

Title: Pobranie definicji rejestru

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

Version: 7

Date: 08/24/24 19:18:18

Table of Contents

Pobranie definicji rejestru

3

Pobranie definicji rejestru

Definicja parametrów:

```
<?php

/**
 * Wyszukuje wpisy w podanym rejestrze
 *
 * @param integer Id rejestru
 * @param string fields pola z rejestru (jedno lub więcej, wymienione po przecinku), których wartości mają zostać zwrócone
 * @param integer limit maksymalna ilość zwracanych wpisów w rejestrze (0 = bez limitu)
 * @param integer offset od którego rekordu mają zostać zwrócone dane (działa jeżeli określono limit większy od 0)
 * @param string json params dane do wyszukiwania typu klucz:wartość (np. {"imie":"jan","nazwisko":"nowak"})
 *
 * @access public
 * @return string json
 */
public function searchRegisterEntries($register_id, $fields, $limit, $offset, $params)

?>
```

Przykłady wywołań:

```
// Plik MyService.php umieszczony w apps/edokumenty.
// MyService.php
<?php

define('EDOK_API_LOGIN', 'developer');
define('EDOK_API_PASSWORD', 'developer');
define('DEFAULT_ENTITY_SYMBOL', 'demo');

require_once('./classes/eDokumentyApi/EDokApiClient.inc');

$options = array(
    'location' => 'http://{host}:{port}/eDokumentyApi.php',
    'uri' => "eDokumentyAPI",
    'encoding' => 'UTF-8'
);

$client = new EDokApiClient(NULL, $options);
$client->setUser(EDOK_API_LOGIN);
$client->setPass(md5(EDOK_API_PASSWORD));
$header = new SoapHeader('eDokumentyAPI', 'entity_symbol', DEFAULT_ENTITY_SYMBOL);
$client->__setSoapHeaders($header);

try {
    $data = array(
        'name__' => 'oskar_reg_2',
    );
    $reg_data = $client->getRegister($data);
    var_dump($reg_data);
} catch(SoapFault $fault) {

    var_dump($fault);

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

```

    }
}

try {
    $data = array(
        'name__' => 'oskar_reg_2',
    );
    $max_date = $client->getMaxDateForRegister($data);
    var_dump($max_date);
} catch(SoapFault $fault) {

    var_dump($fault);

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

try {
    $data = array();
    for ($i = 100; $i--;) {
        $data[] = array(
            'dscrypt' => 'r '.date('d H:m:s').' '.$i,
            'reg_id' => $reg_data['id____'],
            'assdasd' => $i,
            'e3e3e' => '12345',
            'r4r4r4' => md5($i.'aaaa'),
        );
    }
    // może być json
    // $data = json_encode($data);
    $result = $client->addRegisterEntries($reg_data['id____'], $data);
    var_dump($result);
} catch(SoapFault $fault) {

    var_dump($fault);

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

try {
    $data = array(
        'name__' => 'oskar_reg_2',
    );
    $max_date = $client->getMaxDateForRegister($data);
    var_dump($max_date);
} catch(SoapFault $fault) {

    var_dump($fault);

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

```

Powyższy kod udostępniony jest na licencji LGPL <http://www.gnu.org/licenses/lgpl-3.0.txt>