

**Przewodnik użytkownika > Dodatkowe moduły i zakładki**

Informacje o wprowadzenie funkcjonalności:

Wersja systemu	Wersja modułu/funkcji	Data kompilacji	Zmiany	Opis
7.19.15.1	<a href="#">1.9.1</a>	-	zmiana	Obsługa dodatkowego atrybutu dcctid (kategoria dokumentu) dla zakładki na dokumencie
6.72.0	<a href="#">1.9</a>	-	zmiana	Obsługa dodatkowego atrybutu visible dla taga custom_widget
6.44.0	<a href="#">1.8</a>	-	zmiana	Dodano możliwość sterowania dostępem do zakładek w zależności od typu użytkownika
6.38.0	<a href="#">1.7</a>	-	zmiana	Dodano możliwość grupowania zakładek dla panelu pionowego
6.28.0	<a href="#">1.6</a>	-	zmiana	Widok kompaktowy dla pionowego panelu zakładek (compactView)
6.22.0	<a href="#">1.5</a>	-	zmiana	Obsługa tokena {REFRESH_FILTERS} oraz parametru js_refresh_filters do customowych przycisków
6.19.0	<a href="#">1.4</a>	-	zmiana	Obsługa unikalnych ID dla tab CustomModules
6.0.274	<a href="#">1.3</a>	-	zmiana	Obsługa dodatkowego atrybutu tagu tab enabled, explain_how_to_enabled oraz visible
6.0.229	<a href="#">1.2</a>	-	zmiana	Obsługa dodatkowego atrybutu tagu module "always_show_tabs"
6.0.229	<a href="#">1.1</a>	-	zmiana	Obsługa menu dokumentów
6.0.113	<a href="#">1.0</a>	-	zmiana	Obsługa custom_widget w menu

6.0.52(Ready_), 5.4.10	0.9	zmiana	18-05-2018	Każdy Custom moduł może być trzymany w osobnym pliku
5.2.61	0.6	-	zmiana	Dodano token {EXPORT_TO_PRINTER} za pomocą którego można wywołać systemowy dialog drukowania
6.0 (Ready_)	<a href="#">0.8</a>	-	zmiana	Obsługa menu
5.2.33	<a href="#">0.7</a>	02.11.2017	zmiana	Obsługa pionowego panelu zakładek
5.2.12	0.5	29.08.2017	zmiana	Dodano nowy token {EXPORT_TO_XLS} za pomocą którego można eksportować raport do XLS
5.0	0.4	18.11.2015	Zmiana	Dodano filtr (globalny oraz dla raportu) Projektu oraz obsługę parametru {PROJID}
4.8.100, 4.10.16, 4.11.2	0.3	18.11.2015	Zmiana	Dodano możliwość definicji separatora dla przycisków jako tag <separator/>

## Menu

1. [Dodatkowe zakładki](#)
  - 1.1 [Ograniczanie widoczności zakładek](#)
2. [Dodatkowe moduły](#)
  - 2.1 [Definiowanie dodatkowych modułów](#)
  - 2.2 [Ograniczanie widoczności modułów](#)
  - 2.3 [Filtry](#)
  - 2.4 [Ikona modułu](#)

## Dodatkowe zakładki

System eDokumenty umożliwia dodawanie customowych zakładek do kartoteki

- sprawy
- kontrahenta
- urzędzenia
- dokumentu

Zakładki te oparte są na raportach oraz pliku konfiguracyjnym xml znajdującym się w

```
$APP_PATH\var\tpl\tabs
$APP_PATH oznacza /home/edokumenty/public_html/apps/edokumenty
lub
C:\Program files\BetaSoft\eDokumenty\public_html\apps\edokumenty
```

Jeśli katalog var\tpl\tabs jest pusty należy skopiować szablony plików xml z \$APP\_PATH\var\tpl\_default\tabs.

Nazwy obsługiwanych plików xml

- sprawy - process\_tpl.xml

- kontrahenta - contact\_tpl.xml
- urządzenia - device\_tpl.xml
- dokumentu - document\_tpl.xml
- ewidencji - evidence\_tpl.xml
- pracownik - user\_tpl.xml
- produkt - product\_tpl.xml
- profil użytkownika - user\_profile\_tpl.xml

Aby dodać dodatkową zakładkę do jednej z wyżej wymienionych kartotek należy utworzyć odpowiedni raport oraz wyedytować wybrany plik xml dla danej kartoteki.

Definicja pliku xml

```
<?xml version="1.0" encoding="UTF-8"?>
<tabs>
  <tab label="" rep_id="" grp_id="" def_tab="1" alwaysVisible="1" forceOpen="1" prior="0" showFilters="0">
    <buttons>
      <button>
        <id>
          new
        </id>
        <label>
          Nowa
        </label>
        <dscript>
          Nowa
        </dscript>
        <onclick>
          App.openDialogByCls({CLSNAM}, {KEYVAL}, ({afterSubmit:'{AFTER_SUBMIT}', mode:'new'}).toJSONString())
        </onclick>
        <icon>
          new.png
        </icon>
      </button>
      <separator/>
      <button>
        <custom_widget>
          3
        </custom_widget>
      </button>
    </buttons>
  </tab>
</tabs>
```

```
<tab label="zakladka 1" iframe="http://www.onet.pl"></tab>
```

Gdzie:

- rep\_id - identyfikator raportu z systemu eDokumenty
- iframe - url do strony, którą chcemy wyświetlić w eDokumentach w zakładce

Dodatkowo do taga onclick oraz atrybutu iframe można dodawać parametry z postaci danych z aktualnie otwartej kartoteki. Dla przykładu niech posłuży kartoteka produktu oraz dodatkowa zakładka w postaci iframe wyświetlającego katalog zdjęć:

```
<tab label="Zdjęcia" iframe="http://www.moj.serwer/produkty/?symbol={symbol}"></tab>
```

Token {symbol} zostanie zamieniony na symbol produktu z aktualnie otwartej kartoteki.

## Zakładki dla formularza dokumentu

Zakładka dla dokumentu dodatkowo przyjmuje parametry:

**dctpid** (ID typu dokumentu) np.

```
<tab label="zakładka 1" rep_id="1" dctpid="1">
```

Zakładka pojawi się tylko w dokumentach o `dctpid = 1`. Brak tego parametru spowoduje dodanie zakładki dla wszystkich typów. Można też podać więcej identyfikatorów typu po przecinku (np. `dctpid="1,3,5,6,9"`).

**dcctid** (ID kategorii dokumentu) np.

```
<tab label="zakładka 1" rep_id="1" dcctid="2">
```

## Zakładki dla kartoteki sprawy

Zakładka dla sprawy dodatkowo przyjmuje parametr **dos\_id** (ID teczki z wykazu akt) np.

```
<tab label="zakładka 1" rep_id="1" dos_id="1">
```

Zakładka pojawi się tylko w sprawach o `dos_id = 1`. Brak tego parametru spowoduje dodanie zakładki dla wszystkich spraw. Można też podać więcej identyfikatorów typu po przecinku (np. `dos_id="1,3,5,6,9"`).

## Dodatkowe tokeny do użycia w tagu onclick:

Token	Opis
{CLSNAM}	klasa zaznaczonego elementu na liście jeśli zostało podane pole w raporcie
{KEYVAL}	identyfikator zaznaczonego elementu na liście jeśli zostało podane pole w raporcie
{KEYVALS}	identyfikatory zaznaczonych elementów - w przypadku dokumentów: {doc_id:{KEYVALS}}
{AFTER_SUBMIT}	akcja odświeżenia widoku po edycji wpisu lub np. jako dodatkowa akcja na pasku zadań
{MODE}	wartość pobierana z taga <code>&lt;id&gt;&lt;/id&gt;</code> używać jako <code>new</code> , <code>edit</code> , <code>del</code> , <code>delete</code>

Od wersji 4.1-alfa51 istnieje możliwość zadeklarowania przyciski jako custom widget (<http://support.edokumenty.eu/trac/wiki/DeployerGuide/Others/CustomWidgets>). Tag `button` musi zawierać informację w postaci taga `custom_widget` o wartości `id` danego widgeta czyli kolumna `cswgid`. Tabela `custom_widget` powinna zawierać rekord zgodnie z definicją w dokumentacji a wartość w kolumnie `c_path` to

```
c_path = custom/cswgid czyli custom/3
```

gdzie 3 to identyfikator rekordu. Id zaznaczonych rekordów są przekazywane pod parametrem `keyval`.

Od wersji 4.3.23 obsługiwany jest atrybut `def_tab` dla taga `tab`. Oznacza on, że dana zakładka ma być domyślnie otwierana. Działa tylko dla customowych modułów. W przypadku customowych zakładek dla kartotek ten atrybut nie działa.

## Widoczność zakładki

Dodatkowo od wersji 4.6.4, 4.7.2 dla zakładki możemy zdefiniować czy ma ona się pokazać domyślnie na pasku zakładek czy po kliknięciu "więcej...". Dokonujemy tego poprzez dodanie atrybutu `alwaysVisible` dla taga `tab` podobnie jak `def_tab`.

## Wymuszenie otwarcia jako domyślna zakładka

Zdarza się, że chcemy ustawić dodatkową zakładkę jako domyślnie otwartą w danej kartotece podczas jej otwierania. Standardowo ustawienie, która zakładka ma się otworzyć jest przypisane w kartotece na stałe. Zachowanie można to zmienić i wymusić otwarcie dodatkowej zakładki poprzez atrybut `forceOpen="1"`. Atrybut ten można ustawić tylko dla 1 zakładki.

## Ustawienie na odpowiedniej pozycji

Domyślnie zakładki są dodawane w kolejności wpisania ich w definicji xml. Zachowanie to można zmienić i ustawić zakładki w dowolnej kolejności (nawet pomiędzy standardowymi zakładkami) poprzez atrybut `prior` (od 0 do n gdzie n <= liczbie zakładek). Atrybut ten w połączeniu z `forceOpen` umożliwia nam dodanie nowej domyślnie otwartej zakładki na pierwszej pozycji w kartotece.

Od wersji 6.0.7

## Domyślnie schowany panel z filtrami

Aby polepszyć wrażenia estetyczne i dać większe obszar pracy dla użytkownika można domyślnie zwinąć panel z filtrami. Służy te tego atrybut taga `<tab> showFilters="0"`.

Przejdź do [Menu](#)

## Ograniczanie widoczności zakładek

Widoczność Zakładki może również być ograniczona poprzez parametr `grp_id` (np. `grp_id="2,5,10"`) który ograniczy widoczność zakładki wyłącznie do członków wymienionych po przecinku grup.

Przejdź do [Menu](#)

## Dodatkowe moduły

System eDokumenty umożliwia tworzenie własnych modułów w oparciu o podobny mechanizm.

### Definiowanie dodatkowych modułów

W systemie można również skonfigurować w oparciu o ten sam mechanizm własny moduł. Na wersji demonstracyjnej moduły dostępne przez użytkownika *jmamon* "Delegacje" oraz "Urlopy" dla użytkownika *serwis* są utworzone poprzez utworzenie następującego pliku w katalogu `$APP_PATH/var/tpl/CustomModules.xml`

Od wersji 6.0.52 jest możliwość definiowania pojedynczego modułu w jednym pliku w katalogu `$APP_PATH/var/cfg/CustomModules/{module_id}.xml`

Gdzie wartość `{module_id}` zamieniamy na wartość atrybutu `id` w elemencie o nazwie `module`. Listing takiego pliku wygląda dokładnie tak samo jak `tpl/CustomModules.xml`

Edycja tego pliku nie jest konieczna z poziomu konsoli. Dostęp do tego pliku jest możliwy z poziomu edokumentów. W Panelu sterowania szukamy odnośnika Szablony Systemowe. Uruchomi się okienko z plikami *Szablonów Systemowych*

The screenshot shows a software interface with a sidebar on the left and a main content area. The sidebar contains sections: 'Ogólne', 'Sprawy', 'Kalendarz i...', and 'Pozostałe u...'. The main content area is titled 'Panel sterowania' and contains several sections: 'Magazyn' and 'Narzędzia administracyjne'. A file explorer window titled 'Eksplorator szablonów systemowych' is open in the foreground, displaying a list of files. The file 'CustomModules.xml (63 B)' is highlighted with a red box and a red '3' next to it. In the background, the 'Panel sterowania' has a red box around 'Szablony systemowe' with a red '2' next to it.

(Szablony systemowe)

W okienku tym wyszukujemy plik "CustomModules.xml", klikamy na niego, a następnie zapisujemy na dysku. Edytujemy zmiany, po czym usuwamy istniejący na serwerze plik i wgrywamy nową wersję.

```
<?xml version="1.0" encoding="UTF-8"?>
<module id="cModule_U" name="Moduł" label="Mój moduł" icon="fixdefect-48.png" color="#c0392b" right="bswfms.processes" fil
  <toolbar>
  </toolbar>
  <filters>
  </filters>
  <tabs>
    <tab label="AKTYWNE" rep_id="1">
      <buttons>
      </buttons>
    </tab>
    <tab label="WSZYSTKIE" rep_id="2">
```

```

<buttons>
  <button>
    <id>refresh26</id>
    <label>Odśwież</label>
    <dscprt>Odśwież</dscprt>
    <onclick>
      {AFTER_SUBMIT}
    </onclick>
    <icon>sync-24.png</icon>
  </button>
</buttons>
</tab>
</tabs>
</module>

```

- Atrybut `filtersMode` dostępny od wersji 5.1.4, 5.0.184. Za pomocą tego atrybutu możemy zmieniać położenie panela filtrów. Standardowo panel filtrów widoczny jest po lewej stronie (wartość atrybutu `filtersMode = left` nie trzeba go podawać to jest domyślna wartość). Można to zmienić poprzez wpisanie do atrybutu `filtersMode` wartości `right_abs`. Panel filtrów pojawi się z prawej strony i będzie on nachodził na listę umieszczoną w module.

```

<?xml version="1.0" encoding="UTF-8"?>
<modules>
  <module id="Urlopy" name="Urlopy" label="Urlopy"
  icon="travel.png" right="bswfms.custom_modules.vacations">
    <toolbar></toolbar>
    <filters></filters>
    <tabs>
      <tab label="Moje wnioski" rep_id="95">
        <buttons>
          <button>
            <id>new</id>
            <label>Nowy</label>
            <dscprt>Nowy wniosek urlopowy</dscprt>
            <onclick>
              App.openDialogByCls('DOCUMENT', null,
              ({afterSubmit: '{AFTER_SUBMIT}', dctpId:21,dctptp:'CustomDocument', mode:'new'}).toJSONString())
            </onclick>
            <icon>new.png</icon>
          </button>
          <button>
            <id>edit</id>
            <label>Edycja</label>
            <dscprt>Edytuj wniosek</dscprt>
            <onclick>
              App.openDialogByCls('DOCUMENT', {KEYVAL},
              ({afterSubmit: '{AFTER_SUBMIT}', dctpId:21,dctptp:'CustomDocument', mode:'edit'}).toJSONString())
            </onclick>
            <icon>edit.png</icon>
          </button>
          <button>
            <id>delete</id>
            <label>Usuń</label>
            <dscprt>Usuń</dscprt>
            <onclick>
              App.openDialogByCls('DOCUMENT', {KEYVAL},
              ({afterSubmit: '{AFTER_SUBMIT}', dctpId:21,dctptp:'CustomDocument', mode:'del'}).toJSONString())
            </onclick>
            <icon>delete.png</icon>
          </button>
        </buttons>
      </tab>
    </tabs>
  </module>
</modules>

```

```

        <id>report</id>
        <label/>
        <dscrpt>Raport</dscrpt>
        <onclick>>window.open('litescript.php/demo/?script=GenerateReport&ent_id=2&ent_name=Demo S.A.&rep_id=1')</onclick>
        <icon> reports.png</icon>
    </button>
    <button>
        <id>printer</id>
        <label/>
        <dscrpt>OD WERSJI 5.2</dscrpt>
        <onclick>{EXPORT_TO_PRINTER}</onclick>
        <icon>print.png</icon>
    </button>
    <button>
        <id>xls</id>
        <label/>
        <dscrpt>EKSPORT DO XLS DO WERSJI 5.0 WŁĄCZNIE</dscrpt>
        <onclick> window.open('litescript.php/demo/?script=GenerateReport&ent_id=2&ent_name=Demo S.A.&rep_id=1')</onclick>
        <icon>xls.png</icon>
    </button>
    <button>
        <id>xls</id>
        <label/>
        <dscrpt>EKSPORT DO XLS OD WERSJI 5.2</dscrpt>
        <onclick>{EXPORT_TO_XLS}</onclick>
        <icon>xls.png</icon>
    </button>
    <button>
        <id>pdf</id>
        <label/>
        <dscrpt>EKSPORT DO PDF DO WERSJI 5.0 WŁĄCZNIE</dscrpt>
        <onclick> window.open('litescript.php/demo/?script=GenerateReport&ent_id=2&ent_name=Demo S.A.&rep_id=1')</onclick>
        <icon> pdf.png</icon>
    </button>
    <button>
        <id>pdf</id>
        <label/>
        <dscrpt>EKSPORT DO PDF OD WERSJI 5.2</dscrpt>
        <onclick>{EXPORT_TO_PDF}</onclick>
        <icon> pdf.png</icon>
    </button>
</buttons>
</tab>
</tabs>
</module>
</modules>

```

Funkcję `openDialogByCls()` można wywoływać z innymi parametrami zamiast 'DOCUMENT', co spowoduje otwarcie innych typów okien. Poniżej lista najbardziej przydatnych:

- CONTACT - Okienko edycji kontaktu (wpisujemy CONTACT\_EDIT)
- PROCESS - Okienko edycji sprawy
- RCP - Karta pracy
- MEETING - Spotkanie
- EVENT - Termin
- TODO - Zadanie
- PHONECALL - Rozmowa telefoniczna
- DEVICE - Urządzenie



- PRODUCT - Produkt

Przykład zakładania nowego kontaktu (osoba fizyczna):

```
App.openDialogByCls('CONTACT_EDIT', null, ({afterSubmit: '{AFTER_SUBMIT}', phyper: 'true' }).toJSONString())
```

Przykład zabezpieczenia w przypadku gdy raport zwróci zero wyników a na zakładce w module będzie dostępny przycisk Edytuj:

```
var k = {KEYVAL}; if (k) {App.openDialogByCls('DOCUMENT', {KEYVAL}, ({afterSubmit: '{AFTER_SUBMIT}', dctpid:24, dctptp: 'Custom
```

```
var ss={KEYVAL}; if (ss) {App.openDialogByCls('DOCUMENT', ss, ({afterSubmit: 'App.ADocumentslist.refresh()'}).toJSONString())
```

W przypadku gdy zaznaczymy dwa dokumenty:

```
var ss={KEYVALS}; if(ss.length==1) {
App.openDialogByCls('DOCUMENT', ss[0], ({afterSubmit: '{AFTER_SUBMIT}'}).toJSONString());
}else{
if(ss.length>1) {
NewFrame.showInfo('Proszę zaznaczyć tylko jeden dokument do edycji.');
```

```
} else {
NewFrame.showInfo('Proszę zaznaczyć dokument do edycji.');
```

```
}
```

Przykład tworzenia nowej umowy z uzupełnionym klientem na podstawie zestawienia (keyval):

```
<button>
  <id>umowa1</id>
  <label>Nowa umowa</label>
  <dscprt>Nowa umowa</dscprt>
  <onclick>App.openDialogByCls('DOCUMENT', null,
    ({afterSubmit: '{AFTER_SUBMIT}', dctpid:3, dctptp: 'Contract', "contid":{KEYVAL} , mode: 'new'}).toJSONString(
  </onclick>
  <icon>add.png</icon>
</button>
```

## Otwieranie modułu w nowym oknie

Przykład w javascript:

```
window.open('litescript.php/aps/?script=NewWindowModule&ent_id=2&modnam=CustomModule&submodule=cExactoris', '_blank', 'to
```

## Przycisk tworzący sprawę

Przyciski mogą też do listy parametrów obsługiwać klucze z bean-ów, dla przykładu:

Przycisk tworzący sprawę - pełna formatka formatka tworząca sprawę

```
App.openDialogByCls('PROCESS_EDIT', null, ({afterSubmit: '{AFTER_SUBMIT}', clsnam: 'DOSS', keyval:39, mode: 'new' }).toJSONS
```

- keyval przechowuje ID teczki

Przycisk tworzący sprawę - prosta formatka tworząca sprawę

```
App.createDialog('createProcessForm', 'SimpleProcessCreatingForm', './modules
/AProceses/forms/SimpleProcessCreatingForm.inc', 'Zakadanie', '513',
({clsnam: 'DOSS', strpid:351, devcid: '{devcid}', contid: '{contid}'}).toJSONString(), null, 'fast')
```

Opis parametrów przycisku tworzącego sprawę

- createProcessForm - nazwa w kodzie HTML okna
- SimpleProcessCreatingForm - nazwa klasy php'owej
- ./modules/AProcesses/form... - ścieżka do tej klasy
- Sprawa - opis na górnej belce formatki (najczęściej jest nadpisywany przez kod klasy)
- 513 - rodzaj formatki 513 to pełna z przyciskami na górnej belce inne rodzaje (kody) raczej są przeznaczone do wew. wywołań
- ({clsnam:'DOSS',strpid:1253,keyval:776,mode:'new'}).toJSONString() - parametry do formatki w formacie JSON

Ostatnie dwa parametry zawsze są takie same.

Dodatkowo od wersji 4.0 obsługiwany jest parametr {KEYVALS}, który odpowiada ze przekazanie do dialoga wszystkich zaznaczonych elementów z listy z nie tylko pierwszego jak to ma miejsce w przypadku parametru {KEYVAL}.

Parametr strpid jest dostępny w **Ustawienia -> Panel sterowania -> Sprawy -> Wyciąg z wykazu akt -> Kolumna Miejsce (strpid)** W przypadku jeśli kolumna ta nie jest włączona należy kliknąć na ikonę "Widoczne kolumny" (na dole listy) i zaznaczyć ptaszka.

Sprawa automatycznie otrzyma atrybuty id urządzenia oraz id kontrahenta urządzenia. Na razie zaimplementowano w urządzeniu.

## Przekazywanie parametrów

Od wersji 4.4.40, 4.6 oraz 4.5.24 w przypadku przycisku customowego w konfigurowanych zakładkach na kartotekach np. sprawy istnieje możliwość przekazania parametrów z beana sprawy podobnie jak do akcji onClick.

Wymagane jest aby w definicji przycisku w polu Parametry podać odpowiedni klucz - params. Przykład poniżej:

```
{
  "script": "Test.inc",
  "image": "24x24\deadline.png",
  "params": "prc_id: '{prc_id}', dsexid: '{dsexid}'"
}
```

W momencie generowania przycisku na zakładce klucze w nawiasach klamrowych zostaną zastąpione na właściwe atrybuty sprawy. Następnie cały ciąg z klucza params zostanie dołączony do wywołania widgeta. Po stronie widgeta w tablicy \$args będą dostępne klucze jak podano w params - w naszym przypadku będzie to prc\_id oraz dsexid.

Przejdź do [Menu](#)

## Przyciski operujące na rejestrze

W custom module można również operować na dowolnym rejestrze. Aby edytować rekord raport skojarzony z modulem musi posiadać klucze clsnam, keyval. Wówczas działać będzie przekazywanie wartości {KEYVAL}

```
<buttons>
<button>
<id>new</id>
<label>Nowy</label>
<dscprt>Nowy wpis</dscprt>
<onclick>
    App.openDialogByCls('REGISTER_ENTRY', null,
        ({afterSubmit: '{AFTER_SUBMIT}', mode: 'new', cregid: 1}).toJSONString())
</onclick>
<icon>new.png</icon>
</button>
<button>
<id>edit</id>
<label>Edycja</label>
<dscprt>Edytuj wpis</dscprt>
<onclick>
    App.openDialogByCls('REGISTER_ENTRY', {KEYVAL},
        ({afterSubmit: '{AFTER_SUBMIT}', mode: 'edit', cregid: 1}).toJSONString())
</onclick>
```

```

<icon>edit.png</icon>
</button>
<button>
<id>delete</id>
<label>Usuń</label>
<dscprt>Usuń</dscprt>
<onclick>
    App.openDialogByCls('CREGISTER_ENTRY', {KEYVAL},
        ({afterSubmit:'{AFTER_SUBMIT}', mode:'del',cregid:1}).toJSONString())
</onclick>
<icon>delete.png</icon>
</button>
</buttons>

```

### Ograniczanie widoczności modułów

Prawa dostępu do modułów (inaczej widoczność modułów) należy zdefiniować w Panelu sterowania > Definicje uprawnień. Zaleca się utworzenie prawa bswfms.custom\_modules (Moduły dodatkowe). A następnie dodanie prawa podrzędnego do custom\_modules.

Następnym krokiem w procesie jest ustawienie praw dla modułów w pliku *CustomModules.xml* w definicji właściwości *rights* modułu

```

...
<module id="cModule_1" name="Delegacje" label="Delegacje"
icon="processes.png" right="bswfms.custom_modules.delegations">
...

```

gdzie wstawiamy wartość pola *define* danego prawa.

Po zdefiniowaniu praw w tabeli oraz w pliku *CustomModules.xml* należy przejść do menu *Pracownicy*, a następnie wybrać *Grupy*. Jeżeli danej grupy nie ma, to tworzymy ją. Po zapisaniu grupy pojawiają się nowe zakładki. Przechodzimy do zakładki *Pracownicy* gdzie wprowadzamy wybranych przez nas użytkowników, a następnie klikamy w zakładkę *Prawa do systemu*, gdzie ze struktury drzewiastej praw wybieramy odpowiednie prawo i przypisujemy dostęp do prawa.

Przejdź do [Menu](#)

### Filtry

Na chwilę obecną dostępne są następujące filtry, które można zdefiniować bezpośrednio z poziomu xml:

- MonthSelectorTreeFilter - wybór dat z drzewka (parametry {DATE\_FROM} i {DATE\_TO}); zakres dat dla drzewka wyznaczany jest jako przedział od min(documents.sysdat) do max(documents.adddat) dla modułów customowych oraz zakres min(documents.adddat) do max(documents.sysdat) dla modułu Ewidencja. W celu wygenerowania drzewka na większy okres należy dodać do systemu np. notatkę służbową z datą dodania np. 2016-12-12.
- ClearingUnitsFilter - wybór jednostki rozliczeniowej (parametr {ACORID})
- ProjectsFilter - wybór projektu (parametr {PROJID})

```

<filters>
<filter name="Daty" type="MonthSelectorTreeFilter" height="auto" opened="0">
</filter>
<filter name="Jednostka rozliczeniowa" type="ClearingUnitsFilter" height="30px" opened="0">
</filter>
</filters>

```

```

<filters>
<filter name="Data delegacji" type="MonthSelectorTreeFilter" height="auto" opened="0" startDate="SELECT min(start_) FROM
</filter>
</filters>

```

Od wersji 4.5.8 następuje zmiana i dochodzą dwa nowe parametry

- {DATE\_FROM\_RANGE}

- {DATE\_TO\_RANGE}

Odpowiedni przechowują dane do swych odpowiedników bez końcówki RANGE. Modyfikacja ta ma na celu umożliwienie korzystanie z panelu filtrów oraz drzewka gdzie oba parametry były wypełniania przez 2 różne komponenty. Teraz jeśli zdefiniujemy parametr {DATE\_FROM} lub {DATE\_TO} to zostanie od pokazany na panelu filtrów i dodatkowo jeśli dodamy do tpl filter wtedy będą dodatkowe tokeny.

### Filtry statyczne

Filtry statyczne działają zarówno dla modułu jak i pojedynczej zakładki. Definicja takiego filtru jest taka sama jak powyżej z tą różnicą, że nazwa komponentu wskazuje na StaticFilter

```
<filters>
  <filter name="Daty" type="StaticFilter" token="{ANNA}">
    1=2
  </filter>
</filters>
```

Ważne w takim filtrze jest określenie tokena (osadzamy go w raporcie) w tym przypadku token="{ANNA}". Token może mieć dowolną nazwę. Wartość takiego filtra to stała wartość filtrująca. W tym przypadku jest to 1=2 czyli nigdy nic nie pokaże. Jednak można tu wstawić np.

```
d.name__ = 'Pompa'
```

gdzie d to alias na tabelę z raportu a name to kolumna, która występuje w tabeli, która jest ukryta pod d.

Dodatkowe filtry są realizowane za pomocą definiowanych filtrów dla raportów zgodnie z dokumentacją:

[Filtry dla raportów](#)

Przejdź do [Menu](#)

### Ikona modułu / kolor

Ikony w formacie .png,.svg o maksymalnym rozmiarze 48px x 48px wrzucamy do katalogu public\_html/framework/img/PageToolBar. Kolor kodowy modułu, który jest też kolorem tła dla ikony, określamy w atrybucie color tagu module. np.:

```
<module id="cModule_U" name="Moduł" label="Mój moduł" icon="icon.png" color="#c0392b" right="bswfms.processes">
...
</module>
```

### Wywołanie raportu

Aby otworzyć raport w module customowym i przekazać zaznaczone elementy z listy jako parametry do raportu należy dodać wywołanie

```
App.openDialogByCls('REPORT_VIEW', {rep_id}, ({keyval:{KEYVALS}}).toJSONString())
```

gdzie {rep\_id} to id raportu.

W raporcie będzie dostępny parametr {KEYVAL} , który będzie przechowywał zaznaczone elementy dlatego dla wielu zaleca się użycie IN ({KEYVAL}).

Przykład wywołania raportu o rep\_id=23 z parametrem FILTER\_STRING do którego zostaną przekazane po przecinku zaznaczone elementy na liście. W raporcie tym należy umieścić w WHERE parametr {FILTER\_STRING}.

```
<tab label="Przydzielone" rep_id="121">
<buttons>
  <button>
    <custom_widget>1</custom_widget>
  </button>
<button>
  <id>Do maila</id>
  <label>Tabela</label>
  <dscrpt>Otwiera tabelę HTML</dscrpt>
```

```

    <onclick>
      var keyval = {KEYVALS};
      App.openDialogByCls('REPORT_VIEW', 23, ({filter_string:'fkelid IN (' + keyval + ')}).toJSONString());
    </onclick>
    <icon>table24.png</icon>
  </button>
  <button>
    <custom_widget>7</custom_widget>
  </button>
</buttons>
</tab>

```

## Wykonywanie etapów workflow z przycisków

```

<tab label="WSZYSTKIE" rep_id="1">
<buttons>
<button>
<id>selectFR</id>
  <label>Potwierdź</label>
  <dscrpt>Potwierdza etap workflow</dscrpt>
  <onclick>
    App.openDialogByCls('P_WORKFLOW', null,({afterSubmit:'{AFTER_SUBMIT}',clsnam:'PROCESS',keyval:{KEYVAL},next_ptstid:336,
  </onclick>
  <icon>pass-24.png</icon>
</button>
</buttons>
</tab>

<tab label="WSZYSTKIE" rep_id="1">
<buttons>
<button>
<id>selectFR</id>
  <label>Potwierdź</label>
  <dscrpt>Potwierdza etap workflow</dscrpt>
  <onclick>
    App.openDialogByCls('P_WORKFLOW', 345,({afterSubmit:'{AFTER_SUBMIT}',clsnam:'DOCUMENT',keyval:{KEYVAL},ptstid:336}).toJ
  </onclick>
  <icon>pass-24.png</icon>
</button>
</buttons>
</tab>

```

## Moduł operujący na istniejących obiektach

W poniższym przykładzie ewidencjonować można sprawy. Należy zmienić: dsexid - to id z wyciągu z wykazu akt, rep\_id - id raportu

```

<module id="cModule_U2" name="NEW" label="New" icon="fixdefect-48.png" right="bswfms.processes">
  <toolbar>
  </toolbar>
  <filters>
  </filters>
  <tabs>
    <tab label="AKTYWNE" rep_id="1">
      <buttons>
<button>
  <id>newbug</id>
  <label>Zgłoszenie</label>

```

```

<dscrpt>Nowe zgłoszenie. Sprawa zostanie założona w teczce</dscrpt>
<onclick>
App.createDialog('createProcessForm','SimpleProcessCreatingForm','./modules/AProcesses/forms/SimpleProcessCreatingForm')
</onclick>
<icon>fix-defect-24.png</icon>
</button>
<button>
  <id>refresh20</id>
  <label>Odśwież</label>
  <dscrpt>Odśwież</dscrpt>
  <onclick>
  {AFTER_SUBMIT}
  </onclick>
  <icon>sync-24.png</icon>
</button>
</buttons>
</tab>
<tab label="WSZYSTKIE" rep_id="2">
<buttons>
  <button>
    <id>refresh26</id>
    <label>Odśwież</label>
    <dscrpt>Odśwież</dscrpt>
    <onclick>
    {AFTER_SUBMIT}
    </onclick>
    <icon>sync-24.png</icon>
  </button>
</buttons>
</tab>
</tabs>
</module>

```

## Wersja

### 1.9

Dodano obsługę atrybutu visible dla taga custom\_widget podobnie jak dla taga tab

```

<custom_widget visible="{getGlobalSystemConfig(#RR#, #DDD.AAA#)}">
78
</custom_widget>

```

Atrybut ten pozwala za pomocą wpisu w tabeli global\_sys\_conf na sterowanie widocznością przycisku (oprócz możliwości definiowania uprawnień bezpośrednio z GUI) [1.3](#)

### 1.8

Możemy sterować dostępem do zakładek jak i modułu w zależności od typu użytkownika za pomocą atrybutu licenceTypeAccessRule. Atrybut należy uzupełnić regułą: Allow:<typy użytkowników oddzielone przecinkami> np. "Allow:PORTAL,REGULAR" wpisując taką regułę zezwalamy użytkownikom typu REGULAR i PORTAL na dostęp do zakładki lub modułu na którym atrybut został określony. Reguły te możemy dodawać do tagów <tab> oraz <modules>

### !!!WAŻNE!!!

Domyślne zachowanie modułu i zakładki pozwala na wyświetlanie zakładki lub modułu tylko użytkownikom typu REGULAR, pokazanie dla użytkowników typu portal należy jawnie zadeklarować w definicji modułu.

Przykład zastosowania:

```

<module id="moduleName_tab_chart" licenceTypeAccessRule="Allow:REGULAR,PORTAL" label="Wykresy" name="charts" color
  <tabs>
    <tab label="WSZYSTKIE" rep_id="89" licenceTypeAccessRule="Allow:REGULAR"></tab>
    <tab label="Miesięcznie" rep_id="90" licenceTypeAccessRule="Allow:PORTAL"></tab>
  ...

```

## 1.7

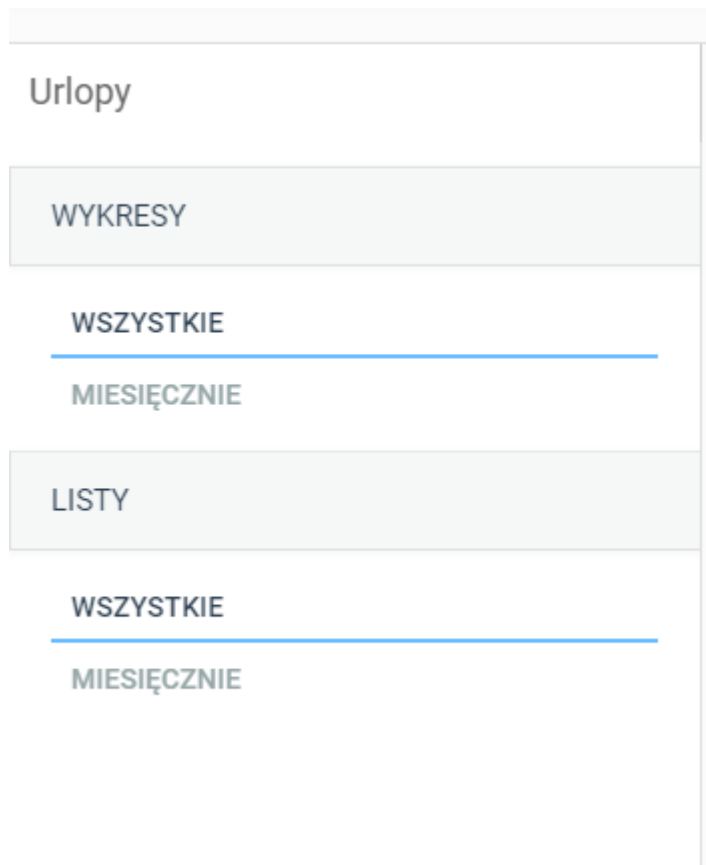
Grupowanie zakładek dla panelu pionowego dokonujemy poprzez dodanie atrybutu groupName dla taga tab. Przykładowa struktura XML

```

<tab id="moduleName_tab_chart" label="Wykresy" rep_id="59">
  <tabs>
    <tab label="WSZYSTKIE" rep_id="89" groupName="Wykresy"></tab>
    <tab label="Miesięcznie" rep_id="90" groupName="Wykresy"></tab>
    <tab label="WSZYSTKIE" rep_id="189" groupName="Listy"></tab>
    <tab label="Miesięcznie" rep_id="190" groupName="Listy"></tab>
  ...

```

Widok poniżej



(Szablony systemowe)

## 1.6

Widok kompaktowy (wąski, dopasowany do lebeli) dla pionowego panelu zakładek (compactView="1")

Przykładowa struktura XML

```

<tab id="moduleName_tab_chart" label="Wykresy" rep_id="89">
  <tabs compactView="1">
    <tab label="WSZYSTKIE" rep_id="89">

```

```

</tab>
<tab id="moduleName_tab_sub_chart" label="Kołowy" rep_id="189">
...

```

### 1.5

Wersja wprowadza możliwość odświeżania panelu filtrów (jeśli filtry zostały zdefiniowane w raporcie - nie dotyczy filtrów zdefiniowanych w pliku XML).

Ogólnie sprawa wygląda dość prosto. Dodany został dodatkowy token (aby zachować wsteczną zgodność) o nazwie {REFRESH\_FILTERS}. Dla przykładu niech posłuży definicja przycisku odświeżającego listę

```

<button>
  <id>refresh26</id>
  <label>Odśwież</label>
  <dscript>Odśwież</dscript>
  <onclick>
    {AFTER_SUBMIT} {REFRESH_FILTERS}
  </onclick>
  <icon>refresh.png</icon>
</button>

```

W tej postaci oprócz odświeżenia listy nastąpi odświeżenie panelu filtrów (tokeny można łączyć).

Drugi przykład wykorzystania to przekazanie funkcji do customowego skryptu - czyli definicja przycisku wygląda np. tak

```

<button>
  <custom_widget>
    71
  </custom_widget>
</button>

```

Wtedy w kodzie php obsługując przekazaną tablicę \$args mamy dodatkowy klucz js\_refresh\_filters. Oczywiście wartość tego klucza może być pusta jeśli raport nie ma filtrów.

Przykładowe dane z tablicy \$args

```

Array
(
    [cswgid] => 71 (string => string)
    [name] => cModule_UcModule_U_PPA (string => string)
    [tab] => cModule_UcModule_U_PPATH80b0babd932952a (string => string)
    [rep_id] => 89 (string => integer)
    [REPORT_NAME] => REP_cModule_UcModule_U_PPATH80b0babd932952aLi_Bean_89 (string => string)
    [js_after] => App.cModule_UcModule_U_PPATH80b0babd932952aLirefresh(); (string => string)
    [js_filters_refresh] => App.cModule_UcModule_U_PPATH80b0babd932952arefreshFilters(); (string => string)
    [keyval] => array() (string => array)
    [listName] => cModule_UcModule_U_PPATH80b0babd932952aLi (string => string)
)

```

### 1.4

Dodatkowy atrybut dla taga tab to atrybut id

```

<tab id="moduleName_tab_chart" label="wykres" rep_id="190">
</tab>

```

Zalecaną konwencją nazewnictwa jest dodanie prefixu jako nazwy modułu. Obecne zastosowanie tego atrybutu to "ręczne" otwieranie modułu customowego np. z przycisku oraz automatycznie zaznaczenie konkretnej zakładki (obecnie tylko głównej - nie z pionowego paska)

Przykład wykorzystania



```
App.openDialogByCls('APPLICATION', 0, ({name:'moduleName', action:'custommodule', selectTab:'moduleName_tab_chart'}).toJSON)
```

Gdzie moduleName to wartość wprowadzona do atrybutu id tagu module

```
<module id="moduleName" name="Kanban" label="Kanban" icon="CallTransfer.svg" color="#c0392b">
...
</module>
```

Opcja umożliwia przekazanie zaznaczenia zakładki również dla poziomego paska zakładek (czyli struktura XML tabs -> tab -> tabs). Aby to zrobić należy użyć wywołania

```
App.openDialogByCls('APPLICATION', 0, ({name:'moduleName', action:'custommodule', selectTab:'moduleName_tab_chart/moduleName'})
```

Czyli dodajemy separator "slash" (/).

Przykładowa struktura XML

```
<tab id="moduleName_tab_chart" label="Wykresy" rep_id="89">
  <tabs>
    <tab label="WSZYSTKIE" rep_id="89">
      <buttons>
        <button>
          <id>refresh26</id>
          <label>{translate(#Mój moduł#, #cModule_U#)}</label>
          <dscrpt>Odśwież</dscrpt>
          <onclick>
            {AFTER_SUBMIT}
          </onclick>
          <icon>refresh.png</icon>
        </button>
      </buttons>
    </tab>
    <tab id="moduleName_tab_sub_chart" label="Kołowy" rep_id="189">
      ...
    </tab>
  </tabs>
</tab>
```

### 1.3

Dodatkowe atrybuty dla taga tab to:

- enabled - umożliwia dodanie zakładki bez jej wizualizacji. W zamian możliwy jest do wyświetlenia tekst zastępczy - atrybut explain\_how\_to\_enabled
- explain\_how\_to\_enabled - tekst informujący jak włączyć funkcjonalność danej zakładki
- visible - flaga czy dana zakładka ma być widoczna - nie ma znaczenia atrybut grp\_id lub prawo

Jako wartości dla tych atrybutów (oprócz explain\_how\_to\_enabled) należy zastosować następującą konstrukcję:

```
<tab label="{translate(#Zakładka#)}" rep_id="90" enabled="{getGlobalSystemConfig(#RR#, #DDD.AAA#)}" explain_how_to_enabled=
</tab>
```

Wartość dla tych atrybutów przechowywane są w bazie w tabeli global\_sys\_conf. Aby je pobrać należy użyć funkcji getGlobalSystemConfig, która jak pierwszy argument przyjmuje wartość w kolumnie objnam (tabela global\_sys\_conf) a drugi argument to wartość w kolumnie varnam. Jako wartość value dla tych dwóch flag należy ustawić 0 lub 1.

### 1.2

Obsługa dodatkowego atrybutu tagu module "always\_show\_tabs". Ustawienie atrybutu always\_show\_tabs="TRUE" powoduje wymuszenie wyświetlenia zakładek górnego panelu

```
<module id="cModule_U2" name="NEW" label="New" icon="fixdefect-48.png" always_show_tabs="TRUE">
...
</module>
```

## 1.1

Obsługa przycisku menu dokumentów

```
<button type="DOCUMENTS_MENU" group="REQUEST_DOCUMENT_GROUP">
  <id>new</id>
  <label>Nowy wniosek</label>
  <dscrpt>Nowy wniosek</dscrpt>
  <icon>new.png</icon>
</button>
```

Dla atrybutu \*group\* możemy podać ID grupy dokumentów (types\_of\_documents\_groups.dctgid) lub GUID (types\_of\_documents\_groups.dgguid) co spowoduje wyświetlenie tylko elementów z tej gałęzi drzewa.

## 1.0

Obsługa customowego przycisku dla opcji menu zdefiniowanej jak dla zmiany 0.8

```
<button>
  <id>pdf2</id>
  <label>asdsad</label>
  <dscrpt>PDF</dscrpt>
  <icon> pdf.png</icon>
  <buttons>
    <button>
      <custom_widget>
        3
      </custom_widget>
    </button>
    <button>
      <id>pdf</id>
      <label/>
      <dscrpt>PDF</dscrpt>
      <onclick> window.open('litescript.php/iyp/?script=GenerateReport&ent_id=2&ent_name=
      <icon> pdf.png</icon>
    </button>
    <button>
      <custom_widget>
        38
      </custom_widget>
    </button>
    <button>
      <id>
        new
      </id>
      <label>
        Nowa
      </label>
      <dscrpt>
        Nowa pmAddTask
      </dscrpt>
      <onclick>
        App.createDialog('pmAddTask', 'pmAddTask', './scripts/pmAddTask.inc', 'Utwórz nowe zadanie')
      </onclick>
      <icon>
        new.gif
      </icon>
    </button>
  </buttons>
</button>
```

## 0.8

Pożądana funkcja przez wielu. Możliwość grupowania przycisków w jeden jako menu. Aby tego dokonać należy wewnątrz sekcji <button> dodać podrzędną sekcję <buttons> jak poniżej

```
<button>
    <id>pdf2</id>
    <label>asdsad</label>
    <dscrpt>PDF</dscrpt>
    <icon> pdf.png</icon>
    <buttons>
        <button>
            <id>pdf</id>
            <label/>
            <dscrpt>PDF</dscrpt>
            <onclick> window.open('litescript.php/iyp/?script=GenerateReport&ent_id=2&ent_name=
            <icon> pdf.png</icon>
        </button>
        <button>
            <custom_widget>
                38
            </custom_widget>
        </button>
        <button>
            <id>
                new
            </id>
            <label>
                Nowa
            </label>
            <dscrpt>
                Nowa pmAddTask
            </dscrpt>
            <onclick>
                App.createDialog('pmAddTask', 'pmAddTask', './scripts/pmAddTask.inc', 'Utwórz nowe zada
            </onclick>
            <icon>
                new.gif
            </icon>
        </button>
    </buttons>
</button>
```

## 0.7

Obsługa pionowego panelu zakładek umożliwia tworzenia zakładek wewnątrz danej głównej zakładki. Użycie tego jest bardzo proste. Wystarczy pod danym tagiem tab (taki tag nie musi posiadać atrybutu rep\_id i żadnych innych tagów pod sobą oprócz tabs) dodać kolejną sekcję z tabs czyli np.

```
<module id="cModule_U2" name="NEW" label="New" icon="fixdefect-48.png" right="bswfms.processes">
    <toolbar>
    </toolbar>
    <filters>
    </filters>
    <tabs>
        <tab label="AKTYWNE">
            <tabs>
                <tab label="AKTYWNE">
                    <buttons>
<button>
    <id>newbug</id>
    <label>Zgłoszenie</label>
```

```

<dscrpt>Nowe zgłoszenie. Sprawa zostanie założona w teczce</dscrpt>
<onclick>
App.createDialog('createProcessForm','SimpleProcessCreatingForm','./modules/AProcesses/forms/SimpleProcessCreatingForm')
</onclick>
<icon>fix-defect-24.png</icon>
</button>
<button>
  <id>refresh20</id>
  <label>Odśwież</label>
  <dscrpt>Odśwież</dscrpt>
  <onclick>
    {AFTER_SUBMIT}
  </onclick>
  <icon>sync-24.png</icon>
</button>
</buttons>
</tab>
<tab label="AKTYWNE">
  <buttons>
<button>
  <id>newbug</id>
  <label>Zgłoszenie</label>
  <dscrpt>Nowe zgłoszenie. Sprawa zostanie założona w teczce</dscrpt>
  <onclick>
App.createDialog('createProcessForm','SimpleProcessCreatingForm','./modules/AProcesses/forms/SimpleProcessCreatingForm')
</onclick>
  <icon>fix-defect-24.png</icon>
</button>
  <button>
    <id>refresh20</id>
    <label>Odśwież</label>
    <dscrpt>Odśwież</dscrpt>
    <onclick>
      {AFTER_SUBMIT}
    </onclick>
    <icon>sync-24.png</icon>
  </button>
  </buttons>
</tab>
</tabs>
</tab>
</tabs>
</module>

```

```
<div onclick="{FILE_EXPLORER}">Eksplorator plików</div>
```

Przejdź do [Menu](#)