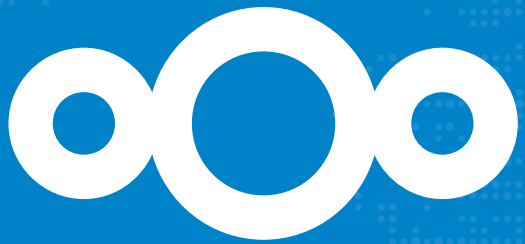


Building Nextcloud Flow

Ambition

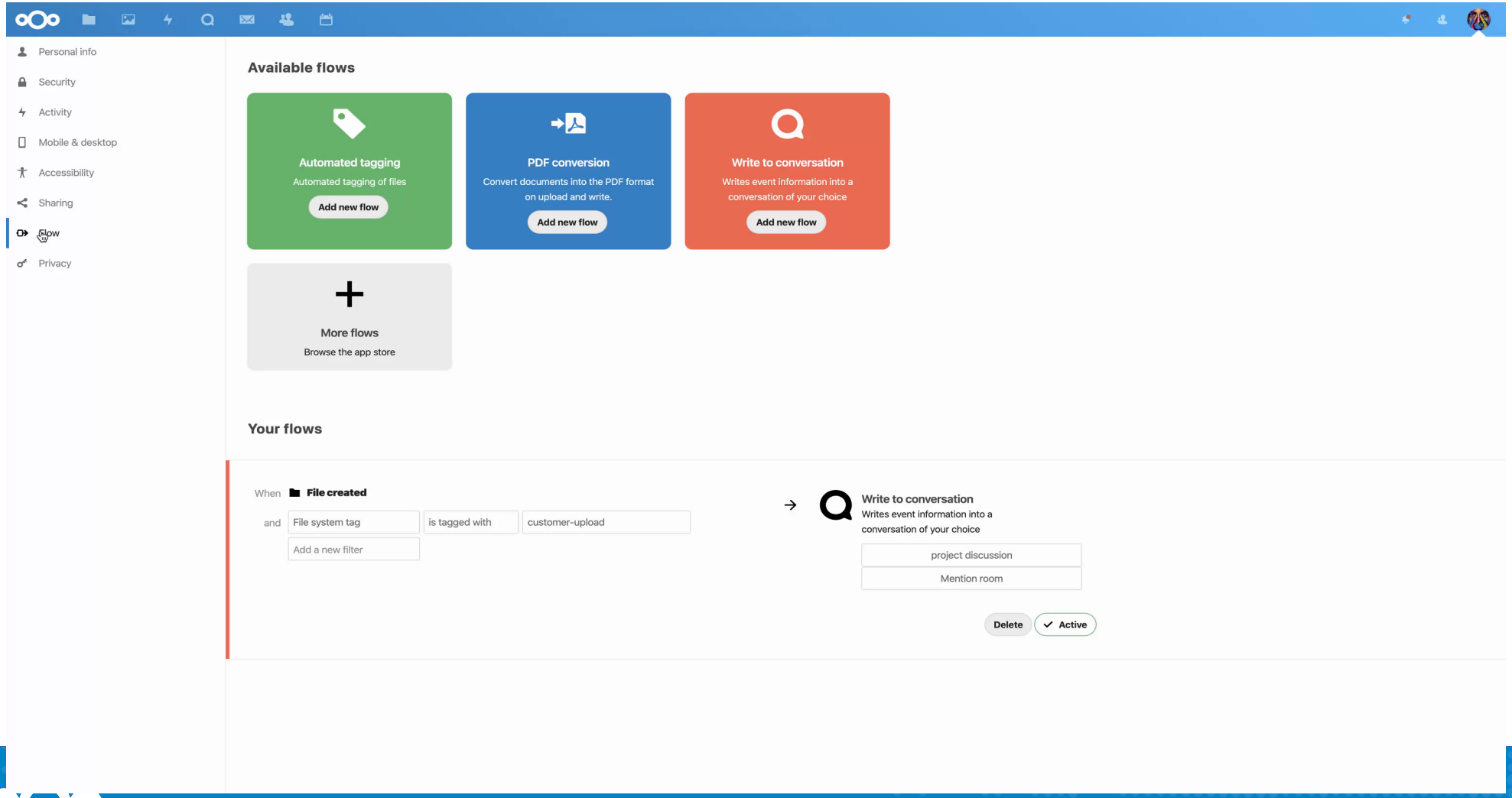
Flexible user-defined event-based task automation

Status Quo (up to Nextcloud 17)

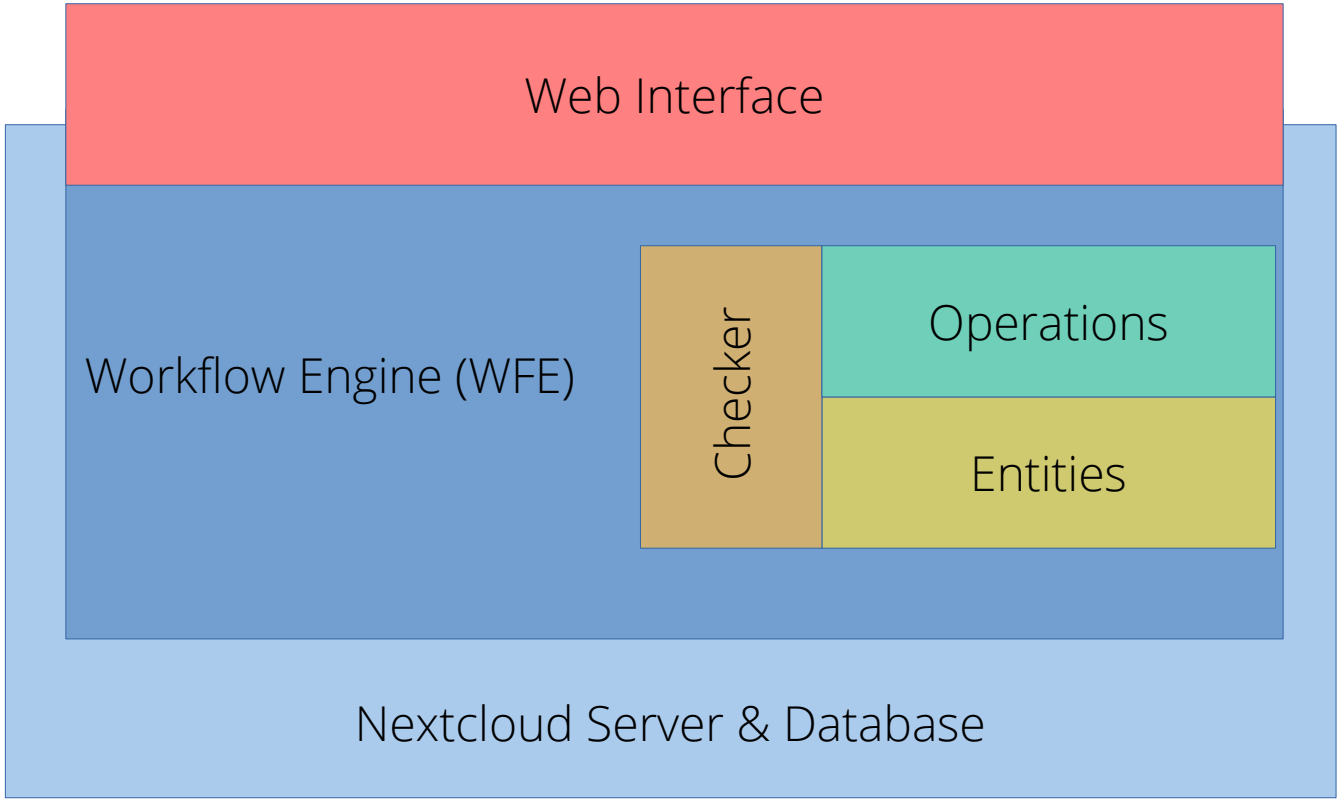
- Limited to files
- Limited to administrators
- Scattered, app-centered settings

Goals

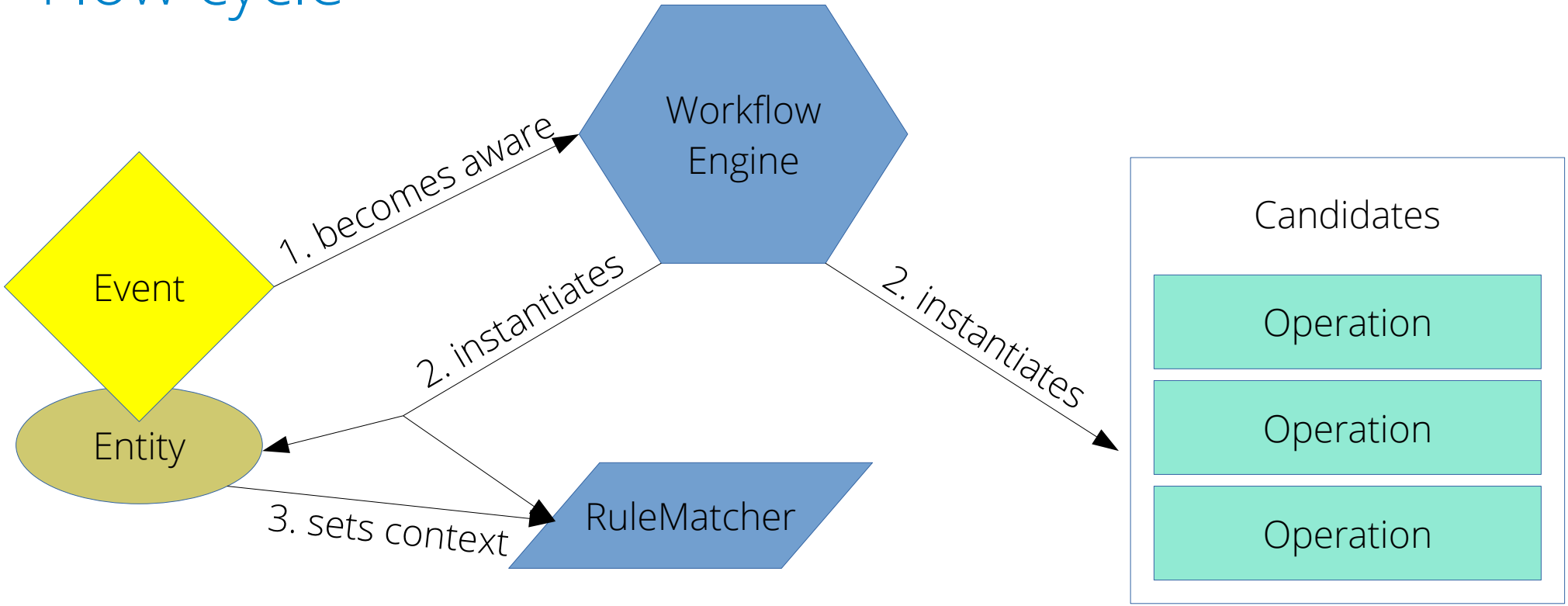
- Unified UI with good UX
- Empower regular users to set up Flows
- Backwards compatible configuration
- More available triggers and actions
- Easily extendable for developers


Personal info
Security
Activity
Mobile & desktop
Accessibility
Sharing
Flow
Privacy
Available flows
Automated tagging
Automated tagging of files
Add new flow
PDF conversion
Convert documents into the PDF format on upload and write.
Add new flow
Write to conversation
Writes event information into a conversation of your choice
Add new flow
More flows
Browse the app store
Your flows
When File created
and File system tag is tagged with customer-upload
Add a new filter
Write to conversation
Writes event information into a conversation of your choice
project discussion
Mention room
Delete
Active

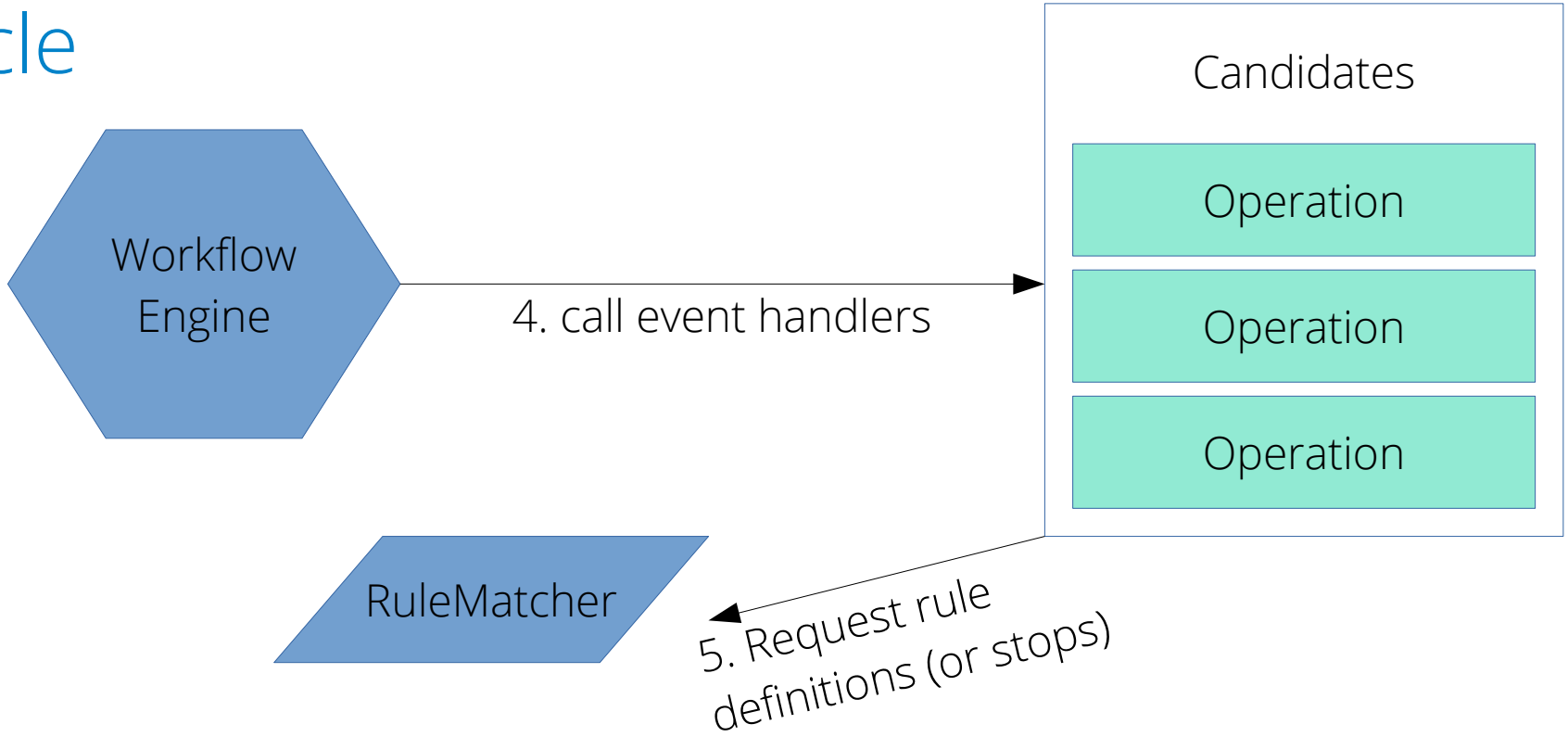
Components


Web Interface
Workflow Engine (WFE)
Checker
Operations
Entities
Nextcloud Server & Database

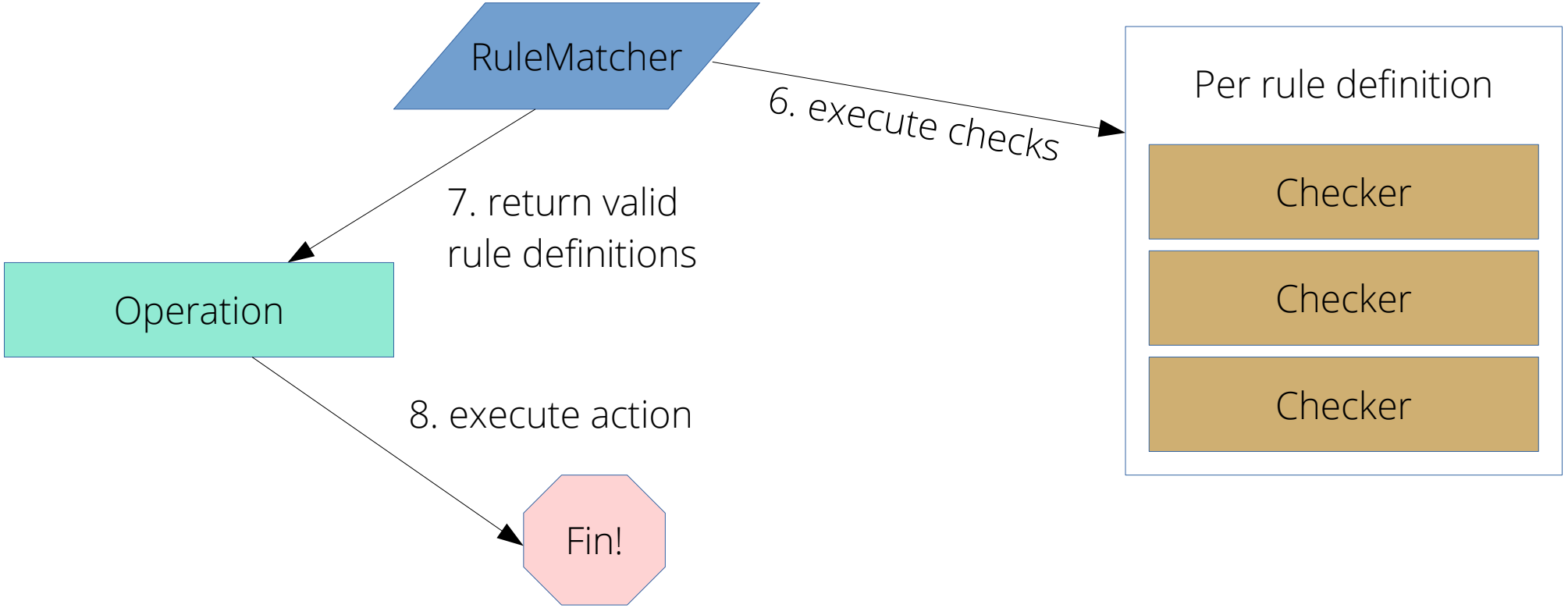
Flow cycle


Workflow Engine
Event
1. becomes aware
2. instantiates
2. instantiates
Entity
3. sets context
RuleMatcher
Candidates
Operation
Operation
Operation

Flow cycle


Workflow Engine
4. call event handlers
Candidates
Operation
Operation
Operation
RuleMatcher
5. Request rule definitions (or stops)

Flow cycle


RuleMatcher
6. execute checks
Per rule definition
Checker
Checker
Checker
7. return valid
rule definitions
Operation
8. execute action
Fin!

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

```php
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.

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

```php
interface IManager {

    const SCOPE_ADMIN = 0;
    const SCOPE_USER = 1;
```

Example: convert to PDF contd.

```php
/**
 * @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.

```php
public function onEvent(string $eventName, Event $event, IRuleMatcher $ruleMatcher): void {
    // .. skipped some precondition check left out
    try {
        // .. skipped some trivial info gathering and sanity checking
        $matches = $ruleMatcher->getFlows( 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 {
        return File::class;
    }
```

class File implements OCP\WorkflowEngine\IEntity

Accesscontrol is an IComplexOperation

```php
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

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

Operations have to register to the WFE

```php
public static function register(IEventDispatcher $dispatcher): void {
    $dispatcher->addListener( eventName: FlowManager::EVENT_NAME_REG_OPERATION, function (GenericEvent $event) {
        $operation = \OC::$server->query( name: Operation::class);
        $event->getSubject()->registerOperation($operation);
        Util::addScript( application: 'spreed',  file: 'flow');
    });
}
```

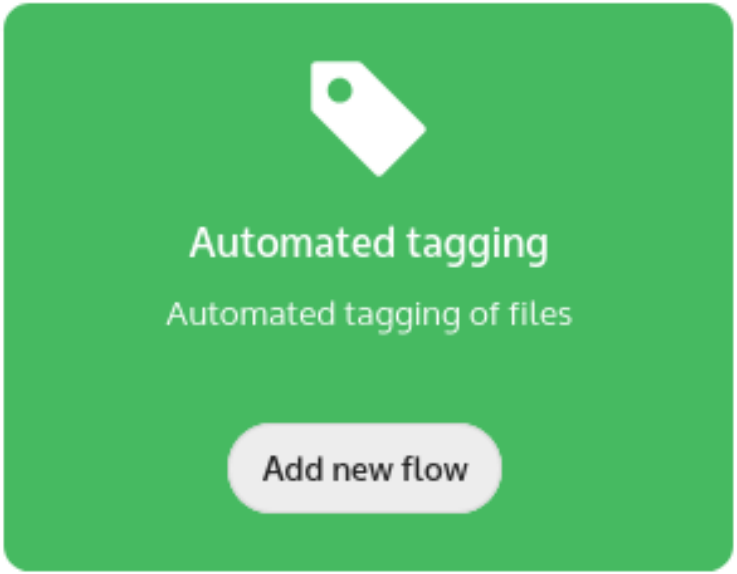
Register a new Operation in the frontend

- 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',
    icon-class: 'icon-tag-white',
    color: 'var(--color-success)',
})
```

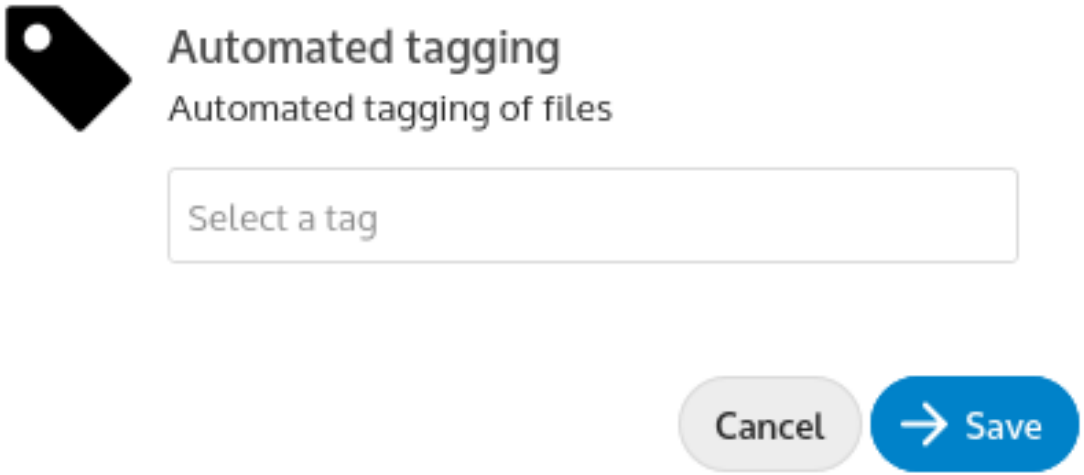
Frontend: Register a new Operation



Frontend: Register a new Operation

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

Frontend: Register a new Operation



When should it be done?

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

Example: File entity

```php
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.

```php
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

```php
/** 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.

```php
public function prepareRuleMatcher(IRuleMatcher $ruleMatcher, string $eventName, Event $event): void {
    if (!$event instanceof GenericEvent && !$event instanceof MapperEvent) {
        return;
    }
    $this->eventName = $eventName;
    $this->event = $event;
    try {
        $node = $this->getNode();
        $ruleMatcher->setEntitySubject($this, $node);
        $ruleMatcher->setFileInfo($node->getStorage(), $node->getInternalPath());
    } catch (NotFoundException $e) {
        // pass
    }
}
```

Legitimation check

```php
public function isLegitimatedForUserId(string $uid): bool {
    try {
        $node = $this->getNode();
        if($node->getOwner()->getUID() === $uid) {
            return true;
        }
        $acl = $this->shareManager->getAccessList($node, recursive: true, currentAccess: true);
        return array_key_exists($uid, $acl['users']);
    } catch (NotFoundException $e) {
        return false;
    }
}
```

Provide additional information

- Entity knows best about the subject to give proper hints
- OCP\WorkflowEngine\EntityContext**IDisplayName**, **IDisplayText**, **IIcon**, **IUrl**

```
public function getUrl(): string {
    try {
        return $this->urlGenerator->linkToRouteAbsolute( routeName: 'files.viewcontroller.showFile', ['fileid' =>
$this->getNode()->getId()]);
    } catch (InvalidPathException $e) {...} catch (NotFoundException $e) {...}
}
```

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

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

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

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

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

Example validator

```php
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

```php
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';
    }
}
```

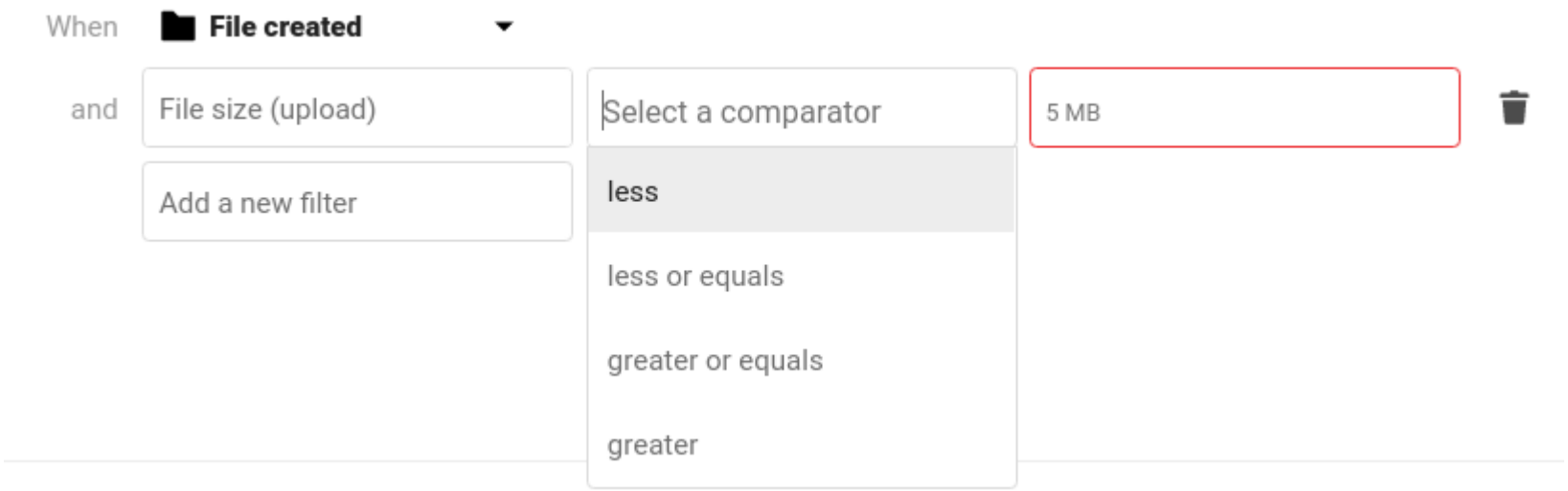
Register a new check in the frontend

- 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.registerCheck({
   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 ? check.value.match(/^[0-9]+[ ]?[kmgt]?
b$/i) !== null : false
},)
```

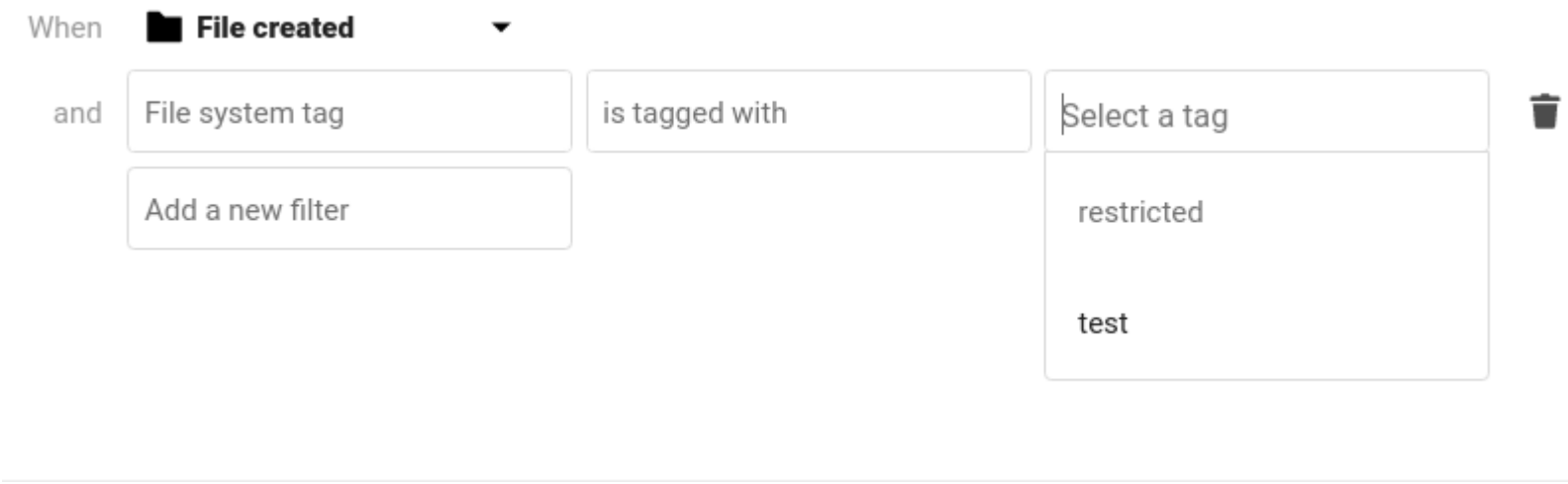
Basic check with validation

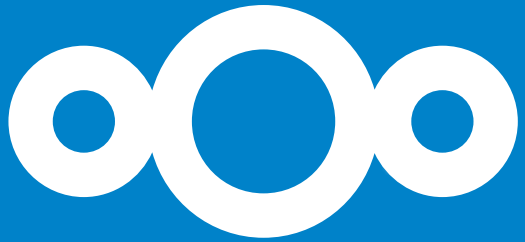


Register a new check with a custom component

```
window.OCA.WorkflowEngine.registerCheck({
   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





Keep your data secure and under your control

Nextcloud GmbH
Hauptmannsreute 44A
70192 Stuttgart

Germany

+49.711-252-4280
sales@nextcloud.com

nextcloud.com