uGiftCert External API

Our external API provides means to operate with certificates from remote systems.

With API you can:

- retrieve list of certificates filtered by any certificate property. If no filter is passed all certificates are returned - method **list($filters = null)**;
- fetch single certificate data by passing certificate code - method **fetch($code)**;
- create certificate, by passing an associative array of data - method **create($data)**

Note: When creating certificates using above method, you can pass field ***qty***, which will result in qty number of certificates being created with same data. If you use this feature, you should make sure that cert_number is passed as pattern and not fixed value. Repeating certificate code is not allowed.

- you can create multiple certificates by passing an array of data arrays - method **massCreate($items)**

Note: Same concerns as for individual certificates are valid here. Each array of data can have qty field and can create multiple certificates. Advantage here is that you can pass different code patterns, amounts and periods.

- you can update single certificate by passing its certificate code and array of update data - method **update($code, $data)**
- you can update multiple certificates by passing an array of data arrays. Each data array should have **cert_number** field. - method **massUpdate($items)**
- you can delete single certificate by passing its code - method **delete($code)**
- you can delete multiple certificates by passing an array of certificate codes - method **massDelete($items)**

Magento currently has 2 versions of its SOAP API, uGiftCert can operate with both.

Those versions differ in how methods are called and in what form arguments should be passed.

V1 example:

```
$client = new SoapClient('http://example.com/api/soap/?wsdl'); // replace
'example.com' with domain name which you need to access
try {
$session = $client->login('user', 'pass'); // replace login and user with
actual magento web-service credentials.

$createData1 = array(
   'cert_number' => '[AN*8]', // if creating single certificate, you can
pass fixed code here. But it should not exist on the system.
   'balance' => 25,
   'currency_code' => 'USD',
   'store_id' => 1,
   'status' => 'P',
   'expire_at' => '2012-06-28' ,// yyyy-mm-dd
```

```
    'sender_name' => 'admin',
    'pin' => '[N*4]',
    'recipient_name' => null,
    'recipient_email' => null,
    'recipient_message' => null,
    'recipient_address' => null,
    'comments' => null
);
$createData2 = array(
    'cert_number' => 'API-[A*5]-[AN*5]', // when creating multiple
certificates, pass a pattern as code.
    'balance' => 26.99,
    'currency_code' => 'USD',
    'store_id' => 1,
    'status' => 'P',
    'expire_at' => '2012-06-28' ,
    'sender_name' => 'admin',
    'pin' => '[N*4]',
    'recipient_name' => null,
    'recipient_email' => null,
    'recipient_message' => null,
    'recipient_address' => null,
    'comments' => null,
    'qty' => 3 // this will create 3 certificates with above settings, only
difference will be auto generated code.
);
$updateData1 = array(
    'cert_number' => 'SBU99MG6', // updating does NOT require passing
cert_number, in fact if it is found, it is ignored.
    'balance' => 25.55,
);
$updateData2 = array(
    'cert_number' => 'API-XQPLE-ER3ER',
    'balance' => 29.99,
);

$certIds = array(
     'VOU-LXCUE-PZ6Y2', // for mass deletion, pass simple array of codes to
be deleted.
     'VOU-MHPDP-559FR'
);

// Filters should be composed in same way as magento collection filters,
// this filter will retrieve all certificates whose code begins with 'API'
string
$filters = array('cert_number' => array('like' => 'API%'));

// passing arguments to call method can be done by passing single value
(like above)
// or an array of values. In second case all array items are treated as
arguments to
```

```
// called method and are passed to it. That is why when we need to pass an
array as
// argument we need to include it in another array first.
// With V1 of Magento API all method names should be namespaced with
preconfigured prefix.
// For uGiftCert extension this is 'ugiftcert'

$result = $client->call($session, 'ugiftcert.list', array($filters));
// $result = $client->call($session, 'ugiftcert.fetch', 'API-XQPLE-ER3ER');
// $result = $client->call($session, 'ugiftcert.create',
array($createData1));
// $result = $client->call($session, 'ugiftcert.massCreate', array(
array($createData1, $createData2) ));
// $result = $client->call($session, 'ugiftcert.update', array('API-XQPLE-
ER3ER', $updateData1));
// $result = $client->call($session, 'ugiftcert.massCreate', array(
array($updateData1, $updateData2) ));
// $result = $client->call($session, 'ugiftcert.delete', 'API-XQPLE-ER3ER');
// $result = $client->call($session, 'ugiftcert.massDelete', array( $certIds
) );

echo '<pre>';
print_r($result);
echo '</pre>';

$client->endSession($session);
} catch (Exception $e) {
    echo $e->getMessage();
}
```

Sample result of above code is:

```
Array
(
    [0] => Array
        (
            [cert_id] => 95
            [cert_number] => API-ALXWB-KXHLR
            [balance] => 26.9900
            [currency_code] => USD
            [pin] => 7757
            [status] => P
            [expire_at] => 2012-06-28
            [recipient_name] =>
            [recipient_email] =>
            [recipient_address] =>
            [recipient_message] =>
            [store_id] => 1
            [sender_name] => admin
        )
```

```
    [1] => Array
        (
            [cert_id] => 97
            [cert_number] => API-CCQEY-HF97E
            [balance] => 26.9900
            [currency_code] => USD
            [pin] => 8325
            [status] => P
            [expire_at] => 2012-06-28
            [recipient_name] =>
            [recipient_email] =>
            [recipient_address] =>
            [recipient_message] =>
            [store_id] => 1
            [sender_name] => admin
        )

    [2] => Array
        (
            [cert_id] => 93
            [cert_number] => API-CRHUR-FLHDR
            [balance] => 26.9900
            [currency_code] => USD
            [pin] => 5748
            [status] => P
            [expire_at] => 2012-06-28
            [recipient_name] =>
            [recipient_email] =>
            [recipient_address] =>
            [recipient_message] =>
            [store_id] => 1
            [sender_name] => admin
        )
)
```

V2 example:

V2 of Magento API was created mainly to improve on integration of the API with non PHP platforms. While V1 of the API accepts any PHP built in data type as argument, and could possibly return any such type, it is not very friendly to non PHP platforms. With V2 of the API arguments passed should be either scalar types (int, string, float, bool), indexed arrays ['one', 'two', 'three'] or objects whose public properties are used. Also in V2 method calling is changed, for method to be callable, it should be declared in WSDL file and general rule of method naming is: *methodPrefix* + upper cased method name. So with prefix **ugiftcert** and method **list** the call becomes **ugiftcertList**. This has the benefit of not using `$client→call($session, 'ugiftcert.list');` but `$client→ugiftcertList($session);`. There is also difference how method arguments are passed, with V2 arguments are declared in WSDL and are being passed directly to method call. No need to wrap in arrays anything and multiple arguments are passed as with any normal function/method call.

So to use V2 API all associative arrays should become stdClass objects with property names the keys of arrays and property values - values of arrays.

```
    try {
        $client = new SoapClient('http://example.org/api/v2_soap/?wsdl=1');
// note usage of different WSDL url
        $session = $client->login('user', 'pass');
        $data = array(
            'balance' => 155.5,
            'status' => 'A',
            'cert_number' => 'VOU-LXCUE-PZ6Y2'
        );
        $data = (object)$data; // easiest way to convert associative array
in object we need is to cast it to one
    // in other programming languages, you should provide a simple value
object as argument.
    // data.balance = 155.5;
    // data.status = 'A';
    // data.cert_number = 'VOU-LXCUE-PZ6Y2';

        $createData1 = array(
            'cert_number' => '[AN*8]',
            'balance' => 25,
            'currency_code' => 'USD',
            'store_id' => 1,
            'status' => 'P',
            'expire_at' => '2012-06-28',
            'sender_name' => 'admin',
            'pin' => '[N*4]',
            'recipient_name' => null,
            'recipient_email' => null,
            'recipient_message' => null,
            'recipient_address' => null,
            'comments' => null
        );
        $createData1 = (object)$createData1;

        $createData2 = array(
            'cert_number' => 'API-[A*5]-[AN*5]',
            'balance' => 26.99,
            'currency_code' => 'USD',
            'store_id' => 1,
            'status' => 'P',
            'expire_at' => '2012-06-28',
            'sender_name' => 'admin',
            'pin' => '[N*4]',
            'recipient_name' => null,
            'recipient_email' => null,
            'recipient_message' => null,
            'recipient_address' => null,
            'comments' => null,
```

```
            'qty' => 3
        );
        $createData2 = (object)$createData2;

        $updateData1 = array(
            'cert_number' => 'API-CRZWF-ANCP2',
            'balance' => 25.55,
        );

        $updateData1 = (object)$updateData1;

        $updateData2 = array(
            'cert_number' => 'E34JU4EE',
            'balance' => 29.99,
        );

        $updateData2 = (object)$updateData2;

        $certIds = array(
            'API-CRZWF-ANCP2',
            'E34JU4EE',
            '8ZR8ERQ9',
            'API-FYEKA-M5ADP',
            'API-DDPLL-NSDXP',
            'API-NWXFY-5ENJP'
        ); // simple indexed arrays need not to be changed.


    // To provide filters they should be constructed as value object
    // with public properties 'filter' and 'complex_filter' (optionally)
        $filter = new stdClass();
        $filter->filter = array(); // filter should be indexed array of
filter objects
        $filter->complex_filter = array();

    // sample status filter
        $status = new stdClass();
        $status->key = 'status'; // filter key or attribute name to filter
        $status->value = 'A';  // filter value or the value of filtered
attribute

    // sample complex filter
        $balance = new stdClass();
        $balance->key = 'balance'; // filter by 'balance' attribute
        $balance->value = new stdClass(); // filtering value should be an
object
        $balance->value->key = 'gt';      // with key - condition to match -
gt = greater than
        $balance->value->value = '30';     // and value to use in condition
matching
```

```
        $filter->filter[] = $status; // add simple filters to filter
property
    $filter->complex_filter[] = $balance; // add complex filters to
complex_filter property

                    $result = $client->ugiftcertList($session, $filter); //
get all 'A'ctive certificates with more than 30 balance (currency
independent)
        //        $result = $client->ugiftcertFetch($session, 'API-FNJBS-
QPMVG');// input valid certificate number for your setup
        //        $result = $client->ugiftcertUpdate($session, 'VOU-
MHPDP-559FR', $data); // input valid certificate number for your setup
        //        $result = $client->ugiftcertCreate($session,
$createData2);
        //        $result = $client->ugiftcertDelete($session, 'API-FNJBS-
QPMVG');// input valid certificate number for your setup
        //        $result = $client->ugiftcertMassCreate($session,
array($createData1, $createData2));
        //        $result = $client->ugiftcertMassUpdate($session,
array($updateData1, $updateData2));
        //      $result = $client->ugiftcertMassDelete($session, $certIds);
        echo '<pre>';
        print_r($result);
        echo '</pre>';

        $client->endSession($session);
    } catch (Exception $e) {
        echo $e->getMessage();
    }
```

Sample result:

```
Array
(
    [0] => stdClass Object
        (
            [cert_number] => VOU-FDZTT-9U4QK
            [currency_code] => USD
            [pin] => 5756
            [status] => A
            [expire_at] => 2011-08-13
            [sender_name] =>
            [store_id] => 1
            [balance] => 150
        )

    [1] => stdClass Object
        (
            [cert_number] => VOU-NRHJJ-FMVRS
            [currency_code] => USD
            [pin] => 3249
```

```
            [status] => A
            [expire_at] => 2011-08-13
            [sender_name] =>
            [store_id] => 1
            [balance] => 150
        )

    [2] => stdClass Object
        (
            [cert_number] => VOU-FTZNQ-V6KK3
            [currency_code] => USD
            [pin] => 8734
            [status] => A
            [recipient_name] => Petar
            [recipient_email] => jamby77@gmail.com
            [recipient_address] =>
            [recipient_message] =>
            [sender_name] =>
            [store_id] => 1
            [balance] => 255
        )
)
```