
Workflows

Release 15.4.1.3

CONTACT Software

29.10.2018



Inhaltsverzeichnis

| | | |
|----------|---|-----------|
| 1 | Einleitung | 1 |
| 2 | Schnittstellen zu anderen Anwendungen | 2 |
| 2.1 | Digitale Signatur (cs.dsig) | 2 |
| 2.2 | Engineering Change Management (cs.ec) | 2 |
| 2.3 | Tasks (cs.taskmanager) | 2 |
| 3 | Monitoring | 3 |
| 3.1 | Workflow-Dienst | 3 |
| 3.2 | E-Mail Benachrichtigungen | 4 |
| 4 | Fehlerbehandlung | 5 |
| 4.1 | Analyse | 5 |
| 4.2 | Behebung | 5 |
| 4.3 | Wiederholung | 6 |
| 5 | Workflow-Vorlagen | 8 |
| 5.1 | Vorlagen transportieren | 8 |
| 6 | Konfiguration | 10 |
| 6.1 | Mappen | 10 |
| 6.2 | Constraints | 12 |
| 6.3 | Formulare | 14 |
| 6.4 | Systemaufgaben | 15 |
| 6.5 | Workflow Designer | 18 |
| 6.6 | Sonstiges | 18 |
| 7 | FAQ | 20 |
| 7.1 | Systemaufgaben werden nicht abgeschlossen | 20 |
| 8 | Glossar | 21 |
| | Stichwortverzeichnis | 26 |

Einleitung

Bemerkung: Die Produktentwicklung stellt hohe Anforderungen an die Prozessunterstützung. Kennzeichnend sind z.B. ein hohes Maß an Parallelität und nicht vorhergesehener Änderungen und Korrekturen im Projektverlauf. Herkömmliche Lösungen stoßen hier an ihre Grenzen. *CONTACT Workflows* berücksichtigt diese Anforderungen umfassend.

CONTACT Workflows unterstützt eine große Bandbreite an Geschäftsprozessen direkt im System. Dabei können interaktive und automatisierte *Aufgaben* miteinander in zeitlicher und logischer Abhängigkeit zu einer Struktur, genannt *Workflow*, kombiniert werden.

Ein *Workflow* ist aktiv oder gestartet, wenn er sich im Status  **Umsetzung** befindet. Eine *Aufgabe* ist aktiv oder gestartet, wenn sie sich im Status  **Umsetzung** befindet.

Ein *Workflow* kann entweder direkt von berechtigten Anwendern oder aus einer von Fachanwendern definierten Vorlage instanziiert werden. Vor dem Start wird eine Konsistenzprüfung durchgeführt, bei der unter anderem sichergestellt wird, dass die Verantwortlichkeit aller interaktiven *Aufgaben* geregelt ist und alle *Systemaufgaben* die notwendigen *Parameter* besitzen.

Für Administratoren gibt es im Umfeld von *CONTACT Workflows* vorrangig zwei Aufgaben: *Monitoring* (Seite 3) und *Fehlerbehandlung* (Seite 5).

Des Weiteren finden Sie in diesem Handbuch technische Details zu einzelnen Komponenten.

Bemerkung: Für Details zur generellen Anwendung und Definition eines *Workflows*, lesen Sie bitte das `wf_user:workflow_user`.

Schnittstellen zu anderen Anwendungen

2.1 Digitale Signatur (cs.dsig)

Die Anwendung *CONTACT Digital Signatures* erweitert *CONTACT Workflows* um einen neuen Aufgabentyp. Um diese abzuschließen, muss ein Verantwortlicher seine digitale Signatur an der Aufgaben oder bestimmten Objekten anbringen.

2.2 Engineering Change Management (cs.ec)


CONTACT Engineering Changes verwendet *CONTACT Workflows* für die Modellierung der einzelnen in einem Engineering Change vorkommenden Prozessschritte („Engineering Change Request“, „Engineering Change Order“ und „Engineering Change Notification“).

2.3 Tasks (cs.taskmanager)

CONTACT Workflows enthält ein Plugin für *CONTACT Tasks* zur Integration seiner Aufgaben.

Bemerkung: Dieses Kapitel enthält Informationen über die technische Funktionsüberwachung von *CONTACT Workflows*. Für Informationen über das Monitoring eigener laufender *Workflows* lesen Sie bitte das `wf_user:workflow_user`.

3.1 Workflow-Dienst

Läuft der *Workflow-Dienst* nicht, hat das auf das laufende System ausschließlich den Effekt, dass *Systemaufgaben* im Status  **Umsetzung** nicht abgearbeitet werden. Ihre *Workflows* pausieren also effektiv, was zu Verzögerungen im Geschäftsprozess führen kann.

Einen Überblick über bevorstehende und fehlgeschlagene Aufträge finden Sie in der Warteschlange (im Navigationsmenü erreichbar unter *Administration/Konfiguration* -> *Administration* -> *Prozesse* -> *Systemaufgabenaufträge*). Erfolgreich abgeschlossene Aufträge werden aus der Warteschlange entfernt.

3.1.1 Impersonation

Die Abarbeitung der *Systemaufgaben* geschieht in einem separaten Unterprozess des Dienstes. Damit Rechte korrekt geprüft und Protokolle korrekt geschrieben werden, wird der Unterprozess mit dem Konto des Benutzers gestartet, der den *Workflow* gestartet hat. Dieser fungiert als eindeutiger *Workflow*-Verantwortlicher, da die Konfiguration auch die Nennung einer Rolle als Verantwortlicher erlaubt. In der Datenbank finden Sie die Benutzer-ID im Attribut `started_by` der Relation `cdbwf_process`.

Warnung: Nach Deaktivierung eines Benutzerkontos schlagen sämtliche *Systemaufgaben* fehl, deren *Workflow* von diesem Benutzer gestartet wurden.

3.1.2 Lizenzprüfung

Für Lizenzprüfungen im Prozess jeder *Systemaufgabe* wird stellvertretend ein Service-Benutzerkonto herangezogen. Welches Konto benutzt wird, kann in den Dienstargumenten mit `--svcuser` festgelegt werden.

CONTACT Workflows liefert zu diesem Zweck das vorkonfigurierte Benutzerkonto `cs.workflow.svcuser` mit.

Bemerkung: Der *Workflow-Dienst* prüft die Lizenzen des *Workflow-Besitzers* bei Ausführung einer *Systemaufgabe*, belegt sie aber nicht dauerhaft, um den interaktiven Betrieb nicht zu stören.

3.1.3 Logging

Das Logverhalten der Unterprozesse kann in `$CADDOK_BASE/etc/wfqueue.conf` festgelegt werden.

3.2 E-Mail Benachrichtigungen

Anwender können auf Wunsch E-Mail Benachrichtigungen bei Eintreten bestimmter Ereignisse erhalten.



Damit diese auch tatsächlich verschickt werden können, muss ein E-Mail Server konfiguriert werden (siehe Umgebungsvariablen `CADDOK_MAIL*` und `CADDOK_SMTP*` in der `site.conf`).

Darüber hinaus besteht die Möglichkeit, durch Setzen der Umgebungsvariable `CADDOK_STOP_EMAIL_NOTIFICATION` (ebenfalls in der `site.conf`) E-Mail Benachrichtigungen aus *CONTACT Workflows* pauschal zu deaktivieren. Dies ist aber in erster Linie für Test- und Entwicklungssysteme gedacht.

Fehlerbehandlung

Um zu beurteilen, ob das Fehlschlagen einzelner *Aufgaben* erwartungskonform ist oder nicht, sollten Sie zunächst mit der Funktionsweise von *Aufgaben* und *Constraints* vertraut sein. Lesen Sie dazu bitte das `wf_user:workflow_user`.

Darüber hinaus können natürlich auch tatsächliche Fehler auftreten, deren Ursache und Behandlung in diesem Kapitel behandelt werden.

Fehler während der Abarbeitung eines *Workflows* oder einer *Aufgabe* resultieren immer in einer Änderung des *Workflow*-Status in  **Fehlgeschlagen**. In einigen Fällen wird auch der Status der *Aufgabe* in  **Verworfen** geändert.

Im Umfeld von *CONTACT Workflows* empfiehlt sich bei unerwarteten Fehlern folgendes Vorgehen:

1. Analyse
2. Behebung
3. Wiederholung

Nicht in jedem Fall sind alle Schritte tatsächlich notwendig.

4.1 Analyse

Ist ein *Workflow* aus ungeklärter Ursache abgebrochen worden, sollten zunächst die letzten Einträge im *Protokoll* des *Workflows* analysiert werden. In den meisten Fällen findet sich hier der Grund für den Abbruch im Klartext.


Unerwartete Fehler können auch einen „Traceback“ ins *Protokoll* schreiben, d.h. die Fehlermeldung aus PowerScript inklusive der letzten Aufrufstellen. Die Analyse von Tracebacks setzt eine gewisse Kenntnis von PowerScript voraus.

Weitere Informationsquellen sind die Serverlogs des *Workflow-Dienstes* und seiner Queue.

4.2 Behebung

Die Fehlerbehebung hängt unmittelbar mit der logischen Ursache zusammen. Einige Fehler erfordern keine administrative Behebung, andere dagegen schon. In diesem Abschnitt finden Sie beispielhaft einige Fehlerklassen.

4.2.1 Unerwarteter Abbruch von Systemaufgaben

Während der Status einiger interaktiver *Aufgaben* von ihren Verantwortlichen explizit in  **Abgelehnt** geändert werden kann, kann dies bei der Abarbeitung von *Systemaufgaben* aus den unterschiedlichsten Gründen ebenfalls passieren.

Ein Beispiel: Eine *Systemaufgabe* „Statusänderung“ soll den Status ihrer *Mappeninhalte* in 200 ändern.

Beispiel Vorbedingungen nicht hergestellt

Befindet sich in unserem Beispiel ein Objekt im *Mappeninhalt*, für welches kein Statusübergang vom seinem aktuellen Status zu 200 definiert ist, wird die *Aufgabe* fehlschlagen.

Wurde vom Verantwortlichen einer Vorgängeraufgabe erwartet, den nötigen Zustand für die Statusänderung herzustellen, liegt der Fehler also bei ihm und es gibt keine Notwendigkeit für administratives Eingreifen.

Beispiel Fehlerhaftes Workflow-Design

Um einen Fehler im Entwurf des *Workflows* handelt es sich beispielsweise, wenn


- die Voraussetzung für die Statusänderung automatisch hergestellt werden hätten sollen oder
- im Fehlerfall einfach keine Statusänderung versucht werden soll.

Solche Fehler müssen durch Anpassung der Vorlage gelöst werden. Basiert der *Workflow* nicht auf einer Vorlage, muss einer der *Workflow*-Verantwortlichen die Korrektur des *Workflows* in einer Kopie vornehmen.

Beispiel Fehler in der Aufgabenlogik

Liegt der Fehler in der Logik der *Aufgabe* selbst, muss er durch einen Entwickler behoben werden. Dies ist in der Regel daran erkennbar, dass das *Protokoll* einen PowerScript-Traceback enthält.

4.2.2 Konfigurationsfehler

Ist eine *Aufgabe* nicht korrekt konfiguriert, kann sie ebenfalls fehlschlagen. Benötigte Attribute oder *Parameter* könnten z.B. nicht gesetzt sein. Diese Regeln werden schon direkt bei Änderung des *Workflow*-Status in  **Umsetzung** geprüft, um solche Fälle frühzeitig zu erkennen. Einige dieser Vorbedingungen können aber noch nach Start des *Workflows* geändert werden.

Konfigurationsfehler haben ihre Ursache im Entwurf des *Workflows* und müssen durch einen *Workflow*-Verantwortlichen gelöst werden.

4.2.3 Systemfehler

Unerwartete Fehler können z.B. durch Customizing oder Programmierfehler verursacht werden. In diesem Fall ist eine weitere Analyse dieser Bausteine erforderlich.

4.3 Wiederholung

Sind die Ergebnisse des fehlgeschlagenen *Workflows* nicht ausreichend, kann eine korrigierte Version des *Workflows* gestartet werden. Basierte der Original-*Workflow* auf einer Vorlage, sollte die Vorlage korrigiert und anschließend neu instanziiert werden.

Sind für die Wiederholung Teile des Original-*Workflows* (die bspw. bereits erfolgreich abgearbeitet wurden) nicht mehr relevant, können diese einfach vor dem Start der Kopie des *Workflows* entfernt werden.

Workflow-Vorlagen

Üblicherweise werden wiederholbare *Workflows* durch Vorlagen gesteuert. Diese können Fachanwender selbst erstellen, ggf. mit Regeln behaften und freigeben.

5.1 Vorlagen transportieren

Da *Workflow*-Vorlagen inhärenter Bestandteil der Geschäftslogik sind, sollten diese wie ein Bestandteil des Customizings behandelt und stets zwischen den Systemen (Entwicklung, Test, Produktion) synchron gehalten werden.

Dies ist mit der folgenden Werkzeugkette möglich:

1. `cs/workflow/updates/tools/export_process.py`
2. `cdbexp`
3. `cdbimp`

Warnung: Um Konflikte zwischen IDs zu vermeiden, müssen Sie Vorkehrungen treffen. Mögliche Lösungen finden Sie unter *Strategien zur Vermeidung von Konflikten zwischen IDs* (Seite 9).

Warnung: Transportiert wird lediglich der *Workflow* mitsamt seiner Struktur (*Aufgaben, Mappen, Constraints...*) und Informationen (*Protokoll, Activities*). Explizit vom Transport ausgeschlossen sind referenzierte Objekte (nicht jedoch die Referenzen selbst), bspw. *Mappeninhalte* und Anhänge zu Beiträgen und Kommentaren im Activity Stream.

5.1.1 Vorgehen

Zuerst sollte der zu transportierende *Workflow* im Quellsystem ausgiebig getestet werden. Am wenigsten fehleranfällig ist es, *Workflows* immer nur in einer festgelegten Reihenfolge zu transportieren (z.B. von der Entwicklung in den Test, vom Test in die Produktion).

Anschließend kann mit dem mitgelieferten Kommandozeilentool `export_process` eine Steuerdatei für `cdbexp` erstellt werden. Diese Datei beinhaltet die Datenbankabfragen, die zum Export des *Workflows* und seiner Struktur benötigt werden:

```
powerscript -m cs.workflow.updates.tools.export_process P00000000 > P.ctl
```

Falls der Aufruf fehlerfrei beendet wird, finden Sie in der Ausgabedatei (im Beispiel `P.ctl`) die Datenbankabfragen. Die Abfragen können als Eingabe für `cdbexp` benutzt werden:

```
cdbexp -c P.ctl -o P.exp
```

Die Ausgabedatei `P.exp` enthält nun die exportierte *Workflow*-Struktur, die im Zielsystem importiert werden kann (bitte beugen Sie Konflikten zwischen IDs vor):

```
cdbimp P.exp
```

5.1.2 Strategien zur Vermeidung von Konflikten zwischen IDs

IDs von *Workflows* (`cdbwf_process.cdb_process_id`) und *Aufgaben* (`cdbwf_task.task_id`) werden in der Standardkonfiguration mit einer fortlaufenden Nummer (P000... bzw. T000...) generiert. Dieser Mechanismus funktioniert nicht systemübergreifend, weshalb beim Transport von *Workflows* und *Aufgaben* Konflikten vorgebeugt werden muss. Im Folgenden werden einige Lösungsansätze skizziert.

Separate Nummernkreise

Die einfachste Möglichkeit ist, in jedem System getrennte Nummernkreise zu verwenden. So könnte jedes System einen dedizierten Prefix für die Nummern verwenden (Bsp.: Entwicklungssystem verwendet für *Workflow*-Nummern X000... statt T000...).

Wird dieser Ansatz verfolgt, muss beim Transport weiter nichts beachtet werden.

Nummern beim Transport separieren

Der Export liegt als Textdatei vor. Mit einem geeigneten Editor können Sie sämtliche potentiell konfigierenden Nummern ersetzen. Diese Lösung ermöglicht unter anderem auch die Nutzung etwas sprechenderer IDs für Vorlagen, da sowohl IDs von *Workflows* als auch *Aufgaben* auch rein textuell definiert sein dürfen.

Nummern zusammenführen

Sollen systemübergreifend einheitliche Nummern für *Workflow*-Vorlagen verwendet werden, sollten zumindest die Nummernkreise der Vorlagen von denen der Instanzen getrennt werden.

Wird dann eine neue Vorlage angelegt, sollte unmittelbar in allen Systemen die gleiche Nummer „reserviert“ werden. Unmittelbar vor dem Transport der fertigen Vorlage kann der Platzhalter in den Zielsystemen dann gelöscht werden.

6.1 Mappen

Dieses Kapitel enthält technische Voraussetzungen in Bezug auf die Nutzung von *Mappen*.

6.1.1 Mappeninhalte

Eine *Mappe* kann Objekte unterschiedlicher Klassen enthalten. Ein Objekt muss alle folgenden Bedingungen erfüllen, um als *Mappeninhalt* verwendbar zu sein:

- Es muss das Attribut `cdb_object_id` besitzen.
- Es muss eine PowerScript-Klasse aufweisen.

Zusätzlich kann ihre PowerScript-Klasse von der Klasse `cs.workflow.briefcases.BriefcaseContent` abgeleitet werden, um einige vordefinierte Event Handler (z.B. für die Objektoperation *Workflow/Neu*) zu verwenden.

Im Standard sind schon einige Klassen (z.B. *Dokumente*) für die Verwendung als *Mappeninhalt* vorkonfiguriert.

Sie können in Ihrer Umgebung weitere Klassen für die Verwendung als *Mappeninhalt* freigeben, indem Sie die oben genannten Bedingungen erfüllen.

Beziehungen

Mappen referenzieren immer einen *Workflow*. Lokale *Mappen* sind darüber hinaus noch mit beliebig vielen *Aufgaben* innerhalb ihres *Workflows* verknüpft.

Die Zuordnung eines Objekts zu einer *Mappe* erfolgt über die Plattformklasse `cdbfolder_content` und hat somit keinen direkten Bezug zum *Workflow*.

Wollen Sie eine Beziehung zwischen Objekten einer Klasse und ihren verknüpften *Workflows* konfigurieren, können Sie dies auf Basis der Klasse `cdbwf_process_content` tun, die für Sie die *Mappeninhalte* pro *Workflow* redundant vorhält.

Diese Klasse kann auch als PowerScript-Referenz verwendet werden. Die entsprechende PowerScript-Klasse ist `cs.workflow.briefcases.BriefcaseReference`.

6.1.2 Automatische Rechtevergabe

Mit *Workflows* besteht die Möglichkeit, die Bearbeitungsrechte der *Mappeninhalte* automatisch an die Verantwortlichen von laufenden *Aufgaben* zu vergeben. Weitere Informationen hierzu finden Sie im Anwenderhandbuch.

Die automatische Rechtevergabe basiert auf einem *Beziehungsrechteprofil* und ist im Standard auf einige vorbestimmte Typen eingeschränkt (z.B. Artikel, Dokumente, usw.). Sie können in Ihrer Umgebung die automatische Rechtevergabe auf weitere Typen erweitern. Dafür legen Sie eine *Beziehung* zwischen der Klasse `cdbwf_briefcase` (als *Referer*) und ihrer Zielklasse (als *Reference*) an.

Verwenden Sie dabei die folgenden Parameter:

Referer `cdbwf_briefcase`



Verknüpfungsklasse `cdbfolder_content`



Reference Ihre Zielklasse

Keymap (Referer) `cdb_object_id=cdb_folder_id`

Keymap (Reference) `cdb_object_id=cdb_content_id`

Beziehungsrechteprofil `cdbwf_assign_rights`

Bei der Vergabe der Rechte sollten Sie darauf achten, dass der erwartete Effekt je nach *Bearbeitungsmodus* auch eintritt. *Inhalte* von  **Info Mappen** sollten Verantwortliche ansehen, aber nicht ändern dürfen (solange die Änderungsrechte nicht über einen anderen Weg gewährt werden), *Inhalte* von  **Edit Mappen** sowohl sehen als auch ändern. Die Standardkonfiguration für Dokumente vergibt beispielsweise diese Berechtigungen:

| <i>Bearbeitungsmodus</i> | Berechtigung |
|---|--|
|  Info | read und read_file |
|  Edit | read, read_file, save, accept, lock und unlock |

Hinweis: Sie können die Rechtevergabe anpassen, um weitere Rechte zu vergeben. Kontaktieren Sie dafür Ihren Systemadministrator.

Berechtigungen auf Mappeninhalten

Damit ein Anwender Objekte in *Mappen* nutzen darf, muss er - je nach *Bearbeitungsmodus* - das Recht `cdbwf_obj_info` bzw. `cdbwf_obj_edit` auf der *Mappe* besitzen. Dies soll verhindern, dass durch einen *Workflow* Rechte erschlichen werden, die dem einrichtenden Benutzer vorher überhaupt nicht gewährt wurden.

Die Rechte `cdbwf_obj_info` und `cdbwf_obj_edit` sind von Rechten auf den der *Mappe* zugeordneten Objekten abgeleitet (über das *Beziehungsrechteprofil* `cdbwf_assign_rights`) und werden immer geprüft, wenn

- ein Objekt zum Inhalt einer *Mappe* hinzugefügt wird,
- der *Bearbeitungsmodus* einer *Mappe* geändert wird,
- eine *Mappe* zu weiteren *Aufgaben* hinzugefügt wird oder
- ein *Workflow* oder eine *Aufgabe* gestartet wird.

Im Auslieferungszustand sind diese Rechte z.B. für Dokumente so eingeschränkt, dass durch die Zuordnung keine Weitergabe von Berechtigungen möglich ist, die der Zuordnende nicht selbst besitzt.

Für bestimmte Objekte kann es gewünscht sein, diese niemals (oder nur durch bestimmte Anwenderkreise) in *Workflows* oder *Mappen*, die Schreibrechte gewähren, zu nutzen. Ein Grund könnte z.B. die Gefahr der Freigabe

der Objekte im *CONTACT Collaboration Portal* sein. In diesem Fall empfiehlt sich eine entsprechende Einschränkung der als *Mappeninhalt* nutzbaren Objekte.

6.1.3 Fehlende „Attachments“-Mappe in Workflow-Vorlagen ergänzen

Falls an einer *Workflow*-Vorlage keine „Attachments“-*Mappe* hinterlegt wurde, wird sie nach einem Update nicht automatisch eingefügt. Mit dem folgenden PowerShell-Code kann die *Mappe* ergänzt werden, falls sie noch nicht existiert (im Codebeispiel für alle *Workflow*-Vorlagen):

```
from cs.workflow.processes import Process

for process in Process.KeywordQuery(is_template="1"):
    process.make_attachments_briefcase()
```

6.2 Constraints

6.2.1 Verwendbare Objektregeln im Workflow Designer

Constraints sind *Objektregeln*, aber nicht alle Regeln eignen sich als *Constraint*. Im *Workflow Designer* stehen nur Regeln zur Verfügung, die für die Definition von *Constraints* freigeschaltet wurden.

Im Auslieferungszustand gilt dies nur für Regeln, deren Name mit `wf-designer:` beginnt. Durch Anpassung der PowerShell-Liste `cs.workflow.designer.catalogs.ConstraintsCatalog.__catalog_rules_conditions__` kann diese Konvention an Ihre Bedürfnisse angepasst werden.

Die Liste enthält SQL-Bedingungen. Im *Workflow Designer* stehen alle Objektregeln zur Verfügung, die mindestens eine dieser *Constraints* erfüllen.

Beispiel

```
from cs.workflow.designer.catalogs import ConstraintsCatalog
catalog_conditions = ConstraintsCatalog.__catalog_rules_conditions__
catalog_conditions.append("name LIKE 'constraint-rule:%'")
```

6.2.2 Eigene Objektregeln für Constraints definieren

Bei Prüfung eines *Constraints* ohne Zuordnung zu einer spezifischen *Mappe* wird die Objektregel gegen die *Aufgabe* des *Constraints* geprüft. Andernfalls erfolgt die Prüfung gegen die *Mappe*.

Beispiel Alle Mappendokumente sind freigegeben

Die Objektregel soll bei Prüfung gegen eine *Mappe* erfüllt sein, wenn die *Mappe* keine oder ausschließlich freigegebene Dokumente enthält. Enthält die *Mappe* mindestens ein nicht freigegebenes Dokument, soll die Objektregel nicht erfüllt sein.

Das Attribut `Content` enthält eine Liste der *Mappeninhalte*. Diese werden mit dem Filter „Dokument ist nicht freigegeben“ weiter eingeschränkt. Die gefilterte Liste muss anschließend leer sein (bzw. ihre Länge muss 0 sein).

Beispiel Alle Aufgabendokumente sind freigegeben

Dieses Beispiel funktioniert analog zum vorhergehenden Beispiel. Allerdings ist diese Regel dazu gedacht, gegen eine *Aufgabe* geprüft zu werden. `Content` enthält in diesem Fall die Inhalte aller *Mappen* der *Aufgabe*.

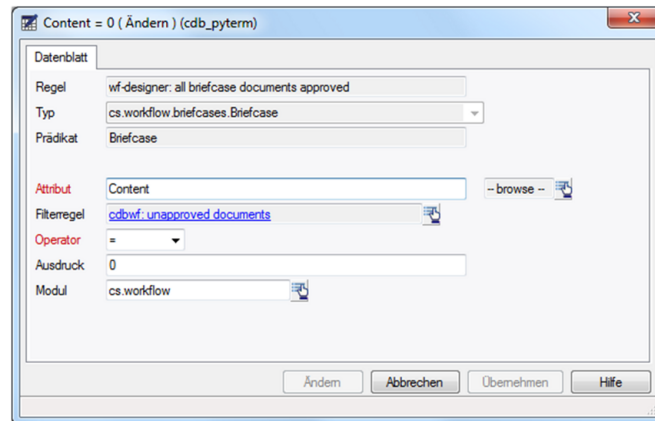


Abb. 6.1: Objektregel Alle Mappendokumente sind freigegeben

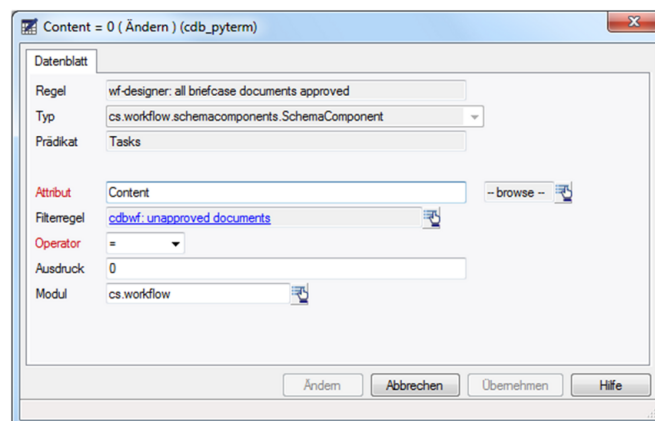


Abb. 6.2: Objektregel Alle Aufgabendokumente sind freigegeben

Um genauer zu unterscheiden, bieten *Aufgaben* auch die Attribute `InfoContent` und `EditContent` an, die nur Inhalte des jeweiligen *Bearbeitungsmodus* liefern.

6.3 Formulare

Formulare nutzen die *Maskenkonfiguration* und werden wie Masken mit dem *Formular* selbst als Kontextobjekt dargestellt. Damit eine Maske als *Formular* genutzt werden kann, muss eine Formularvorlage angelegt, benannt und freigegeben werden.

Die Daten eines *Formulars* werden als JSON (JavaScript Object Notation) in seinem Langtextattribut `cdbwf_form_contents_txt` abgelegt.

6.3.1 Formularvorlagen

Unter *Administration/Konfiguration* -> *Katalogverwaltung* -> *Prozesse* -> *Formularvorlagen* können neue Formularvorlagen angelegt oder bestehende gefunden werden. Folgende Daten sind bei der Neuanlage relevant:

Name Benennung des *Formulars* in allen Login-Sprachen. Die Benennung sollte den Einsatzzweck klar verständlich machen.

Maske Name einer *Maskenkonfiguration*. Diese gibt sowohl Aussehen als auch den Attributvorrat des *Formulars* vor. Außerdem kommen die üblichen Mechanismen (Kataloge, Pflichtfelder und Vorbelegungen) der Maske zur Anwendung.

Warnung: Es werden nur Masken, keine Maskenverbünde unterstützt.

Warnung: Es werden nur explizite Pflichtfelder ausgewertet. Da *Formulare* kontextlos sind, werden keine Pflichtfelder aus dem Data Dictionary angewandt.


Formularvorlagen können in *Workflows* ausschließlich im Status  **Freigegeben** verwendet werden. Die Rücknahme der Freigabe hat keine Auswirkung auf bereits aus der Vorlage instanziierte *Formulare*.



Abb. 6.3: Statusnetz einer Formularvorlage

6.3.2 Zugriffsrechte

Warnung: In diesem Abschnitt werden die Berechtigungen im Auslieferungszustand beschrieben. Die Berechtigungen in Ihrem System können davon abweichen.

Zugriffsrechte auf Formularvorlagen

| Bedingung | Recht | Gewährt für |
|-------------------------|---|---|
| Status Neu | <ul style="list-style-type: none"> • Vollzugriff | <ul style="list-style-type: none"> • <i>Administrator</i> • <i>Administrator: Master Data</i> • <i>cdbwf: Process Administrator</i> • <i>cdbwf: Process Library Manager</i> |
| Status nicht Neu | <ul style="list-style-type: none"> • Lesen • Statusänderung | <ul style="list-style-type: none"> • <i>Administrator</i> • <i>Administrator: Master Data</i> • <i>cdbwf: Process Administrator</i> • <i>cdbwf: Process Library Manager</i> |

Zugriffsrechte auf Formularen

Das Leserecht für *Formulare* ist grundsätzlich nicht eingeschränkt.

Zugriffsrechte für *Formulare* werden Verantwortlichen einer *Aufgabe* durch die jeweils relevante Zuordnung zu einer *Mappe* gewährt.

Verantwortliche eines *Workflows* erhalten die gleichen Rechte wie auf den *Aufgaben*. Die Berechtigungen auf *Formulare* werden über das Beziehungsrechteprofil *Workflow components* auf das *Workflow-Recht edit schema* abgebildet.

6.4 Systemaufgaben

Bitte beachten Sie, dass *Systemaufgaben* durch den *Workflow-Dienst* ausgeführt werden.

6.4.1 Systemaufgaben konfigurieren

Im Auslieferungszustand sind einige *Systemaufgabendefinitionen* vorkonfiguriert. Sie können Ihre Umgebung um weitere Definitionen erweitern. Die notwendigen Schritte dafür werden im *Beispiel: Eigene Systemaufgabe erstellen* (Seite 17) erläutert.

Definition einer Systemaufgabe anlegen

Zuerst müssen Sie eine neue *Systemaufgabendefinition* anlegen. Den Zugang dazu finden Sie im Navigationsmenü unter *Administration/Konfiguration -> Katalogverwaltung -> Prozesse -> Systemaufgabendefinitionen*.

In der Neuanlagemaske können Sie die folgenden Felder eintragen:

Identifikator Eine eindeutige Bezeichnung

Name (de) Deutscher Name

Name (en) Englischer Name

Vollqualifizierter Python-Name Der Name eines aufrufbaren PowerScript-Objekts (typischerweise eine Funktion), welches die Logik der *Systemaufgabe* implementiert.

Systemaufgaben implementieren

Zunächst können Sie die PowerScript-Funktion für die neue *Systemaufgabendefinition* schreiben. Diese muss unter dem eingetragenen Python-Namen aufrufbar sein.

Das System ruft für jede aktive *Systemaufgabe* dieser Definition die Funktion mit den folgenden Parametern auf:

task Die *Systemaufgabe* als `cdb.objects.Object`.

content Ein PowerScript-Dictionary, das die *Mappeninhalte* enthält. Die Schlüssel sind die möglichen *Bearbeitungsmodi* („info“, „edit“) eines Objekts. Die Werte sind Listen mit den Objekten der jeweiligen Inhalte.

****kwargs** Die restlichen Parameter entsprechen den in der Definition konfigurierten *Parameter* (siehe *Definition von Parametern* (Seite 16)).

Fehlerbehandlung

Implementierungen von *Systemaufgaben* sollten alle möglichen Fehler abfangen und in eine für den Benutzer lesbare Meldung umwandeln.


Soll die *Systemaufgabe* nach dem Fehler abgebrochen werden, kann an Stelle des ursprünglichen Fehlers ein Objekt der Klasse `cs.workflow.systemtasks.TaskCancelledException` geworfen werden. Der Fehler wird dann vom System protokolliert.

Um den gesamten *Workflow* abzubrechen, kann die Klasse `cs.workflow.systemtasks.ProcessAbortedException` verwendet werden.

Darstellung im Workflow Designer anpassen

Mit der Operation *Bild importieren* im Kontextmenü einer *Systemaufgabendefinition* können Sie die Darstellung von *Systemaufgaben* dieser Definition im *Workflow Designer* bestimmen.

Definition von Parametern

Für jede *Systemaufgabendefinition* kann eine Liste an erforderlichen *Parameter* angegeben werden. Jeder dieser *Parameter* muss für die Abarbeitung der *Systemaufgabe* existieren, ansonsten reagiert diese direkt mit einer Statusänderung in  **Abgelehnt**.

Bemerkung: Um *Parameter* im *Workflow Designer* anzeigen und editieren zu können, braucht dieser ein passendes Plugin. Kontaktieren Sie dafür bitte Ihren Systemadministrator.

Konfiguration für mitgelieferte Systemaufgabe „Statusänderung“

Die mitgelieferte *Systemaufgabe* „Statusänderung“ kann Objekte je nach Konfiguration automatisch entsperren. Dazu muss die *Einstellung* `cs.workflow.status_change_unlock` für den Klassennamen des Objekts (im „Bereich“ der Einstellung) einen für den `unlock` Parameter der Methode `cdb.objects.Object.ChangeState` gültigen Wert enthalten. Dabei werden keine Oberklassen berücksichtigt, sondern nur die exakten Klassennamen der Objekte.

Im Auslieferungszustand ist dies jeweils für die Klassen *document*, *model* und *cdb_wsp* mit dem Wert *1* (entsperren, wenn das Recht gewährt wird) definiert.

6.4.2 Beispiel: Eigene Systemaufgabe erstellen

In diesem Beispiel erstellen wir eine eigene *Systemaufgabe*, die *Mappeninhalte* kopiert und dabei die an ihr konfigurierten *Parameter* für die Vorbelegung der Kopieroperationen verwendet.

Bemerkung: Für das Beispiel nehmen wir an, dass der namespace des Systems demo ist und dass ein Kundenmodul namens `demo.plm` existiert.

Definition anlegen

Für unsere Variante der *Systemaufgabe* Kopie werden wir die folgenden Parameter wählen:

Identifikator `demo_copy_objects`

Name (de) Demo Objekte kopieren

Name (en) Copy Objects Demo

Vollqualifizierter Python-Name `demo.plm.workflow.demo_copy_objects`

Anschließend können Sie noch *ein Bild für die neue Definition importieren* (Seite 16).

Funktion implementieren

Legen Sie die Datei `workflow.py` im Kundenmodul `demo.plm` mit dem folgenden Inhalt an:

```
# coding: utf-8
"Example of a custom workflow system task"
from cdb import constants
from cdb import util
from cdb.objects import operations
from cs.workflow.systemtasks import TaskCancelledException

def demo_copy_objects(task, content, **kwargs):
    """
    Copy all objects in content["info"] (using kwargs to preset the copy
    operation) and put them in the "edit" briefcase.

    Only allows for exactly one "edit" briefcase.
    """
    if len(task.EditBriefcases) != 1:
        msg = util.get_label(
            "cdbwf_only_one_out_briefcase") % task.GetDescription()
        raise TaskCancelledException(msg)
    briefcase = task.EditBriefcases[0]

    for obj in content["info"]:
        copyobj = operations.operation(constants.kOperationCopy,
                                      obj,
                                      operations.form_input(obj, **kwargs))

        # unlock new object
        try:
            cd = obj.GetClassDef()
            if constants.kOperationUnlock in [
                oi.get_opname() for oi in cd.getOperationInfos()
            ]:
                operations.operation(constants.kOperationUnlock, copyobj)
        except RuntimeError:
            task.addProtocol(
                "cannot unlock object %s" % obj.GetDescription(),
```

```
msgtype=protocols.MSGINFO)

operations.operation(constants.kOperationNew,
                    FolderContent,
                    cdb_folder_id=briefcase.cdb_object_id,
                    cdb_content_id=copyobj.cdb_object_id)
```

Test

Melden Sie sich erneut am System an. Im *Workflow Designer* steht Ihnen nun die neue *Systemaufgabe* bei der Anlage von *Aufgaben* zur Verfügung.

Bemerkung: Die *Parameter* für diese *Systemaufgabe* können Sie zunächst nur über das Datenblatt der *Systemaufgabe* festlegen. Weitere Informationen finden Sie unter *Definition von Parametern* (Seite 16).

6.5 Workflow Designer

6.5.1 Katalogfilter

Die folgenden im *Workflow Designer* verwendeten Kataloge besitzen nicht vom Anwender änderbare Standard-Filter:

| Katalog | Standard-Filter |
|---|------------------------------|
| cs.workflow.designer.catalogs.ConstraintsCatalog | [„name LIKE ,wf-designer:%“] |
| cs.workflow.designer.catalogs.FormTemplateCatalog | [„status = 20“] |
| cs.workflow.designer.catalogs.OperationCatalog | [„name LIKE ,CDB_%“] |

Filter sind Python-Listen oder -Sets mit SQL Where-Bedingungen. Diese werden zur Laufzeit „OR“-verknüpft angewandt und dann mit den Benutzereingaben kombiniert.

Siehe auch Workflows API für Details zur Anpassbarkeit dieser Filter.

6.6 Sonstiges

6.6.1 Vorzeitig abschließen

Die Option *Vorzeitig abschließen* für *Aufgaben* vom Typ *Genehmigung* und *Prüfung* ist im *Workflow Designer* standardmäßig deaktiviert.

Um die Option zu aktivieren, setzen Sie die Property `wffo` auf `true`.

6.6.2 Optionale Performanceoptimierungen

Im Auslieferungszustand prüft der *Workflow Designer* beim Start einige Berechtigungen, um interaktive Elemente in Abhängigkeit davon ein- oder auszublenden. Dies widerspricht dem ansonsten üblichen Bediennmuster, dass Elemente grundsätzlich angeboten werden und Berechtigungen erst bei tatsächlicher Interaktion geprüft werden.

Um diese initialen Rechteprüfungen (für Schreibrechte, das Leserecht auf dem *Workflow* selbst wird selbstverständlich weiterhin geprüft) zu deaktivieren, können Sie die Standardeinstellung `cs.workflow.designer/check_access_proactively` auf 0 stellen oder dies gezielt für einzelne Benutzer als Benutzereinstellung entsprechend festlegen.

Des Weiteren wird folgende Optimierung der Konfiguration der Beziehungsrechteprofile empfohlen:

- Entfernen der Zuordnung des Beziehungsrechteprofils `Workflow components` zu den Beziehungen `cdbwf_p2task`, `cdbwf_ag2task`, `cdbwf_p2component` und `cdbwf_p2aggr`.
- Einrichten einer neuen Beziehung `cdbwf_p2all_components`, die einen *Workflow* mit allen Objekten der Klasse `cdbwf_process_component` seiner `cdb_process_id` verknüpft (unabhängig von ihrer Einordnung in die Struktur). Diese Beziehung nutzt dann das Beziehungsrechteprofil `Workflow components`, um die Rechteprüfungen auf Strukturelementen zu beschleunigen.

Bemerkung: Ein SQL-Skript zur Einrichtung dieser empfohlenen Änderungen liegt unter `cs/workflow/updates/tools/optional_performance_optimization.sql`.

7.1 Systemaufgaben werden nicht abgeschlossen

Prüfen Sie, ob der *Workflow-Dienst* läuft.

Läuft der Dienst und einzelne *Systemaufgaben* werden trotzdem nicht abgeschlossen, kann es sein, dass die dazugehörigen Jobs fehlgeschlagen sind. Sie können die Jobs wie folgt erneut ausführen:

```
UPDATE mq_wfqueue SET cdbmq_state='W' WHERE cdbmq_state='F'
```

Abschlussaufgabe



Besondere *Sammelaufgabe*, die nach dem regulären Ende des *Workflows* ausgeführt wird. Der Begriff wird im *Workflow Designer* synonym für die *Aufgaben* innerhalb dieser *Sammelaufgabe* verwendet.

Abschlussaufgaben werden typischerweise für Ausnahmebehandlung oder Benachrichtigungen verwendet.


- Beispiel

Aufgabe



Element eines *Workflows*, das Eingabedaten mit einer Aufgabenbeschreibung kombiniert. Die Beschreibung ist für interaktive Aufgaben textuell, für *Systemaufgaben* eine PowerScript-Funktion. Eingabedaten werden über *Mappen* mit der Aufgabe verknüpft. Das Verhalten einer Aufgabe wird durch ihren wf_user:task_types bestimmt.



- wf_user:create_tasks
- wf_user:task_olc

Anwender können mit *CONTACT Tasks* eine Übersicht über ihre Aufgaben im Status  **Umsetzung** sehen und diese bearbeiten.

Bearbeitungsmodus



Steuert die wf_admin:briefcase_contents_access. Mögliche Werte:

| Datenbankwert | Name | Gewährte Rechte (Beispiel) |
|---------------|---|--|
| 0 |  Info | Lesen (read, read_file) |
| 1 |  Edit | Schreiben (accept, lock, save, unlock) |

- Details

- Einrichtung

Constraint



Mit Constraints können Sie Bedingungen definieren, die unmittelbar nach der Statusänderung einer *Aufgabe* in **Umsetzung** geprüft werden. Existiert mindestens ein nicht erfülltes Constraint zur *Aufgabe*, wird die *Aufgabe* direkt in den Status **Verworfen** versetzt. Constraints sind **Objektregeln**.

- Verwendung
- Einrichtung

Formular



Formulare ermöglichen die Erfassung und Weiterverarbeitung von strukturierten Daten (Metadaten) in *Workflows* und ihren *Aufgaben*.

Formulare werden immer aus einer *Vorlage* instanziiert.

- Verwendung
- Einrichtung

Mappe



Mappen sammeln Ein- oder Ausgabedaten (siehe *Mappeninhalt*), entweder für einen gesamten *Workflow* („globale Mappe“) oder einzelne *Aufgaben* („lokale Mappe“).

Die Zuordnung einer Mappe zu einem *Workflow* oder einer *Aufgabe* enthält zusätzlich einen *Bearbeitungsmodus*, der eine zusätzliche Rechtevergabe regelt.

- Verwendung
- Einrichtung

Mappeninhalt Objekt, das einer *Mappe* als Inhalt zugeordnet ist. Mögliche Inhalte sind *Formulare*, Dateien, oder Fachobjekte, die als Ein- oder Ausgabedaten eines *Workflows* oder einer *Aufgabe* dienen.

Parameter

Filterparameter *Systemaufgaben* können Parameter enthalten, die die Ausführung als zusätzliche Eingaben beeinflussen. Parameter, die an der *Systemaufgabendefinition* konfiguriert sind, sind dabei zwingend erforderlich.

Protokoll Das Protokoll eines *Workflows* sammelt Meldungen (über den regulären Ablauf oder unerwartete Ereignisse) samt Zeitpunkt, die für Monitoring und nachträgliche Analyse hilfreich sind.

- Weitere Informationen

Sammelaufgabe




Sammelaufgaben sind Container für *Aufgaben*, die selbst keine eigene Logik abgesehen von der Ausführungsreihenfolge ihrer untergeordneten *Aufgaben* besitzen.

Sequenzielle Sammelaufgaben arbeiten ihre untergeordneten *Aufgaben* strikt nacheinander ab, parallele gleichzeitig. Den Status **Abgeschlossen** erreichen Sammelaufgaben erst wenn sich alle untergeordneten *Aufgaben* in einem finalen Status (**Abgeschlossen**, **Abgelehnt** oder **Verworfen**) befinden.

Workflows selbst verhalten sich wie eine sequenzielle Sammelaufgabe.

Im *Workflow Designer* werden Sammelaufgaben nicht explizit dargestellt. Übereinander dargestellte *Aufgaben* befinden sich in einer gemeinsamen parallelen Sammelaufgabe, hintereinander dargestellte *Aufgaben* entweder direkt im *Workflow* oder einer sequenziellen Sammelaufgabe.

Systemaufgabe

Eine *Aufgabe*, die nicht interaktiv von einem Anwender, sondern durch eine PowerScript-Funktion ausgeführt wird. Die Abarbeitung von Systemaufgaben im Status  **Umsetzung** geschieht asynchron durch den *Workflow-Dienst*.

- Verwendung
- Einrichtung

Systemaufgabendefinition Logik, Name und Aussehen einer *Systemaufgabe* werden durch ihre Definition bestimmt. Jede *Systemaufgabe* muss eine Definition über ihr Attribut `task_definition_id` referenzieren.

Ihr Systemadministrator kann neue Definitionen hinzufügen und so den Vorrat an nutzbaren Typen von *Systemaufgaben* erweitern. Lesen Sie dazu den Abschnitt im Administrationshandbuch.

Vorlage Eine Vorlage ist ein Objekt, das nicht selbst verwendet werden kann, aber kopiert wird, um ein nutzbares Objekt zu erzeugen. Dieser Kopiervorgang wird „instanziiieren“ genannt.

Objekte, die aus Vorlagen erstellt werden können, sind z.B. *Workflows* oder *Formulare*.

Workflow

Ein Workflow enthält zeitliche und logische Abhängigkeitsstrukturen zwischen *Sammelaufgaben* und interaktiven und automatischen *Aufgaben*. Er kann unter Umständen einen gesamten Geschäftsprozess abbilden.


Workflows können auch *Vorlagen* sein.



- Menüzugang: *Prozesse* -> *Workflows*

Workflow Designer

Webanwendung, die das grafische Design eines *Workflows* unterstützt.

- Verwendung

Workflow-Besitzer Der Anwender, der einen *Workflow* in den Status  **Umsetzung** versetzt, gilt fortan als dessen „Besitzer“. Effektiv zeichnet er sich durch diese Aktion persönlich verantwortlich für die Korrektheit des *Workflow*-Designs zu diesem Zeitpunkt.

Workflow-Dienst Dieser Dienst führt *Systemaufgaben* im Status  **Umsetzung** aus. Er ist als „Message Queue“ implementiert, d.h. sobald eine *Systemaufgabe* in den Status  **Umsetzung** gesetzt wird, wird ein Job angelegt, welcher die Ausführung durch den Dienst steuert.

Der Name des Dienstes ist `cs.workflow.services.WFServer`, der der Queue `wfqueue`.

- Weitere Informationen
- [Message Queues](#)
- [Dienste](#)

Workflow-Kategorie Kategorien sollen eine logische Gruppierung ihrer *Workflows* ermöglichen. Insbesondere bei der Vorauswahl von *Workflow*-Vorlagen kann eine sinnvolle Kategorisierung helfen. Denkbare Kategorienamen wären z.B. *Ausschreibung*, *Engineering Change Order* oder *Request for Information*.

- Menüzugang *Administration/Konfiguration* -> *Katalogverwaltung* -> *Prozesse* -> *Workflow-Kategorien*

Abbildungsverzeichnis

| | | |
|-----|--|----|
| 6.1 | Objektregel <i>Alle Mappendokumente sind freigegeben</i> | 13 |
| 6.2 | Objektregel <i>Alle Aufgabendokumente sind freigegeben</i> | 13 |
| 6.3 | Statusnetz einer Formularvorlage | 14 |

Tabellenverzeichnis

A

Abschlussaufgabe, [21](#)
Aufgabe, [21](#)

B

Bearbeitungsmodus, [21](#)

C

CADDOK_MAIL*, [4](#)
CADDOK_SMTP*, [4](#)
CADDOK_STOP_EMAIL_NOTIFICATION, [4](#)
cdbfolder_content, [10](#)
cdbwf_process_content, [10](#)
Constraint, [22](#)
cs.workflow.briefcases.BriefcaseReference, [10](#)

F

Filterparameter, [22](#)
Formular, [22](#)

M

Mappe, [22](#)
Mappeninhalt, [22](#)

P

Parameter, [22](#)
Protokoll, [22](#)

S

Sammelaufgabe, [22](#)
Systemaufgabe, [23](#)
Systemaufgabendefinition, [23](#)

U

Umgebungsvariable
CADDOK_MAIL*, [4](#)
CADDOK_SMTP*, [4](#)
CADDOK_STOP_EMAIL_NOTIFICATION, [4](#)
cdbfolder_content, [10](#)
cdbwf_process_content, [10](#)
cs.workflow.briefcases.BriefcaseReference, [10](#)

V

Vorlage, [23](#)

W

Workflow, [23](#)
Workflow Designer, [23](#)
Workflow-Besitzer, [23](#)
Workflow-Dienst, [23](#)
Workflow-Kategorie, [23](#)