

Marathon API

Table of Contents

Marathon API.....	1	get_orders.....	23
Version history.....	2	create_order.....	24
Introduction.....	3	change_order.....	29
General.....	4	change_client_status.....	30
Media functions.....	5	create_order_direct.....	31
get_clients.....	5	delete_order.....	33
get_products.....	6	get_invoice.....	34
get_agreements.....	7	Media database functions.....	35
get_agreement_details.....	8	get_placements.....	35
get_collective_mediatypes.....	9	get_insertion_dates.....	36
get_mediatypes.....	10	get_sizes.....	37
get_medias.....	11	get_price.....	38
get_discount_codes.....	12	Project functions.....	39
get_surcharge_codes.....	13	get_proclients.....	39
get_units.....	14	get_project.....	40
get_campaign.....	15	get_projects.....	41
get_campaigns.....	16	create_project.....	42
create_campaign.....	17	get_feecodes.....	43
get_plan.....	18	get_employee.....	44
get_changed_plans.....	19	get_timereports.....	45
create_plan.....	20	create_timereport.....	46
scratch_plan.....	21	get_proinvoice.....	47
get_order.....	22	get_proinvoices.....	48

Version history

Version	Description	Author	Date
1	Marathon API created from earlier API:s.	Gunnar Weiner	2017-12-29
2	New function get_mediatypes.	Gunnar Weiner	2018-01-31
3	New returned fields in get_project and new function get_employee.	Gunnar Weiner	2018-02-02
4	New function get_collective_mediatypes and filter on collective mediatype in get_mediatypes.	Gunnar Weiner	2018-02-02

Introduction

This document contains a description of the Marathon standard API. Before using the API some configuration has to be done by Kalin Setterberg to open it up.

This document contains some basic information needed to use the API. It also contains descriptions of all available functions.

Some functions are used for extracting data from base registers like clients, products, agreements and medias. These functions extract data from the base registers in the corresponding register in "Backoffice/Base registers" in the local Marathon implementation.

Some functions are used for extracting basic data from the media database maintained by the supplier of Marathon (Kalin Setterberg). To be able to use these API:s a special agreement with Kalin Setterberg is needed.

General

The API is basically a REST API where XML-data is sent to the Marathon server and XML-data is returned.

The XML-data should be sent to the Marathon server with HTTPS POST to the url you receive from Kalin Setterberg when the API is configured for you. The following illustrates the structure of the url.

`https://<marathon server address>/cgi-bin/<program>?<password>`

For each function in this document you will find an example of the input XML-data.

The password in the input XML-data is created in Marathon in "System/Integration API" where it is also possible to restrict the password to certain functions.

A reply xml is received directly from the marathon server. The reply xml has the following format if the result is OK.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<marathon>
  <status>OK</status>
  xxx
</marathon>
```

xxx is replaced with fields relevant to the function. These fields are described for each function.

Format of the reply xml if anything goes wrong:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<marathon>
  <status>ERROR</status>
  <message>company_id: missing</message>
</marathon>
```

Media functions

get_clients

Input: filter on client name

Output : active clients

Filtering can be done on the name. The search pattern can start and/or end by an asterisk. Without asterisk exact match is required. All searches are ignoring case.

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>get_clients</type>
  <company_id>DS</company_id>
  <filter_client_name>*o</filter_client_name>
</marathon>
```

Reply:

```
<client id="VOLV" name="Volvo AB" internal_name="Volvo" />
```

get_products

Input: client id

Output: active products

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>get_products</type>
  <company_id>DS</company_id>
  <client_id>VOLV</client_id>
</marathon>
```

Reply:

```
<product id="LAST" name="Lastbilar" />
<product id="PERS" name="Personbilar" />
```

get_agreements

Input: client id

Output: list of allowed agreements

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>get_agreements</type>
  <company_id>DS</company_id>
  <client_id>DS</client_id>
</marathon>
```

Reply:

```
<agreement id="VOLV-1" name="Volvo 30 dgr, 2 %" />
```

get_agreement_details

Input: agreement id

Output: agreement details

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>get_agreement_details</type>
  <company_id>DS</company_id>
  <agreement_id>DS</agreement_id>
</marathon>
```

The function returns all fields currently in the client agreement (as of 29/10/12).

get_collective_mediatypes

Input: -

Output: all collective mediatypes

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>get_collective_mediatypes</type>
  <company_id>DS</company_id>
</marathon>
```

Reply:

```
<collective_mediatype id="ON" name="Online" />
<collective_mediatype id="PR" name="Print" />
<collective_mediatype id="TV" name="TV" />
```

get_mediatypes

Input: collective_mediatype_id (optional)

Output: mediatypes

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>get_mediatypes</type>
  <company_id>DS</company_id>
  <collective_mediatype_id></collective_mediatype_id>
</marathon>
```

Reply:

```
<mediatype id="22" name="Online Display" collective_mediatype_id="ON" />
<mediatype id="23" name="TV" collective_mediatype_id="TV" />
```

get_medias

Input: filtering on media name, media type, collective media type and country

Output: active medias

Filtering can be done on the name with the field filter_media_name. Filtering can be done with a wild card (*) in the beginning and at the end of the filter string.

Filtering can also be done on the media type code, collective media type code and country code with the fields filter_media_type, filter_collective_media_type and filter_country. More than one code can be filtered resulting in all medias with any of the codes returned. When filtering on more than one code the codes should be delimited by comma or space.

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>get_medias</type>
  <company_id>DS</company_id>
  <filter_media_name>dagens*</filter_media_name>
  <filter_media_type>01,02</filter_media_type>
  <filter_collective_media_type>PRIN</filter_collective_media_type>
  <filter_country>SE</filter_country>
</marathon>
```

Reply:

```
<media id="DN" name="Dagens Nyheter">
  <publisher_name>Bonnier</publisher_name>
  <commission type="FORM" calc="pct">4.08</commission>
  <commission type="INFO" calc="pct">1.00</commission>
</media>
```

get_discount_codes

Input: -

Output: all active discount codes

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>get_discount_codes</type>
  <company_id>DS</company_id>
</marathon>
```

Reply:

```
<discount_code id="SP" name="Special discount" />
<discount_code id="VO" name="Volume discount" />
```

get_surcharge_codes

Input: -

Output: active surcharge codes

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>get_surcharge_codes</type>
  <company_id>DS</company_id>
</marathon>
```

Reply:

```
<surcharge_code id="100" name="Colour" />
<surcharge_code id="101" name="Delivery" />
```

get_units

Input: -

Output: active unit codes

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>get_units</type>
  <company_id>DS</company_id>
</marathon>
```

Reply:

```
<unit id="MD" name="MD" />
<unit id="CP" name="CPM" />
```

get_campaign

Input: campaign id

Output: campaign data

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>get_campaign</type>
  <company_id>DS</company_id>
  <campaign_id>10001</campaign_id>
</marathon>
```

Reply:

```
<campaign_id>10001</campaign_id>
<name>Test campaign</name>
<PO_number>1234</PO_number>
<max_ctc>1000000</max_ctc>
<client_id>VOLV</client_id>
<active>true</active>
<plan_id>001001</plan_id>
<plan_id>001002</plan_id>
<project_id>VOLV0001</project_id>
<project_id>VOLV0002</project_id>
```

get_campaigns

Input: client id

Output: list of campaigns

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>get_campaigns</type>
  <company_id>DS</company_id>
  <client_id>VOLV</client_id>
</marathon>
```

Reply:

```
<campaign_id>10001</campaign_id>
<campaign_id>10002</campaign_id>
```


create_campaign

Input: campaign data

Output: campaign id

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>create_campaign</type>
  <company_id>DS</company_id>
  <campaign_id></campaign_id>
  <name>Test campaign</name>
  <PO_number>1234</PO_number>
  <max_ctc>1000000</max_ctc>
  <client_id>VOLV</client_id>
  <active>true</active>
</marathon>
```

Reply:

```
<campaign_id>10001</campaign_id>
```

get_plan

Input: plan id

Output: plan details

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>get_plan</type>
  <company_id>DS</company_id>
  <plan_number>2001</plan_number>
  <include_cancelled>>false</include_cancelled>
</marathon>
```

get_changed_plans

Input: date

Output: list of plans

This file requests a list of changed plans and orders since a specific date.

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>get_changed_plans</type>
  <company_id>DS</company_id>
  <from>2016-10-01</from>
</marathon>
```

Returned fields:

```
<plan>
  <plan_id>002489</plan_id>
  <plan_changed>true</plan_changed>
  <order>
    <order_id>103765</order_id>
    <order_changed>true</order_changed>
  </order>
</plan>
```

create_plan

Input: plan data

Output: plan id

The following file creates a new plan. By providing a plan number a present plan can be changed.

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>create_plan</type>
  <company_id>DS</company_id>
  <plan_id></plan_id>
  <name>Test plan</name>
  <cuid>123456</cuid>
  <PO_number>1234</PO_number>
  <PO_number_same>J</PO_number_same>
  <campaign_id>10001</campaign_id>
  <client_id>VOLV</client_id>
  <product_id>VOLV</product_id>
  <agreement_id>VOLV-1</agreement_id>
  <owner_id>GW</owner_id>
  <comment>Comment 1</comment>
  <comment>Comment 2</comment>
  <comment>Comment 3</comment>
  <comment>Comment 4</comment>
  <comment>Comment 5</comment>
</marathon>
```

Reply:

```
<plan_id>10001</plan_id>
```

scratch_plan

Input: plan id

Output: -

This function deletes all orders in an active media plan with only preliminary orders.

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>scratch_plan</type>
  <company_id>DS</company_id>
  <plan_number>2001</plan_number>
</marathon>
```

get_order

Input: order id

Output: order details

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>get_order</type>
  <company_id>DS</company_id>
  <order_number>2001</order_number>
</marathon>
```

get_orders

Input: client, media and/or insertion period

Output: list of orders

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>get_orders</type>
  <company_id>DS</company_id>
  <client_id>VOLV</client_id>
  <media_id>DAIN</media_id>
  <from_insertion_date>2017-01-01</from_insertion_date>
  <to_insertion_date>2017-12-31</to_insertion_date>
</marathon>
```

Reply:

```
<order_number>2001</order_number>
<order_number>2011</order_number>
```

create_order

Input: order data

Output: -

The order is placed in the tab Order Import in Media/Plans.

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>create_order</type>
  <company_id>DS</company_id>
  <external_system>Adform</external_system>
  <external_order_number>1234</external_order_number>
  <media_id>DN</media_id>
  <section_id>xxx</section_id>
  <placement_id>20100101-001154</placement_id>
  <placement_name>Motor</placement_name>
  <client_id>VOLV</client_id>
  <product_id>C</product_id>
  <agreement_id>VOLV-1</agreement_id>
  <client_reference>PO1234</client_reference>
  <client_contact>Sune Larsson</client_contact>
  <plan_number>1001</plan_number>
  <plan_name>New cars</plan_name>
  <cuid>123456</cuid>
  <status>P</status>
  <colour>0</colour>
  <unit>MD</unit>
  <column>2</column>
  <height>100</height>
  <module>22B</module>
  <format></format>
  <ad_type></ad_type>
  <comment></comment>
  <currency>DKK</currency>
  <agency_ctc>900</agency_ctc>
  <insertion>
    <insertion_date>2010-10-20</insertion_date>
    <end_date></end_date>
    <client_reference>1234</client_reference>
    <price_row>
      <price_code>000</price_code>
      <number_of_units>2500000</number_of_units>
      <gross>1000</gross>
      <discount>
        <code>FR</code>
        <percent>10,5</percent>
        <amount>105</amount>
      </discount>
    </price_row>
  </insertion>
</marathon>
```



```
<comment>This is a comment!</comment>
</price_row>
</insertion>
</marathon>
```

Occurs 1 and 1- mean that the field is mandatory if the "block" that contains the field is present.

If more characters than the specified data type can hold is sent the data will be truncated.

If gross is not available, gross can be set to net and the discounts can be omitted.

Agency commission (byråprovision, informationsersättning mm) is automatically picked up from the media register when the order is created and should therefore not be considered a discount.

When creating an order from an entry in the order queue a callback file is created in XML/EXOR/<external system>". This callback file can be sent to the external system by a separate process. Please contact Kalin Setterberg for configuration of this.

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <type>callback_order</type>
  <company_id>DS</company_id>
  <external_system>Adform</external_system>
  <external_order_number>1234</external_order_number>
  <marathon_order_number>100100</marathon_order_number>
  <marathon_budget_number>20200</marathon_budget_number>
  <net>1000.00</net>
  <ctc>1030.00</ctc>
</marathon>
```

Field name	Occurs	Data type	Value (if used)	Comment
password	1	10A	A valid password	
type	1	25A	"create_order"	
company_id	1	4A	A valid company id	
external_system	1	10A	"Admaker", "Adform", "Trinity", "GMInternal", "Mediamind" or "Unicorn".	
external_order_number	0-1	30A	Free text	
media_id	0-1	10A	A valid media id in one of three formats: 1. Local media id (4 char). 2. Media id in media database (6 char). 3. "company id-local media id".	Manually entered when creating the order in Marathon if not present.
section_id	0-1	6A	A valid section id.	
placement_id	0-1	15A	A valid placement id in the media database	
placement_name	0-1	100A		
client_id	0-1	4A	A valid client id	Manually entered when creating the order in Marathon if not present.
product_id	0-1	4A	A valid product id	Manually entered when creating the order in Marathon if not present.
agreement_id	0-1	8A	A valid agreement id	Manually entered when creating the order in Marathon if not present.
client_reference	0-1	50A		
client_contact	0-1	50A		
plan_number	0-1	6N	A valid plan number	Manually entered when creating the order in

				Marathon if not present.
plan_name	0-1	50A	Free text	Manually entered when creating the order in Marathon if not present.
cuid	0-1	50A	Free text	
status	0-1	1A	D, P	Defaults to P if not present.
colour	0-1	1N	0, 1, 2 or 3	
unit	0-1	2A	MM, MD or SA. For Internet IM, CL, AC, FP, DA, MO, WE can be used.	Retrieved from the media database if not present
column	0-1	3N		
height	0-1	3N		
module	0-1	3A	A module in the format 99 or 99B	Cannot be used together with format
format	0-1	30A		Cannot be used together with module
ad_type	0-1	25A	Free text	Used by online
comment	0-1	280A		The field is splitted into 4 fields with 70 characters
currency	1	3A	A valid currency id (in Marathon).	
agency_ctc	0-1	9N+2 dec		
insertion	1-			
insertion_date	1	AAAA-MM-DD	A valid date	
end_date	0-1	AAAA-MM-DD	A valid date	
client_reference	0-1	50A		
price_row	0-			
price_code	0-1	3A	A valid price code id or "000"	
number_of_units	0-1	9N		
gross	1	9N		Retrieved from the media database if not present.
discount	0-4			The yearly discounts in Marathon is added to the order in addition to the discounts provided.
code	1	2A or 4A*	A valid discount	

			id	
percent	1	3N+4dec		
amount	1	9N		
comment	0-3	50A	Comment shown on the invoice to the client	

* If the source system cannot handle discount codes with national characters character 3 and 4 can be used to express the national characters. The character 3 is used for character 1 and character 4 is used for character 2. As an example FÅ can be expressed as FA_O. The O "converts" A to Å. The following combinations are handled: AO=Å, AE=Ä, OE=Ö, AX=Æ, OX=Ø.

change_order

Input: external system, external order number and status

Output: -

This function is used to change status of an order created with the API function create_order. It is only possible to change status if the fields external_system and external_order_number matches exactly one order.

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>change_order</type>
  <company_id>DS</company_id>
  <external_system>Adform</external_system>
  <external_order_number>1234</external_order_number>
  <status>D</status>
</marathon>
```

The status for the order will be changed even if it is already created on a media plan. In that case the order is changed both in the order queue and in the media plan.

It is not possible to change status back from definitive (D) to preliminary (P).

change_client_status

Input: order id and client status

Output: -

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>change_client_status</type>
  <company_id>DS</company_id>
  <order_id>10001</order_id>
  <client_status>B</client_status>
</marathon>
```

create_order_direct

Input: order data

Output: order id

This function is used to create an order directly on a specific plan. The fields are much the same as in create_order. See description above. The main difference is that fields that only are stored in the plan cannot be used with this function. The fields that can be used are shown below in the example.

Unlike create_order a present order can be changed by providing an order number in order_id. If an invoiced price row is changed the difference is put in a new price row.

If no commission fields are present the commission on the media is used (like in create_order).

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>create_order_direct</type>
  <company_id>DS</company_id>
  <order_id></order_id>
  <plan_number>1001</plan_number>
  <product_id>C</product_id>
  <external_order_number>1234</external_order_number>
  <PO_number>76812</PO_number>
  <cuid>123456</cuid>
  <media_id>DN</media_id>
  <section_id>1</section_id>
  <headline>Annonsrubrik</headline>
  <placement_id>20100101-001154</placement_id>
  <placement_name>Motor</placement_name>
  <unit>MD</unit>
  <status>P</status>
  <agreement_id>VOLV-1</agreement_id>
  <colour>0</colour>
  <column>2</column>
  <height>100</height>
  <format></format>
  <ad_type></ad_type>
  <comment>Order comment</comment>
  <currency>DKK</currency>
  <agency_ctc>900</agency_ctc>
  <insertion>
    <insertion_date>2016-11-20</insertion_date>
    <end_date></end_date>
    <PO_number>1234</PO_number>
    <price_row>
      <price_code>000</price_code>
      <number_of_units>2500000</number_of_units>
    </price_row>
  </insertion>
</marathon>
```

```
<gross>1000</gross>
<discount>
  <code>FR</code>
  <percent>10,5</percent>
  <amount>105</amount>
</discount>
<commission type="INFO" calc="pct">1.00</commission>
<commission type="BETG" calc="pct">1.00</commission>
<commission_calculation_order>1</commission_calculation_order>
<comment>This is a comment!</comment>
</price_row>
</insertion>
</marathon>
```


delete_order

Input: order id

Output: -

The function is used to delete an order. An invoiced order cannot be deleted.

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>delete_order</type>
  <company_id>DS</company_id>
  <order_id>10001</order_id>
</marathon>
```

get_invoice

Input: invoice number

Output: pdf

This file gets the pdf of a client invoice created in the media system.

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>get_invoice</type>
  <company_id>DS</company_id>
  <invoice_number>1001</invoice_number>
  <watermark_copy>true</watermark_copy>
</marathon>
```

Media database functions

get_placements

Input: media id and insertion date

Output: placements

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>get_placements</type>
  <company_id>DS</company_id>
  <media_id>DN</media_id>
  <insertion_date>2010-10-10</insertion_date>
</marathon>
```

media_id can be specified in the following three formats.

1. Local media id (4 char).
2. Media id in media database (6 char).
3. "company id"-"local media id".

Reply (three examples):

```
<placement id="20100101-001154" name="Motor" weekday="MTOTF-S" unit="MM" />
<placement id="20100101-000125" name="Textsida" weekday="MTOT---" unit="SA"
size="1/1" />
<placement id="20100101-000067" name="Textsida" weekday="MTOT---" unit="MD" />
```

get_insertion_dates

Input: media, month and placement

Output: valid insertion dates

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>get_insertion_dates</type>
  <company_id>DS</company_id>
  <media_id>DN</media_id>
  <placement_id>20100101-001154</placement_id>
  <year>2010</year>
  <month>10</month>
</marathon>
```

media_id can be specified in the following three formats.

1. Local media id (4 char).
2. Media id in media database (6 char).
3. "company id"-"local media id".

Reply:

```
<insertion_date insertion_date="2010-10-01" material_date="2010-09-29 16:00"
order_date="2010-09-28 12:00" />
<insertion_date insertion_date="2010-10-03" material_date="2010-09-30 12:00"
order_date="2010-09-29 12:00" />
```

get_sizes

Input: media and placement

Output: valid sizes

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>get_sizes</type>
  <company_id>DS</company_id>
  <media_id>DN</media_id>
  <placement_id>20100101-001154</placement_id>
</marathon>
```

media_id can be specified in the following three formats.

1. Local media id (4 char).
2. Media id in media database (6 char).
3. "company id"-"local media id".

Reply (three examples):

```
<size unit="MM" columns="1" width="29" height_min="5" height_max="360" />
<size unit="MM" columns="2" width="61" height_min="5" height_max="360" />
```

```
<size unit="SA" name="1/1" width="300" height_min="400" height_max="400"
width_bleed="336" height_bleed="420" extra_bleed="4" />
```

```
<size unit="MD" name="22" width="80" height_min="80" height_max="80" />
<size unit="MD" name="22B" width="80" height_min="60" height_max="60" />
```

get_price

Input: media, placement, client, ad size and insertion date.

Output: smallest size that fits the ad and price

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>get_price</type>
  <company_id>DS</company_id>
  <media_id>DN</media_id>
  <placement_id>20100101-001154</placement_id>
  <client_id>VOLV</client_id>
  <colour>0</colour>
  <width>50</width>
  <height>100</height>
  <insertion_date>2010-10-20</insertion_date>
</marathon>
```

media_id can be specified in the following three formats.

1. Local media id (4 char).
2. Media id in media database (6 char).
3. "company id"-"local media id".

colour=0 is b/w, 1 is black+1 colour, 2 is black+2 colours and 3 is full colour.

width and height must be specified if unit=MM or MD on the placement. The system chooses the cheapest ad where the given size fits. If unit=MM the number of columns and the height is returned in the fields column and height. If unit=MD the module code is returned in the field module.

Reply:

```
<column>2</column>
<height>100</height>
<module>22B</module>
<price currency="SEK" gross="8532" discount="-853" commission="-154" fee="230"
insertion_fee="30" ctc="7785 />
<agreement_identified>true</agreement_identified>
```

The above is a non valid example. If unit=MM only column and height is returned. If unit=MD only module is returned.

Project functions

get_proclients

Input: filtering of client name, timereporting

Output: list of active clients

This function returns active clients in the job system with active projects that are open for time reporting.

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>get_proclients</type>
  <company_id>DS</company_id>
  <filter_client_name></filter_client_name>
  <timereporting>true</timereporting>
</marathon>
```

Reply:

```
<client id="VOLV" name="Volvo" internal_name="Volvo" />
```

get_project

Input: project id and period for retrieving hours

Output: basic project data

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>get_project</type>
  <company_id>DS</company_id>
  <client_id>VOLV</client_id>
  <project_no>0001</project_no>
  <from_date>2017-12-01</from_date>
  <to_date>2017-12-31</to_date>
</marathon>
```

Reply:

```
<project_name>New cars</project_name>
<project_leader_id>GW</project_leader_id>
<job_type_id>1001</job_type_id>
<fee_total>45.50</fee_total>
<fee_feecode feecode_id="010">10.00</fee_feecode>
<fee_feecode feecode_id="020">35.50</fee_feecode>
<fee_employee employee_id="GUWE">45.50</fee_employee>
```


get_projects

Input: client id and timereporting

Output: list of active projects

This function returns all active projects for the supplied client that are open for time reporting.

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>get_projects</type>
  <company_id>DS</company_id>
  <client_id>VOLV</client_id>
  <timereporting>true</timereporting>
</marathon>
```

Reply:

```
<project id="0001" name="New cars" />
```

create_project

Input: project data

Output: project id

Without project_no a new project is created if automatic project numbers is used.
Otherwise project_no must be present and if the project already exists it is updates.

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>create_project</type>
  <company_id>DS</company_id>
  <client_id>VOLV</client_id>
  <project_no>0001</project_no>
  <project_name>New cars</project_name>
  <project_leader_id>GW</project_leader_id>
  <job_type_id>1001</job_type_id>
</marathon>
```

Reply:

```
<project_no>0001</project_no>
```

get_feecodes

Input: timereporting

Output: list of active fee codes

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>get_feecodes</type>
  <company_id>DS</company_id>
  <timereporting>true</timereporting>
</marathon>
```

Reply:

```
<feecode id="010" name="Analys/Research" />
```

get_employee

Input: employee

Output: first non completely reported day the last two month

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>get_employee</type>
  <company_id>DS</company_id>
  <employee_id>GW</employee_id>
</marathon>
```

Reply:

```
<first_non_complete_date>2018-01-28</first_non_complete_date>
```

get_timereports

Input: employee and period

Output: list of timereports

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>get_timereports</type>
  <company_id>DS</company_id>
  <employee_id>GW</employee_id>
  <from_date>2017-12-01</from_date>
  <to_date>2017-12-31</to_date>
</marathon>
```

Reply:

```
<timereport>
  <employee_id>GW</employee_id>
  <date>2017-12-28</date>
  <client_id>VOL</client_id>
  <project_no>0001</project_no>
  <feecode_id>010</feecode_id>
  <hours>2,5</hours>
  <comment>Meeting with Claus</comment>
</timereport>
```

create_timereport

Input: timereport data

Output: -

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>create_timereport</type>
  <company_id>DS</company_id>
  <employee_id>GW</employee_id>
  <date>2017-12-28</date>
  <client_id>VOL</client_id>
  <project_no>0001</project_no>
  <feecode_id>010</feecode_id>
  <hours>2,5</hours>
  <comment>Meeting with Claus</comment>
</marathon>
```

get_proinvoice

Input: invoice number

Output: invoice data (no pdf)

This function returns data about a client invoice created in the job system.

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>get_proinvoice</type>
  <company_id>DS</company_id>
  <invoice_number>1001</invoice_number>
</marathon>
```

Reply:

```
<invoice>
  <invoice_number>1001</invoice_number>
  <invoice_date>2017-09-01</invoice_date>
  <due_date>2017-09-30</due_date>
  <booking_date>2017-09-01</booking_date>
  <client_id>VOLV</client_id>
  <project_no>0001</project_no>
  <base_currency>
    <currency>SEK</currency>
    <fee>2500</fee>
    <purchases>1000</purchases>
    <other>100</other>
    <pre_invoiced>-2000</pre_invoiced>
    <vat>400</vat>
    <total>2000</total>
  </base_currency>
  <invoice_currency>
    <currency>EUR</currency>
    <excl_vat>160</excl_vat>
    <vat>40</vat>
    <total>200</total>
  </invoice_currency>
</invoice>
```

get_proinvoices

Input: client and project

Output: list of invoices

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<marathon>
  <password>xyz</password>
  <type>get_proinvoices</type>
  <company_id>DS</company_id>
  <client_id>VOL</client_id>
  <project_no>0001</project_no>
</marathon>
```

Reply:

```
<invoice_number>1001</invoice_number>
```