ItkAutoImageCompressor

Entwicklerdokumentation

André Fischer

30.03.2026

Inhaltsverzeichnis

1 Technischer Überblick

Konfigurations-Plugin-Name: ItkAutoImageCompressor → Keys `ItkAutoImageCompressor.config.*`

Auslöser: MediaUploadSubscriber auf **`Shopware\\Core\\Content\\Media\\Event\\MediaUploadedEvent`**.

Kern: ImageCompressorService delegiert an **ImageCompressorEngineInterface** (Default-Alias: **MultiEngine**), das je nach MIME-Typ **JpegoptimEngine**, **OptipngEngine** oder **CwebpEngine** aufruft.

1.1 Symfony / Shopware für Einsteiger

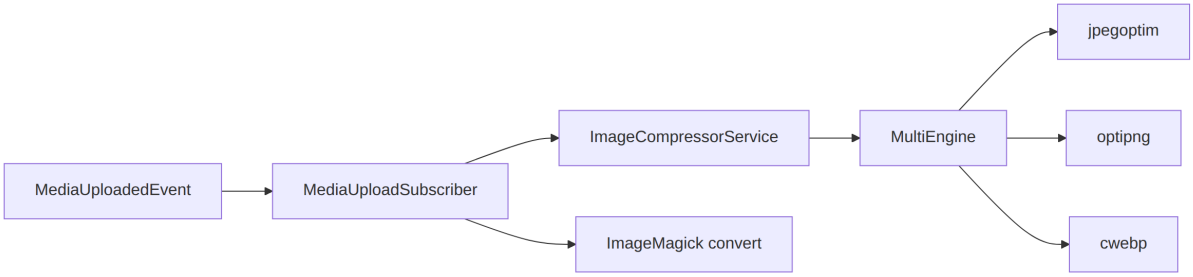
- **Event Subscriber:** Eine Klasse mit **`getSubscribedEvents()`** registriert sich beim **Event-Dispatcher**. Wenn Shopware nach einem Upload **MediaUploadedEvent** wirft, ruft der Kernel deine Methode **onMediaUploaded** auf – **ohne** dass du den Upload-Code von Shopware bearbeiten musst.

- **SystemConfigService:** Liest Werte aus der Datenbank-Tabelle der Plugin-Konfiguration (was du im Admin unter Plugin-Einstellungen siehst). Keys sind `ItkAutoImageCompressor.config.fieldName`.

- **Compiler-Pass / Tags:** Die Engines tragen das Tag `shopware.image_compressor.engine`. Wenn du eigene Bundles schreibst, könntest du weitere Engines registrieren (in diesem Repo nutzt `services.xml` primär den Alias auf **MultiEngine**).

- **Subprocess:** Die Engines starten echte CLI-Tools wie **jpegoptim**, **optipng** und **cwebp** aus PHP heraus. Das ist bewusst **kein** reines PHP-GD/Imagick-Konzept, sondern setzt auf System-Binaries für bessere Kompression und mehr Formatschalter.

1.2 Ablauf als Diagramm



Hinweis: Resize und viele „Verbesserungen“ laufen über **convert** im Subscriber bzw. in den Engines (**applyImageImprovements**), nicht nur über die drei reinen Kompressions-Binaries.

---

2 Alle PHP-Klassen (`src/`)

Klasse	Rolle
ItkAutoImageCompressor	Plugin-Bootstrap
Subscriber\MediaUploadSubscriber	Event → Pfad auflösen, Filter (Größe, MIME, Ordner), Backup, Aufruf **ImageCompressorService**
Service\MediaPathService	Berechnet aus Dateiname, Extension und Upload-Zeit den erwarteten Shopware-Medienpfad; kann den Pfad auch gegen die echte Verzeichnisstruktur testen und notfalls per Glob den tatsächlichen Speicherort finden
Core\Content\ImageCompressor\Config\ImageCompressorConfig	Liest **alle** SystemConfig-Keys, Defaults, Validierung, **getExcludeFolders()**, **isFolderExcluded()** mit **LIKE-ähnlichen Mustern** (**%** → Regex)
Core\Content\ImageCompressor\ImageCompressorService	Orchestrierung Komprimierung, Format-Checks, **supportsFormat()**
Core\Content\ImageCompressor\Engine\ImageCompressorEngineInterface	Engine-Vertrag
…\Engine\JpegoptimEngine	Subprocess **jpegoptim**
…\Engine\OptipngEngine	Subprocess **optipng**
…\Engine\CwebpEngine	Subprocess **cwebp**

Klasse	Rolle
...\Engine\MultiEngine	Dispatch nach MIME
Core\Content\Image Compressor\Struct\CompressionResult	Ergebnisobjekt
...\Exception\ImageCompressionException	Basisklasse
...\Exception\EngineNotAvailableException	Tool fehlt / nicht ausführbar
Command\OptimizeImageCommand	itk:image:optimize
Command\OptimizeAllMediaCommand	itk:image:optimize-all
Command\CheckResizeRulesCommand	itk:image:check-resize
Command\TestPluginCommand	itk:image:test

---

3 Services (`Resources/config/services.xml`)

- Alias **ImageCompressorEngineInterface** → **MultiEngine**
- Engines sind mit `shopware.image_compressor.engine` getaggt (falls Erweiterung durch weitere Bundles genutzt wird)
- Logger `image_compressor` → `var/log/image_compressor.log`
- Logger `image_resize_small` → `var/log/image_resize_small.log`
- **ImageCompressorConfig** bekommt **SystemConfigService** injiziert; optional Sales Channel über **`setSalesChannelId()`** im Subscriber wenn **Context** eine **SalesChannelContext** ist

---

4 Ablauf `MediaUploadSubscriber::onMediaUploaded` (kurz)

1. Plugin aktiv + Media-Entity + Datei auf Disk
2. Dateigröße in [min, max] (Byte; KB aus Config × 1024)
3. MIME von **`supportsFormat()`**

4. Medienordner-Name: `isFolderExcluded()`
5. Optional Backup
6. **ImageCompressorService**: wendet Resize und Bildverbesserungen an, ruft danach die passende Engine zur eigentlichen Komprimierung auf und erzeugt bei aktivierter WebP-Konvertierung zusätzlich das WebP-Derivat
7. Bei Fehler: loggen, Upload bleibt in Shopware bestehen (Soft-Fail)

---

5 Erweiterung

- Eigene Engine: Interface implementieren, als Service definieren und taggen oder **MultiEngine** erweitern
- Eigene Events: Subscriber nach **MediaUploadedEvent** chainen
- Performance: Batch-Commands außerhalb der Peak-Zeiten nutzen

---

6 Abhängigkeiten Server

- **jpegoptim**, **optipng**, **cwebp** im PATH des **gleichen Users** wie PHP-FPM / PHP-CLI
- Auf manchen Hostings: `open_basedir` / `proc_open` prüfen