs` or DI container:

```
csharp
```

```
using RenderProcess.Content;
```

```
// In ConfigureServices or service registration:
```

```
services.AddSingleton<FileContentLoader>();
```

```
services.AddSingleton<WebContentLoader>();
```

```
services.AddSingleton<MediaContentLoader>();
```

```
services.AddSingleton<APIContentLoader>();
```

```
services.AddSingleton<DynamicEnvironmentLoader>();
```

## 3. Configuration

Set API Keys (Optional but recommended for full features):

```
powershell
```

```
# Windows
```

```
setx NASA_API_KEY "your-api-key-here"
```

```
setx OPENWEATHER_API_KEY "your-key"
```

```
setx WEATHER_API_KEY "your-key"
```

```
# Linux/Mac
```

```
export NASA_API_KEY="your-api-key-here"
```

```
export OPENWEATHER_API_KEY="your-key"
```

```
export WEATHER_API_KEY="your-key"
```

**API Key Sources:**

- NASA API: <https://api.nasa.gov/> (Free)
- OpenWeatherMap: <https://openweathermap.org/api> (Free tier)

- WeatherAPI: <https://www.weatherapi.com/> (Free tier)

## Feature Matrix

Feature	File	Web	Media	API
Local Files	✓	✗	✓	✗
Network	✗	✓	✗	✓
Caching	✗	✓	✗	✓
Authentication	✗	✓	✗	✓
Progress	✓	✗	✓	✗
Thumbnails	✓	✗	✓	✗
Metadata	✓	✓	✓	✓
Retry Logic	✗	✗	✗	✓

## Usage Examples

### File Loader

```
csharp
var loader = new FileContentLoader(logger);
var result = await loader.LoadAsync("code.cs", cancellationToken);

if (result.ContentType == FileContentType.Code)
{
    webView.NavigateToString(result.Html); // Syntax highlighted
    Console.WriteLine($"Lines: {result.LineCount}");
}
```

### Web Loader

```
csharp
```

```
var loader = new WebContentLoader(logger);
var nasa = loader.GetSourceById("nasa_apod");
var (success, html) = await loader.BuildEmbedHtmlAsync(nasa, ct);

if (success)
    webView.NavigateToString(html);
```

## Media Loader

```
csharp

var loader = new MediaContentLoader(logger);
var result = await loader.LoadMediaAsync("video.mp4", ct);

webView.NavigateToString(result.PreviewHtml);
```

## API Loader

```
csharp

var loader = new APIContentLoader(logger);
var request = new APIRequest
{
    Url = "https://api.example.com/data",
    Method = "GET",
    EnableCaching = true
};

var response = await loader.FetchAsync(request, ct);
if (response.Success)
    ProcessData(response.Data);
```

## Dynamic Environments

```
csharp

var envLoader = new DynamicEnvironmentLoader(logger, webLoader, apiLoader);
var environments = envLoader.GetAvailableEnvironments();

var dayNight = environments.First(e => e.Id == "day_night_cycle");
var html = await dayNight.Generator(ct);
webView.NavigateToString(html);
```

---

## Configuration Options

### File Loader

```
csharp

loader.MaxFileBytes = 128 * 1024 * 1024; // 128 MB
loader.EnableThumbnails = true;
loader.EnableMetadataExtraction = true;
loader.EnableSyntaxHighlighting = true;
loader.ThumbnailMaxSize = 512;
```

### Web Loader

```
csharp

loader.CacheMinutes = 30;
loader.RequestTimeoutSeconds = 60;
loader.EnableCaching = true;
loader.EnableRateLimiting = true;
```

### Media Loader

```
csharp

// Minimal configuration needed
// FFmpeg optional for enhanced metadata
```

### API Loader

```
csharp

loader.CacheMinutes = 15;
loader.MaxRetries = 5;
loader.TimeoutSeconds = 45;
```

---

## Performance Guidelines

### Recommended Settings by Use Case

#### High-Performance Desktop (Gaming/Workstation):

```
csharp
```

```
fileLoader.MaxFileBytes = 256 * 1024 * 1024;  
fileLoader.EnableContentAnalysis = true;  
webLoader.CacheMinutes = 15;  
apiLoader.MaxRetries = 5;
```

## Standard Desktop:

```
csharp
```

```
fileLoader.MaxFileBytes = 100 * 1024 * 1024;  
fileLoader.EnableContentAnalysis = true;  
webLoader.CacheMinutes = 30;  
apiLoader.MaxRetries = 3;
```

## Low-End/Mobile:

```
csharp
```

```
fileLoader.MaxFileBytes = 50 * 1024 * 1024;  
fileLoader.EnableContentAnalysis = false;  
webLoader.CacheMinutes = 60;  
apiLoader.MaxRetries = 2;
```

---

## Troubleshooting

### Common Issues

#### 1. Files Not Loading

- Check file path is correct
- Verify file format is supported
- Check MaxFileBytes limit

#### 2. Web Sources Failing

- Verify internet connection
- Check API keys are set
- Increase timeout: `loader.RequestTimeoutSeconds = 120`

### 3. Out of Memory

- Reduce MaxFileBytes
- Disable thumbnails: `EnableThumbnails = false`
- Disable content analysis: `EnableContentAnalysis = false`

### 4. API Rate Limiting

- Increase cache duration: `CacheMinutes = 60`
- Reduce request frequency
- Check API key quotas

---

## File Structure After Deployment

```
RenderProcess/  
└─ Content/  
    ├── FileContentLoader.cs  ✓ Deployed  
    ├── WebContentLoader.cs  ✓ Deployed  
    ├── MediaContentLoader.cs ✓ Deployed  
    └─ APIContentLoader.cs    ✓ Deployed
```

---

## Security Notes

1. **Never hardcode API keys** - Always use environment variables
2. **Validate file paths** - Check for path traversal attacks
3. **Limit file sizes** - Prevent memory exhaustion
4. **Sanitize HTML** - Before rendering in Web View
5. **Use HTTPS** - For all web requests

---

## Upgrading

### From Version 1.x

1. Backup existing loaders
2. Run deployment script



3. Update DI registrations
4. Test with existing content
5. Configure new features

## Migration Notes

- FileContentLoader: Compatible, new features auto-enabled
  - WebContentLoader: 30+ new sources added
  - MediaContentLoader: Enhanced but backward compatible
  - APICContentLoader: New loader, requires registration
- 

## Support

### Getting Help

1. Check **Complete Documentation.pdf** first
2. Review troubleshooting section
3. Check example code in documentation
4. Verify configuration settings



### Reporting Issues




Include:

- Loader version (2.0.0)
  - Error messages
  - Configuration settings
  - Sample code
- 

## Version History

### 2.0.0 (Current)

-  50+ file formats
-  50+ web sources

-  15+ dynamic environments
-  Complete API support
-  Production ready

## 1.x (Legacy)

- Basic file loading
  - Limited web sources
  - No API support
- 

## Deployment Checklist

- ☐ Run `Deploy-ContentLoaders.ps1 -DryRun`
  - ☐ Review deployment plan
  - ☐ Run actual deployment
  - ☐ Rebuild RenderProcess project
  - ☐ Update DI container
  - ☐ Set API keys (optional)
  - ☐ Test with sample files
  - ☐ Test web sources
  - ☐ Test dynamic environments
  - ☐ Verify performance
- 

## Next Steps

1. **Deploy** using provided script
  2. **Configure** API keys for enhanced features
  3. **Test** with your content
  4. **Optimize** settings for your use case
  5. **Explore** 15+ dynamic environments
  6. **Integrate** into your UI
- 

**Package Version:** 2.0.0

**Last Updated:** October 2024

**Status:**  Production Ready

**License:** Part of RainmeterManager Project