Für Entwickler: Ereignisse

Fileshelf lässt sich mit eigenen Plugins erweitern – genau so, wie es die Pro-Version tut. Alle Plugins der Gruppe `fileshelf` lädt Fileshelf, bevor es seine Ereignisse feuert. Die Klassen liegen in `Graup\Component\Fileshelf\Administrator\Event`.

> Die Ereignisse sind die öffentliche Schnittstelle von Fileshelf. Name, Argumente und öffentliche Methoden ändern sich nur mit einer neuen Hauptversion.

Die Ereignisse

Ereignis	Klasse	Wann	Was ein Plugin tun kann
`onFileshelfResolveToken`	`ResolveTokenEvent`	Die Anfrage trägt einen Token (`t`).	`grant()` überspringt Zugriffsebene und Passwort, `reject()` sperrt mit Grund `token`.
`onFileshelfAccess`	`AccessEvent`	Nach Status, Token, Zugriffsebene und Passwort; auch in Listen (Kontext `list`).	`deny($reason, $message, $redirect)` sperrt. In Listen nichts zählen oder schreiben.
`onFileshelfBeforeSend`	`BeforeSendEvent`	Zugriff erteilt, vor dem Zähler.	`setFile()` tauscht die Datei; Fileshelf prüft sie danach erneut gegen die Ablage.
`onFileshelfAfterCount`	`AfterCountEvent`	Nach dem Hochzählen, vor dem ersten Byte.	Nur Meldung; `getRequestId()` verbindet mit dem nächsten Ereignis.
`onFileshelfAfterSend`	`AfterSendEvent`	Nach dem Senden, auch bei Abbruch.	`isCompleted()`, `getBytesSent()`, `getMethod()`.
`onFileshelfDownloadDisplay`	`DownloadDisplayEvent`	Detailseite, je Position.	`addHtml()` mit fertig maskiertem HTML.

Ereignis	Klasse	Wann	Was ein Plugin tun kann
`onFileshelfDashboard`	`DashboardEvent`	Dashboard im Backend.	`addTile($id, $title, $html, $order, $area)` (`$area`: `status`, `content`, `insight` oder `setup`); ab 1.8.0 `addAttention($id, $text, $link, $count, $label)` für „Braucht Ihre Aufmerksamkeit“, nur bei Handlungsbedarf mit Ausweg; eigenes Recht vorher prüfen.
`onFileshelfAdminPages`	`AdminPagesEvent`	Dashboard und jede Plugin-Seite.	`addPage($name, $title, $right, $icon)` meldet eine Seite `view=page&page=…` an; Fileshelf prüft das Recht.
`onFileshelfAdminPage`	`AdminPageEvent`	Die Plugin-Seite wird angezeigt.	`addHtml()`, `addAction($action, $label, $icon, $confirm)`.
`onFileshelfAdminAction`	`AdminActionEvent`	Eine Aktion der Seite läuft; Token, Recht und Bestätigung sind geprüft.	`handled($message, $type)`.
`onFileshelfFilesInUse`	`FilesInUseEvent`	Werkzeuge, Download-Formular und Medienmanager fragen, welche Dateien der Ablage Plugins gehören.	`claim($path)` für eigene Dateien (z. B. ältere Versionen); sie gelten nie als verwaist.
`onFileshelfSitePages`	`SitePagesEvent`	Menüpunkt „Erweiterungsseite“ und jeder Aufruf einer solchen Seite.	`addPage($name, $title, $login)` meldet eine Seite der Website an; mit `$login` leitet Fileshelf Gäste zur Anmeldung.
`onFileshelfSitePage`	`SitePageEvent`	Die Seite wird angezeigt.	`addHtml()` mit fertig maskiertem HTML, `setTitle()`, `getOptions()` (Menüpunkt).
`onFileshelfSiteAction`	`SiteActionEvent`	Ein Formular der Seite wurde abgeschickt; Token, Seite und Anmeldung sind geprüft.	`handled($message, $type, $redirect)`; Fileshelf maskiert die Meldung.

Ereignis	Klasse	Wann	Was ein Plugin tun kann
`onFileshelfCategoryDisplay`	`CategoryDisplayEvent`	Kategorieseite, Positionen `beforeList` und `afterList`.	`addHtml()` mit fertig maskiertem HTML; `getTotal()` nennt die Zahl der gelisteten Downloads.
`onFileshelfActionLog`	`ActionLogEvent`	Seit 1.3.0: Fileshelf oder ein Plugin meldet eine Aktion fürs Aktionsprotokoll (Passwort, Freigabe, Import).	Reine Meldung: `getAction()`, `getKind()` (`download`, `category`, `import`), `getItemId()`, `getTitle()`, `getDetails()`. Das Plugin „Aktionsprotokoll - Fileshelf“ schreibt sie.
`onFileshelfImportSources`	`ImportSourcesEvent` (Pro)	Seit 1.3.0: Die Seite „Import“ sammelt ihre Quellen.	`addSource()` mit einer Klasse, die `SourceInterface` des Import-Plugins erfüllt (Kategorien, Downloads, Prüfung der Tabellen).
`onFileshelfResolveScopes`	`ResolveScopesEvent`	Von der Pro-Version (Limits) gefeuert, sobald ein Limit einen Scope-Typ eines anderen Plugins nutzt; die Free-Version feuert es nicht.	`addScope($type, $id)`.

Der Download kommt als schreibgeschütztes `DownloadRecord` (ohne Passwort-Hash), nie als Table-Objekt. Eigene Sperrgründe beginnen mit `x-<plugin>-`; die Website zeigt dann nur die Nachricht.

Beispiel

```
final class Example extends CMSPlugin implements SubscriberInterface
{
    public static function getSubscribedEvents(): array
    {
        return ['onFileshelfAccess' => 'onAccess'];
    }

    public function onAccess(AccessEvent $event): void
    {
```

```
        if ($event->getContext() !== 'list' && !$this->allowed($event->getUser())) {
            $event->deny('x-example-hours', 'Downloads sind nur werktags möglich.');
        }
    }
}
```

Formular-Reiter am Download fügen Plugins wie bei Beiträgen mit `onContentPrepareForm` (Formular `com_fileshelf.download`) ein und speichern sie in `onContentAfterSave` (Kontext `com_fileshelf.download`).

Seiten auf der Website

Mit dem Menüpunkttyp **Fileshelf → Erweiterungsseite** bekommt ein Plugin eine eigene Seite auf der Website, ohne eigene Komponente. Formulare dieser Seite schicken `task=extension.action`, `page`, `action` und das Formular-Token an `Route::_('index.php?option=com_fileshelf')` – die Adresse immer über `Route::_()` bauen, eine relative Adresse stimmt unter suchmaschinenfreundlichen URLs nicht. Reiter am Kategorieformular hängen Plugins mit `onContentPrepareForm` an das Formular `com_categories.categorycom_fileshelf`; Fileshelf lädt die Gruppe `fileshelf` dafür.

English version

Revision #6
Created 2026-11-10 08:11:28 UTC by Norbert Graup
Updated 2026-11-11 20:56:56 UTC by Norbert Graup