

Wikiprint Book

Title: Przewodnik wdrożeniowca > eDokumentyApi

Subject: Archiwum - eDokumenty - elektroniczny system obiegu dokumentów, workflow i CRM -
DeployerGuide/Others/eDokumentyApi

Version: 97

Date: 06/21/26 23:40:06

Table of Contents

Przewodnik wdrożeniowca > eDokumentyApi

3

[Przewodnik wdrożeniowca](#) > eDokumentyApi

System eDokumenty udostępnia API (Application Programming Interface) dzięki któremu jesteśmy w stanie przeprowadzać podstawowe czynności bez konieczności logowania do systemu.

Usługa jest zabezpieczona rozszerzeniem WSSecurity protokołu SOAP i wymaga podania w nagłówkach wywołania XML nazwy użytkownika i hasła.

Hasło i użytkownik to specjalne dane, które należy wprowadzić do pliku config.inc pod kluczami

```
define('EDOK_API_LOGIN', 'edok_api_user');
define('EDOK_API_PASSWORD', 'edok_api_pass');

// Dodatkowa stała która umożliwia pominięcie autentykacji (FALSE - nie sprawdza danych EDOK_API_LOGIN i EDOK_API_PASSWORD
// domyślnie ustawiona na TRUE, można wysłać GET lub dodatkowe nagłówki WSS
// define('EDOK_API_AUTH_MODE', TRUE);
```

wartości stałych w powyższym przykładzie są tylko danymi prezentacyjnymi i nie powinno się ich używać na produkcyjnej bazie.

Stałe te mogą mieć dowolne wartości ważne jednak aby te same wartości podać przy wywołaniu usługi SOAP w kliencie.

Usługa jest dostępna pod adresem:

<http://{host}:{port}/apps/edokumenty/classes/eDokumentyApi/EDokApiServer.php>

Opcjonalnie od wersji 0.9.8 usługa jest dostępna pod skróconym adresem:

<http://{host}:{port}/eDokumentyApi.php>

Wartość {host} oraz {port} należy zamienić odpowiednimi wartościami zgodnymi z konfiguracją serwera instalacyjnego systemu eDokumenty.

Dokumentacja API znajduje się pod adresem

<http://{host}:{port}/apps/edokumenty/classes/eDokumentyApi/DokumentacjaAPI.txt>

Podgląd dokumentacji API

```
<?php
// Dokumentacja Webservice - API systemu eDokumenty w wersji 1.1.0
Dokumentacja Webservice - API systemu eDokumenty w wersji 1.1.0

Usługa jest dostępna pod adresem:

http://{host}:{port}/apps/edokumenty/classes/eDokumentyApi/EDokApiServer.php

Opcjonalnie od wersji 0.9.8 usługa jest dostępna pod skróconym adresem:

http://{host}:{port}/eDokumentyApi.php

Usługa jest zabezpieczona rozszerzeniem WSSecurity protokołu SOAP i wymaga
podania w nagłówkach wywołania XML nazwy użytkownika i hasła.

/**
 * Tworzy nowy dokument
 *
 * @param data Array - Tablica z parametrami
 * @param data['dscrpt'] String - opis dokumentu
 * @param data['dctpid'] Int - typ dokumentu (types_of_documents:dctpid)
 * @param data['target'] Int - identyfikator stanowiska (organization_units:orunid)
 * @param data['prc_id'] Int - identyfikator sprawy (processes:prc_id)
 * @param data['state_'] Int - rodzaj dokumenty (1-wychodzący, 2-przychodzący, 3-wewnętrzny)
```

```

* @param data['from_contact_symbol'] String - symbol nadawcy (contacts:symbol)
* @param data['to_contact_symbol'] String - symbol odbiorcy (contacts:symbol)
*
* @return Int - id dokumentu jeśli sukces (documents:doc_id), 0 w razie niepowodzenia
*
* @throws Exception - SoapFault
*/
Int createDocument(Array data)

/**
* Dodaje pracownika do uprawnionych w dokumencie
*
* @param doc_id Int - id dokumentu (documents:doc_id)
* @param usr_id Int - id pracownika (users:usr_id)
* @param rights String - poziom uprawnień (r - odczyt, w - zapis, m - zarządzanie)
* @param prionl Bool - udostępni tylko uprawnionym użytkownikom
*
* @return Int - 1 jeśli sukces , 0 w razie niepowodzenia
*
* @throws Exception - SoapFault
*/
Int grantUserAccessToDocument(Int doc_id, Int usr_id, String rights, Bool prionl)

/**
* Dodaje grupę do uprawnionych w dokumencie
*
* @param doc_id Int - id dokumentu (documents:doc_id)
* @param grp_id Int - id grupy (groups:grp_id)
* @param rights String - poziom uprawnień (r - odczyt, w - zapis, m - zarządzanie)
* @param prionl Bool - udostępni tylko uprawnionym użytkownikom
*
* @return Int - 1 jeśli sukces , 0 w razie niepowodzenia
*
* @throws Exception - SoapFault
*/
Int grantGroupAccessToDocument(Int doc_id, Int grp_id, String rights, Bool prionl)

/**
* Dodaje załącznik do dokumentu
*
* @param fileContent String - Tablica z parametrami
* @param fileName String - nazwa pliku
* @param documentId Int - identyfikator dokumentu (documents:doc_id)
* @param contentTransferEncoding String - kodowanie przesyłanej treści pliku (tylko base64)
*
* @return Int - id pliku jeśli sukces, 0 w razie niepowodzenia
*
* @throws Exception - SoapFault
*/
Int addAttachmentToDocument(String fileContent, String fileName, Int documentId)

/**
* Ustawia meta text załącznika (używany przy wyszukiwaniu)
*
* @param int $fileid ID załącznika
* @param string $text
*
* @return bool true on success, false otherwise
*
* @since 1.0.1

```

```

* @throws Exception - SoapFault
*/
bool function setFileMetaText($fileid, $text)

/**
 * Tworzy nowy dokument typu raport dobowy z kasy
 * W przypadku jeśli dany dokument już istnieje następuje
 * jego aktualizacja.
 * Warunkiem sprawdzenia są 2 parametry acorid oraz datedr.
 *
 * @param data Array - Tablica z parametrami
 * @param data['datedr'] Date - data raportu w formacie YYYY-MM-DD (np 2010-01-01)
 * @param data['netto_'] Numeric(12, 2) - wartość netto obrotów
 * @param data['vatval'] Numeric(12, 2) - wartość vat
 * @param data['brutto'] Numeric(12, 2) - wartość brutto obrotów
 * @param data['crdpay'] Numeric(12, 2) - wartość jaka została zapłacona kartami
 * @param data['clncnt'] Int - ilość klientów
 * @param data['acorid'] Int - jednostka rozliczeniowa (organization_units:orunid) jeśli isclun = TRUE
 *
 * @return Int - id dokumentu jeśli sukces (documents:doc_id), 0 w razie niepowodzenia
 *
 * @throws Exception - SoapFault
*/
Int createFKDayReport(Array data)

/**
 * Tworzy nową sprawę
 *
 * @param data Array - Tablica z parametrami
 * @param data['dscrpt'] String - opis sprawy
 * @param data['briefcase_symbol'] String - symbol teczki
 * @param data['dsexid'] int - identyfikator teczki (doss_extract_list:dsexid)
 * @param data['orunid'] int - identyfikator jednostki organizacyjnej, pod którą zostanie utworzona sprawa (organization_u
 * @param data['rspoid'] int - identyfikator osoby odpowiedzialnej w sprawie (organization_units:orunid)
 * @param data['prtpid'] int - identyfikator procedury (procedures_def:prtpid)
 * @param data['comnts'] String - uwagi
 * @param data['contact_symbol'] String - symbol kontaktu (contacts:symbol)
 *
 * @return Int - id sprawy jeśli sukces (processes:prc_id), 0 w razie niepowodzenia
 *
 * @throws Exception - SoapFault
*/
Int createProcess(Array data)

/**
 * Zwraca dane sprawy
 *
 * @param prc_id Int - Id sprawy (processes:prc_id)
 *
 * @return Array - tablica danych sprawy jeśli sukces, SoapFault w razie niepowodzenia
 *
 * @throws Exception - SoapFault
*/
Array getProcess(Int prc_id)

/**
 * Tworzy nowy kontakt

```

```

*
* @param data Array - Tablica z parametrami
* @param data['name_1'] String - nazwa kontaktu
* @param data['name_2'] String - skrót kontaktu
* @param data['nip___'] Int - numer NIP kontaktu
* @param data['symbol'] String - symbol kontrahenta
* @param data['notes_'] String - uwagi
* @param data['ph_num'] String - numery telefonów
* @param data['faxnum'] String - numery faxów
* @param data['email_'] String - adresy email oddzielone przecinkiem
* @param data['websit'] String - strona www kontaktu
* @param data['regon_'] String - numer REGON
* @param data['pesel_'] String - numer PESEL
* @param data['bldnum'] String - numer domu
* @param data['fltnum'] String - numer mieszkania
* @param data['distkm'] Numeric(12, 2) - dystans
* @param data['powiat'] String - nazwa powiatu
* @param data['gmina_'] String - nazwa gminy
* @param data['woj___'] String - nazwa województwa ze słownika
* @param data['post__'] String - poczta
* @param data['countr'] String - kraj
* @param data['city__'] String - miasto
* @param data['code__'] String - kod_pocztowy
* @param data['street'] String - nazwa ulicy
*
* @return Int - id kontaktu jeśli sukces (contacts:contid), 0 w razie niepowodzenia
*
* @throws Exception - SoapFault
*/
Int createContact(Array data)

/**
* Zwraca dane kontaktu wraz z adresem
*
* @param contid Int - Id kontaktu (contacts:contid)
*
* @return Array - tablica danych kontaktu jeśli sukces, SoapFault w razie niepowodzenia
*
* @throws Exception - SoapFault
*/
Array getContact(Int contid)

/**
* Szuka w bazie kontaktu/ów o podanych parametrach
*
* @param data Array - Tablica z parametrami w formie klucz => wartość, gdzie klucz to nazwa pola z bazy z tabeli contacts
* natomiast wartość to ciąg tekstowy, który szukamy i który ma się znaleźć w kluczu
*
* @return Array - tablica identyfikatorów kontaktów - tablica może być pusta jeśli nie znalazł, SoapFault w razie niepowo
*
* @throws Exception - SoapFault
*/
Array searchContacts(Array data)

/**
* Tworzy nowe zdarzenie
*
* @param data Array - Tablica z parametrami

```

```

* @param data['dscrpt'] String - opis zdarzenia
* @param data['trmtyt'] String - typ zdarzenia (TODO - zadanie, PHONECALL - rozmowa tel., MEETING - spotkanie)
* @param data['type__'] String - typ rozmowy tel. jeśli trmtyt = PHONECALL (IN - przychodząca, OUT - wychodząca, INT - wewnętrzna)
* @param data['start_'] String - data rozpoczęcia w formacie YYYY-MM-DD HH:MM:SS np 2010-01-01 08:00:00
* @param data['duratn'] String - czas trwania np 2h30m - co oznacza 2 godziny 30 minut
* @param data['usr_id'] Mixed - identyfikator pracownika (może być tablica identyfikatorów), któremu zleca się wykonanie
* @param data['contid'] Int - identyfikator kontaktu (contacts:contid) pod którym zostanie utworzone zdarzenie
* @param data['prc_id'] Int - identyfikator sprawy (processes:prc_id) do której zostanie dołączone zdarzenie
* @param data['adduid'] Int - identyfikator pracownika (users:usr_id) który zleca zdarzenie, musi należeć do jednostki i
*
* @return Int - id event jeśli sukces (events:evntid), 0 w razie niepowodzenia
*
* @throws Exception - SoapFault
*/
Int createEvent(Array data)

/**
* Zwraca dane zdarzenia
*
* @param evntid Int - Id zdarzenia (events:evntid)
*
* @return Array - tablica danych zdarzenia jeśli sukces, SoapFault w razie niepowodzenia
*
* @throws Exception - SoapFault
*/
Array getEvent(Int evntid)

/**
* Zwraca listę zdarzeń według zadanych parametrów
*
* @param data Array - Tablica z parametrami
* @param from__ Date - data w formacie YYYY-MM-DD (np 2010-01-01) z jakiego dnia ma pobrać zdarzenia
* @param to____ Date - data w formacie YYYY-MM-DD (np 2010-01-10) do jakiego dnia ma pobrać zdarzenia, jeśli brak parametru
* @param usr_id Int - Id pracownika (users:usr_id)
* @param contid Int - Id kontaktu (contacts:contid)
*
* @return Array - tablica zdarzeń (może być pusta jeśli nie ma takich, które spełniają kryterium wyszukiwania)
* jeśli sukces, SoapFault w razie niepowodzenia
*
* @throws Exception - SoapFault
*/
Array getEvents(Array data)

/**
* Zwraca dane etapów danej procedury.
* Identyfikator procedury można pobrać wykonując $client->getProcess($prc_id) i w zwróconej
* tablicy będzie kolumna procid.
*
* @param Int procid - Id procedury (procedures:procid)
*
* @return Array - tablica z danymi etapów jeśli pusta tzn, że procedura nie ma etapów
*
* @throws Exception - SoapFault
*/
Int getProceduresStages(Int procid)

/**

```

```

* Wykonuje dany etap procedury oraz aktywuje następny
*
* @param Int procid - Id procedury (procedures:procid)
* @param Int ptstid - Id etapu z wskazanej procedury jakie zostanie wykonany jako załatwiony
* @param Int next_ptstid - Id etapu z tabeli stages_def
*         ("ptstid" jest etapem decyzyjnym to "next_ptstid" będzie wybrane
*         jako następny etap jeżeli jest następnikiem etapu "ptstid")
*
* @return Boolean - TRUE w przypadku powodzenia w każdym innym jest zwracany wyjątek SoapFault
*
* @throws Exception - SoapFault
*/
Int completeStage(Int procid, Int ptstid, Int next_ptstid = NULL)

/**
* Zwraca dane dotyczące typu dokumentu na podstawie jednego z poniższych parametrów
*
* @param data Array - Tablica z parametrami
* @param dctpid Int - ID typu dokumentu
* @param dctptp String -
*
* @return Array
*
* @throws Exception - SoapFault
*/
Array getDocumentTypeData(Array data)

/**
* Zwraca dane dotyczące jednostki organizacyjnej na podstawie jednego z poniższych parametrów
*
* @param data Array - Tablica z parametrami
* @param orunid Int - ID jednostki organizacyjnej
* @param usr_id Int - ID jednostki organizacyjnej
* @param usrnam String - Nazwa użytkownika
* @param fullnm String - Imię i Nazwisko użytkownika
*
* @return Array
*
* @throws Exception - SoapFault
*/
Array getOrganizationUnit(Array data)

/**
* Ustawienie cechy
*
* @param featid Int - Identyfikator cechy (features.featid)
* @param tblnam String - Identyfikator tekstowy obiektu do którego jest przypisana cecha
*         (contacts - Kontakty, types_of_documents_view - Dokumenty, processes - Sprawy, events - Zdarzenia,
*         users - Pracownicy, devices - Urządzenia, depository - Produkty, resources - Zasoby)
* @param tbl_id Int - Identyfikator obiektu, klucz główny wskazany w tblnam, gdzie tblnam to nazwa tabeli czyli dla contacts
*         wyjątkiem są dokumenty tam tbl_id wskazuje na doc_id z tabeli documents
* @param value Mixed - Wartość cechy. W przypadku cech wyboru definiowanych przez użytkownika (oznaczonych atrybutem multiple)
*         Dodatkowo aby usunąć wartość z cechy value === NULL (nie pusty string a NULL)
*
* @return int - 1 sukces, 0 w razie niepowodzenia
*
* @throws Exception - SoapFault
*/

```

```
Int setFeatureValue(Int featid, String tblnam, Int tbl_id, Mixed value)
```

```
?>
```

Przykład wywołania API z poziomu PHP w systemie eDokumenty

Hasło powinno być zakodowane przy użyciu funkcji md5

```
// Plik MyService.php umieszczony w apps/edokumenty.
// MyService.php
<?php
// stałe są w config.inc i są to dane potrzebne do autentykacji usługi
require_once(' ../../framework/lib/util/Translator/Translator.inc');
require_once('./config.inc');

// klient SOAP systemu eDokumenty wysyłający nagłówki WSSecurity zmienić na
// odpowiednią ścieżkę w zależności od lokalizacji pliku MyService.php
require_once('./classes/eDokumentyApi/EDokApiClient.inc');

// wartość {host}:{port} zmienić na odpowiednią dla serwera
$ops = array(
    'location' => 'http://{host}:{port}eDokumentyApi.php',
    'uri' => "eDokumentyAPI",
    'encoding'=>'UTF-8'
);

// Dodatkowo od wersji 3.3 autentykacja nie wymaga nadpisania klienta Soap. Wystarczy do url z adresem serwisu dodać param
// a1 - login,
// a2 - hasło,
// a3 - symbol jednostki
/*
$ops = array(
    'location' => 'http://{host}:{port}eDokumentyApi.php?a1='.EDOK_API_LOGIN.'&a2='.md5(md5(EDOK_API_PASSWORD)).'_SOAP_eDok
    'uri' => "eDokumentyAPI",
    'encoding'=>'UTF-8'
);

// bezpośrednie wywołanie klienta Soap
$client = new SoapClient(NULL, $ops);
*/
$client = new EDokApiClient(null, $ops);
$client->setUser(EDOK_API_LOGIN); // ten sam co w config.inc
$client->setPass(md5(EDOK_API_PASSWORD)); // to samo co w config.inc

$header = new SoapHeader('eDokumentyAPI', 'entity_symbol', DEFAULT_ENTITY_SYMBOL);
$client->__setSoapHeaders($header);

// Tworzy kontakt
$contid = NULL;

try {
    $data = array(
        'name_1' => 'SOAP TEST'.date('d H:m:s'),
        'name_2' => 'SOAPTEST',
        'nip__' => 1111111111,
        'street' => 'Główna',
        'symbol' => 'FGH99'
    );
};
```

```

    $contid = $client->createContact($data);
    var_dump($contid);
} catch(SoapFault $fault) {

    var_dump($fault);

    if ($fault->faultcode < 100) {
        trigger_error("SOAP Fault: (faultcode: {$fault->faultcode}, faultstring: {$fault->faultstring})", E_USER_ERROR);
    }
}

// Tworzy sprawę
$prc_id = NULL;

try {
    $data = array(
        'dscrip' => 'SOAP TEST '.date('d H:m:s'),
        'briefcase_symbol' => 'DK.023',
        'orunid' => 49,
        'rspoid' => 54,
        'fxtrid' => 60,
        'prtpid' => 1,
    );
    $prc_id = $client->createProcess($data);
    var_dump($prc_id);
} catch(SoapFault $fault) {

    var_dump($fault);

    if ($fault->faultcode < 100) {
        trigger_error("SOAP Fault: (faultcode: {$fault->faultcode}, faultstring: {$fault->faultstring})", E_USER_ERROR);
    }
}

// tworzy dokument w sprawie
if ($prc_id) {

    $doc_id = NULL;

    try {
        $data = array(
            'dscrip' => 'SOAP TEST '.date('d H:m:s'),
            'prc_id' => $prc_id,
            'dctpid' => 1,
            'target' => 54,
            'from_contact_symbol' => '12345',
            'to_contact_symbol' => '54321',
        );
        $doc_id = $client->createDocument($data);
        var_dump($doc_id);
    } catch(SoapFault $fault) {

        var_dump($fault);

        if ($fault->faultcode < 100) {
            trigger_error("SOAP Fault: (faultcode: {$fault->faultcode}, faultstring: {$fault->faultstring})", E_USER_ERROR);
        }
    }
}

```

```

// dodaje załącznik do utworzonego dokumentu
// o treści Testowy plik i nazwie dokument.txt
if ($doc_id) {
    try {
        $filePath = './test.pdf';
        $content = base64_encode(file_get_contents(filePath));

        $dd = $client->addAttachmentToDocument($content, basename(filePath), $doc_id);
        var_dump($dd);

    } catch(SoapFault $fault) {

        var_dump($fault);

        if ($fault->faultcode < 100) {
            trigger_error("SOAP Fault: (faultcode: {$fault->faultcode}, faultstring: {$fault->faultstring})", E_USER_ERROR);
        }
    }
}

// zwraca dane (tablica) o sprawie wszystko z tabeli processes gdzie prc_id = $prc_id
try {

    $data = $client->getProcess($prc_id);
    var_dump($data);

} catch(SoapFault $fault) {

    var_dump($fault);

    if ($fault->faultcode < 100) {
        trigger_error("SOAP Fault: (faultcode: {$fault->faultcode}, faultstring: {$fault->faultstring})", E_USER_ERROR);
    }
}

?>

```

W przypadku jeśli chcemy utworzyć własnego klienta SOAP dla innego języka programowania np. JAVA należy skontaktować się z działem oprogramowania celem wyszczególnienia specyfikacji takiego klienta.