

Wikiprint Book

Title: Automatyzacja procesów workflow

Subject: eDokumenty - elektroniczny system obiegu dokumentów, workflow i CRM - DeployerGuide/Customization/ProcessAutomation

Version: 133

Date: 11/21/24 21:32:35

## Table of Contents

<i>Automatyzacja procesów workflow</i>	3
<i>Podstawowe informacje</i>	3
<i>API komend workflow</i>	3
<i>Dla zaawansowanych</i>	6
<i>Wykorzystanie własności, danych wejściowych i przypisań</i>	7
<i>Trochę teorii</i>	7

## Automatyzacja procesów workflow

### Podstawowe informacje

Procedury workflow oparte są o notację BPMN i uwzględniają wszystkie najważniejsze elementy tej notacji. Składają się na nią:

- Etapy (Czynności - bloczki)
- Przejścia (strzałki)
- Decyzje (diament powodujący wyświetlenie decyzji dla użytkownika)
- Warunki (diament dokonujący ewaluacji warunków SQL)
- Złączenia (JOIN - w przypadku wymagania spełnienia poprzednich etapów)



Konfiguracja procedur pozwala tworzyć mapy procesów odnoszące się zarówno do dokumentów jak i spraw. Przykłady wykorzystania dostępne są tutaj:

[Wykorzystanie procedur](#)

### API komend workflow

W akcjach etapów można używać komend które będą wykonane w czasie aktywacji danego etapu. Komendy wybiera się z listy wyboru określając dodatkowe parametry np.

```
target="20",dscprt="Wezwanie, uwaga!"
status="4",controlQuery="SELECT status = 3 FROM processes WHERE prc_id=$prc_id"
```

Nie zaleca się na obecnym etapie stosowania znaków specjalnych w wartościach parametrów, między nawiasami powinna się znaleźć wartość liczbowa ("5") lub tekstowa bez znaków specjalnych ("Wezwanie do wykonania etapu")

### Przełącz dokument

Komenda służy do automatycznego przekazywania dokumentu na wybrane stanowiska za pomocą komendy oraz procedury.

- to = "1" - parametr wskazujący do kogo ma zostać przekazany oryginał , jeśli parametru nie będzie, lub będzie pusty oryginał zostaje.

*Parametry:*

- dw = "2,3,4,5" - do wiadomości
- udw = "6,7,8" - ukryte do wiadomości

Każda z tych wartości to orunid (najlepiej sprawdzić w orgtree\_view)

### Sprawdź czy pole jest wypełnione

Komenda służy do sprawdzania czy dane pole formularza jest wypełnione. Przyjmuje 2 parametry i oba są wymagane.

*Parametry:*

- field="featid|8" lub "symbol" - pole które ma sprawdzić
- alert = "Wypełnij pole symbol" - wiadomość w przypadku pustej wartości w polu

W przypadku parametru field może on obsługiwać zarówno cechy jak i inne pola. W przypadku cechy wpisujemy featid|id cechy z tabeli features dla której będzie wykonywana komenda. Jeśli podamy inny klucz wtedy zostanie od sprawdzony w danych formularza. Klucz wtedy powinien się nazywać tak samo jak nazwa kolumny w bazie w tabeli documents lub processes. Komenda nie pozwala na jednoczesne sprawdzanie dla kilku pól jednak jest możliwość ustawienia kilku tych samych komend z innymi parametrami dla akcji przed lub po zapisie etapu.

W przypadku jeśli cena będzie wpisana etap zostanie zaznaczony jako wykonany.

### Sprawdź prawdziwość warunku SQL

Komenda służy do sprawdzania warunku SQL. Komenda może być użyta tylko dla definicji etapu. Przyjmuje 3 parametr

*Parametry:*

- query ="SELECT cena IS NOT NULL FROM table WHERE prc\_id = {PRC\_ID}/doc\_id = {DOC\_ID}" - sql
- alert = "Wypełnij pole cena" - wiadomość w przypadku niespełnienia sql
- success = "Pole cena wypełnione" - wiadomość w przypadku spełnienia sql

W przypadku jeśli cena będzie wpisana etap zostanie zaznaczony jako wykonany.

### Utwórz zdarzenie

Komenda służy do utworzenia zadania dla zalogowanego pracownika lub innego wpisanego w parametrze emp\_id jako usr\_id. Zgodnie z [#2390](#). komenda przyjmuje dwa parametry opis zadania oraz emp\_id (zgodnie z wymogami komenda akceptuje parametr {LOGGED\_USER} który wskazuje na zalogowanego pracownika)

*Parametry:*

- dscript="Aktualizacja dokumentacji wdrożenia"
- emp\_id="{LOGGED\_USER}" (lub usr\_id z tabeli users)
- trmty="TODO" - typ zdarzenia, może być do wyboru MEETING, PHONECALL, ALARM
- start\_="featid|81"; - na kiedy ma dodać zdarzenie, wartość jest pobierana z cechy dlatego definicja np featid|81
- end\_="featid|81"; - data zakończenia
- interval="+ 4 days" - interval za jaki utworzyć się zdarzenie, można tworzyć wstecz poprzez -
- ptstid="2" - id etapu z tabeli stages\_def lub (ptstid="next") dla następnego etapu (ten parametr nie jest wymagany)

### Sprawdź czy istnieje dokument

Komenda służy do sprawdzania czy w danej sprawie występuje konkretny dokument. Zgodnie z [#2082](#). komenda przyjmuje dwa parametry typ dokumenty (id z tabeli types\_of\_documents) oraz status (wartość z kolumny state\_ z tabeli documents)

*Parametry:*

- dctpId="9"
- state\_="2"

### Zmień status

Obecnie przyjmuje tylko jeden parametr jest to klucz główny tpstid z tabeli types\_of\_processes\_states, który oznacza na jaki status zostanie zmieniony status sprawy lub dokumentu.

*Parametry:*

- tpstid="9"

Czyli do kolumny params w tabeli action\_commands wpisujemy np. tpstid="9" lub w interfejsie użytkownika w słownikach

### Uprawnij grupę pracowników do sprawy

Komenda działa niezależnie od tego czy jest ustawiona dla procedury czy etapu. Uprawnia ona wpisaną grupę w parametrach na określoną maskę do sprawy. Jeśli dana grupa pracowników została już wcześniej uprawniona wtedy dostajemy komunikat o tym jednak zama sprawa, komenda, etap jest zapisywana.

*Parametry:*

- grpnam="Pracownicy" - nazwa grupy z tabeli groups
- attrib="rwnd" - maska uprawnień według specyfikacji [Integracja z systemem Subjekt?](#)

Oba parametry są wymagane czyli ciąg z parametrami wygląda np. grpnam="Pracownicy",attrib="rwnd"

Znaczenie poszczególnych flag:

- r - Odczyt
- w - Zapis zadań i dokumentów
- d - Oglądanie wszystkich dokumentów
- n - Powiadomianie o nowych dokumentach, zadaniach i komentarzach
- m - Zarządzanie (karta ogólne i Uprawnienia)

### Utwórz dokument

Komenda ta tworzy dokument dla etapu procedury (nie obsługuje utworzenie dokumentu dla procedury). Przyjmuje również tylko 2 parametry i jest nim klucz główny dctpId z tabeli types\_of\_documents oraz opis jaki zostanie użyty w generowanym dokumencie.

*Parametry:*

- dctpId="5" -id typu dokumentu
- dctpTp="Note" -typ dokumentu
- dscrpt="Wezwanie do wykonania etapu procedury" -opis dokumentu
- do="1" -id jednostki do której przekazać dokument(orunid)
- dw="2,3,4" -(dw = do wiadomości) jeden lub więcej id (rozdzielone znakiem interpunkcyjnym, zwanym przecinkiem) jednostki do której przekazać kopie dokumentu(orunid)

Przy standardowej instalacji taki parametr utworzy dokument typu notatka służbowa.

Do kolumny params w tabeli action\_commands wpisujemy np. dctpId="5",dscrpt="Wezwanie do wykonania etapu procedury" lub w interfejsie użytkownika w słownikach

Późniejsza implementacja będzie również uwzględniać parametry z formularza np nazwę etapu, procedury, lub klucze główne jak np prc\_id, contid itd. Pełna funkcjonalność parametrów już niebawem ;)

### Utwórz zadanie dla następnego etapu

Komenda tworzy zadanie dla następnego etapu procedury, o ile w kolejnym etapie jest określone stanowisko i jest na nim user. Po załatwieniu zadania automatycznie odznacza się etap procedury. Po załatwieniu etapu procedury, automatycznie zaznaczane jest zadanie jako załatwione.

*Parametry:*

- brak możliwości określenia - mile widziane zgłoszenia w tym temacie

### Utwórz przypomnienie

Komenda tworzy zdarzenie typu przypomnienie o danej procedurze/etapie dla wskazanego pracownika.

*Parametry:*

- usr\_id = dla kogo zostanie utworzone przypomnienie;
- start\_ = kiedy ma się pojawić przypomnienie;
- dscrpt = opis przypomnienia;

Wszystkie parametry są wymagane.

### Przełącz dokument do przełożonego

Komenda służy do automatycznego przekazywania dokumentu do przełożonego pracownika wskazanego w parametrze emp\_id.

*Parametry:*

- emp\_id = id użytkownika którego przełożony otrzyma dokument

### Zarejestruj dokument

Komenda służy do automatycznej rejestracji dokumentu według ustalonej procedury.

*Parametry:*

- reg\_id = id rejestru z tabeli registers w którym zostanie zarejestrowany dokument - używany wówczas gdy z góry znamy właściwy dziennik

*Opcjonalnie można użyć innych parametrów, wówczas system obliczy właściwy dziennik (np. wtedy kiedy dokumenty mogą być rejestrowane w różnych dziennikach - np. jednostek rozliczeniowych):*

- orunid = id jednostki organizacyjnej po której zostanie wyszukany dziennik (tabela registers kolumna orunid), w przypadku jeśli chcemy aby wartość orunid była pobrana bezpośrednio z formularza np pole jednostka rozliczeniowa na formularzu faktury parametr ten powinien wyglądać w następujący sposób orunid="{acوريد}"

- regtyp = typ rejestru w danym orunid
- type\_\_ = typ rejestru (1 = wychodzący, 2 = przychodzący, 3 = wewnętrzny)

### Utwórz załącznik z szablonu

Komenda automatycznie tworzy załącznik do dokumentu na podstawie podanego id szablonu (tabela templates kolumna tpl\_id).

*Parametry:*

- tpl\_id = id szablonu z tabeli templates, parametr jest wymagany

### Wysyłanie powiadomienia do grupy

Komenda pozwala na wysłanie powiadomienia na wewnętrzny komunikator.

*Parametry:*

- dscript = treść powiadomienia
- grp\_id = id grupy z tabeli groups do której zostanie wysłane powiadomienie
- grpnam = nazwa grupy (jeśli nie podano grp\_id) do której zostanie wysłane powiadomienie

### Informacja dodatkowa

Wszystkie parametry mają możliwość pobierania wartości w następujący sposób

- z cechy (nie dotyczy cech z list wyboru definiowanych przez użytkownika)
  - emp\_id = "featid|81"
  - emp\_id = "featid::81"
- bezpośrednio z otwartego formularza
  - emp\_id = "{rspuid}" gdzie wartość {rspuid} zostanie zamieniona na wartość w kolumnie (polu) odpowiedniej tabeli - dla formularza dokumentu documents dla sprawy processes. W podanym przypadku jest to pole rspuid (osoba odpowiedzialna) z tabeli processes.
- jako wynik zapytania SQL
  - emp\_id = "SQL::SELECT usr\_id FROM users WHERE usr\_id = featid|81"
  - emp\_id = "SQL::SELECT usr\_id FROM users WHERE usr\_id = featid::81"
  - emp\_id = "SQL::SELECT usr\_id FROM users WHERE usr\_id = {emp\_id}"

Dodatkowo obsługiwane są specjalne wartości parametrów, które można również dodawać do zapytań SQL

- {LOGGED\_USER} - id zalogowanego pracownika
- {ENT\_ID} - id jednostki
- {PKEYVALUE} - wartość klucza głównego z tabeli documents/processes

czyli może być

- emp\_id = "SQL::SELECT rspuid FROM processes WHERE prc\_id = {PKEYVALUE}"

Dodatkowo dla dokumentów różnych typów można podawać nazwy kolumn z dodatkowych tabel np dla faktur tabela vatnote itd.

### Dla zaawansowanych

W workflow biorą udział następujące tabele:

- procedures\_def - tabela procedur - przechowuje informacje o procedurze np. Zatwierdzenie faktury kosztowej
- stages\_def - tabela etapów - przechowuje definicje poszczególnych etapów np. Akceptacja Prezesa
- stages - instancje etapów - przechowuje informacje o zapisanych etapach konkretnych procesów: spraw, dokumentów
- proc\_actions - akcje powiązane z procedurami lub z etapami, wykonują się przed lub po zapisie np. beforeStageChange
- action\_commands - komendy wykonywane przez system na akcjach - wybierane spośród zawartych w katalogu commands - można dodać parametry, które dodają się do standardowych dwóch Obiektu Akcji oraz obiektu encji powiązanej z wykonywaną akcją np. Dokument albo Sprawa

## Wykorzystanie własności, danych wejściowych i przypisań

Własności służą do zdefiniowania dodatkowych atrybutów procedury - można je traktować jako zmienne procedury.

Dane wejściowe służą tym samym czym odczyt standardowego wejścia w konsoli czy programie (czyli pobraniu od użytkownika znaków)

Przypisania służą nadaniu wartości dla zmiennych procedury jak również nadaniu wartości atrybutom etapu którego dotyczą.

- Tworzymy własność typu array dla procedury np. Opiekun
- Tworzymy etap w którym ustalimy Opiekuna (np. Określenie opiekuna) w zakładce Dane wejściowe wpisujemy Nazwę dodanej własności (Opiekun). W efekcie tego w etapie pojawia się link monitorujący "Edytuj dane"
- W etapie który chcemy zlecić Opiekunowi dodajemy Przypisanie: własność: {stages.orgarr} (czyli tablica wykonujących zadanie workflow) wyrażenie: {procedures.Opiekun}

Przypisanie też możemy użyć bez konieczności pobierania danych od użytkownika, możemy je pobrać z bazy danych. Dla tego przykładu gdybyśmy chcieli pobrać Opiekuna klienta którego dotyczy sprzedaż dodalibyśmy Przypisanie z własnością: {stages.orgarr} i wyrażeniem:

```
SELECT o.orunid FROM contacts c JOIN processes USING(contid) JOIN orgtree_view o ON o.usr_id = c.macrtk WHERE prc_id = {
```

### Trochę teorii

Tworzenie prostych procesów workflow nie wymaga dużego przygotowania, ale do tworzenia bardziej zaawansowanych modeli konieczna jest minimalna znajomość teoretycznych zasad rządzących przepływem procesów.

[Podstawy teoretyczne](#)