WebWeaver®

DigiOnline GmbH

WebWeaver API

Stand: 04.01.2018

Diese Beschreibung stellt den aktuellen Stand verschiedener interner Strukturen zusammen, die fuer die Benutzung der API von WebWeaver Suite benoetigt werden. Sie erhebt weder Anspruch auf Vollstaendigkeit noch auf Richtigkeit. Details koennen sich ohne Vorankuendigung aendern.

Einleitung

Ueberblick

SOAP vs. JSON-RPC

Die API ist ueber SOAP und JSON-RPC ansprechbar. Funktionsweise und -umfang sind identisch. Die Wahl des Uebertragungsweges sollte allein anhand des Clients getroffen werden.

Funktionsweise

Es wird immer eine Kette von Befehlen (Batch) in einem API-Call geschickt.

Sitzung

Normalerweise sollte eine typische Sitzung wir folgt ablaufen:

Erster Aufruf:	login()
Weitere Aufrufe:	set_session()
	set_focus()
	(...)
Letzter Aufruf:	set_session()
	logout()

Einzelne Requests

Wird in Ausnahmefaellen keine Session benoetigt, kann der Parameter "is_volatile" gesetzt werden. Ein explizierter Logout ist dann nicht notwendig.

Online / Quickmessages

Bei einem normalen Login wird der Nutzer auf der Plattform als "online" angezeigt. Es wird erwartet, dass er Quickmessages zugestellt bekommt. Andernfalls muss beim Login der Parameter "is_online" auf 0 gesetzt werden.

Beispiele

Alle SOAP-Beispiel-Aufrufe sind PHP-Code. Diese setzen eine aktuelle PHP-Version mit aktiviertem SOAP-Support voraus. Diese Beispiele sollten sich aber relativ problemlos in jeder anderen (SOAP-faehigen) Sprache nachbauen lassen. JSON-RPC-Beispiel-Aufrufe sind fertige JSON-Objekte.

Return Codes und Fehlermeldungen

Bei falsch formatierten Befehlen wird eine SOAP-Exception bzw. ein JSON-Fehler generiert. Ansonsten wird in der Antwort zu jedem Befehl ein Wert "return" mitgegeben:

OK	Ausfuehrung OK und abgeschlossen
RESUME	Ausfuehrung nur teilweise: Beim naechsten Aufruf weiter
RETRY	Ausfuehrung uebersprungen: Beim naechsten Aufruf neu
ERROR	Fehler in diesem Befehl, weiter mit naechstem Befehl
FATAL	Fehler und Abbruch weiterer Befehle

Bei den Return-Codes "ERROR" und "FATAL" werden zusaetzlich die Werte "error" mit einer kurzen Fehlerbeschreibung und "errno" mit einer eindeutigen Fehlernummer zurueckgeliefert. "RESUME" liefert den "offset" fuer den naechsten Aufruf.

Schnittstellen-Hilfe

URL (HTML, lesbar)

https://HOST/wws/559185.php
(Beispiel: https://www.webweaver.de/wws/559185.php)

URL (XML)

https://HOST/soap.php?help
(Beispiel: https://www.webweaver.de/soap.php?help)

Beschreibung

Liefert die aktuelle Liste aller Objekte, Methoden und Parameter.

SOAP

Funktionsweise

Alle Aufrufe werden ueber die Methode "request()" getaetigt. Diese Methode erwartet als Parameter ein mehrdimensionales Array mit den nacheinander auszufuehrenden Befehlen. Die Antwort ist wieder ein mehrdimensionales Array mit einem Sub-Array pro Befehl und einem einem zusaetzlichen Sub-Array, welches u.a. die Session-ID enthaelt, die beim naechsten Aufruf zu verwenden ist.
Es sollten moeglichst viele Befehle in einem Aufruf gebuendelt werden.

Eingabe

Normalerweise wird beim Aufruf ein assoziatives Array uebergeben. Sollte Ihre Umgebung diese Art von Arrays nicht unterstuetzen, so kann jedes Set [Parameter=>Value] durch ein Sub-Array [Parameter,Value] ersetzt werden.

Ausgabe

Die Ausgabe erfolgt als assoziatives Array. Falls es beim Entgegennehmen zu Problemen kommt, koennen Sie das Array serialisiert als Zeichenkette zurueckliefern lassen. set_options(['output'=>'serialized'])

Endpoint

https://HOST/soap.php?wsdl
(Beispiel: https://www.webweaver.de/soap.php?wsdl)

JSON-RPC

Spezifikation

http://www.jsonrpc.org/specification

Endpoint

https://HOST/jsonrpc.php
(Beispiel: https://www.webweaver.de/jsonrpc.php)

Beispiele

Basis-Script

SOAP / PHP

```
<?
$client=new SoapClient('https://www.webweaver.de/soap.php?wsdl',
  array('encoding'=>'ISO-8859-1','trace'=>1,
  'features'=>SOAP_SINGLE_ELEMENT_ARRAYS,'exceptions'=>0));
$result=$client->request(); // Aufruf
print_r($result);
?>
```

JSON-RPC (jQuery)

```
var url='https://www.webweaver.de/jsonrpc.php';
var request = [{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "get_information",
  "params": {}
}];
$.post(url,JSON.stringify(request),function(response) {
  alert(JSON.stringify(response));
},'json').fail(function(evt) {
alert('finished\n'+JSON.stringify(evt));
});
```

Beschreibung

Dieser Block wird fuer alle Aufrufe verwendet, der eigentlich Aufruf variiert von Beispiel zu Beispiel.

Login

SOAP / PHP

```
$result=$client->request(array(
  array('method'=>'login','login'=>'test@webweaver.de','password'=>'pw_test'),
));
```

SOAP / PHP alternativ

```
$result=$client->request(array(
  array(array('method','login'),array('login','test@webweaver.de'),
       array('password','pw_test')),
));
```

JSON-RPC

```
[{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "login",
  "params": {
    "login": "test@webweaver.de",
    "password": " pw_test"
  }
},{
  "jsonrpc": "2.0",
  "id": 2,
  "method": "get_information",
  "params": {}
}]
```

Beschreibung

Fuehrt den Login auf der Plattform durch. Das Passwort wird PLAIN uebergeben. Ausgegeben werden u.a. die Basisrechte des Users und alle Institutionen/Gruppen/Klassen inkl. der Rechte, in denen der User Mitglied ist.

Quickmessages lesen und neuen User anlegen

SOAP / PHP

```
$result=$client->request([
 ['method'=>'set_session','session_id'=>'255783429914332594183105 11'],
 ['method'=>'set_focus','object'=>'messenger'],
 ['method'=>'read_quick_messages'],
 ['method'=>'set_focus','login'=>'info@test.webweaver.de','object'=>'administration'],
 ['method'=>'add_user','user_name'=>'test','type'=>0,'fullname'=>'Test'],
]);
```

JSON-RPC

```
[{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "set_session",
  "params": {
    "session_id": "255783429914332594183105 11",
  }
},{
  "jsonrpc": "2.0",
  "id": 2,
  "method": "set_focus",
  "params": {
    "object": "messenger",
  }
},{
  "jsonrpc": "2.0",
  "id": 3,
  "method": "read_quick_messages"
},{
  "jsonrpc": "2.0",
  "id": 4,
  "method": "set_focus",
  "params": {
    "login": "info@test.webweaver.de",
    "object": "administration",
  }
},{
  "jsonrpc": "2.0",
  "id": 5,
  "method": "add_user",
  "params": {
    "user_name": "test",
    "type": 0,
    "fullname": "Test",
  }
},{
  "jsonrpc": "2.0",
  "id": 6,
  "method": "get_information",
}]
```

Beschreibung

1. Session-ID setzen
2. Focus auf den privaten Messenger
3. Lesen alle verfuegbaren Quickmessages
4. Focus auf die Institutions-Adminstration
5. Neuen User "test" mit Rolle 0 anlegen

Vertrauensbeziehung (Trust Relationship)

Damit das Passwort eines Nutzers nicht lokal abgelegt werden muss, kann ueber die API eine Vertrauensbeziehung angelegt werden, die das Passwort beim Login ersetzt.

Beziehung aufbauen

SOAP / PHP

```
$result=$client->request([
 ['method'=>'login','login'=>$l,'password'=>$p,'get_properties'=>[]],
 ['method'=>'set_focus','object'=>'trusts'],
 ['method'=>'register_master','remote_title'=>$t,'remote_application'=>$a],
 ['method'=>'logout']
]);
```

Beschreibung

trusts::register_master() legt eine neue Vertrauensbeziehung an. "remote_title" wird dem Nutzer in der Uebersicht seiner Vertrauensbeziehungen angezeigt. Ebenso der optionale Parameter "remote_ident". "remote_application" ist der interne Name der Applikation, die die Vertrauensbeziehung anlegt und sollte eindeutig sein (z. B. eine URL) und nur ASCII-Zeichen enthalten. Als Rueckgabe wird u. a. das Token geliefert:
['trust']['token'] => 'qVUlXz2qCOqOURjmKG3i5u5j154XwDgj6TD7FQPWuyeKAVesN4TZ1iSc5hrWhXBy'

Login

SOAP / PHP

```
$url=false;
$result=$client->request([
 ['method'=>'get_nonce']
]);
if($result and ($result[0]['return']==='OK') and ($nonce=$result[0]['nonce'])) {
 $salt='Salz (ASCII)';
 $algo='sha1';
 $token='qVUlXz2qCOqOURjmKG3i5u5j154XwDgj6TD7FQPWuyeKAVesN4TZ1iSc5hrWhXBy';
 $hash=sha1($nonce['key'].$salt.$token);
 $result=$client->request([
  ['method'=>'login','login'=>$l,'algorithm'=>'sha1','nonce_id'=>$nonce['id'],
   'salt'=>$salt,'hash'=>$hash,'application'=>$a,'get_properties'=>[]],
  ['method'=>'set_focus','object'=>'trusts'],
  ['method'=>'get_url_for_autologin','locale'=>'de'],
  ['method'=>'logout'],
 ]);
 if($result and ($result[0]['return']==='OK') and ($result[1]['return']==='OK')
    and ($result[2]['return']==='OK')) {
  $url=$result[2]['url'];
 }
}
echo $url;
```

Beschreibung

Der erste Request fordert eine Nonce vom Server an. Der zweite Request fuehrt den Login anhand des SHA1-Hashes vom Key der Nonce, einem Salz und dem Token durch. Der Parameter "application" muss mit "remote_application" beim Aufbau der Vertrauensbeziehung identisch sein. Anschiessend kann die API wie nach einem "normalen" Login genutzt werden. Im Beispiel wird eine URL angefordert, die bei Aufruf im Browser den Nutzer einloggt.