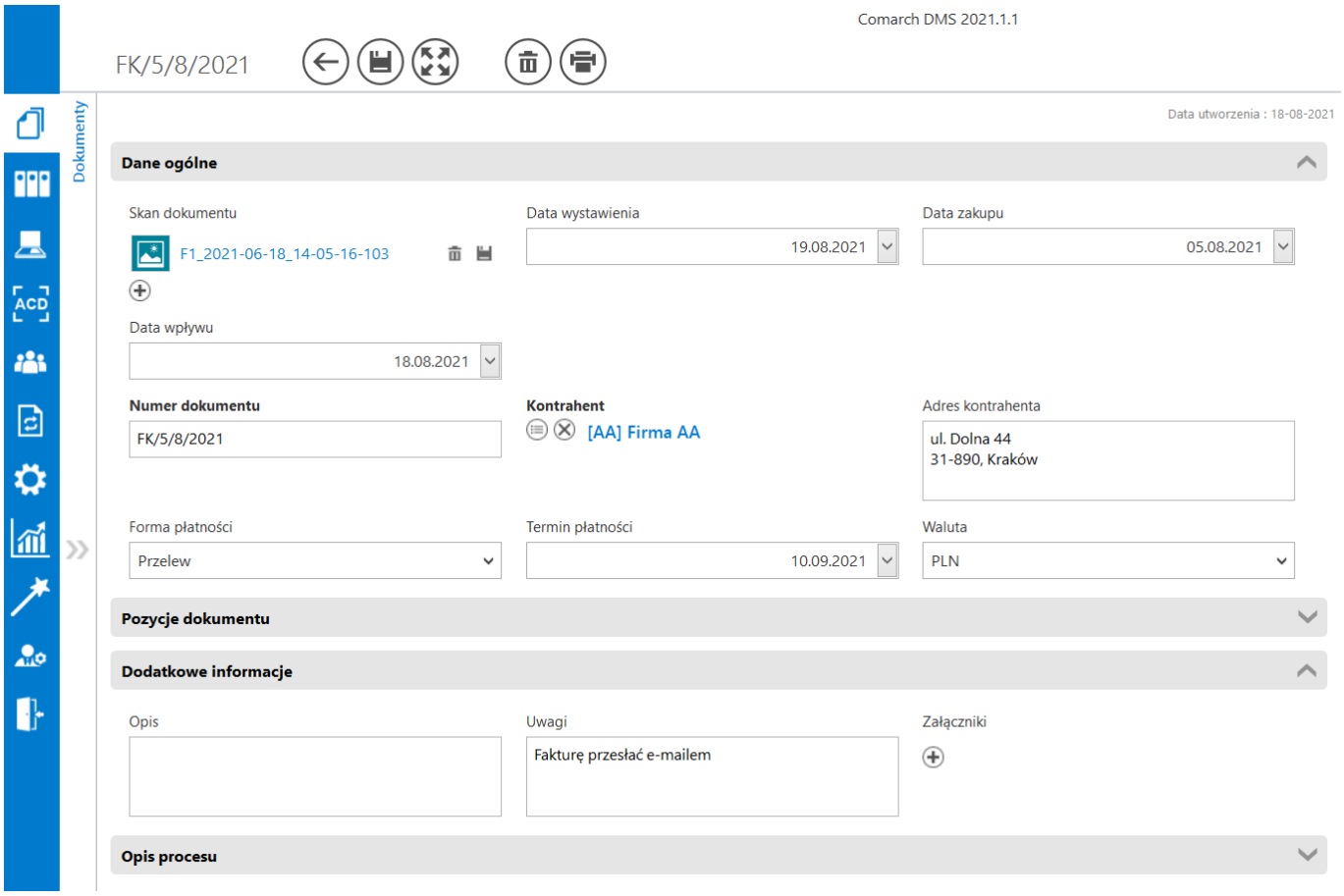
Konfiguracja sekcji dokumentu

Konfiguracja karty obiegu w Comarch DMS odbywa się poprzez definiowanie sekcji dokumentu. Każda z sekcji prezentowanych na dokumencie możne być zwinięta lub rozwinięta w zależności od aktualnych potrzeb operatora pracującego z kartą obiegu oraz od ustawienia prezentacji sekcji w panelu użytkownika.

Na nowo dodanym typie obiegu tworzona jest automatycznie pierwsza sekcja o nazwie ‘Karta obiegu’. Możliwa jest zmiana nazwy sekcji, w przykładzie poniżej nazwa ‘Karta obiegu’ została zamieniona na ‘Dane ogólne’.



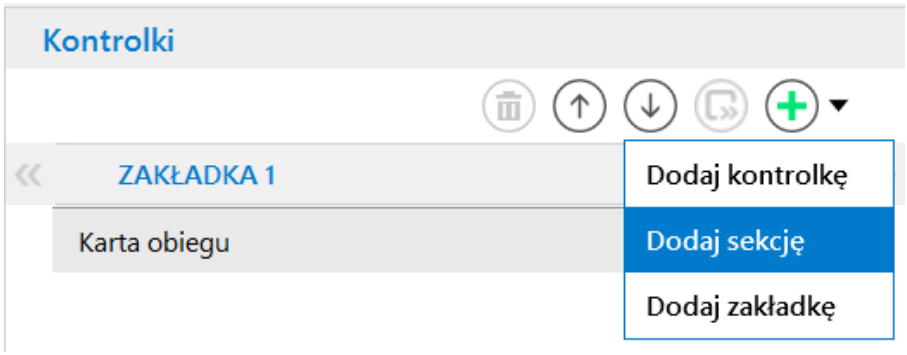
Dokument Comarch DMS w nowej prezentacji z podziałem na sekcje

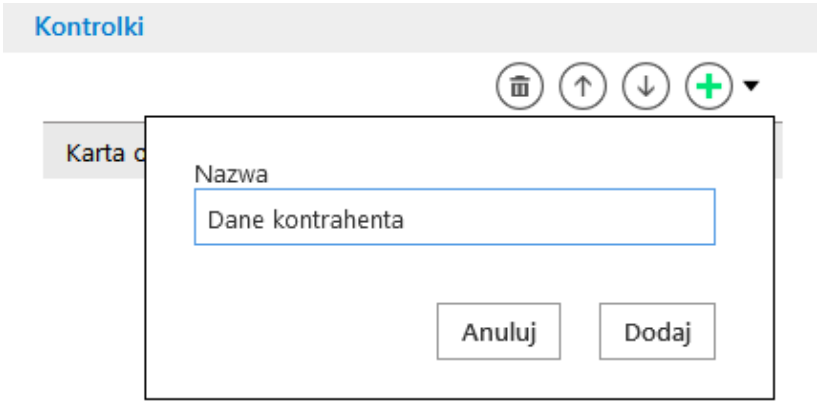
Uwaga
W wersjach Comarch DMS nowszych od wersji 2019.1 na wszystkich

istniejących definicjach obiegów dokumentów dodawana jest sekcja o nazwie ‘Karta obiegu’, na której prezentowane są wszystkie zdefiniowane wcześniej kontrolki. W przypadku Repozytoriów dodawana jest sekcja ‘Repozytorium’.

Konfiguracja sekcji

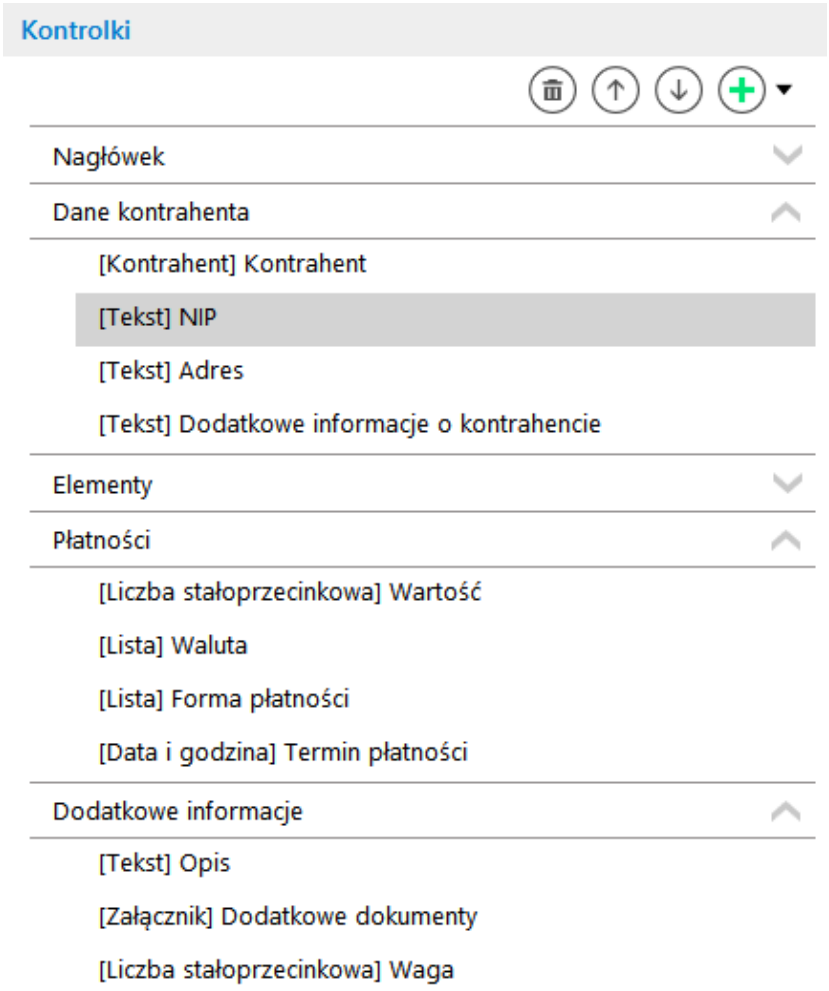
Aby dodać nową sekcję należy na liście kontrolek rozwinąć menu obok plusa i wybrać opcję „Dodaj sekcję”, uzupełnić nazwę sekcji i wybrać przycisk Dodaj .





Dodawanie nowej sekcji na definicji typu obiegu dokumentów

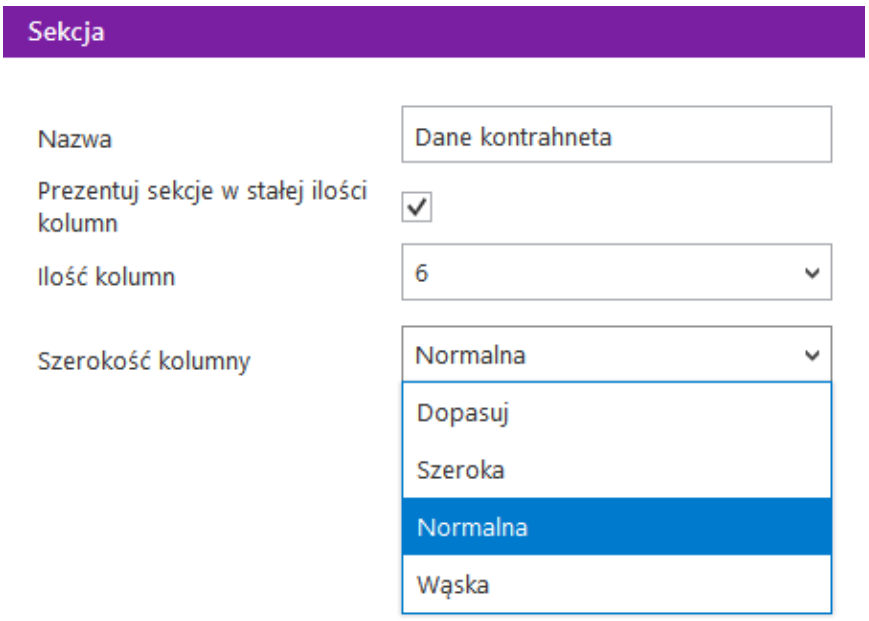
Dodana sekcja zostanie wyświetlona na liście kontrolek i sekcji.



Lista kontrolek na definicji typu obiegu, podział dokumentu na sekcje

W aktualnej wersji Comarch DMS konfiguracja liczby kolumn w której prezentowana jest karta obiegu zostaje przeniesiona z ustawień obiegu na ustawienia poszczególnych sekcji karty obiegu. W ustawieniach sekcji można określić, czy liczba zmienia się dynamicznie w zależności od szerokości karty obiegu, czy jest stała. Dla każdej sekcji ustawienia liczby kolumn i ich szerokości są osobne.

Uwaga
Po konwersji Comarch DMS domyślnie utworzona sekcja ‘Karta obiegu’ będzie prezentowana zgodnie z dotychczasowymi ustawieniami liczby kolumn, w których jest prezentowana.



Właściwości sekcji

Prezentuj sekcję w stałej ilości kolumn – zaznaczenie parametru spowoduje, że w ramach sekcji kontrolki będą wyświetlane w liczbie określonej w parametrze *Ilość kolumn*.

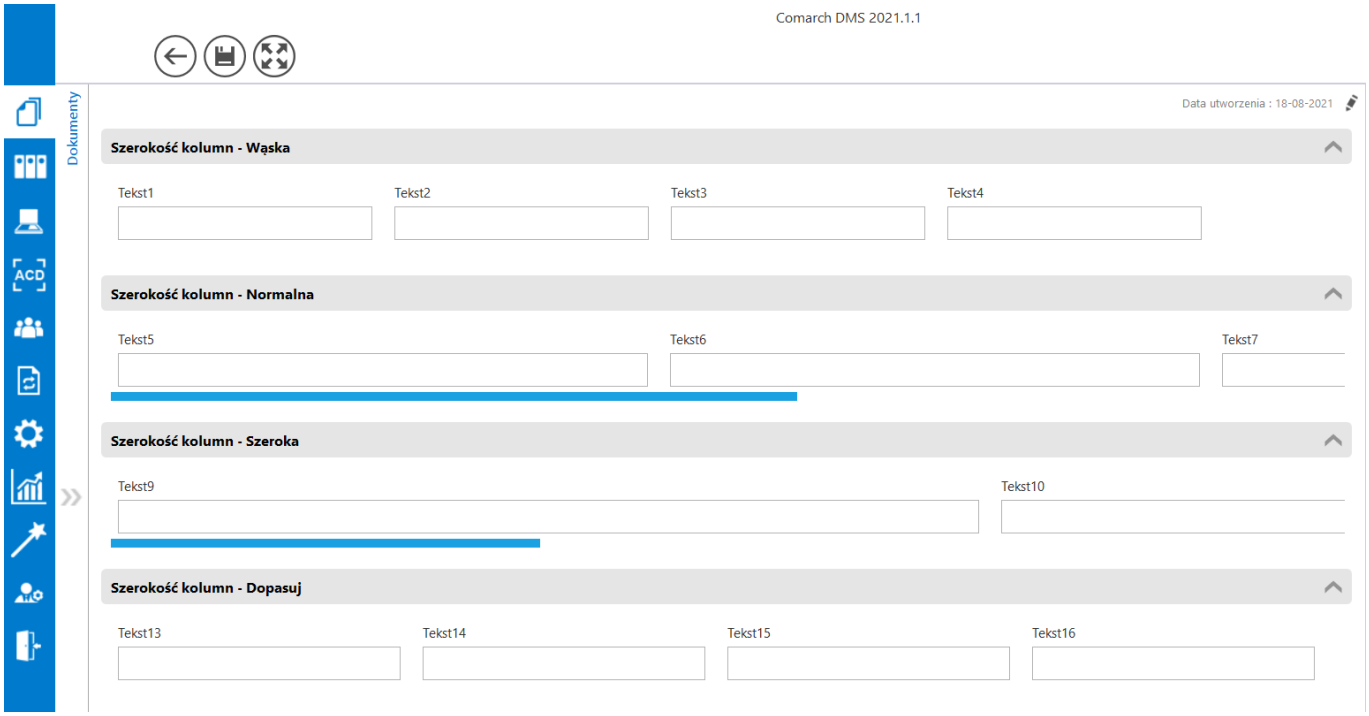
Ilość kolumn – określa liczbę kolumn, w której będą wyświetlane kontrolki w ramach sekcji. Maksymalna liczba kolumn określana jest w pliku web.config w kluczu MaximumNumberOfControlCardColumns.

Szerokość kolumny – określa szerokość kolumn wyświetlanych w ramach sekcji. Dostępne są wartości:

- Dopasuj – wybranie wartości spowoduje, że szerokość prezentowanych kolumn będzie różna w zależności od szerokości karty obiegu,
- Wąska,
- Normalna,
- Szeroka.

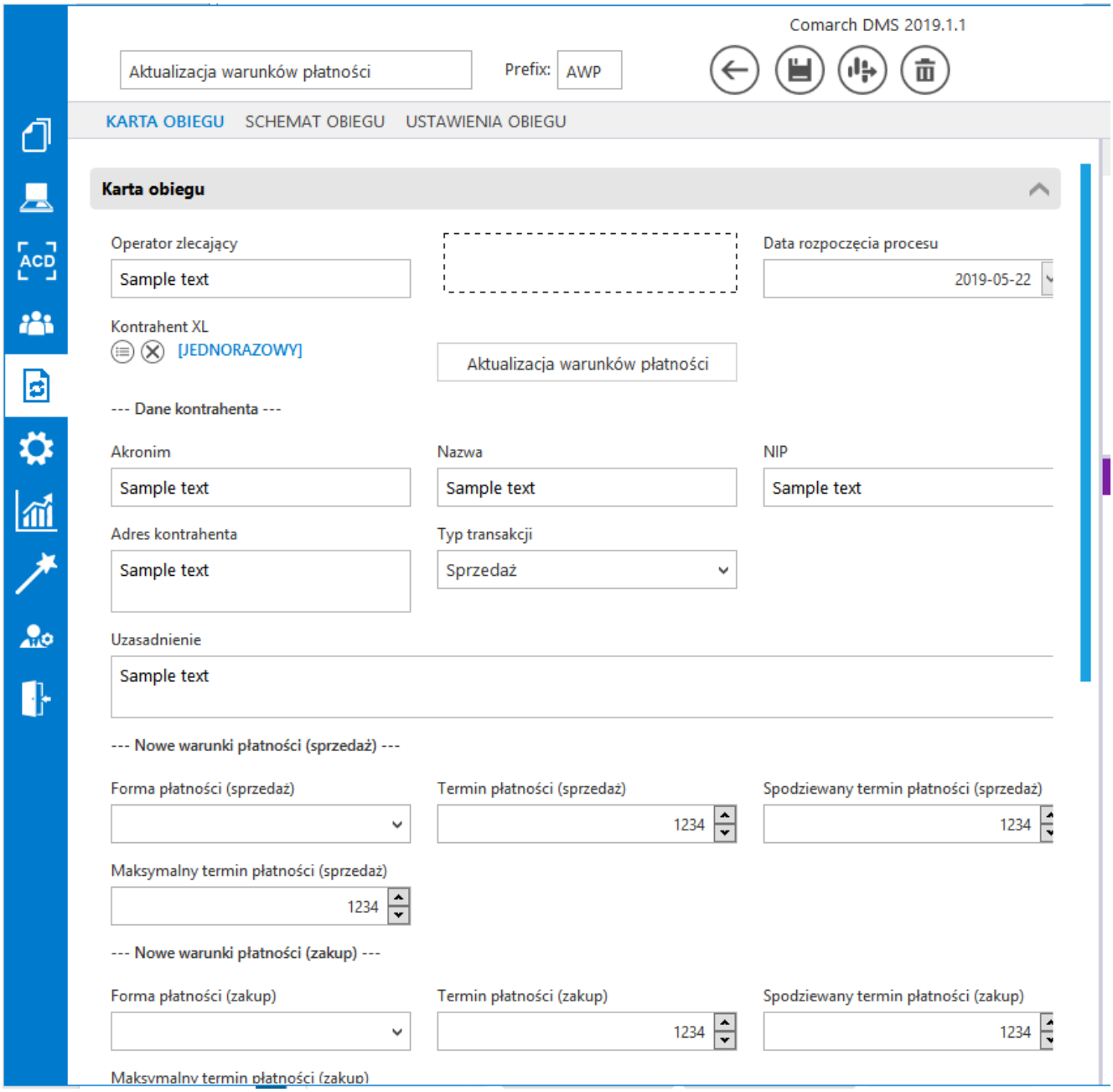
Przyklad
Zdefiniowano testową kartę obiegu, w której dodano cztery

sekcje. W każdej sekcji określony został inny rozmiar kolumn w ramach sekcji. W pierwszej sekcji kolumny wyświetlone są w rozmiarze wąskim, następnie w normalnym, w trzeciej sekcji określono rozmiar szeroki i w ostatniej wybrano opcję dopasuj.



Analogicznie do kontrolek, sekcje mogą być ukrywane na dokumencie na poszczególnych etapach procesu, konfiguracja odbywa się na zakładce *Kontrolki* we właściwościach etapu.

W lewej części okna definicji typu obiegu wyświetlany jest podgląd karty obiegu w podziale na sekcje wraz ze wszystkimi zdefiniowanymi kontrolkami.



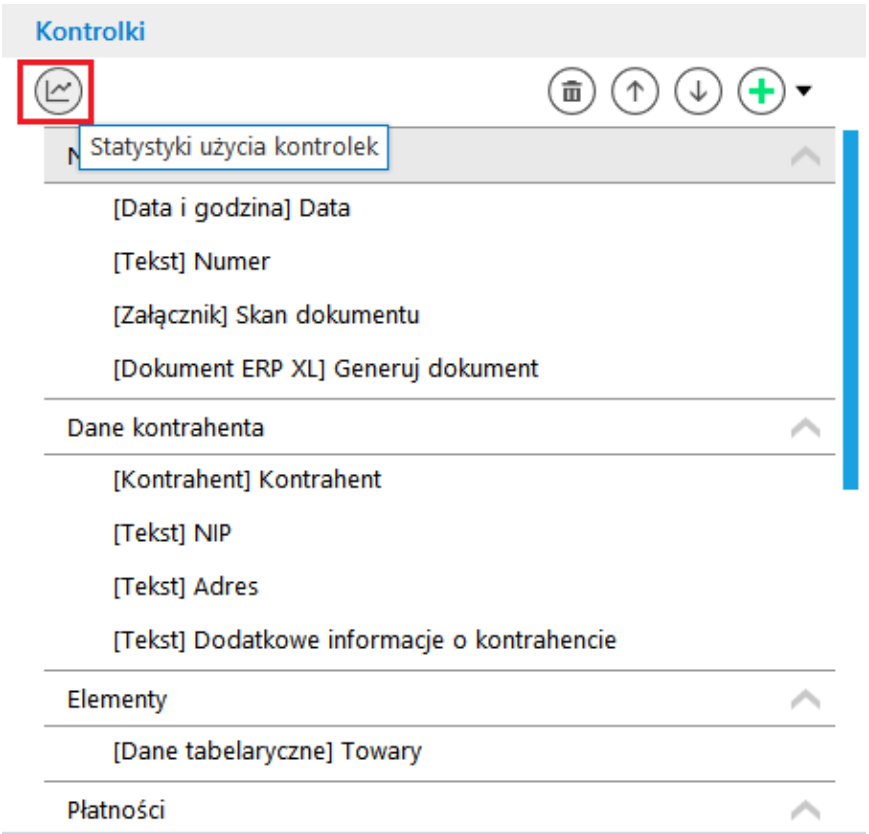
Podgląd karty obiegu

Analiza wykorzystania i uzupełniania kontrolek

na karcie obiegu

W Comarch DMS dostępna jest funkcjonalność analizy wykorzystania i uzupełniania kontrolek na karcie obiegu. Przycisk „Statystyki użycia kontrolek” dostępny jest na definicji typu obiegu.

Podczas pracy z definicją typu obiegu operator posiadający uprawnienia administratora ma możliwość wyświetlenia listy kontrolek z karty obiegu wraz z informacją, kiedy ostatnio kontrolka była używana na dokumentach. Na podstawie tych danych administrator może zadecydować czy kontrolka nadal jest potrzebna na karcie obiegu, czy może ją usunąć.



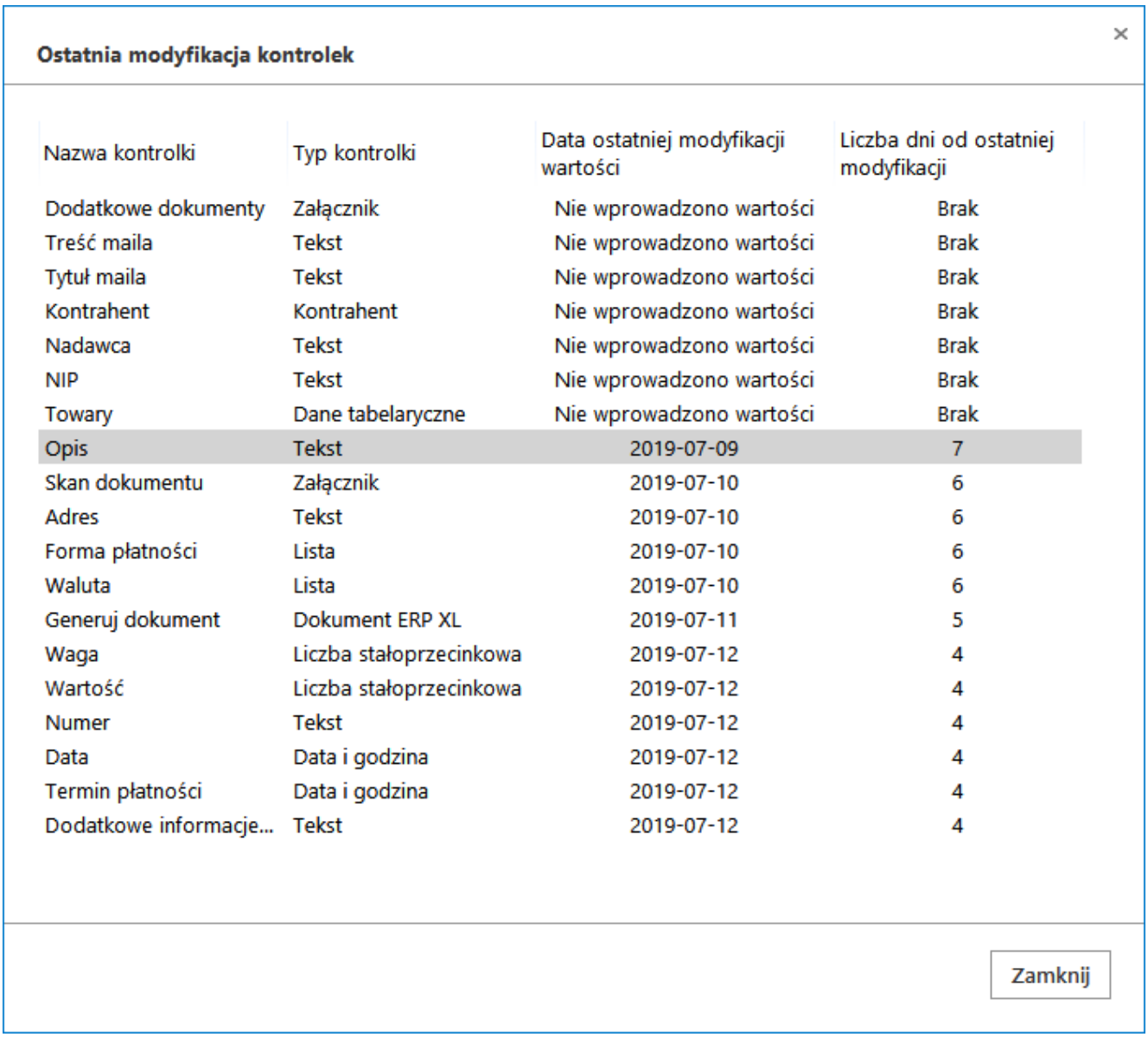
Lista kontrolek na definicji typu obiegu dokumentów, przycisk Statystyki użycia kontrolek

Przycisk ‘Statystyki użycia kontrolek’ wywołuje okno ‘Ostatnia modyfikacja kontrolek’. W oknie prezentowane są wszystkie kontrolki znajdujące się na definicji typu obiegu wraz z

informacją o liczbie dni od ostatniej modyfikacji kontrolki.

Ostatnia modyfikacja kontrolek zawiera kolumny:

- Nazwa kontrolki – prezentowana jest lista nazw kontrolek znajdujących się na definicji typu obiegu.
- Typ kontrolki – prezentuje typ dodanej kontrolki.
- Data ostatniej modyfikacji wartości – wyświetla datę ostatniej modyfikacji wartości w kontrolce, w przypadku, gdy kontrolka nie była używana, wyświetlana jest informacja „Nie wprowadzono wartości”.
- Liczba dni od ostatniej modyfikacji – wyświetla liczbę dni, które minęły od ostatniej modyfikacji wartości kontrolki, gdy kontrolka nie była używana, wyświetlana jest informacja „Brak”.



Okno „Ostatnia modyfikacja kontrolek”

Na liście kontrolek w pierwszej kolejności wyświetlane są kontrolki, które nie były używane na dokumentach w obiegu, następnie kontrolki od największej do najmniejszej liczby dni które upłynęły od ostatniej modyfikacji kontrolki.

Modelowanie procesów za pomocą C#

Uwaga
W wersji 2020.0.0 Comarch DMS mechanizm modelowania procesów metodą C# jest wersją beta.
Od wersji 2021.1.0 modelowanie procesów w języku C# jest objęte licencją Premium.

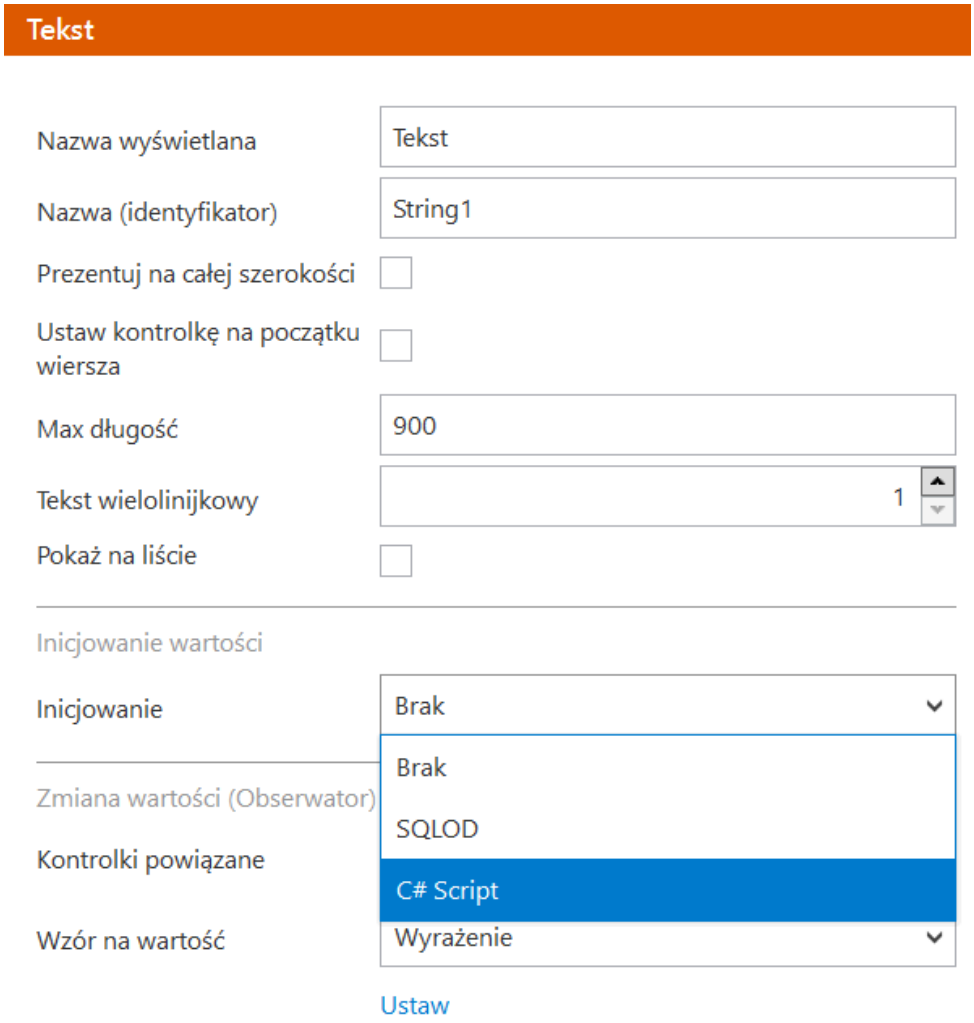
Oprogramowano funkcjonalność tworzenia własnych zdarzeń w języku C#. W aktualnej wersji możliwość modelowania zdarzeń w języku C# dostępna jest w kontrolkach:

- Dane tabelaryczne,
- Dane tabelaryczne, kolumna typu lista,
- Data i godzina,
- Dokument elektroniczny,
- Komunikat,
- Kontrahent,
- Liczba całkowita,
- Liczba rzeczywista,
- Liczba stałoprzecinkowa,
- Lista,
- Tekst,
- Towar,
- Własna akcja.

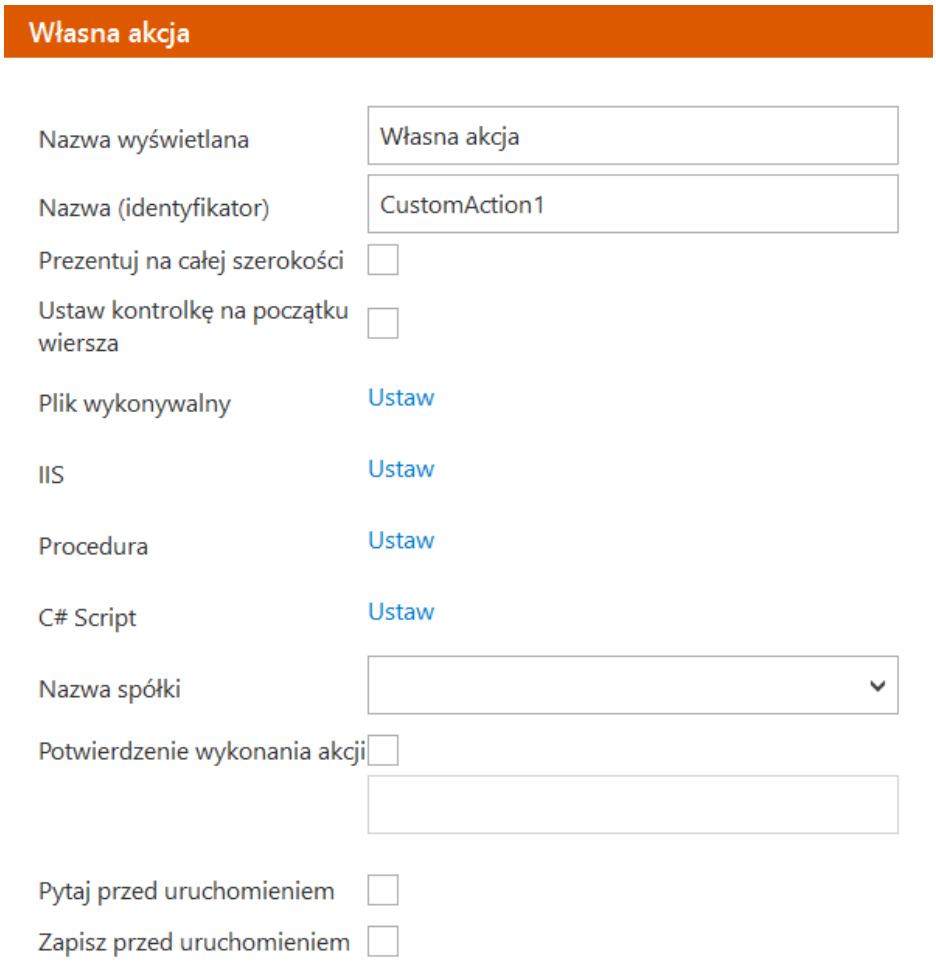
W większości kontrolek możliwe jest tworzenie skryptów służących zarówno inicjowaniu wartości w kontrolkach, jak i obserwowaniu wartości innych kontrolek. Obserwowanie wartości umożliwia „reagowanie” na zmiany. Dla kontrolek typu dane tabelaryczne oraz własna akcja możliwa jest również „obserwacja samej siebie”.

Aby włączyć funkcjonalność należy przy instalacji lub aktualizacji systemu zaznaczyć parametr „Wersja Premium” lub w w pliku Web.config wartość klucza *PremiumFunctionality* ustawić na „True” i wprowadzić odpowiedni klucz licencji. <add key=”PremiumFunctionality” value=”false” />

Włączenie edytora C# skutkuje pojawieniem się dodatkowych opcji na definicjach kontrolek.



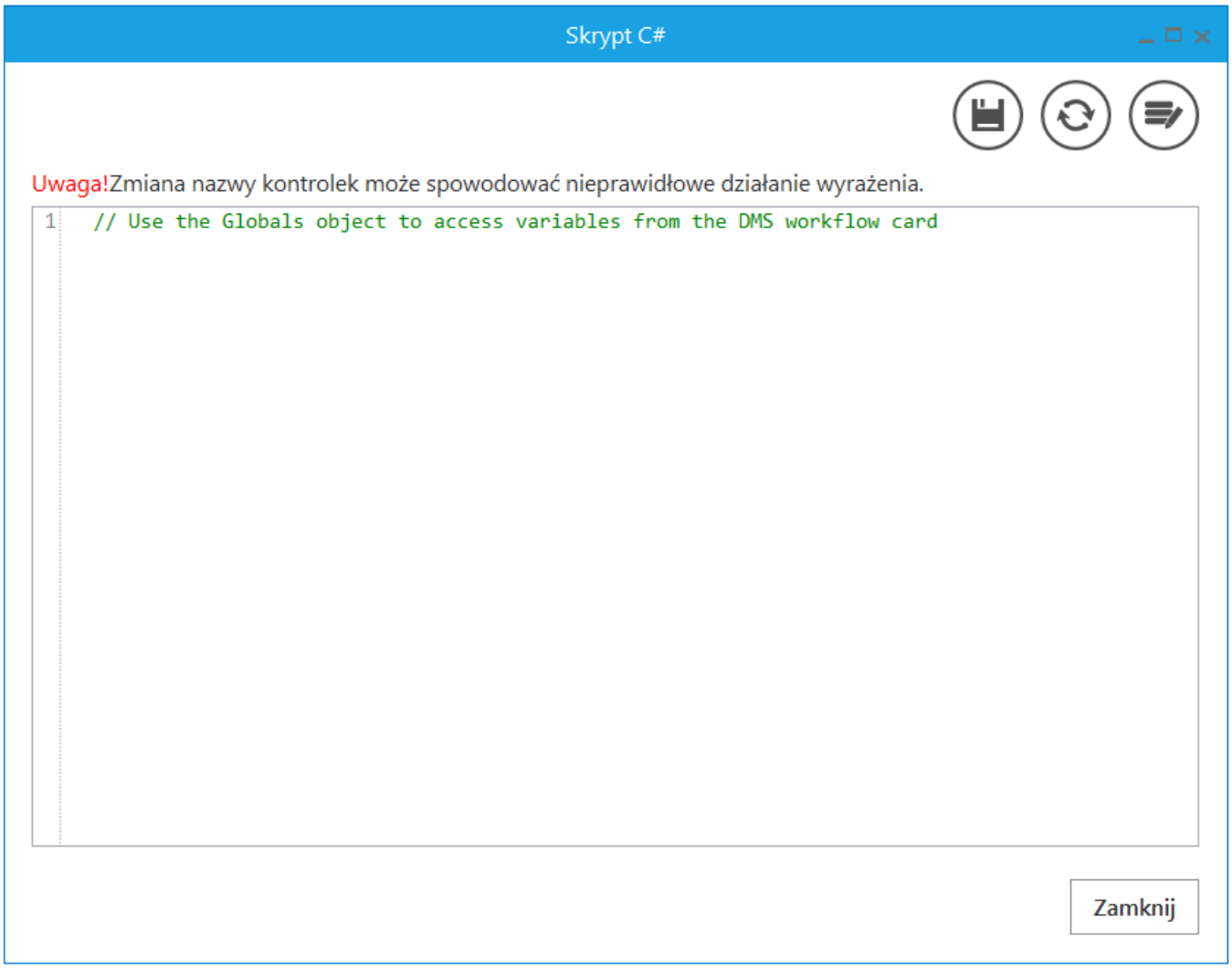
Pozycja C# Script w opcjach inicjowania wartości kontrolki typu tekst



Pozycja C# Script w kontrolce typu Własna akcja

Po wskazaniu inicjowania wartości za pomocą C# Script i

kliknięciu w uruchamiana jest formatka, w której należy wpisać kod C#.



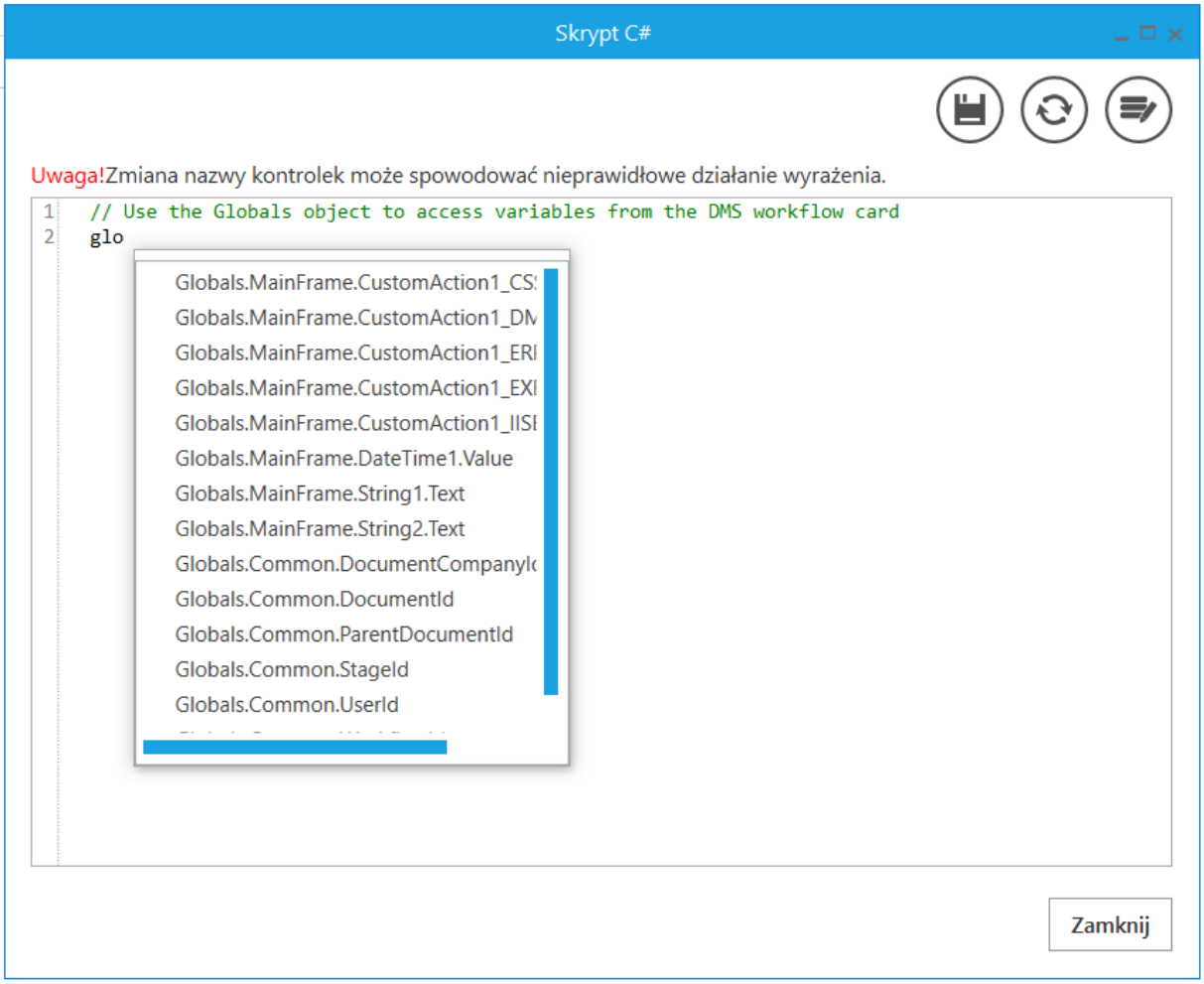
Okno Skrypt C#

Okno Skrypt C# składa się z pola, w którym należy wpisać kod, a następnie skompilować i zapisać przyciskiem .

Kod po skompilowaniu zapisywany jest w formie biblioteki w lokalizacji **…\bin\.scriptsc_cache** w katalogu, w którym zainstalowano aplikację web Comarch DMS.

W polu Referencje należy wpisać nazwy bibliotek, z których będzie korzystał kod C#. Domyślną lokalizacją, w której należy umieścić biblioteki jest **…\bin\scripts_bin** w katalogu, w którym zainstalowano aplikację web Comarch DMS.

W edytorze skryptów C# wyświetlane są podpowiedzi zawierające obiekty kontrolek karty obiegu (Globals.MainFrame.*) oraz zmienne (Globals.Common.*).



Podpowiedzi wyświetlane w edytorze skryptów C#

Modelowanie procesów za pomocą skryptu C# – podstawowe informacje

Użytkownik modelujący typ obiegu metodą C# ma do dyspozycji obiekt „Globals”, dzięki któremu ma dostęp do takich informacji jak np. identyfikator dokumentu Poprzez ten obiekt ma również dostęp do wartości kontrolek znajdujących się na karcie obiegu.

Obiekt „Globals” zawiera następujące informacje (właściwości):

- Właściwość „Common” – w której znajdują się następujące

identyfikatory:

- UserId (typu int) Identyfikator operatora,
- WorkflowId (typu int) Identyfikator typu obiegu,
- DocumentId (typu int?) Identyfikator dokumentu,
- StageId (typu int?) Identyfikator etapu,
- DocumentCompanyId (typu int) Identyfikator spółki,
- ParentDocumentId (typu int?) Identyfikator właściciela dokumentu.

Typ zakończony „?” (int?) oznacza, że oprócz wartości numerycznych zmienna może przechowywać również wartość NULL (pustą). Przykładowa sytuacja to brak wpisanej wartości w kontrolce typu liczba całkowita.

- Właściwość „MainFrame” – zawiera nazwy identyfikatorów kontrolek z karty obiegu.

W kontrolce Dane tabelaryczne dostępne są następujące funkcje i właściwości:

- MainFrame.<nazwaDT>.Items[Index]… – odwołanie do wartości kontrolki
  - Funkcje
    - MainFrame.<nazwaDT>.AddRow() – dodaj kolejny wiersz.
    - MainFrame.<nazwaDT>.AddRows(count) – dodaj wiersze, gdzie „count” oznacza ile wierszy ma zostać dodane.
    - MainFrame.<nazwaDT>.RemoveRow(index) – usuń wiersz, gdzie „index” to wiersz, który ma zostać usunięty.
    - MainFrame.<nazwaDT>.Clear() – usuwa wszystkie wiersze.
  - Właściwości
    - MainFrame.<nazwaDT>.Column – informacja o modyfikowanej komórce – numer kolumny.

- MainFrame.<nazwaDT>.Row – informacja o modyfikowanej komórce – numer wiersza.
- MainFrame.<nazwaDT>.RowCount – informacja o ilości wierszy w DT
  .

Wskazówka
W tablicach C# – pierwszy wiersz to 0

Uwaga
Podczas pierwszego uruchomienia operacji zamodelowanej mechanizmem C# może nastąpić opóźnienie działania mechanizmu. Wynika to z konieczności załadowania biblioteki dll do pamięci komputera.

Ze względu na konieczność zachowania nomenklatury nazewniczej języka C#, nie jest możliwe używanie polskich znaków diakrytycznych oraz spacji w identyfikatorach kontrolek.

Użytkownik może modyfikować wartości kontrolek poprzez modyfikację właściwości „Value” lub „Text”. Np. **Globals.MainFrame**.Liczba.Value = 10;

Możliwe jest to jednak tylko dla kontrolek, dla których został wprowadzony kod C#, czyli zostało wywołane zdarzenie OnInit lub OnChange (obserwacja). Pozostałe właściwości są tylko do odczytu.

Przyklad
Przepisanie wartości z kontrolki typu liczba całkowita o nazwie Liczba do kontrolki typu tekst o nazwie Tekst:
Globals.MainFrame.Tekst.Text
= **Globals.MainFrame**.Liczba.Value.**ToString**();

Przyklad
Pobranie wartości z kontrolki skladnik i zapisanie tej wartości w kontrolce suma powiększonej o 1.

```
var a = Globals.MainFrame.skladnik.Value ?? 0; // jeżeli pusta wartość zapisz 0
a = a + 1;
Globals.MainFrame.suma.Value = a;
```

Przyklad
Przepisanie wartości z kontrolki typu tekst do kontrolki typu liczba całkowita, pod warunkiem, że wpisana wartość jest liczbą.

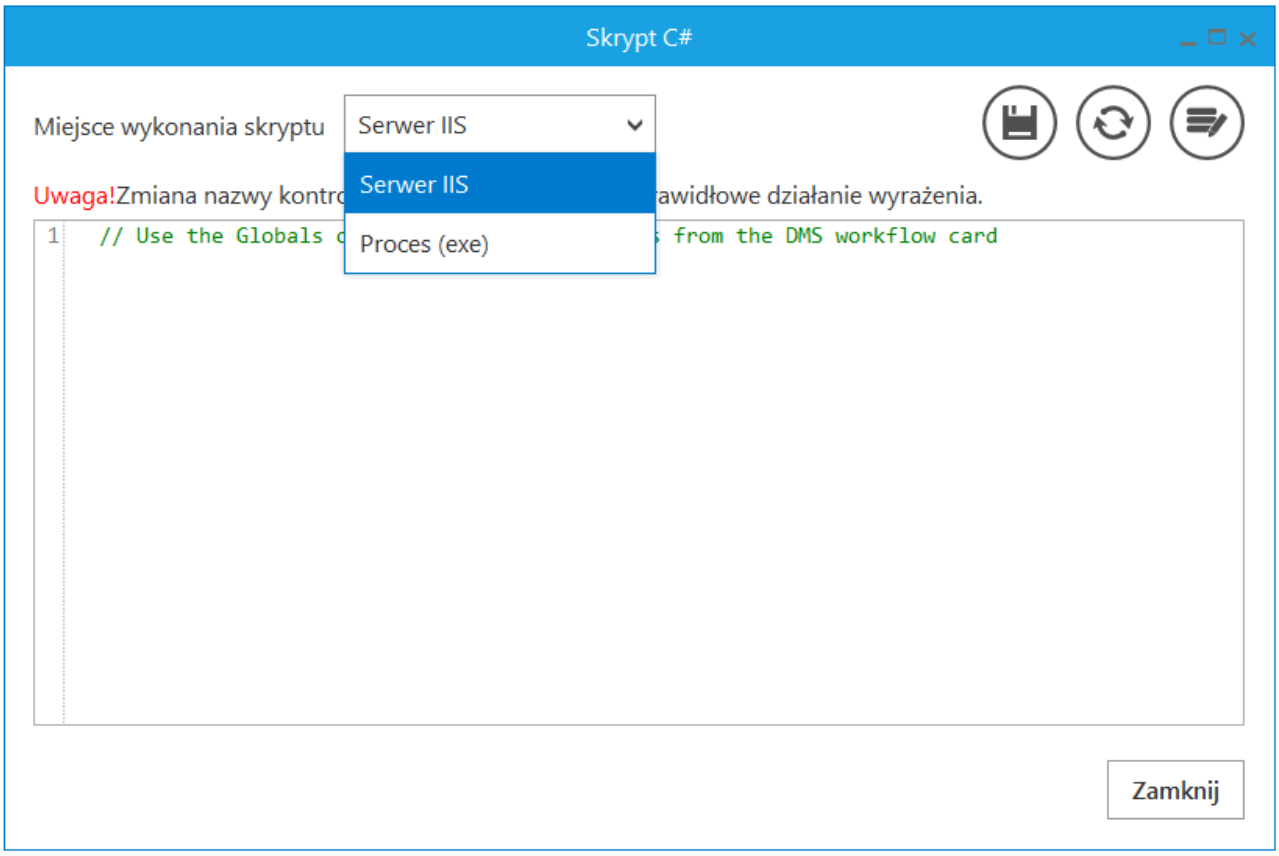
```
if (IsNumeric (Globals.MainFrame.Wartosc_tekst.Text))
{
Globals.MainFrame.Liczba.Value = Int32.Parse(Globals.MainFrame.Wartosc_tekst.Text);
}

public static bool IsNumeric(string value)
{
return value.All(char.IsNumber);
}
```

Przyklad
Przykład pobrania informacji o identyfikatorze operatora i wyświetlenia jej w kontrolce typu tekst.

```
var info = „Identyfikator operatora: ” + Globals.Common.UserId;
Globals.MainFrame.Informacja.Text = info;
```

Dla kontrolki „Własna akcja” został oprogramowany przełącznik, dzięki któremu można wskazać miejsce uchronienia skompilowanego kodu C#. Istnieje możliwość uruchomienia kodu w ramach procesu IIS lub w ramach niezależnego procesu (jednowątkowego). Uruchomienie na niezależnym procesie zalecane jest w przypadku obsługi API, które nie powinno działać na procesach wielowątkowych, jakim jest proces IIS.



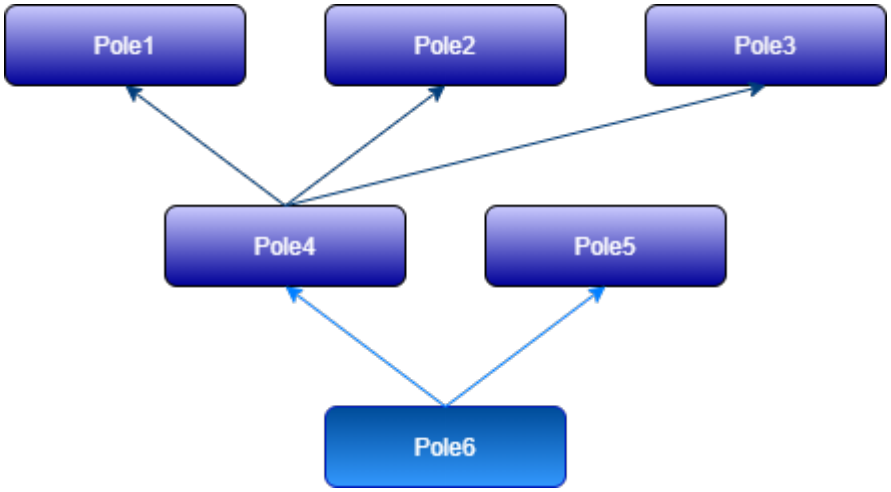
Ustawienie miejsca uruchomienia kodu C# dla kontrolki „Własna akcja”

Parametry ^SenderControlName oraz ^InitSenderControlName w mechanizmie obserwowania i zmiany wartości kontrolki na postawie innej kontrolki.

W mechanizmie modelowania procesów metodami SQL i C# w zdarzeniu obserwacji wartości kontrolek dotychczas dostępny był parametr ^SenderControlName. W aktualnej wersji

udostępniono parametr ^InitSenderControlName. Poniżej przykłady zastosowania obu parametrów:

- **^SenderControlName** – zwraca nazwę identyfikatora kontrolki będącej na najniższym poziomie mechanizmu obserwacji, po zmianie wartości w grupie kontrolek. Mechanizm użycia parametru ^SenderControlName prezentuje poniższy przykład.

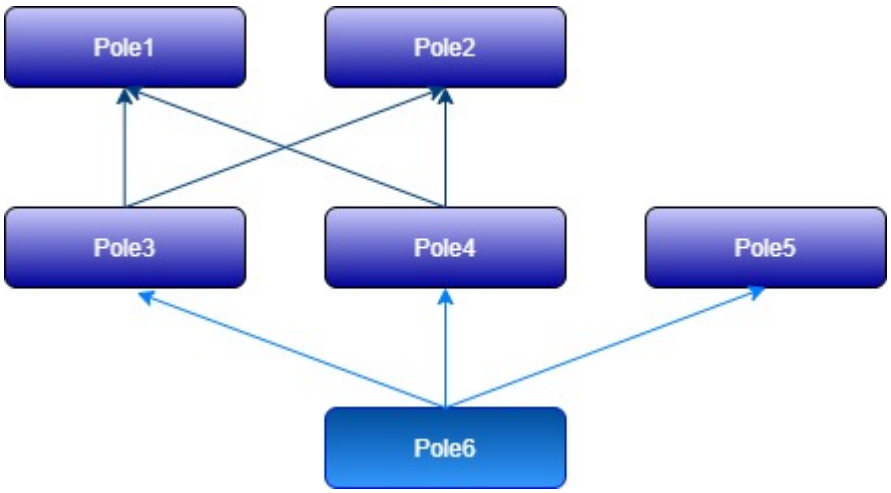


Kod obsługujący pobranie wartości dla kontrolki Pole6 może rozróżnić, w której grupie obserwacyjnej zmieniła się wartość pola. Jeżeli operator lub metoda inicjująca zmieniła wartości kontrolek Pole1, Pole2, Pole3 obserwowanych przez kontrolkę Pole4 (najniższy poziom dla kontrolki Pole6) w parametrze ^SenderControlName zostanie ustawiony identyfikator Pole4. Natomiast po zmianie wartości w kontrolce Pole5, ^SenderControlName przyjmie identyfikator kontrolki Pole5. Przykładowy kod dla obserwatora w kontrolce Pole6:

```
if @^SenderControlName@ =’Pole4´
select 'Zmieniono wartości w polach Pole1, Pole2 lub Pole3´
if @^SenderControlName@ = 'Pole5´
select 'Zmieniono wartość w polu Pole5´
```

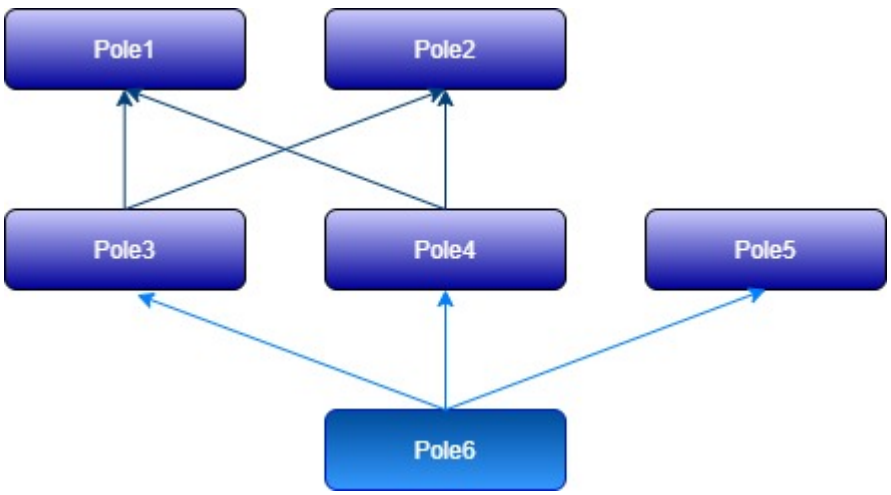
Jeżeli w grupie obserwacyjnej na najniższym poziomie

występują dwie lub więcej kontrolek, nie można określić, która z tych kontrolek zostanie zainicjowana jako ostatnia, dlatego kod obsługujący tego typu sytuacje powinien sprawdzić nazwy identyfikatorów we wszystkich kontrolkach ostatniego poziomu.




```
if @^SenderControlName@ =’Pole3′ or
@^SenderControlName@ = 'Pole4′
select 'Zmieniono wartości w polach Pole1 lub Pole2′
if @^SenderControlName@ = 'Pole5′
select 'Zmieniono wartość w polu Pole5′
```

- **^InitSenderControlName** – zwraca nazwę identyfikatora kontrolki, w której nastąpiła zmiana wartości lub kliknięcie w przycisk w przypadku kontrolki typu ‘Własna akcja’.



W powyższym przykładzie po zmianie wartości w polach Pole1, Pole2 lub Pole5 kod obsługujący tę sytuację w kontrolce Pole6 może wyglądać następująco:

```
if @^InitSenderControlName@ =’Pole1′ or @^InitSenderControlName@ = 'Pole2′
select 'Zmieniono wartości w polach Pole1 lub Pole2′
if @^InitSenderControlName@ = 'Pole5′
select 'Zmieniono wartość w polu Pole5′
```

Najczęstszą sytuacją, w której wykorzystywany jest parametr ^SenderControlName lub ^InitSenderControlName jest własna akcja, która dodaje wartości do kontrolki typu Dane tabelaryczne.

Procedury

W Comarch DMS dostępne są procedury (wyzwalacze), które rozszerzają funkcjonalność modelowania procesów o możliwość wykonywania dodatkowych akcji uruchamiane poprzez określone zdarzenie występujące podczas pracy z aplikacją. Aktualnie Comarch DMS wyposażony jest w następujące wyzwalacze:

- **Baza Comarch DMS**:
  - **do.OnAfterDocumentCreated** – procedura uruchamiana po każdym utworzeniu dokumentu w Comarch DMS.
  - **do.OnAfterDocumentSave** – procedura uruchamiana po każdej aktualizacji dokumentu Comarch DMS:
    - po kliknięciu w ikonę zapisu,
    - w przypadku, gdy na typie obiegu ustawiono automatyczny zapis po zainicjowaniu dokumentu,

        - po kliknięciu ‘Utwórz’ w celu utworzenia dokumentu w masowym skanowaniu,
        - po kliknięciu w ‘Generuj’ w celu wygenerowania dokumentu w punktach ACD,
        - przy próbie zapisu dokumentu w obiegu utworzonego z poziomu kolumny typu Dokumenty w obiegu w kontrolce typu Dane tabelaryczne.
    - **do.OnBeforeDocumentShow** – procedura uruchamiana przed wyświetleniem szczegółów dokumentu Comarch DMS:
        - w przypadku utworzenia nowego dokumentu za pomocą ikony  ,
        - w przypadku wyświetlenia istniejącego dokumentu,
        - po kliknięciu ‘Utwórz’ w celu utworzenia dokumentu w masowym skanowaniu,
        - po kliknięciu w ‘Generuj’ w celu wygenerowania dokumentu w punktach ACD.
    - **do.OnDocumentPropagation** – procedura uruchamiana jest podczas każdego przekazania dokumentu do kolejnego etapu.
- **Bazy Comarch ERP XL, Comarch ERP Optima, Comarch ERP Altum** – podczas propagacji dokumentu Comarch DMS można wykonać procedurę na bazie systemu Comarch ERP:
    - **CDN.OnDocumentPropagation** – procedura uruchamiana jest podczas każdego przekazania dokumentu do kolejnego etapu.

Uwaga
Procedura **CDN.OnDocumentPropagation** na bazach systemów Comarch ERP jest wykonywana wyłącznie w instalacji jednospółkowej.
W każdej z procedur (wyzwalaczy) obsłużono wyjątki (Try-Catch) z możliwością przekazania informacji do aplikacji jako komunikat (MessageBox). Ponadto wyjątek zwrócony poprzez instrukcję THROW spowoduje cofnięcie transakcji. Możliwość tą można wykorzystać np. do zatrzymania przekazania dokumentu do

kolejnego etapu (procedura OnDocumentPropagation) w sytuacji, gdy nie został spełniony warunek przekazania. Kliknięcie „Przekaż do kolejnego etapu” może spowodować wyświetlenie komunikatu zdefiniowanego w procedurze i zatrzymanie akcji przekazania.

Budowa procedur (wyzwalaczy) rozszerzających funkcjonalność modelowania procesów (budowa procedury zostanie wyświetlona po kliknięciu w nazwę):

do.OnDocumentPropagation

Procedura jest wywoływana podczas przekazania dokumentu do następnego etapu.

```
CREATE PROCEDURE do.OnDocumentPropagation

@WorkflowId as int = NULL,

@StageId as int = NULL,

@DocumentId as int = NULL,

@PropagatedById as int = NULL,

@DocumentOwnerId as int = NULL,

@IsAutomaticPropagation as bit = NULL

AS

BEGIN

    declare @test int

    – Tutaj można obsłużyć dodatkowe akcje użytkownika.

END
```

Opis argumentów:

- **@WorkflowId** – int, id Obiegu dokumentów (DSH_Id),
- **@StageId** – int, id etapu (DDS_Id),
- **@DocumentId** – int, id dokumentu (DWD_ID),
- **@PropagatedById** – int, id użytkownika przekazującego dokument (DCD_Id),
- **@DocumentOwnerId** int, id właściciela dokumentu (DCD_Id) – przyjmuje zawsze wartość 0,
- **@IsAutomaticPropagation** bit, wartość prawdziwa, jeśli dokument był przekazany przez propagację czasową.

do.OnAfterDocumentSave

Procedura jest wykonywana podczas zapisu dokumentu.

```
CREATE PROCEDURE do.OnAfterDocumentSave

@DocumentId as int = NULL, – DWD_ID z tabeli DF_Work

@SavedBy as int = NULL – DCD_ID z tabeli DF_ConfOSDictionary

AS

BEGIN

    declare @test int;

    – Tutaj można obsłużyć dodatkowe akcje użytkownika.

END
```

Opis argumentów:

- **@DocumentId** – id dokumentu (DWD_ID),
- **@SavedBy** – id użytkownika zapisującego

dokument (DCD_Id).

do.OnAfterDocumentCreated

Procedura jest wykonywana przy zapisie nowego dokumentu.

```
CREATE PROCEDURE do.OnAfterDocumentCreated

@DocumentId as int = NULL, – DWD_ID z tabeli DF_Work

@SavedBy as int = NULL – DCD_ID z tabeli DF_ConfOSDictionary

AS

BEGIN

    declare @test int;

    – Tutaj można obsłużyć dodatkowe akcje użytkownika.

END
```

Opis argumentów:

- **@DocumentId** – id dokumentu (DWD_ID),
- **@SavedBy** – id użytkownika zapisującego dokument (DCD_Id).

do.OnBeforeDocumentShow

Procedura uruchamiana jest przed wyświetleniem szczegółów dokumentu DMS

```
CREATE PROCEDURE do.OnBeforeDocumentShow

@documentId as int = NULL, – DWD_ID z tabeli DF_Work
```

```
    @workflowId as int = NULL, – DFH_ID z tabeli
    DF_HeadDokumentFlow

    @userId as int = NULL – DCD_ID z tabeli
    DF_ConfOSDictionary

    AS

    BEGIN

        begin try

        declare @test int

        – Tutaj można obsłużyć dodatkowe akcje
        użytkownika.

        end try

        begin catch

            DECLARE @ErrorMessage varchar(max) = 'W
            procedurze do.OnBeforeDocumentShow wystąpił
            błąd: ' + ERROR_MESSAGE();

            –THROW 50001, @ErrorMessage ,1;

            declare    @ErrorSeverity    int    =
            ERROR_SEVERITY(), @ErrorState int =
            ERROR_STATE()

            raiserror (@ErrorMessage, @ErrorSeverity,
            @ErrorState);

        end catch

    END
```

Opis argumentów:

- **@DocumentId** – identyfikator dokumentu

(DWD_Id),
- **@WorkflowId** – identyfikator typu obiegu (DFH_Id),
- **@UserId** – identyfikator operatora (DCD_Id).

Przyklad
Przykład zastosowania procedury **do.OnAfterDocumentCreated**

W przykładzie zastosowano funkcję skalarną do.ModWorkflowName zwracającą nazwę typu obiegu. W procedurze do.OnAfterDocumentCreated dodano warunek, który ogranicza wywołanie procedury do nazwy typu obiegu. Oznacza to, że procedura zostanie uruchomiona każdorazowo, gdy w ramach wskazanego typu obiegu zostanie utworzony nowy dokument i zostanie podjęta próba jego zapisu w bazie. W przypadku, gdy kontrolka, której identyfikator wskazano w procedurze ma uzupełnioną wartość, dokument zostanie zapisany. W przypadku, gdy w kontrolce, której identyfikator określono w procedurze, nie zostanie uzupełniona wartość, dokument nie zostanie zapisany i zostanie wyświetlony komunikat o treści określonej w poleceniu THROW w procedurze.

```
USE [nazwa_bazy_DMS]

GO

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

ALTER PROCEDURE [do].[OnAfterDocumentCreated]

@DocumentId as int = NULL, – DWD_ID z tabeli DF_Work

@SavedBy as int = NULL – DCD_ID z tabeli DF_ConfOSDictionary
```

```
AS

BEGIN

    BEGIN TRY

    declare @string varchar(max)=”;

    if  do.ModWorkflowName  (@DocumentId)=’Nazwa_typu_obiegu’
    /*nazwa typu obiegu, w którym nastąpi blokada*/

        Begin

        select  @string=  isnull  (w.DWC_ValueString,”)  from
        do.DF_Work d

        join do.DF_ConfDSHead i on d.DWD_DSHId=i.DSH_ID

        join do.DF_ConfCFCardDokFlow k on k.DKO_DSHId=i.DSH_ID

        join do.DF_WorkCF w on w.DWC_DKOId = k.DKO_ID

        where           DWD_ID=@DocumentId           and
        k.DKO_Name=’Identyfikator’  /* identyfikator kontrolki,
        dla której wymagane jest uzupełnienie wartości*/

        if @string =”

        THROW 50001, 'Pole Dokument jest wymagane’ ,1; – treść
        wyświetlanego komunikatu

        End

    END TRY

BEGIN CATCH

DECLARE @ErrorMessage varchar(max) = ERROR_MESSAGE();

–THROW 50001, @ErrorMessage ,1;

declare @ErrorSeverity int = ERROR_SEVERITY(), @ErrorState int
= ERROR_STATE()
```

```
raiserror (@ErrorMessage, @ErrorSeverity, @ErrorState);

end catch

END
```