

Title: Automatyzacja procesów workflow

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

Version: 133

Date: 05/17/24 11:06:12

## Table of Contents

<i>Automatyzacja procesów workflow</i>	3
<i>Podstawowe informacje</i>	3
<i>API komend workflow</i>	3

## 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"
```

#### Przełącz dokument

Komenda służy do automatycznego przekazywania dokumentu na wybrane stanowiska.

*Parametry:*

- to = "1" - parametr wskazujący do kogo ma zostać przekazany oryginał , jeśli parametru nie będzie, lub będzie pusty oryginał zostaje.
- dw = "2,3,4,5" - do wiadomości
- udw = "6,7,8" - ukryte do wiadomości

Wszystkie wartości w parametrach to orunid z widoku orgtree\_view.

```
}} {{ #!html
```

#### 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 {{ {{ #!html

#### Sprawdź prawdziwość warunku SQL

}} Komenda służy do sprawdzania warunku SQL. "Parametry:" \* query = "SELECT cena IS NOT NULL FROM table WHERE prc\_id = {PKEYVALUE}" - zapytanie SQL \* alert = "Wypełnij pole cena" - wiadomość w przypadku niespełnienia sql \* success = "Pole cena wypełnione" - wiadomość w przypadku spełnienia sql {{ {{ #!html

#### Utwórz zdarzenie dla pracownika

}} Komenda służy do utworzenia zdarzenia określonego typu (domyślnie zadanie) dla pracownika. "Parametry:" \* dscprt = "Aktualizacja dokumentacji wdrożenia" \* emp\_id = "{LOGGED\_USER}" - id pracownika lub pracowników oddzielone znakiem przecinka, którym zostanie przypisane to zdarzenie (usr\_id z tabeli users) \* grp\_id = "2" - id grupy lub grup pracowników oddzielone znakiem przecinka, którzy zostaną dodani do zadania (grp\_id z tabeli groups) \* orunid = "11" - identyfikator stanowiska lub stanowisk osób oddzielone znakiem przecinka, które zostaną dodane do zdarzenia (Pracownicy -> Struktura organizacyjna -> ID) \* 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) \* witinf = "t" - poinformuj mnie jeśli zadanie (dotyczy tylko trmty = TODO) zostało wykonane - parametr nie jest wymagany domyślnie przyjmuje wartość t aby nie powiadamiać należy wpisać f lub FALSE \* prev = "" - poprzednik(i) - identyfikator

zadania w projekcie (kolejne wartości należy oddzielać znakiem przecinka) \* next = "" - następnik(i) - identyfikator zadania w projekcie (kolejne wartości należy oddzielać znakiem przecinka) Parametry dscrpt i emp\_id są wymagane. {{{ #!html

### Sprawdź czy istnieje dokument

}}} Komenda służy do sprawdzania czy w danej sprawie występuje konkretny dokument. "Parametry:" \* dctp\_id = "9" - id typu dokumentu ze słownika (Ustawienia -> Panel sterowania -> Dokumenty -> Typy dokumentów -> kolumna id) \* state\_ = "2" - identyfikator rodzaju dokumentu 1 - wychodzący, 2 - przychodzący 3 - wewnętrzny {{{ #!html

### Zmiana statusu

}}} Komenda zmienia status zlecenia (sprawy/dokumentu) na podany w parametrze. Jeśli komenda jest wykonywana w kontekście procedury dokumentu wtedy dodatkowo jest zmieniany stan załatwienia dokumentu na załatwiony. "Parametry:" \* tpstid = "9" - id statusu ze słownika statusów dla odpowiedniej klasy (Ustawienia -> Panel sterowania -> Ogólne -> Statusy -> kolumna id) {{{ #!html

### Uprawnij grupę pracowników do sprawy

}}} Komenda służy do nadania lub aktualizacji uprawnień grupie lub pracownikom do sprawy. Jeśli grupa lub pracownik zostali już wcześniej uprawnieni wtedy następuje aktualizacja poziomu uprawnień. "Parametry:" \* grp\_id = "2" - id grupy, którą chcemy uprawnnić do sprawy (grp\_id z tabeli groups) - parametr zamienny z grpnam \* usr\_id = "2,3,4" - id pracowników, których chcemy uprawnnić \* grpnam = "Pracownicy" - nazwa grupy która została przydzielona do sprawy (Pracownicy -> Grupy) - parametr zamienny z grp\_id \* attrib = "rwnd" - maska uprawnień Znaczenie poszczególnych flag maski uprawnień: \* r - odczyt \* w - zapis zadań i dokumentów \* d - oglądanie wszystkich dokumentów \* n - powiadamianie o nowych dokumentach, zadaniach i komentarzach \* m - zarządzanie (karta ogólna i Uprawnienia) {{{ #!html

### Uprawnij grupę pracowników do dokumentu

}}} Komenda służy do nadania (aktualizacji) uprawnień grupie pracowników lub pracownikom do dokumentu. Jeśli grupa lub pracownik zostali już wcześniej uprawnieni wtedy następuje aktualizacja poziomu uprawnień. "Parametry:" \* grp\_id = "2" - id grupy, którą chcemy uprawnnić do dokumentu (grp\_id z tabeli groups) - parametr zamienny z grpnam \* usr\_id = "2,3,4" - id pracowników, których chcemy uprawnnić \* grpnam = "Pracownicy" - nazwa grupy która została przydzielona do dokumentu (Pracownicy -> Grupy) - parametr zamienny z grp\_id \* attrib = "rwnd" - maska uprawnień Znaczenie poszczególnych flag maski uprawnień: \* r - odczyt \* w - zapis \* m - zarządzanie {{{ #!html

### Utwórz dokument

}}} Komenda ta tworzy dokument określonego typu. Komenda przyjmuje następujące parametry: "Parametry:" \* dctp\_id = "5" - id typu dokumentu (Ustawienia -> Panel sterowania -> Typy dokumentów -> kolumna ID \* dctp\_tp = "Note" - nazwa klasy typu dokumentu (dctp\_tp z tabeli types\_of\_documents) parametr zamienny z dctp\_id \* dscrpt = "Wezwanie do wykonania etapu procedury" - treść dokumentu \* do = "1" - id stanowiska na jakie zostanie przekazany dokument (orunid z tabeli organization\_units) \* dw = "2,3,4" - (dw = do wiadomości) jeden lub więcej id (rozdzielone znakiem interpunkcyjnym, zwanym przecinkiem) jednostki do której przekazać kopie dokumentu (orunid z tabeli organization\_units) {{{ #!html

### Utwórz przypomnienie

}}} Komenda tworzy zdarzenie typu przypomnienie o danej procedurze/etapie dla wskazanego pracownika. "Parametry:" \* usr\_id = "2" - dla kogo zostanie utworzone przypomnienie (Pracownicy -> Karta pracowników -> kolumna id) \* orunid = "1" - identyfikator stanowiska osoby dla której chcemy dodać przypomnienie (Pracownicy -> Struktura organizacyjna -> ID), parametr zamienny z usr\_id w przypadku podania obu parametrów pierwszeństwo ma parametr usr\_id \* start\_ = "SQL::SELECT now()" - na kiedy ma ustawić przypomnienie \* dscrpt = "Przypomnienie o zatwierdzeniu etapu" - opis przypomnienia; {{{ #!html

### Przeładaj dokument do przełożonego

}}} Komenda służy do automatycznego przekazywania dokumentu do przełożonego pracownika wskazanego w parametrze emp\_id. "Parametry:" \* emp\_id = "{LOGGED\_USER}" - id użytkownika którego przełożony otrzyma dokument domyślnie id zalogowanego (usr\_id z tabeli users) {{{ #!html

### Zarejestruj dokument

}}} Komenda służy do automatycznej rejestracji dokumentu według ustalonej procedury. "Parametry:" \* reg\_id = "2" - 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) "Opcjonalnie można użyć innych parametrów, wówczas system obliczy właściwy dziennik (np. wtedy kiedy dokumenty mogą być rejestrowane w różnych dziennikach - np. jednostek rozliczeniowych):" \* orunid = "{acrid}" - 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="{acrid}" \* regtyp = "RegOfVatNotes" - typ rejestru w danym orunid (kolumna regtyp z tabeli registers) \* type1\_\_ = "1" - typ rejestru (1 = wychodzący, 2 = przychodzący, 3 = wewnętrzny) {{{ #!html

## Utwórz załącznik z szablonu

}}} Komenda automatycznie tworzy załącznik do dokumentu na podstawie podanego id szablonu (tabela templates kolumna tpl\_id). "Parametry:" \* tpl\_id = "23" - id szablonu z tabeli templates, parametr jest wymagany {{{ #!html

## Wysyłanie powiadomienia

}}} Komenda pozwala na wysłanie powiadomienia na zadany sposób. "Parametry:" \* dscrpt = "Zebranie zarządu" - treść powiadomienia \* grp\_id = "2" - id grupy, której pracownicy otrzymają powiadomienie (kolumna grp\_id z tabeli groups) \* grpnam = "Zarząd" - zamienny parametr do grp\_id, nazwa grupy (kolumna grpnam z tabeli groups) \* usr\_id = "2" - id użytkownika (users.usr\_id) do którego zostanie wysłane powiadomienie \* orunid = "1" - identyfikator stanowiska lub stanowisk osób oddzielone znakiem przecinka, które zostaną dodane do odbiorców powiadomienia (Pracownicy -> Struktura organizacyjna -> ID) \* type!\_\_ = "Communicator" - rodzaj powiadomienia, obecnie wspierane są Communicator - wewnętrzny komunikator, Document - notatka służbowa, Mail - powiadomienie zostanie wysłane na adres mailowy zapisany w kartotece pracownika Aby komenda zadziałała musi być podany jeden z 3 parametrów: grp\_id, grpnam, usr\_id lub orunid. W przypadku podania wszystkich parametrów określających odbiorców, lista odbiorców jest łączona z poszczególnych wartości parametrów. {{{ #!html

## Wyślij wiadomość email

}}} Komenda pozwala na wysłanie wiadomości email do określonych odbiorców. Dodatkowo istnieje możliwość automatycznego załączenia załączników dokumentu do email. "Parametry:" \* from!\_\_ = "SQL::SELECT email FROM users WHERE usr\_id = {LOGGED\_USER}" - pole od kogo domyślnie jest wstawiane eDokumenty \* to!\_\_!\_\_ = "prezes@firma.eu" - pole do w przypadku parametru liczbowego zostanie wstawiony adres email kontaktu o identyfikatorze podanym w parametrze np to!\_\_!\_\_ = "{contid}" \* cc!\_\_!\_\_ = "kierownik@firma.eu" - pole kopia w przypadku parametru liczbowego zostanie wstawiony adres email kontaktu o identyfikatorze podanym w parametrze np cc!\_\_!\_\_ = "featid::98" \* bcc!\_\_!\_\_ = "archiwumfirma.eu" - pole kopia ukryta w przypadku parametru liczbowego zostanie wstawiony adres email kontaktu o identyfikatorze podanym w parametrze np bcc!\_\_!\_\_ = "{contid}" \* subjct = "Rekrutacja zakończona" - temat wiadomości zwykły tekst bez znaczników html. Pole jest wymagane. \* body!\_\_ = "Zakończono proces rekrutacji. W załączniku dostępne są wyniki" - treść wiadomości zwykły tekst bez znaczników html. Pole jest wymagane. \* tpl\_id = "1" - identyfikator szablonu (templates.tpl\_id). Jeśli szablon jest typu html parametr body!\_\_ zostanie pominięty a treścią maila będzie wygenerowany dokument z szablonu. W przypadku szablonu typu RTF zostanie on dodany jako załącznik - parametr body nie zostanie pominięty. \* attach = "1" - flaga oznaczająca czy do wysyłanej wiadomości mają zostać dołączone wszystkie załączniki jakie są w dokumencie. Działa tylko w przypadku procedury przypisanej do dokumentu. Spośród parametrów to!\_\_!\_\_, cc!\_\_!\_\_ oraz bcc!\_\_!\_\_ wystarczy aby tylko jeden był podany aby wiadomość została wysłana. {{{ #!html

## Dodaj wyjątek do kalendarza pracownika

}}} Komenda dodaje wyjątek do kalendarza pracownika (zasobu) z atrybutami dzienny, niepracujący. "Parametry:" \* usr\_id = "{LOGGED\_USER}" - identyfikator pracownika (users.usr\_id), któremu zostanie dodany wyjątek do kalendarza. W przypadku jest pracownik o podanym identyfikatorze nie posiada kalendarza (Zasoby -> Kalendarze zasobów) system automatycznie utworzy kalendarz, który będzie dziedziczył po domyślnym kalendarzu systemowym \* dscrpt = "Wyjazd na urlop" - opis dodawanego wyjątku. Uwaga - należy zapewnić unikalność nazw wyjątków w obrębie jednego kalendarza \* from!\_\_ = "2011-08-10" - data rozpoczęcia obowiązywania wyjątku w formacie YYYY-MM-DD np 2011-08-10 \* to!\_\_!\_\_ = "2011-08-20" - data zakończenia obowiązywania wyjątku w formacie YYYY-MM-DD np 2011-08-20 {{{ #!html

## Zamknij sprawę

}}} Komenda zamyka sprawę na której wykonywany jest etap procedury. Komenda działa tylko i wyłącznie w kontekście sprawy.. "Parametry:" brak parametrów {{{ #!html

## Utwórz projekt z szablonu

}}} Komenda generuje strukturę spraw oraz zadań zgodnie z zadaniem identyfikatorem szablonu projektu. "Parametry:" \* ptplid = "1" - identyfikator szablonu projektu (projects\_templates.ptplid) \* dsexid = "45" - identyfikator teczki z wyciągu z wykazu akt \* contid = "677" - domyślny identyfikator kontrahenta (contacts.contid) \* start\_ = "2011-08-20" - data rozpoczęcia Wszystkie parametry są wymagane ---- {{{ #!html

}}} "Informacja dodatkowa" Wszystkie parametry mają możliwość pobierania wartości w następujący sposób: \* z cechy \* emp\_id = "featid|81" \* emp\_id = "featid::81" \* dscrpt = "Pracownik featid::81:string prosi o udzielenie urlopu okolicznościowego" \* bezpośrednio z otwartego formularza \* emp\_id = "{rspuid}" gdzie wartość {rspuid} zostanie zamieniona na wartość w kolumnie (polu) odpowiedniej tabeli - dla formularza dokumentu documents (plus dodatkowe tabele) dla sprawy processes. W podanym przypadku pole rspuid (osoba odpowiedzialna w sprawie) jest wartością z kolumny rspuid 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 = {rspuid}" \* emp\_id = "SELECT usr\_id FROM users WHERE usr\_id = {rspuid}" (dla wersji > 3.5) \* predefiniowane parametry \* {LOGGED\_USER} - id zalogowanego użytkownika \* {ENT\_ID} - id jednostki na której pracuje użytkownik \* {PKEYVALUE} - wartość klucza głównego (id dokumentu/sprawy) patrz klucz główny \* {CURRENT\_DATE} - aktualna data (dla wersji > 3.5) \* {CURRENT\_TIME} - aktualny czas (dla wersji > 3.5) \* {LOGGED\_ORUNID} - identyfikator stanowiska zalogowanej osoby (dla wersji > 3.5) Predefiniowane parametry można używać w następujący sposób \* emp\_id = "SQL::SELECT rspuid FROM processes WHERE prc\_id = {PKEYVALUE}" \* emp\_id = "{LOGGED\_USER}" Sposób definicji parametrów można łączyć np.: \* emp\_id = "SQL::SELECT usr\_id FROM users WHERE usnam = featid:81 AND adddat > '{adddat}':timestamp AND usr\_id != {LOGGED\_USER} AND ent\_id = {ENT\_ID}" lub można wykonywać

działania (przykład teoretyczny nigdzie nie występuje taki parametr :)) \* netto\_ = "SQL::SELECT {brutto}::int \* featid::89" Do parametru określonego za pomocą cechy np featid::89 można dodać specjalny modyfikator - string czyli featid::89::string - pozwala to na pobranie wartości tekstowej cechy zdefiniowanej jako lista pracowników lub lista adresów. W wyniku działania tego modyfikatora otrzymamy na nazwę urządzenia wybranego w cenie. Dodatkowo dla dokumentów różnych typów można podawać nazwy kolumn z dodatkowych tabel np dla faktur tabela vatnote itd. {{{ #!html  
}}}  
---- == 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 == Potężne możliwości silnika workflow systemu eDokumenty możliwe są m.in. dzięki wykorzystaniu parametrów i zmiennych które mogą być dynamicznie przetwarzane podczas wykonywania procedury. Dane mogą być pobierane od użytkownika, ale również przetwarzane przez sam workflow. === Dane wejściowe === Dane wejściowe służą tym samym czym odczyt standardowego wejścia w konsoli czy programie (czyli pobranu od użytkownika znaków). Można je pobierać z różnych formantów (pól tekstowych, list wyboru, list pracowników). Najciekawszą opcją jest opcja SELECT która pozwala zdefiniować dowolną kwerendę SQL zwracającą potrzebną nam w danym etapie listę (np. kierowników, księgowych, zasobów itp). Przykładowa lista dla atrybutu CZŁONEK ZARZĄDU potrzebna do wyboru osoby podpisującej umowę: {{{ SELECT orunid as value, fullnm || ' - ' || ndenam as caption FROM orgtree\_view WHERE orunid IN (3,14,15,16) }}} Inny przykład to pobranie identyfikatora stanowiska, wystarczy w tym celu wybrać opcję orunid[].  
[[Image(parametry1.PNG)]]  
[[BR]] === Przypisania === Przypisania służą nadaniu wartości dla zmiennych procedury jak również nadaniu wartości atrybutom etapu którego dotyczą. Najczęściej wykorzystuje się przypisanie stanowisk wykonujących etap poprzez przypisanie do własności {stages.orgarr} tablicy (UWAGA! dane muszą być typem tablicowym, w kwerendach należy pamiętać o rzutowaniu). Patrz przykład:  
[[Image(parametry2.PNG)]]  
[[BR]] Tak więc dane wejściowe typu array o nazwie "Akceptant" zostały przypisane do własności {stages.orgarr} (czyli tablicy wykonujących zadanie workflow). 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 (ze sprawy) dodalibyśmy Przypisanie własności {stages.orgarr} wartości wyrażenia SQL: {{{ SELECT ARRAY[o.orunid] FROM contacts c JOIN processes USING(contid) JOIN orgtree\_view o ON o.usr\_id = c.macrtk WHERE prc\_id = {processes.prc\_id} }}} === Własności === Własności służą do zdefiniowania dodatkowych atrybutów procedury - można je traktować jako zmienne procedury dostępne we wszystkich etapach jak również w parametrach akcji(komend). Najczęściej zdefiniujemy własność kiedy chcemy aby nadać jej określoną wartość a później wykorzystywać np. w warunkach do sterowania przebiegiem workflow. Np. Zdefiniujemy własność "Czy jest przedpłata", którą napelnimy wartością zależną od wyniku zapytania SQL. Następnie wykorzystamy tą własność w warunku. === Kilka słów o zmiennych === W zapytaniach SQL można używać następujących wyrażeń, które zostaną zastąpione odpowiednimi wartościami: \* {PRC\_ID} - prc\_id sprawy której dotyczy procedura \* {DOC\_ID} - doc\_id dokumentu którego dotyczy procedura \* {SOP\_ID} - id etapu/czynności \* {STAGES.PTSTID} - id definicji etapu \* również zawartości obiektów podlegających workflow sprawy i dokumentu np.: \* {processes.rsuid} - id osoby odpowiedzialnej za sprawę \* {documents.adduid} - id osoby tworzącej dokument W dalszej części umieszczone zostały użyteczne konstrukcje przy budowaniu workflow:  
[wiki:DeployerGuide/Customization/ProcessAutomation/Examples Przykłady zapytań] == 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. [wiki:DeployerGuide/Customization/ProcessAutomation/Theory Podstawy teoretyczne]