Restaurant Search by Zipcode

http://www.foodarena.ch/serve_search_xml.php?criteria=<zipcode>

DTD

```
<!ELEMENT search_results (restaurant* | EMPTY)>
<!ELEMENT restaurant (picture, title, address, phone,
description, current_delivery_hrs, delivery_only,
offer_delivery, current_opening_hrs, min_order_value,
card_payment_option, delivery_charge, delivery_time,
preparation_time, is_open>
<!ATTLIST restaurant id ID #REQUIRED>
<!ELEMENT picture (link)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT address (#PCDATA)>
<!ELEMENT phone(#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT current_dekivery_hrs (#PCDATA)>
<!ELEMENT delivery_only (#PCDATA)>
<!ELEMENT offer_delivery (#PCDATA)>
<!ELEMENT current_opening_hrs (#PCDATA)>
<!ELEMENT min_order_value (#PCDATA | EMPTY)>
<!ELEMENT card_payment_option (#PCDATA | EMPTY)>
<!ELEMENT delivery_charge (#PCDATA | EMPTY)>
<!ELEMENT delivery_time (#PCDATA | EMPTY)>
<!ELEMENT preparation_time (#PCDATA)>
<!ELEMENT is_open (#PCDATA)>
<!ELEMENT cuisine_id_csv (#PCDATA)>
```

Notes:

- title, address, phone and description elements are URL encoded. This is to ensure backward compatibility with foodrena's existing XML clients.
- Opening and delivery hours may consist of two sets of times to allow for restaurant closure in the middle of the day. If this is the case, the two time sets will be separated by a comma. If second time period is not shown after comma, then the restaurant does not operate two sets of opening times.
- If search query returns no results, search_results element will be empty.

Get Restaurant's Menu by Restaurant ID

http://www.foodarena.ch/serve_menu_xml.php?id=<restaurant id>

DTD

```
<!ELEMENT menu (menu_group, combo_menu_group*)>
<!ELEMENT menu_group (group_title, group_order,
menu_item)>
<!ATTLIST menu_group group_id ID #REQUIRED>
<!ELEMENT group_title (#PCDATA)>
<!ELEMENT group_order (#PCDATA)>
<!ELEMENT menu_item (item_title, item_description,
item_number, item_picture, item_price+, item_extra*>
<!ATTLIST menu_item item_id ID #required>
<!ELEMENT item_title (#PCDATA)>
<!ELEMENT item_description (#PCDATA | EMPTY)>
<!ELEMENT item_number (#PCDATA)>
<!ELEMENT item_picture (#PCDATA | EMPTY)>
<!ELEMENT item_price (item_price_desc,
item_price_collect, item_price_deliver)>
<!ATTLIST item_price item_price_id ID #REQUIRED>
<!ELEMENT item_price_desc (#PCDATA | EMPTY)>
<!ELEMENT item_price_collect (#PCDATA)>
<!ELEMENT item_price_deliver (#PCDATA)>
<!ELEMENT item_extra (item_extra_name,
item_extra_price)>
<!ATTLIST item_extra extra_id ID #REQUIRED>
<!ELEMENT item_extra_name (#PCDATA)>
<!ELEMENT item_extra_price (#PCDATA)>
<!ELEMENT combo_menu_group (combo_group_title,
combo_item_groups+)>
<!ATTLIST combo_menu_group combo_groups_id ID
#REQUIRED>
<!ELEMENT combo_group_title (#PCDATA)>
<!ELEMENT combo_item_groups (combo_item_groups_title,
combo_item_groups_price, combo_item_group)>
<!ATTLIST combo_item_groups item_groups_id ID
#REQUIRED>
<!ELEMENT combo_item_groups_title (#PCDATA)>
<!ELEMENT combo_item_groups_price (#PCDATA)>
<!ELEMENT limited_availability (start_time, end_time,
days)>
<!ATTLIST limited_availability currently_available
value(1|0) #REQUIRED>
<!ELEMENT start_time (#PCDATA)>
<!ELEMENT end_time (#PCDATA)>
<!ELEMENT days (#PCDATA)>
```

```
<!ELEMENT combo_item_group (combo_item_group_title,
combo_item_id+)>
<!ATTLIST combo_item_group id ID #REQUIRED>
<!ATTLIST combo_item_group num_items CDATA #REQUIRED>
<!ELEMENT combo_item_group_title (#PCDATA)>
<!ELEMENT combo_item_id (#PCDATA)>
<!ATTLIST combo_item_id item_price_id ID #required>
```

NOTES:

- Combo offers: Note the difference between ‘combo_menu_groups’ and ‘combo_menu_group’. A combo offer can consist of many a ‘combo_menu_group’. These are contained by the ‘combo_menu_groups’ element. It is the combo_menu_group that contains the items that are offered as part of that group.
- The ‘group_order’ element should be used to ensure menu groups appear to the user in the correct order. If the group order is set to zero, the order is assumed to be that of the XML.
- Each standard menu_group element contains menu items. Each menu item may have several different prices (eg, small, medium, large pizza). Each price should be selected depending on if the order is for delivery or collection.

Get Possible Delivery or Collection Times

http://www.foodarena.ch/serve_future_times_xml.php?rest_id=<restaurant id>&deliv_check=<true or false>

DTD

```
<!ELEMENT future_times (time*)>
<!ELEMENT time (#PCDATA)>
```

NOTES:

- Possible points in time that the order can be prepared or delivered for (depending on deliv_check url variable)
- Time element contents are unix timestamps.

Get Registered User Details

http://www.foodarena.ch/serve_user_xml.php?email=<user's email>&passwd=<user's password>

DTD

```
<!ELEMENT user_result (id, name, vorname, strasse, nr,
floor_nr, plz, ort, kanton, tel, mobile_tel,
newsletter, gender, dob, points, lang)>
<!ELEMENT id (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT vorname (#PCDATA)>
<!ELEMENT strasse (#PCDATA)>
<!ELEMENT nr (#PCDATA)>
<!ELEMENT floor_nr (#PCDATA | EMPTY)>
<!ELEMENT plz (#PCDATA)>
<!ELEMENT ort (#PCDATA)>
<!ELEMENT kanton (#PCDATA)>
<!ELEMENT tel (#PCDATA)>
<!ELEMENT mobile_tel (#PCDATA | EMPTY)>
<!ELEMENT newsletter (#PCDATA)>
<!ELEMENT gender (#PCDATA)>
<!ELEMENT dob (#PCDATA | EMPTY)>
<!ELEMENT points (#PCDATA)>
<!ELEMENT lang (#PCDATA)>
```

NOTES:

- If more than 5 requests have been made within a 15min period by the same IP, further requests will not be served to that IP for an additional 15 minutes.

Sending an Order

Orders are sent in a multi-dimensional POST var array called 'order' to the following URL:

http://www.foodarena.ch/send_order_external.php

This is the POST data for the order stored in an post var array called: 'order'. This order only has one item with some two extras. If there are more than one item then the order array will just have more elements... so for example, $_POST['order'][0] to $_POST['order'][9] for 10 items in the order.

```
Array
(
    [0] => Array
        (
            [id] => 9614
            [item_number] => 7401
            [price] => 13
            [price_desc] => Medium
            [name] => Hawaii & stuff
            [qnty] => 1
            [total] => 13
            [extras] => Array
                (
                    [0] => Array
                        (
                            [extra_id] => 123
                            [name] => onionss
                            [extra_price] => 1.00
                        )

                        [1] => Array
                        (
                            [extra_id] => 456
                            [name] => mushroomss
                            [extra_price] => 2.00
                        )

                    )

          )

)
```

This is an example post var that completes the order details....

```
[company] =>
[name] => my surname
[vorname] => my firstname
[strasse] => my street
[nr] => 22
[floor_nr] => 1
[plz] => 333
[ort] => my city
[email] => email@email.com
[DOB_day] =>
[DOB_month] =>
```

```
[DOB_year] =>
[phone] => 12345
[special_instructions] => my test message to restaurant
[user_id] => 16
[deliv_check] => true
[future_time] => 1260197100
[deliv_cost] => 4.00
[id] => 91
[order_from] => iphone
[payment_type] => cash
[voucher_code] => tst10pct
[referral_credit_id] => 1
[referral_discount_credit] => 6.00
[oauth_consumer_key] => consumer key code
[device_id] => abcd1234
[app_version] => 1.2
[card_handling_pc] => 2.5
```

Send order XML response

```
<!ELEMENT order_result (order_sent, error*,
missing_field*, order_code?, user_account_auth?)>
<!ELEMENT order_sent (#PCDATA)>
<!ELEMENT error (#PCDATA)>
<!ELEMENT missing_field (#PCDATA)>
<!ELEMENT order_code (#PCDATA)>
<!ELEMENT user_referral_code (#PCDATA)>
<!ELEMENT user_account_auth (#PCDATA)>
```

If order is sent successfully, an order code will be returned and `order_sent` will be `true`.

If order is NOT sent successfully, then `order_sent` element will be `false` and one or more `error` elements will describe the nature of the error. In this case the `order_code` element will not be present.

`user_account_auth` element will contain an OAuth access token and key, if an account has been created automatically. An OAuth consumer key must have been provided when sending the order for this to happen.

`user_referral_code` contains the user’s referral code

The post data vars that must be sent (even if empty):

```
- order
- id
- deliv_cost
- deliv_check
- future_time
- plz
- name
- vorname
- strasse
- nr
- ort
- email
- phone
- special_instructions
```

NOTE:

- Setting the 'order_from' field will allow foodarena to track orders coming from your service by referring to the tag provided.
- 'customer_paid' should be set to 1 for card orders and 0, or left blank, for cash orders
- future_time is one of the times provided as a result of the serve_future_times_xml.php described above.
- The 'id' shown in the 'order' array above, is the PRICE_ID of the item, NOT the item_id.
- If POST var fields are missing, they will be returned in the XML result as missing_field elements.
- 'payment_type' may either be set to 'cash' for orders where the customer will pay on delivery of their order, or 'card' if the user intends to make a card payment. If the 'payment_type' is set to 'card', the order will be placed into a pending state, where it is awaiting payment. When a successful payment is made, an automatic callback from the payment handler updates the order status, and the order is then sent.
- 'voucher_code' can be left blank if no voucher is being used. If it has a value set, the voucher will be checked for validity. If the voucher code sent is not valid, the order will not be sent, and an error message will result. Please check voucher codes before sending the final order, by using the 'check voucher code' facility, which is explained later in this document.
- 'referral_discount_credit' is optional, but must be specified if a discount has been applied to this order as a result of referral credits shown as being available in the user's account. Availability of credits can be ascertained by making a request for the user's data, as detailed in API guide document under 'Get User Data'.
- ‘referral_credit_id’ is optional, but must be specified if a discount has been applied to this order as a result of referral credits shown as being available in the user's account. Referral credit id can be obtained in the user’s data in field, ‘available_referral_credit_id’.
- ‘device_id’ is optional, but where present identifies a unique device. Usually a mobile device id that identifies a particular mobile phone etc. It may be used to enforce ‘one use per user’ type rules, such as with certain voucher codes.
- 'oauth_consumer_key' is optional, but required to produce an OAuth access token in the response, if a new user account has automatically been created. The OAuth access token can be used for instant access to the created account.
- ‘app_version’ is optional. When used it should contain the software version of the sending application.
- ‘card_handling_pc’ must be sent if online payment method is used. It should contain the card handling fee used for an order as a percentage. For example for a fee of 2.5% the card_handling_pc should be set to 2.5. The card handling fee to use can be obtained from the foodarena API. See API docs.

Change an Order's Payment Type to Cash

Sometimes it may be necessary to update an order's payment type. For example, if the user originally intended to pay by card, but then decided to pay by cash instead. In this instance the order would need its 'payment_type' updated to 'cash' before the order can be sent, as it would otherwise just be left in a pending state, where a card payment is expected before the order can be sent.

POST data is sent to the following URL:

http://www.foodarena.ch/update_order_payment_type.php

The following POST vars are required:

- `order_code`
- `payment_type`

XML Response:

```
<!ELEMENT update_result (order_updated, order_code,
error*, missing_field*)>
<!ELEMENT order_updated (#PCDATA)>
<!ELEMENT order_code (#PCDATA)>
<!ELEMENT error (#PCDATA)>
<!ELEMENT missing_field (#PCDATA)>
```

If the order is updated successfully, `order_updated` will be set to `true`.

If the order was previously in a pending state, awaiting payment, and the `payment_type` is updated to `cash`, the order will then be sent automatically.

Online Payment

The order must be sent to foodarena BEFORE the user is forwarded for payment collection, and a positive response with an order code identifier must be received. This is to ensure the order exists in foodarena's database in a state that is 'awaiting payment'.

If the user is forwarded before the order exists at foodarena, they will be returned back to the referring page, as the order will be unknown and therefore unable to be processed for payment.

NOTE: A card handling charge might need to be added to the user's total order amount. This can be different for individual payment types. Please see the API guide for details of how to use the API to retrieve the correct card handling charge amount.

The following POST parameters must be included when the order is sent to foodarena.

`payment_type` - Mandatory. This must be set to the users chosen online payment 'method. Valid values:

`VIS` - Visa
`ECS` - Mastercard
`PAP` - Paypal
`PFC` - Postfinance Card
`PEF` - Postfinance E-finance
`ESY` - Swisscom Easypay
`TWT` - TWINT

Upon receipt of a success response from foodarena after sending an order, the user may now be forwarded to the following URL:

```
https://www.foodarena.ch/cc_payment_forwarder.php?
external=1&order_number=<order code>
```

Where `order code` is the order number relating to the customer's order, contained in the initial order send response (above). The 'hash' character in the order code should be URL encoded (%23). For instance: #123456789 url encoded becomes %23123456789

Once the user has been forward to foodarena, foodarena will handle the order send to completion. No further interaction with the user is necessary beyond this point.

Register a New User

User details are sent as POST vars to the following URL:

http://www.foodarena.ch/serve_reg_usr.php

The following POST vars are required:

- `email`
- `name`
- `vorname`
- `password`
- `password_conf`
- `strasse`
- `nr`
- `plz`
- `ort`

If successful, the following XML will be returned (example).

```
<result>
      <status>success</status>
      <user_id>123</user_id>
</result>
```

The `user_id` returned is the user ID of the newly created user.

If unsuccessful, the following XML will be returned (example).

```
<result>
      <status>failure</status>
      <msg>User already exists</msg>
</result>
```

The `msg` element shows the type of error that occurred.

Check Voucher Code

Vouchers can be checked by sending voucher code data (GET or POST) to the following url:

http://www.foodarena.ch/serve_check_voucher_xml.php

Required vars:

- `voucher_code` the voucher code you wish to check
- `restaurant_id` the restaurant id you wish to check against (probably of the current order)
- `order_value` the value of the order to check against (some vouchers may specify a minimum order value)

Optional vars:

- `usage_time` the unix timestamp time to check against. If empty, the current time will be used.
- `device_id` if using a mobile device, this should be provided. It is used to enforce ‘single use only’ vouchers, by checking against previously used vouchers.

- `device` The requesting device. Possible values: `iphone`, `android`

Voucher check XML response DTD

```
<!ELEMENT voucher_check_result (result?,
discount_amount?, error*)>
<!ELEMENT result (#PCDATA)>
<!ELEMENT discount_amount (#PCDATA)>
<!ELEMENT error (#PCDATA)>
```

If the voucher code is valid (it exists, meets minimum order value, it hasn't expired etc), then the `result` element will contain a positive value. A negative value indicates an invalid voucher. Codes for result element are shown below:

` 1` = Valid voucher
`-1` = Invalid voucher
`-2` = Order value too low
`-3` = User not eligible to use this voucher (single use only / new users only / etc)

If the voucher is valid, the discount_amount element will show the discount amount to apply for the voucher being tested. The amount is a positive figure, probably to be deducted from the user's total order value.

One or more `error` elements, if present, will contain errors such as missing fields.