

Building Flows



## Personal

- Personal info
- Security
- Activity
- Mobile & desktop
- Accessibility
- Sharing
- Workflows

## Administration

- Overview
- Basic settings
- Support
- Sharing
- Security
- Theming
- Groupware
- Activity
- Workflows
- Logging
- System

## Workflows

**Files access control**  
Block access if a file is accessed

**Automated tagging**  
Automated tagging of files

**Run a script**  
Convert a file to PDF using Libreoffice

**Notify me**  
Send a Nextcloud Notification

**Convert to PDF**  
Convert a file to PDF using Libreoffice

**Create a card in Deck**  
Attach the file to a new card

**Send a message in Talk**  
Send a message to a conversation

**Send an email**  
Get notifications by email

▲ Show less

When **File updated** ▼

and

File system tag

is tagged with

confidential

Add a new filter



**Automated tagging**  
Automated tagging of files



restricted

✗ The configuration is invalid



## Personal

Personal info

Security

Activity

Mobile & desktop

Accessibility

Sharing

Workflows

## Administration

Overview

Basic settings

Support

Sharing

Security

Theming

Groupware

Activity

Workflows

Logging

System

## Workflows



**Files access control**  
Block access if a file is accessed



**Automated tagging**  
Automated tagging of files



**Run a script**  
Convert a file to PDF using Libreoffice



**Notify me**  
Send a Nextcloud Notification



**Convert to PDF**  
Convert a file to PDF using Libreoffice



**Create a card in Deck**  
Attach the file to a new card



**Send a message in Talk**  
Send a message to a conversation



**Send an email**  
Get notifications by email

▲ Show less

When **File updated**

and File system tag

is tagged with

confidential

Add a new filter



**Automated tagging**  
Automated tagging of files



restricted

✗ The configuration is invalid

Mobile & desktop

Accessibility

Sharing

Workflows

## Administration

Overview

Basic settings

Support

Sharing

Security

Theming

Groupware

Activity

Workflows

Logging

System

### Files access control

Block access if a file is accessed

### Automated tagging

Automated tagging of files

### Run a script

Convert a file to PDF using Libreoffice



### Notify me

Send a Nextcloud Notification



### Convert to PDF

Convert a file to PDF using Libreoffice



### Create a card in Deck

Attach the file to a new card



### Send a message in Talk


Send a message to a conversation



### Send an email

Get notifications by email

▲ Show less

When  **File updated** ▼

and

File system tag

is tagged with

confidential

Add a new filter



### Automated tagging

Automated tagging of files



restricted

✗ The configuration is invalid

When **File updated** ▼

and

File system tag

is tagged with

confidential

Add a new filter



**Automated tagging**  
Automated tagging of files



restricted

✕ The configuration is invalid

When  **File updated** ▼

and

File system tag

is tagged with

confidential

Add a new filter



**Automated tagging**

Automated tagging of files



restricted

✕ The configuration is invalid

When  **File updated** ▼

and

File system tag

is tagged with

confidential

Add a new filter



**Automated tagging**

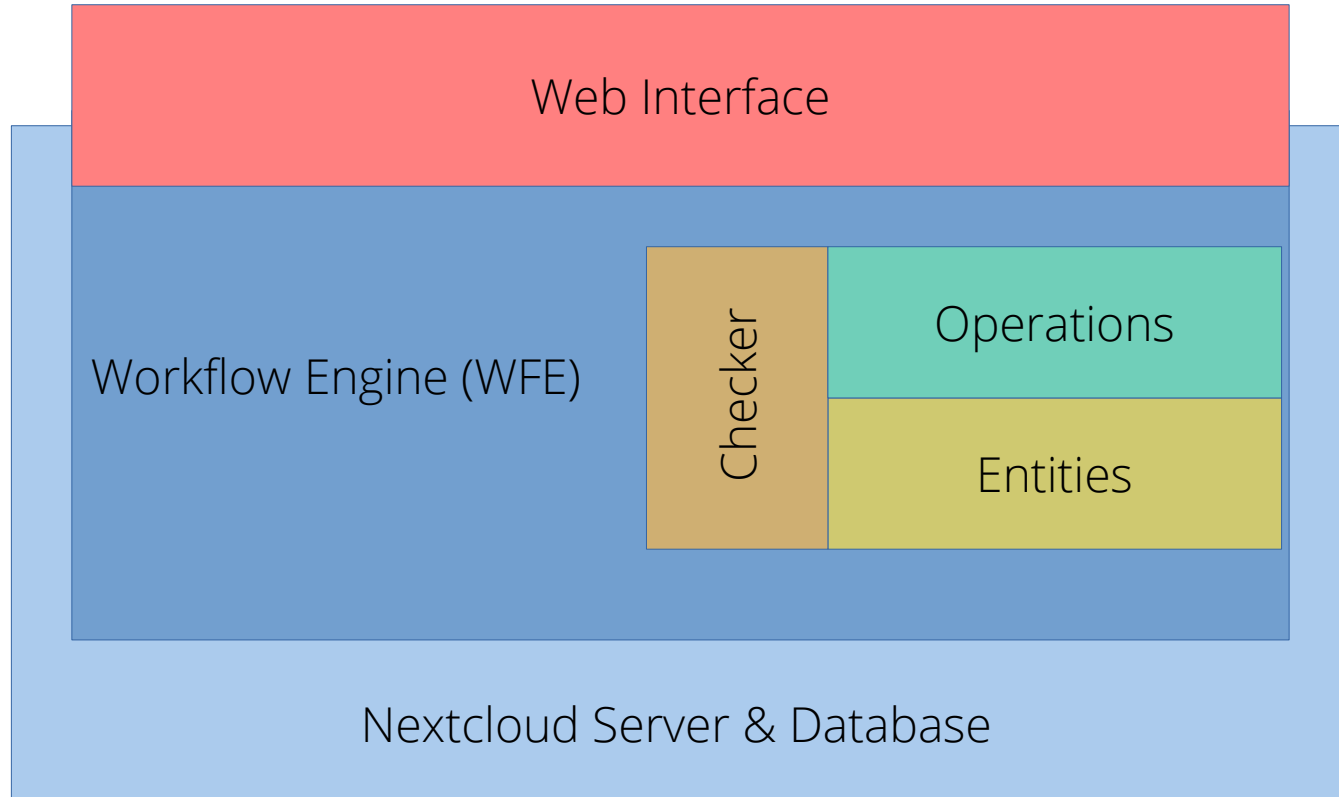
Automated tagging of files



restricted

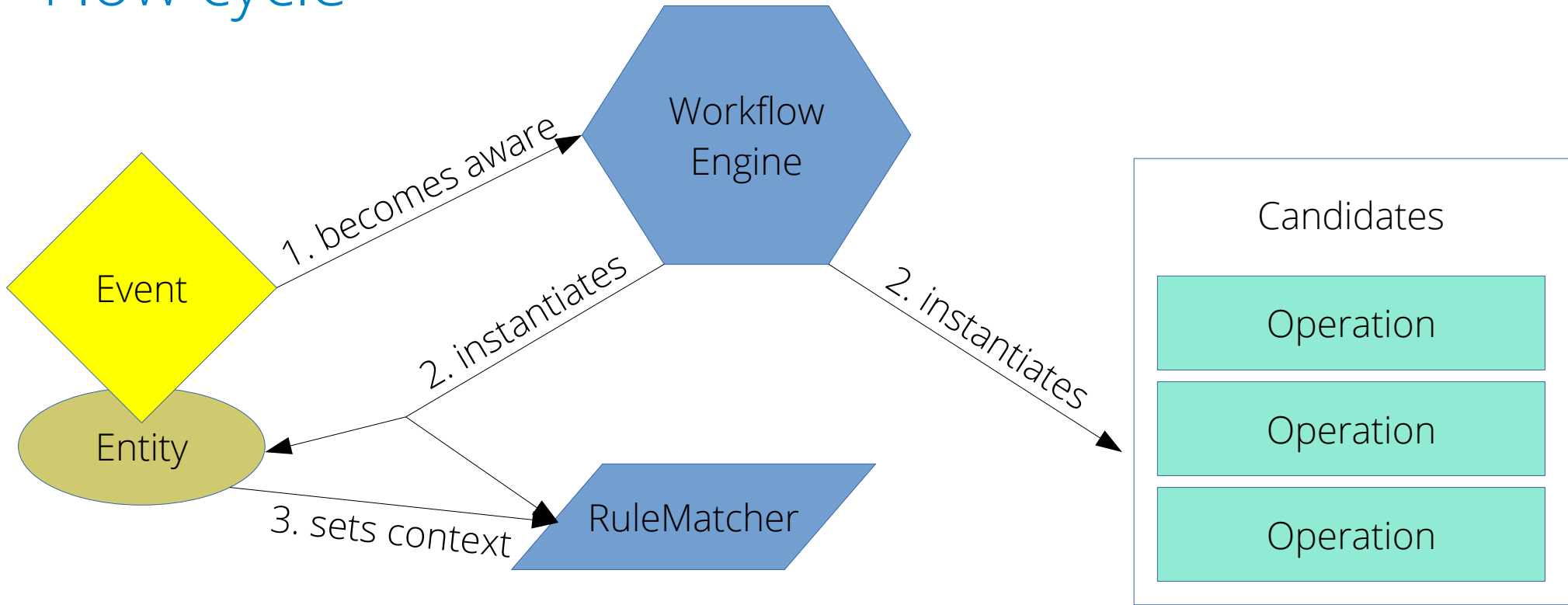
✕ The configuration is invalid

# Components

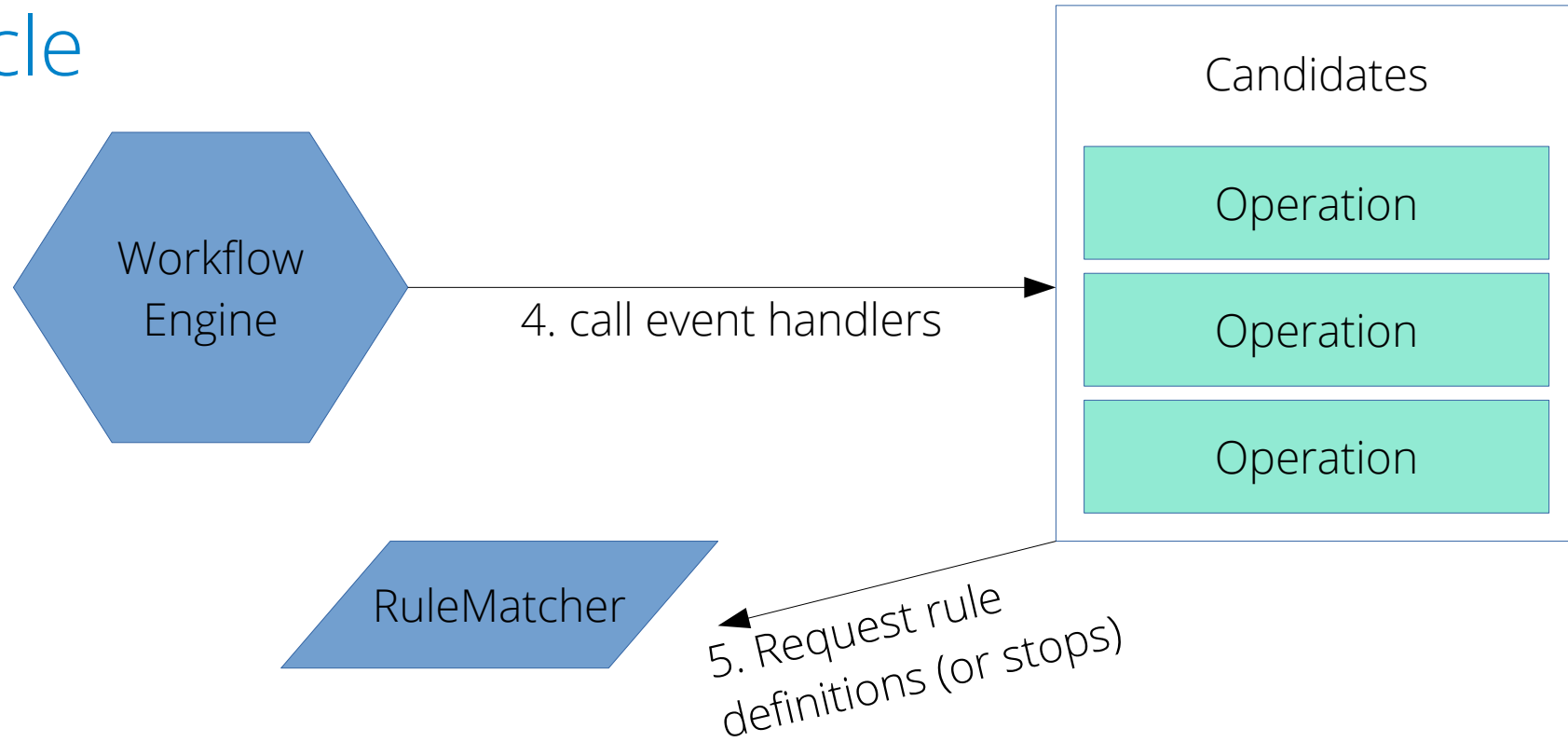




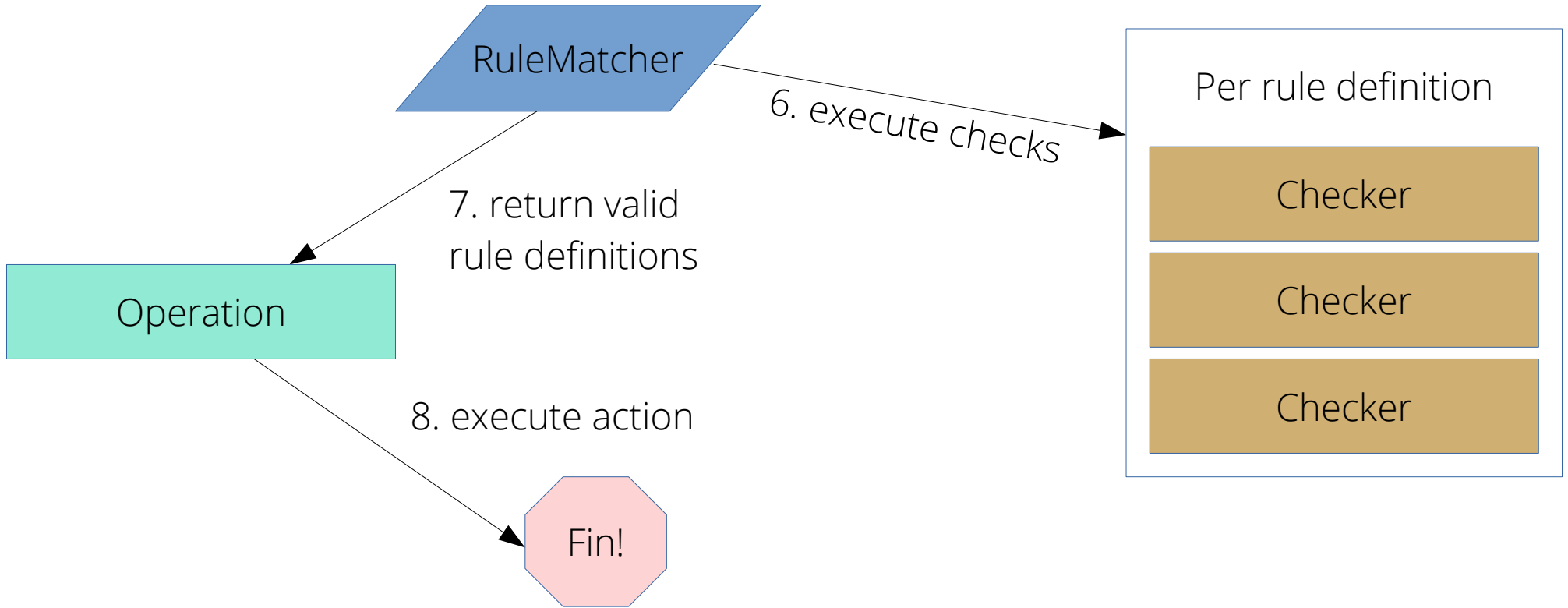
# Flow cycle



# Flow cycle



# Flow cycle



# The Engine

- Accepts **registrations of components**
- Sets up event listeners, forwards to **Operations**
- **Provides information** to the user interface
- Manages rules in the **database**

# What should be done?

- OCP\WorkflowEngine\**IOperation**
  - User facing information
  - Scope
  - Validation
  - Event handler
- **ISpecificOperation** – limited to specific events
- **IComplexOperation** – own listening logic

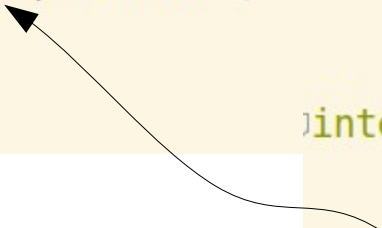
# Example: convert to PDF

```
public function getDisplayName(): string {  
    return $this->l->t( text: 'PDF conversion');  
}  
  
public function getDescription(): string {  
    return $this->l->t( text: 'Convert documents into the PDF format on upload and write.');}  
  
public function getIcon(): string {  
    return \OC::$server->getURLGenerator()->imagePath( appName: 'workflow_pdf_converter', file: 'app.svg');}
```

# Example: convert to PDF contd.

```
public function isAvailableForScope(int $scope): bool {  
    return true;  
}
```

```
interface IManager {  
  
    const SCOPE_ADMIN = 0;  
    const SCOPE_USER = 1;  
}
```



# Example: convert to PDF contd.

```
/**
 * @throws \UnexpectedValueException
 * @since 9.1
 */
public function validateOperation(string $name, array $checks, string $operation): void {
    if(!in_array($operation, haystack: Operation::MODES)) {
        throw new \UnexpectedValueException($this->l->t( text: 'Please choose a mode.'));
    }
}
```

MODES = [  
'keep;preserve',  
'keep;overwrite',  
'delete;preserve',  
'delete;overwrite',  
] : array



# Example: convert to PDF contd.

```
public function onEvent(string $eventName, GenericEvent $event, IRuleMatcher $ruleMatcher): void {
    try {
        // ... some logic that gathers info and returns on unsupported mimetypes
        $matches = $ruleMatcher->getMatchingOperations( class: Operation::class, returnFirstMatchingOperationOnly: false);
        $originalFileMode = $targetPdfMode = null;
        foreach($matches as $match) {
            $fileModes = explode( delimiter: ';', $match['operation']);
            if($originalFileMode !== 'keep') {...}
            if($targetPdfMode !== 'preserve') {...}
            if($originalFileMode === 'keep' && $targetPdfMode === 'preserve') {...}
        }
        if(!empty($originalFileMode) && !empty($targetPdfMode)) {
            $this->jobList->add( job: Convert::class, [
                'path' => $node->getPath(),
                'originalFileMode' => $originalFileMode,
                'targetPdfMode' => $targetPdfMode,
            ]);
        }
    } catch(\OCP\Files\NotFoundException $e) {
    }
}
```

# PDF Converter is an ISpecificOperation

```
public function getEntityId(): string {  
    class File implements OCP\WorkflowEngine\IEntity  
        return File::class;  
}
```

# Accesscontrol is an IComplexOperation

```
public function getTriggerHint(): string {  
    return $this->l->t( text: 'File is accessed');  
}  
  
public function onEvent(string $eventName, GenericEvent $event, IRuleMatcher $ruleMatcher): void {  
    // Noop  
}
```

# ... registers event listeners itself

```
public function registerHooksAndListeners() {  
    Util::connectHook( signalClass: 'OC_Filesystem', signalName: 'preSetup', $this, slotName: 'addStorageWrapper');  
    \OC::$server->getEventDispatcher()->addListener(...);  
}
```

# Operations have to register to the WFE

```
class Application extends \OC\AppFramework\App {  
  
    public function __construct() {...}  
  
    public function registerHooksAndListeners() {  
        \OC::$server->getEventDispatcher()->addListener( eventName: IManager::EVENT_NAME_REG_OPERATION,  
            function (GenericEvent $event) {  
                $operation = \OC::$server->query( name: Operation::class);  
                $event->getSubject()->registerOperation($operation);  
                \OC_Util::addScript( application: 'my_app', file: 'wfe-settings');  
            }  
        );  
    }  
}
```

# Register a new Operation in the frontend

- Checks by default have a standard input field
- Different operators can be specified
- They can have additional validation in the frontend
- Custom placeholders
- Provide a custom vue component

# Register a new Operation in the frontend

```
window.OCA.WorkflowEngine.registerOperator({  
  id: 'OCA\\FilesAutomatedTagging\\Operation',  
  color: 'var(--color-success)',  
})
```

# Frontend: Register a new Operation



## Automated tagging

Automated tagging of files



# Frontend: Register a new Operation

```
window.OCA.WorkflowEngine.registerOperator({  
  id: 'OCA\\FilesAutomatedTagging\\Operation',  
  color: 'var(--color-success)',  
  operation: '',  
  options: Tag  
})
```

# Frontend: Register a new Operation



**Automated tagging**  
Automated tagging of files



Select a tag to assign

→ Save

# When should it be done?

- OCP\WorkflowEngine\I**Entity**
  - User facing information
  - Announce available events
  - Providing event context

# Example: File entity

```
public function getName(): string {  
    return $this->l10n->t( text: 'File');  
}  
  
public function getIcon(): string {  
    return $this->urlGenerator->imagePath( appName: 'core', file: 'categories/files.svg');  
}
```

## Example: File entity contd.

```
public function getEvents(): array {
    $namespace = '\OCP\Files::';
    return [
        new GenericEntityEvent($this->l10n->t( text: 'File created'), eventName: $namespace . 'postCreate' ),
        new GenericEntityEvent($this->l10n->t( text: 'File updated'), eventName: $namespace . 'postWrite' ),
        new GenericEntityEvent($this->l10n->t( text: 'File renamed'), eventName: $namespace . 'postRename' ),
        new GenericEntityEvent($this->l10n->t( text: 'File deleted'), eventName: $namespace . 'postDelete' ),
        new GenericEntityEvent($this->l10n->t( text: 'File accessed'), eventName: $namespace . 'postTouch' ),
        new GenericEntityEvent($this->l10n->t( text: 'File copied'), eventName: $namespace . 'postCopy' ),
        new GenericEntityEvent($this->l10n->t( text: 'Tag assigned'), eventName: MapperEvent::EVENT_ASSIGN ),
    ];
}
```

# Describing Events

```
/** Interface IEntityEvent ...*/  
interface IEntityEvent {  
    /** returns a translated name to be presented in the web interface. ...*/  
    public function getDisplayName(): string;  
  
    /** returns the event name that is emitted by the EventDispatcher, e.g.: ...*/  
    public function getEventName(): string;  
}
```

- Implementation in \OCP\WorkflowEngine\**GenericEntityEvent**
- Requires EventDispatcher mechanism

# Example: File entity contd.

```
public function prepareRuleMatcher(IRuleMatcher $ruleMatcher, string $eventName, GenericEvent $event): void {
    switch ($eventName) {
        case 'postCreate':
        case 'postWrite':
        case 'postDelete':
        case 'postTouch':
            $ruleMatcher->setEntitySubject($this, $event->getSubject());
            break;
        case 'postRename':
        case 'postCopy':
            $ruleMatcher->setEntitySubject($this, $event->getSubject()[1]);
            break;
        case MapperEvent::EVENT_ASSIGN:
            if(!$event instanceof MapperEvent || $event->getObjectType() !== 'files') {
                break;
            }
            $nodes = $this->root->getById((int)$event->getObjectId());
            if(is_array($nodes) && !empty($nodes)) {
                $node = array_shift( &array: $nodes);
                $ruleMatcher->setEntitySubject($this, $node);
            }
            break;
    }
}
```

# How to narrow down events?

- OCP\WorkflowEngine\**ICheck**
  - No user facing info – has its own UI component
  - Scope
  - Supported Entities (can be any)
  - Validation method
  - Execution method



# Examples

```
public function supportedEntities(): array {  
    return [ File::class ];  
}
```

```
public function isAvailableForScope(int $scope): bool {  
    return true;  
}
```

```
public function supportedEntities(): array {  
    return [];  
}
```

```
public function isAvailableForScope(int $scope): bool {  
    return $scope === IManager::SCOPE_ADMIN;  
}
```

# Example validator

```
public function validateCheck($operator, $value) {  
    if (!in_array($operator, ['is', '!is'])) {  
        throw new \UnexpectedValueException($this->l->t( text: 'The given operator is invalid'), code: 1);  
    }  
  
    if (!$this->groupManager->groupExists($value)) {  
        throw new \UnexpectedValueException($this->l->t( text: 'The given group does not exist'), code: 2);  
    }  
}
```

# Example executor

```
public function executeCheck($operator, $value) {  
    $user = $this->userSession->getUser();  
  
    if ($user instanceof IUser) {  
        $groupIds = $this->getUserGroups($user);  
        return ($operator === 'is') === in_array($value, $groupIds);  
    } else {  
        return $operator !== 'is';  
    }  
}
```


# Register a new check in the frontend

- Checks by default have a standard input field
- Different operators can be specified
- They can have additional validation in the frontend
- Custom placeholders
- Provide a custom Vue.js component

# Register a new check in the frontend

```
window.OCA.WorkflowEngine.registerOperator({
  class: 'OCA\\WorkflowEngine\\Check\\FileSize',
  name: t('workflowengine', 'File size (upload)'),
  operators: [
    { operator: 'less', name: t('workflowengine', 'less') },
    { operator: '!greater', name: t('workflowengine', 'less or equals') },
    { operator: '!less', name: t('workflowengine', 'greater or equals') },
    { operator: 'greater', name: t('workflowengine', 'greater') }
  ],
  placeholder: (check) => '5 MB',
  validate: (check) => check.value.match(/^([0-9]+[ ]?[kmgT]?b$/i) !== null
},)
```

# Basic check with validation


When  **File created** ▼

and File size (upload)

Add a new filter

Select a comparator


- less
- less or equals
- greater or equals
- greater

5 MB 


# Register a new check with a custom component

```
window.OCA.WorkflowEngine.registerOperator({  
    class: 'OCA\\WorkflowEngine\\Check\\FileSystemTags',  
    name: t('workflowengine', 'File system tag'),  
    operators: [  
        { operator: 'is', name: 'is tagged with' },  
        { operator: '!is', name: 'is not tagged with' }  
    ],  
    component: FileSystemTag  
})
```

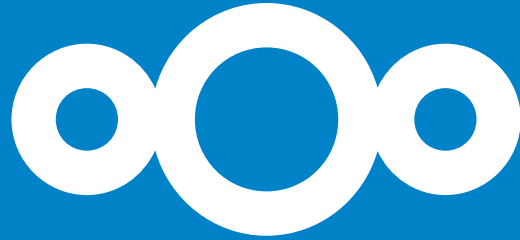
# Check with a custom component

When  **File created** ▼

and

File system tag	is tagged with	Select a tag	
Add a new filter		restricted  test	






# Keep your data secure and under your control

Nextcloud GmbH  
Hauptmannsreute 44A  
70192 Stuttgart

Germany

 +49.711-252-4280  
 [sales@nextcloud.com](mailto:sales@nextcloud.com)

 [nextcloud.com](https://nextcloud.com)