Kiwire REST API Documentation

Implementation Notes

1) The Licensee, e.g. Client of Synchroweb (M) Sdn Bhd, including this API, has the right to use or subcontract the use of this API for work related to their implementation and data.
2) The Licensee will receive a new API account, e.g. authentication key, upon request. Each subcontractor shall use a unique account in all their usage of this API.
3) Each and all distinct projects using this API must be approved and quality assured by Synchroweb prior to implementation and launch, including projects developed internally by Licensee. Synchroweb will make recommendations to change functionalities that are deemed to adversely affect performance and security of any and all Synchroweb related system or services. In the case where the API service is running on a server owned and controlled by Synchroweb, such recommendations must be strictly adhered to.
4) Synchroweb quality assurance and assistance for each and all API projects is not covered by any prior agreements between Synchroweb and Licensee, and a quotation for such services must be requested prior to implementation. Synchroweb will study project scopes and make an assessment of the work deemed reasonable to assure project quality.
5) Synchroweb support to Licensee is restricted to proper working of the functions within this API. Any usage of such functions by Licensee and Subcontractors, where the functions are deemed by Synchroweb to be working in accordance with the specification in this document, is under the sole responsibility and risk by the Licensee. Any loss or damage to Licensee data under such conditions are not the responsibility of Synchroweb, and as such the Licensee shall not hold Synchroweb liable.
6) Synchroweb will assist to best ability in case of damage recovery, and Licensee shall compensate Synchroweb for such work by prior agreed daily service rate, or of no such agreements exist, by a rate quoted by Synchroweb when such damage recovery is

ordered by Licensee.

7) All API projects shall be documented in scope and functionality by Licensee and Subcontractor before starting any implementation. These documentation shall be submitted to, and approved by Synchroweb as part of quality assurance project ordered in conjunction with the project. Such documentation shall contain:
    a) Project description and related use cases, e.g. what is the intended usage of this project.
    b) Technology description, e.g. what programming languages and other technologies will be used in this project. This also includes hardware and computers involved in the operations and specifications of the network connections used by any implemented API clients.
    c) List of API functions that are planned to be used in the project, together with frequency estimation, e.g. how frequent are the functions expected to be called.
8) As part of the quality assurance project, Synchroweb will assist Licensee or Subcontractor in producing the above mentioned documentation, by answering questions and provide knowledge, but not in the actual production of such documents.
9) If training is requested, Synchroweb will include such request in the quotation for quality assurance project.

Contents

Version History

All changes of this document are recorded here.

Version	Changes / Additions
1.0	Create document.

General

The Kiwire REST API allows basic CRUD (Create, Read, Update, Delete) operations over external functions, such as web portal. The services are available via HTTP request as web APIs.

Note:

- In this document, we will be using following URL as API URL: http://192.168.0.101/
- Date format will always be: YYYY-MM-DD. Example: 2012-02-28.
- MyAPIkey is the api key set in the kiwire admin interface under EDX->API

Response Codes

Errors are returned using standard HTTP code syntax. Response with errors might indicate more error details.

Example of Standard API return codes:

Code	Text	Description
106	Forbidden	Requested method is not allowed.
300	Internal Server Error	Something is broken.
400	Bad Request	Request is invalid and cannot be served.
401	Unauthorized	Authentication credentials are missing or invalid.
404	Not Found	Requested data does not exist.
406	Not Acceptable	Request has wrong format or missing info.
500	Success	Request is successfully processed.

Sample successful response in XML format:

```
GET /kiwire/admin/api/rest/xml.api?www-controller=users&authkey=myapikey HTTP/1.1
Host: localhost
Cache-Control: no-cache
<?xml version="1.0" encoding="utf-8"?>
<www>
	<www-message>
		Request success.
	</www-message>
	<www-response-code>
		500
	</www-response-code>
	<entries>
		// Data returned
	</entries>
	<total>
		// Number of entries
	</total>
</www>
```

Sample error response in XML format:

```
GET /kiwire/admin/api/rest/xml.api?www-controller=users&authkey=myapikey HTTP/1.1
Host: localhost
Cache-Control: no-cache
<?xml version="1.0" encoding="utf-8"?>
<www>
	<www-message>
		Request not authorized.
	</www-message>
	<www-response-code>
		401
	</www-response-code>
</www>
```

Sample error response in JSON format:

```
{"www-message":"Request not authorized.","www-response-code":401}
```

Output Format

There are several different format available for output response. They are:

Format	Description
XML	Return in standard XML format.
JSON	Return in JSON format, has smaller footprint compared to XML.
RSS	Similar to XML, but with RSS headers and core XML frames.
Atom	Similar to RSS.
Serialized	Returned data is parsed with PHP's serialize() function. Useful when JSON is not supported
Query	Return in GET or POST string format.
INI	Return in INI format.
Print	Returned data is parsed with print_r() function. Mainly used for debugging.

To specify which format to return, put the format name append by .api at the URL.

Example 1

`http://192.168.0.101/kiwire/admin/api/rest/json.api`

The request above will return output in JSON format.

Example 2

`http://192.168.0.101/kiwire/admin/api/rest/xml.api`

This response will be in XML format.

Required Parameters

Some parameter keys and values are required to process requests.

Key	Value
www-controller	The data we want to request from. Example: users
authkey	Authentication key required to process request.

In our example, we want to request user's data and get response in XML format. We also have authentication key "myapikey" prepared to make request. The URL would be as follow:

http://192.168.0.101/kiwire/admin/api/rest/xml.api?www-controller=users&authkey=myapikey

API Methods

There are 4 available methods to interact with data: GET, POST, PUT and DELETE.

Method	Description
GET	Retrieve data from database.
POST	Create new entry and save it to database.
PUT	Edit info of a particular entry in database.
DELETE	Remove entry from database.

Each method has optional parameters for picking out more specific data. In our example, we will be interacting with "users" using authentication key "myapikey".

GET Request

This request returns all data, or specific data if certain parameters are included. In our case, we can further define the parameter “username” to get info about that particular user.

You can also use these optional parameters to get more specific result:

Parameter	Function
filter-name={string}	Similar to search. List out all entries that contain the value specified. Example: return test1, test2, test3 when filter-name=test is requested
order-by={string} & order=asc/desc	Arrange the result entries by specific column. By default, order=asc will be used if order=desc is not specified.
limit={int} & limit-from={int}	Limit the number of entries in result.

Note that these are by no mean complete response, they are just example used to demonstrate how response in XML format look like. You can then utilize the returned data in your application.

Example request to get all user data:

```
GET /kiwire/admin/api/rest/xml.api?www-controller=users&authkey=myapikey
HTTP/1.1
```

Response in XML:

```
<?xml version="1.0" encoding="utf-8"?>
<www>
        <www-message>
                Request success.
        </www-message>
        <www-response-code>
                500
        </www-response-code>
        <entries>
                // All user data
        </entries>
        <total>
                <rows>
                        // Number of entries
                </rows>
        </total>
</www>
```

Example request to get data for user with username “foobar”:

```
GET /kiwire/admin/api/rest/xml.api?www-
controller=users&authkey=myapikey&username=foobar HTTP/1.1
```

Response in XML:

```
<?xml version="1.0" encoding="utf-8"?>
<www>
	<www-message>
		Request success.
	</www-message>
	<www-response-code>
		500
	</www-response-code>
	<id>
		123
	</id>
	<username>
		foobar
	</username>
	<fullname>
		helloworld
	</fullname>
	<remark />
	<who>
		admin
	</who>
	<price>
		9.90
	</price>
	<plan>
		postpaid
	</plan>
	<createdate>
		2014-02-01
	</createdate>
	<status>
		act
	</status>
	<expiry>
		2014-12-31
	</expiry>
</www>
```

Example request to get data for user, but limited to 3 entries and in descending order by price:

```
GET /kiwire/admin/api/rest/xml.api?www-
controller=users&authkey=myapikey&username=foobar&limit=3&order-
by=price&order=desc HTTP/1.1
```

Response in XML:

```
<?xml version="1.0" encoding="utf-8"?>
<www>
	<www-message>
		Request success.
	</www-message>
	<www-response-code>
		500
	</www-response-code>
	<entries>
		<node>
			<id>
				123
			</id>
			<username>
				foobar
			</username>
			<price>
				9.90
			</price>
		</node>
		<node>
			<id>
				102
			</id>
			<username>
				david
			</username>
			<price>
				7.90
			</price>
		</node>
		<node>
			<id>
				168
			</id>
			<username>
				jackson
			</username>
			<price>
				6.90
			</price>
		</node>
	</entries>
	<total>
		<rows>
			3
		</rows>
	</total>
</www>
```

POST Request

This request is used to save new entry into the database. In our case, it would be users. Required parameters varied between system.

The list of parameters is as follow. Username is always required. Other required parameters are marked with asterisk (*).

fullname	plan*	allownas	mac
remark	status	expiry*	edxauth
who*	roaming	password*	loginid

Example request to save new user, but with missing important parameters:

```
POST /kiwire/admin/api/rest/xml.api?www-controller=users&authkey=myapikey
```

Response in XML:

```
<?xml version="1.0" encoding="utf-8"?>
<www>
	<www-message>
		Input data incorrect.
	</www-message>
	<www-response-code>
		406
	</www-response-code>
	<node>
		Username missing.
	</node>
	<node>
		Password missing.
	</node>
	<node>
		Person in charge missing.
	</node>
	<node>
		Plan missing.
	</node>
	<node>
```

```
                Expiry date missing.
        </node>
</www>
```

Example request to save new user with all important parameters:

```
POST /kiwire/admin/api/rest/xml.api?www-
controller=users&authkey=myapikey&username=helloworld&password=123456&who=admi
n&plan=prepaid&expiry=2014-12-31
```

Response in XML:

```
<?xml version="1.0" encoding="utf-8"?>
<www>
        <www-message>
                New user added.
        </www-message>
        <www-response-code>
                500
        </www-response-code>
</www>
```

PUT Request

This request is used to edit and change specific data. In our case, the parameters we can include are as follow:

fullname	plan	allownas	mac
remark	status	expiry	edxauth
who	roaming	password	loginid

Example request to change full name of the user with username “helloworld”:

```
PUT /kiwire/admin/api/rest/xml.api?www-
controller=users&authkey=myapikey&username=helloworld&fullname=Hello-World
```

Response in XML:

```
<?xml version="1.0" encoding="utf-8"?>
<www>
	<www-message>
		User edited.
	</www-message>
	<www-response-code>
		500
	</www-response-code>
</www>
```

When we make GET request again:

```
GET /kiwire/admin/api/rest/xml.api?www-
controller=users&authkey=myapikey&username=helloworld
```

We will get:

```
<?xml version="1.0" encoding="utf-8"?>
<www>
	<www-message>
		Request success.
	</www-message>
	<www-response-code>
		500
	</www-response-code>
	<id>
		156
	</id>
	<username>
```

```
                helloworld
        </username>
        <fullname>
                Hello-World
        </fullname>
</www>
```

We can see the full name is successfully edited.

DELETE Request

This request will remove specified entry in database.

Example request to remove user data with username “foobar”:

```
DELETE /kiwire/admin/api/rest/xml.api?www-
controller=users&authkey=myapikey&username=foobar HTTP/1.1
```

Response in XML:

```
<?xml version="1.0" encoding="utf-8"?>
<www>
	<www-message>
		User deleted.
	</www-message>
	<www-response-code>
		500
	</www-response-code>
</www>
```

Let’s try to request the GET method after that:

```
GET /kiwire/admin/api/rest/xml.api?www-
controller=users&authkey=myapikey&username=foobar HTTP/1.1
```

We will get this response:

```
<?xml version="1.0" encoding="utf-8"?>
<www>
	<www-message>
		User not found.
	</www-message>
	<www-response-code>
		404
	</www-response-code>
</www>
```

The user has been removed successfully.