

Modelowanie procesów za pomocą C# – wprowadzenie

Informacje ogólne

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.

Od wersji 2023.1.0 za pomocą C# można tworzyć skrypty do pracy automatycznej (zob. [Konfiguracja automatycznego trybu pracy \(robotyzacja procesów\)](#)).

Comarch DMS został wyposażony w mechanizm wspomagający modelowanie procesów oparty o kompilator C#. Wersja kompilatora jak również wersja języka C# jest uzależniona od Microsoft .NET Framework. Zalecana jest aktualizacja Microsoft .NET Framework do najnowszej dostępnej wersji.

Wskazówka

Kod C# ma zastosowanie w Comarch DMS m. in. przy **weryfikacji kontrahenta na białej liście podatników** (zob. [Weryfikacja kontrahenta na białej liście podatników](#))

Środowisko kompilacji C#

Wynikowymi bibliotekami kompilatora Comarch DMS są zawsze biblioteki **dll**.

Po skompilowaniu, biblioteki są automatycznie konfigurowane do pracy na serwerze IIS. Dzięki temu funkcjonalność jest

dostępna dla wszystkich platform (aplikacji stacjonarnej, webowej, mobilnych iOS i Android).

Miejsca przechowywania bibliotek:

1. **{Apl. www (serwerowa)}\bin\.scriptcs_cache** – katalog do którego są kopiowane biblioteki po kompilacji kodu C#.
2. **{Apl. www(serwerowa)}\bin\Scriptcs_bin** – miejsce na biblioteki dll używane przez kod C# jako referencje.

Przed kompilacją skryptów C# na komputerze z IIS należy **nadać użytkownikowi IIS_IUSRS prawo zapisu do katalogu .scriptcs_cache**

Okno kompilacji języka C# jest dostępne z poziomu właściwości wybranych kontroltek. Kontrolki obsługują poniższe zdarzenia:

1. Init (zdarzenie OnInit)
2. Obserwator (zdarzenie OnChange)
3. Wciśnięcie przycisku (zdarzenie OnClick)

Jeżeli zaistnieje potrzeba skorzystania z biblioteki referencyjnej w której znajdują się funkcje do wywołania w kodzie C#, bibliotekę należy wgrać do katalogu **bin\Scriptcs_bin**. Nazwę tej biblioteki należy podać w oknie „Referencje” (zob. poniżej punkt *Język C# na definicjach kontroltek*).

Istnieje możliwość skorzystania z referencji bibliotek com (.NET) zarejestrowanych na serwerze IIS. W takim przypadku, w oknie „Referencje” należy wpisać nazwę biblioteki bez rozszerzenia dll.

Kompilacja kodu C# odbywa się po naciśnięciu ikony  lub kliknięciu przycisku „Kompiluj i zapisz”. Elementy te znajdują

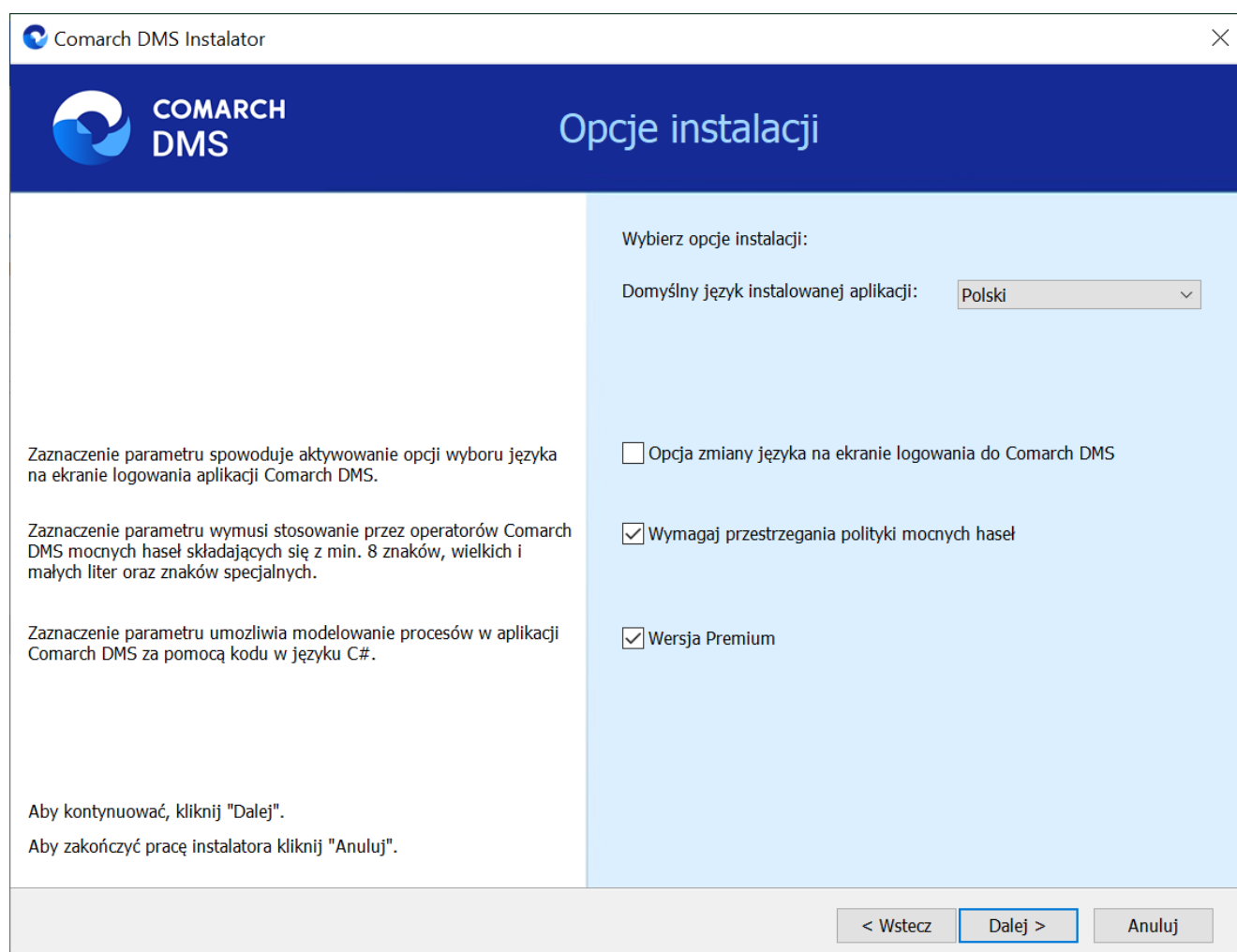
się w oknie edycji kodu C#. (zob. poniżej punkt *Język C# na definicjach kontrolek*)

Jeżeli skompilowane biblioteki dll zostaną usunięte z katalogu .scriptcs_cache, to przed pierwszym użyciem funkcjonalności (podniesienie dokumentu DMS) zamodelowanej metodą C# nastąpi automatyczne skompilowanie bibliotek.

Włączenie funkcjonalności

Aby **włączyć** funkcjonalność należy:

- przy instalacji lub aktualizacji systemu **zaznaczyć** parametr „**Wersja Premium**”



The screenshot shows the 'Comarch DMS Instalator' window with the title 'Opcje instalacji'. The window has a dark blue header with the Comarch DMS logo. The main area is divided into two columns. The left column contains three paragraphs of text explaining the effects of selecting certain options: 1. 'Zaznaczenie parametru spowoduje aktywowanie opcji wyboru języka na ekranie logowania aplikacji Comarch DMS.' 2. 'Zaznaczenie parametru wymusi stosowanie przez operatorów Comarch DMS mocnych haseł składających się z min. 8 znaków, wielkich i małych liter oraz znaków specjalnych.' 3. 'Zaznaczenie parametru umożliwi modelowanie procesów w aplikacji Comarch DMS za pomocą kodu w języku C#.' Below these paragraphs are instructions: 'Aby kontynuować, kliknij "Dalej".' and 'Aby zakończyć pracę instalatora kliknij "Anuluj".' The right column is titled 'Wybierz opcje instalacji:' and contains a dropdown menu for 'Domyślny język instalowanej aplikacji:' set to 'Polski'. Below this are three checkboxes: 'Opcja zmiany języka na ekranie logowania do Comarch DMS' (unchecked), 'Wymagaj przestrzegania polityki mocnych haseł' (checked), and 'Wersja Premium' (checked). At the bottom right are three buttons: '< Wstecz', 'Dalej >' (highlighted with a blue border), and 'Anuluj'.

Comarch DMS Instalator

COMARCH DMS

Opcje instalacji

Wybierz opcje instalacji:

Domyślny język instalowanej aplikacji: Polski

Zaznaczenie parametru spowoduje aktywowanie opcji wyboru języka na ekranie logowania aplikacji Comarch DMS.

Zaznaczenie parametru wymusi stosowanie przez operatorów Comarch DMS mocnych haseł składających się z min. 8 znaków, wielkich i małych liter oraz znaków specjalnych.

Zaznaczenie parametru umożliwi modelowanie procesów w aplikacji Comarch DMS za pomocą kodu w języku C#.

Aby kontynuować, kliknij "Dalej".

Aby zakończyć pracę instalatora kliknij "Anuluj".

☐ Opcja zmiany języka na ekranie logowania do Comarch DMS

☒ Wymagaj przestrzegania polityki mocnych haseł

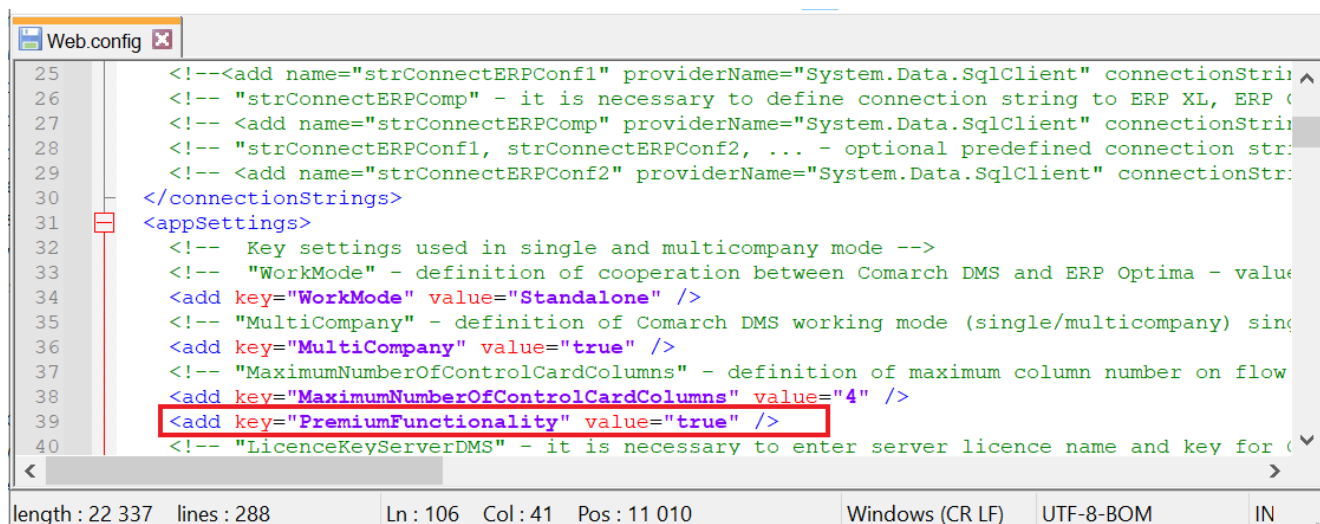
☒ Wersja Premium

< Wstecz Dalej > Anuluj

Wybór opcji „Wersja Premium” podczas instalacji Comarch DMS

lub

- w pliku **Web.config** wartość klucza **PremiumFunctionality** ustawić na „**True**” i wprowadzić odpowiedni klucz licencji.



```
25 <!--<add name="strConnectERPConf1" providerName="System.Data.SqlClient" connectionStrin
26 <!-- "strConnectERPComp" - it is necessary to define connection string to ERP XL, ERP (
27 <!-- <add name="strConnectERPComp" providerName="System.Data.SqlClient" connectionStrin
28 <!-- "strConnectERPConf1, strConnectERPConf2, ... - optional predefined connection str
29 <!-- <add name="strConnectERPConf2" providerName="System.Data.SqlClient" connectionStr
30 </connectionStrings>
31 <appSettings>
32 <!-- Key settings used in single and multicompany mode -->
33 <!-- "WorkMode" - definition of cooperation between Comarch DMS and ERP Optima - value
34 <add key="WorkMode" value="Standalone" />
35 <!-- "MultiCompany" - definition of Comarch DMS working mode (single/multicompany) sing
36 <add key="MultiCompany" value="true" />
37 <!-- "MaximumNumberOfControlCardColumns" - definition of maximum column number on flow
38 <add key="MaximumNumberOfControlCardColumns" value="4" />
39 <add key="PremiumFunctionality" value="true" />
40 <!-- "LicenceKeyServerDMS" - it is necessary to enter server licence name and key for (
```

Parametr „PremiumFunctionality”

Jeżeli operator **nie posiada odpowiedniej licencji**, a zaznaczył parametr „**PremiumFunctionality**”, wówczas **przy próbie logowania** w oknie logowania zostanie **wyświetlona na czerwono informacja** o braku licencji Premium, a dostęp do Comarch DMS **nie będzie możliwy**.

**COMARCH
DMS**

×

Spółka

Dom

Login

Administrator

Hasło

Brak wymaganej licencji Comarch DMS Premium.

Zmień język

Zmień hasło

☐ zapamiętaj logowanie



Okno logowania do aplikacji desktop z informacją o braku licencji Premium

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

Zasady pisania kodu C#

Kod C# dzięki któremu można programować inicjowanie wartości w poszczególnych kontrolkach programuje się w udostępnianych przez kontrolki edytorach na zasadzie skryptu. Można przyjąć, że obszar edytora to „wnętrze” metody obsługującej określone zdarzenie.

W zależności od edytora kontrolki (wybranego zdarzenia), administrator ma dostęp do listy właściwości, które można przywołać wpisując nazwę „**Global**”.

Obiekt „**Global**” skupia w sobie wszystkie inne obiekty, które można wykorzystać podczas pisania obsługi zdarzenia. Poprzez ten obiekt użytkownik ma również **dostęp do wartości kontrolek znajdujących się na karcie obiegu**.

W przypadku zdarzeń „**Inicjowanie**”, każda kontrolka posiada obiekt „**Common**” z następującymi właściwościami:

- **Globals.Common.DocumentCompanyId** – Id spółki powiązanej z kontrolką, np. DocumentCompanyId (typu int)
- **Globals.Common.DocumentId** – Id dokumentu DMS, np. DocumentId (typu int?)
- **Globals.Common.ParentDocumentId** – Id właściciela dokumentu, np. ParentDocumentId (typu int?)
- **Globals.Common.StageId** – Id etapu w którym znajduje się dokument, StageId (typu int?)
- **Globals.Common.UserId** – Id zalogowanego operatora, np. UserId (typu int)
- **Globals.Common.WorkflowId** – Id typu obiegu, np. WorkflowId (typu int)

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

W przypadku zdarzeń „**Obserwacja**”, każda z kontrolek posiada

dodatkowo właściwość:

- **Globals.Common**.SenderControlName – Identyfikator kontrolki inicjującej zdarzenie

Uwaga

Wszystkie właściwości udostępniane przez obiekt „**Common**” są tylko do odczytu.

Obiekt „**Global**” posiada również obiekt „**MainFrame**”.

Po nazwie „**MainFrame**” występują **nazwy identyfikatorów kontrolek, następnie właściwości lub metody**.

Elementy udostępniane przez obiekt „**MainFrame**” **zależą od kontrolki**, dla której pisze się obsługę zdarzenia, **zdarzeń** oraz od **innych zdefiniowanych na karcie obiegu kontrolek**.

W przypadku obiektu „**MainFrame**”, dostęp do właściwości odbywa się w oparciu o następującą ścieżkę:

Globals.MainFrame.Identyfikator kontrolki.Nazwa właściwości lub nazwa metody

Przykład

Przykładowa linia kontrolki o identyfikatorze „Pole1” pozwalająca na wprowadzenie lub zmianę wartości kontrolki za pomocą właściwości „Text”:

```
Globals.MainFrame.Pole1.Text = „NR/123/2020”;
```

Język C# na definicjach kontrolek

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”.

Tekst

Nazwa wyświetlana	Numer dokumentu
Nazwa (identyfikator)	Numer_dokumentu
Prezentuj na całej szerokości	☐
Ustaw kontrolkę na początku wiersza	☐
Max długość	900
Tekst wielolinijkowy	1  
Pokaż na liście	☐

Inicjowanie wartości

Inicjowanie	C# Script ▼ Brak SQLOD C# Script
Inicjuj przy każdym otwarciu dokumentu	

Zmiana wartości (Obserwator)

Kontrolki powiązane	Dodaj
Wzór na wartość	Wyrażenie ▼

[Ustaw](#)

Pozycja C# Script w opcjach inicjowania wartości kontrolki typu tekst

Własna akcja

Nazwa wyświetlana	Własna akcja
Nazwa (identyfikator)	Wlasna_akcja
Prezentuj na całej szerokości	☐
Ustaw kontrolkę na początku wiersza	☐
Plik wykonywalny	Ustaw
IIS	Ustaw
Procedura	Ustaw
Procedura ERP	Ustaw
C# Script	Ustaw
Nazwa spółki	
Potwierdzenie wykonania akcji	☐
Pytaj przed uruchomieniem	☐
Zapisz przed uruchomieniem	☐

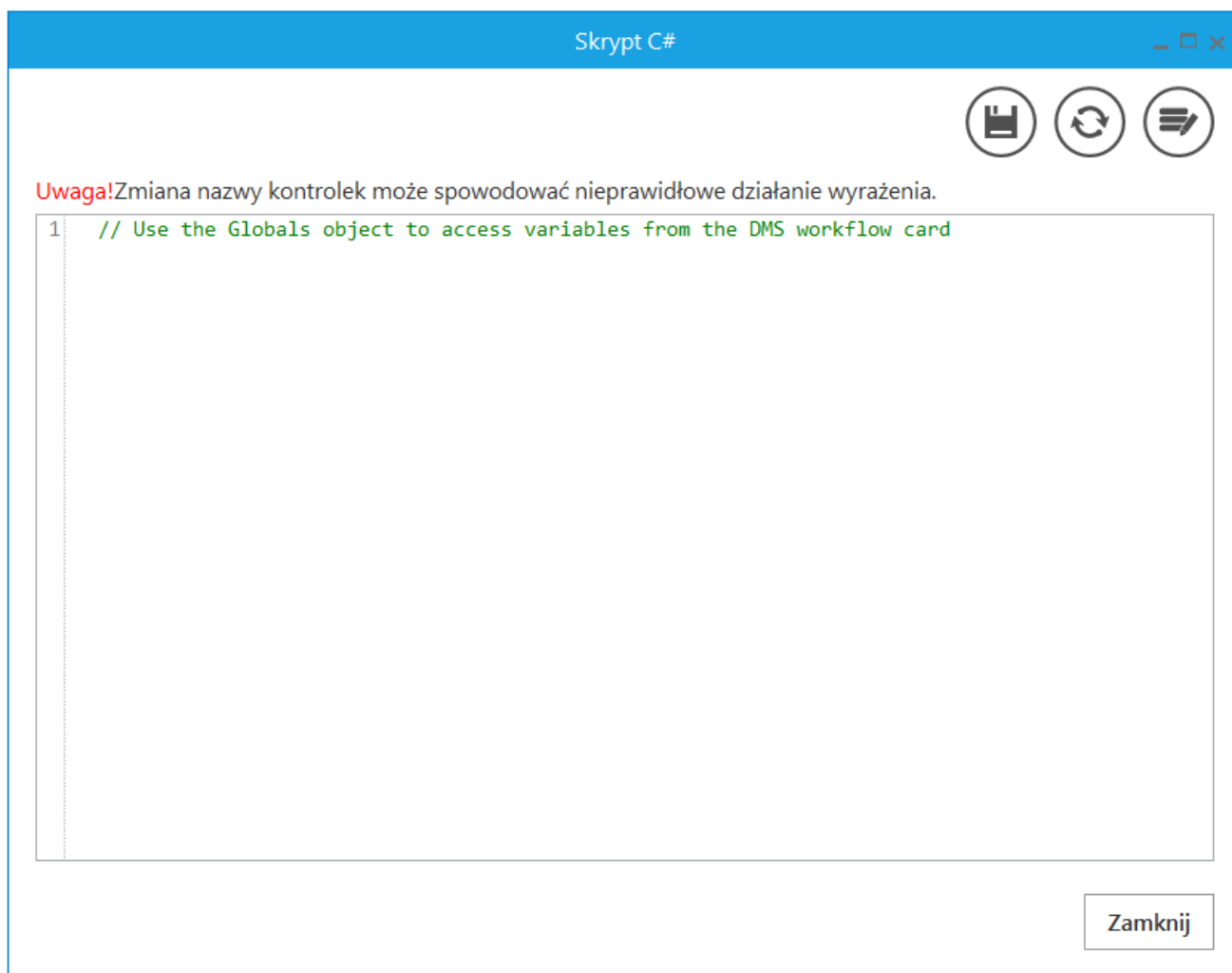
Pozycja C# Script w kontrolce typu Własna akcja

Po **wskazaniu** inicjowania wartości za pomocą **C# Script** i

kliknięciu w

[Ustaw](#)

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  **[Kompiluj i zapisz]**.

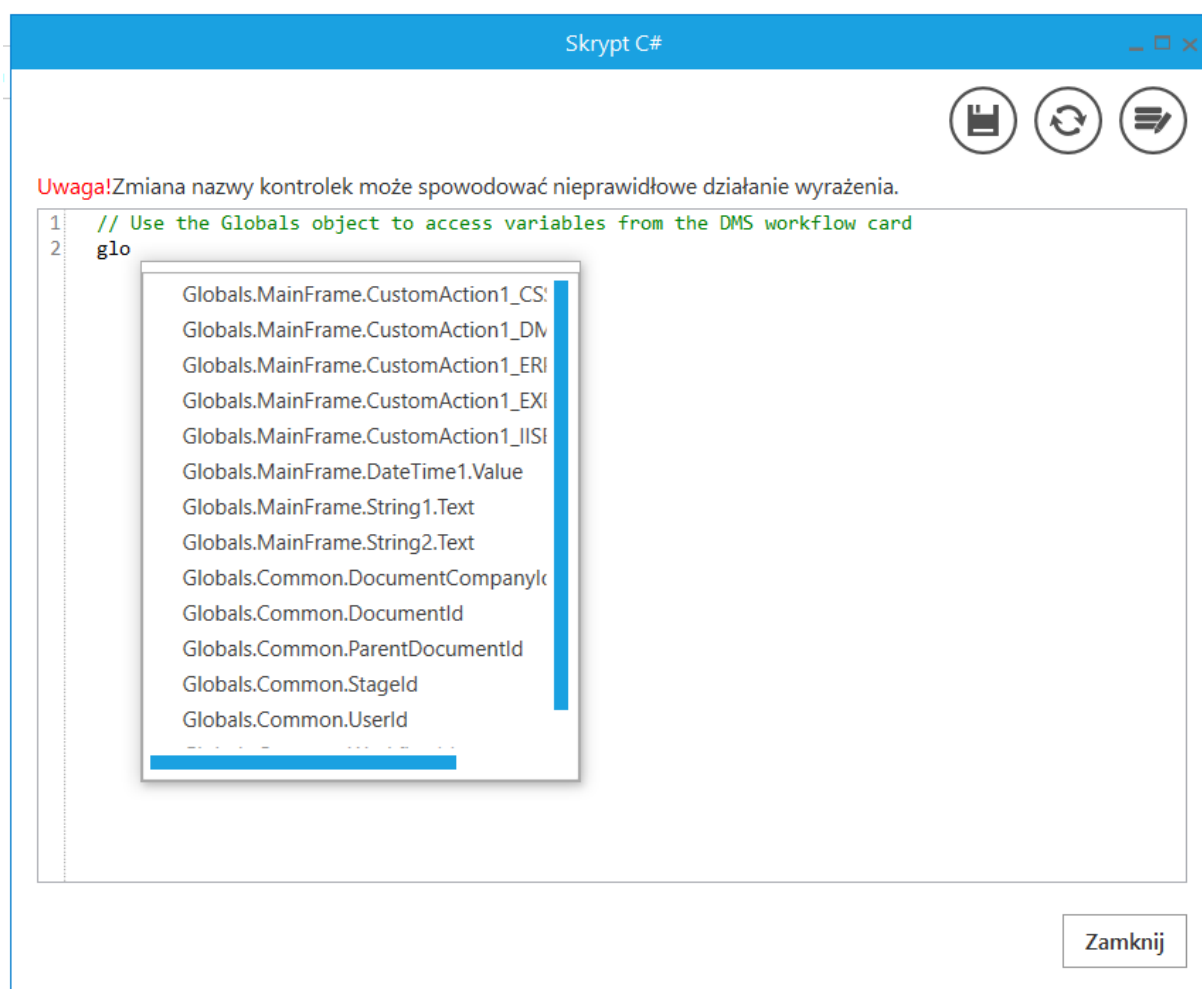
Kod po skompilowaniu **zapisywany jest w formie biblioteki** w lokalizacji **...\bin\.scriptsc_cache** w katalogu, w którym zainstalowano aplikację serwerową (dawna web) Comarch DMS.

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 serwerową (dawna web)Comarch DMS.

Warunkiem koniecznym do poprawnego działania jest **nadanie operatorowi IIS_USER uprawnień pełnej kontroli do katalogu .scriptscache**, pozwoli to na **dynamiczny zapis** skompilowanych plików.

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



Podpowiedzi wyświetlane w edytorze skryptów C#

W kontrolce typu 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 nazewnicznej języka C# **nie jest możliwe** używanie polskich znaków diakrytycznych oraz spacji w identyfikatorach kontroltek.

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

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

Przykład

Przepisanie wartości z kontrolki typu liczba całkowita o nazwie (identyfikatorze) *Liczba* do kontrolki typu tekst o nazwie (identyfikatorze) *Tekst*:

```
Globals.MainFrame.Tekst.Text  
= Globals.MainFrame.Liczba.Value.ToString();
```

Przykład

Pobranie wartości z kontrolki składnik 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;
```

Przykład

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);
}
```

Przykład

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

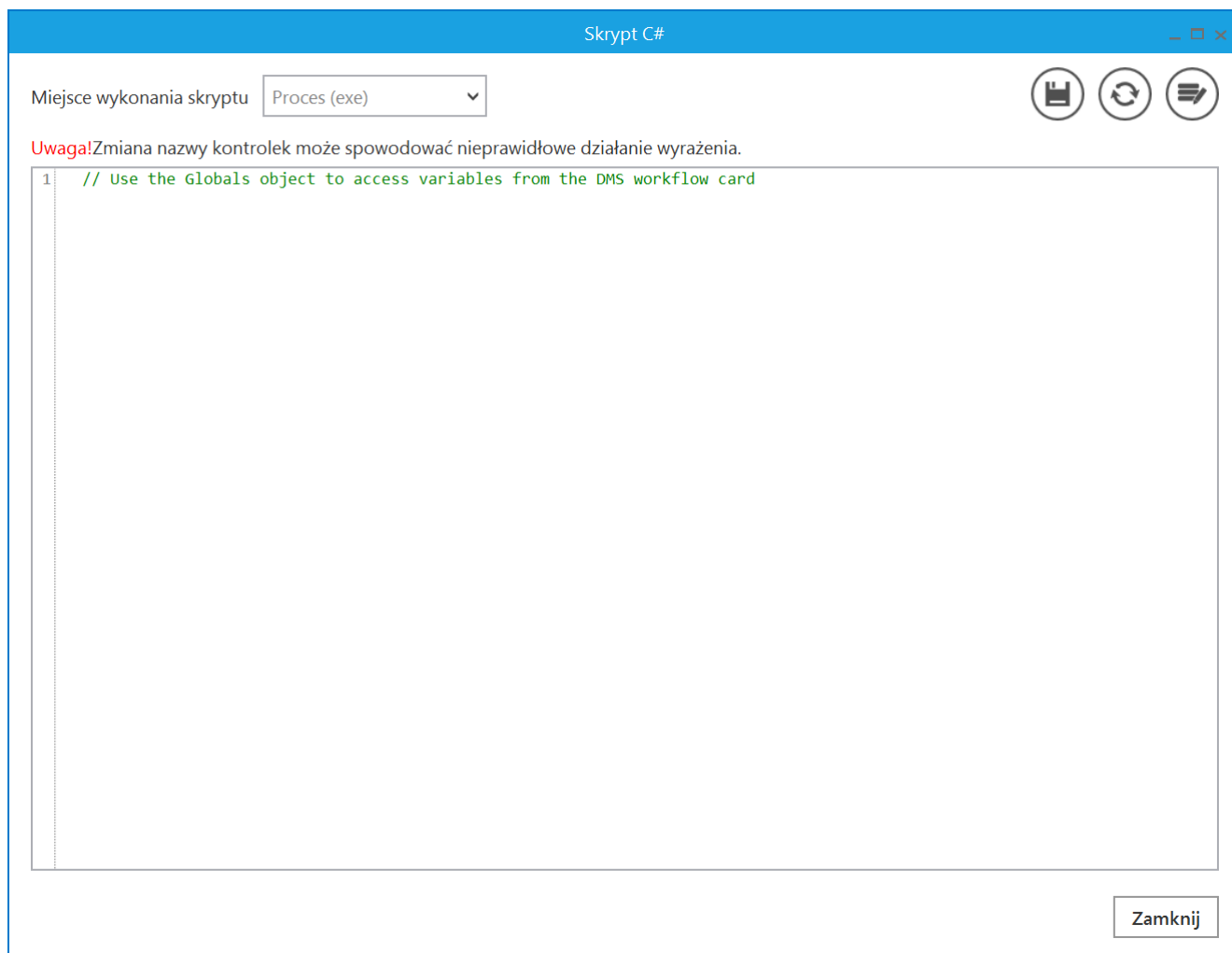
```
var info = „Identyfikator operatora:
” + Globals.Common.UserId;
Globals.MainFrame.Informacja.Text = info;
```

Od wersji 2024.3.0 zablokowano możliwość zmiany miejsca wykonywania skryptów C# – dostępna jest jedynie opcja Proces (exe).

Uwaga

W przypadku wcześniejszych wersji uruchomienie skryptu na niezależnym procesie exe jest zalecane w przypadku obsługi

API, które nie powinno działać na procesach wielowątkowych takich jak proces IIS.



Okno definiowania procedury wywołania kontrolki typu Własna akcja w języku C#

[Powrót do początku artykułu](#)