v.1.26. July 2024

Booking Manager XML API and Availability Service Description

Booking Manager XML API is a standard SOAP Web service that enables clients to connect their data with external systems and services and to extend the automation benefits of the Booking Manager System.

- It is used by charter agencies to publish availabilities of Charter Operators and automate booking processes by connecting Booking Manager to their own web sites to allow for seamless online booking experience for end users or integrating it with their CRM solutions in order to automate office procedures.
- Charter operators use the web service to connect Booking Manager with their book keeping software and save time and effort in accounting department.
- Third Party Software tools also use Booking Manager API to display richer information, for example boat tracking systems are able to visualise client contact information directly on the map showing the boat location in real time.

Description of the individual Web service methods in this manual is separated into three distinct chapters/categories depending on the particular use case:

- For Charter Agencies and entities wanting to synchronise yacht information and availability, Chapter 1 is relevant. It describes how to retrieve the basic details about Companies in the system, yachts (resources) and connected data about destinations. It also describes a way to create and manipulate reservations in the system directly via Web service calls.
- Chapter 2 describes methods for exporting the invoices produced in Booking Manager System and methods for manipulating and exporting payments. It is used primarily to automate export of accounting data to external Book keeping software.
- Chapter 3 describes all other data manipulation methods such as detailed reservation manipulation, crew list entry, address book access and todo notes manipulation. These methods are useful for plugin applications that communicate with Booking Manager, such as Check-in software or yacht position tracking software tools.

Web service description definition is located at:
https://www.booking-manager.com/cbm_web_service2/services/CBM?wsdl

Table of Contents

1. Availability Retrieval and Manipulation Methods

1.1. getCompanies

```
public String getCompanies(long _userId,String _username,String _password) throws
RemoteException;
```

This function retrieves the list of all the Charter operators in the Booking Manager System who manage their resources (yachts) availability in real time. This list provides the id's of the companies that are available for synchronisation. The method returns a single String with xml response in the following format:

```
<root>
<company id="" name="" address="" city="" zip="" country="" telephone1=""
telephone2="" fax1="" fax2="" mobile1="" mobile2="" vatcode="" email="" web=""
availability="" checkoutnote="">
      <rating average="4.4" reviews="14" />
</company>
</root>
```

Each *company* section holds a record of a single realtime charter fleet with relevant details of the company:

1. *id* - the company id in the system – this information is important for subsequent retreival of fleet and availability using other methods
2. *name* – the legal name of the company
3. *address* – the street name and number
4. *city* – the city
5. *zip* – the zip code of the city
6. *country* - the country name
7. *telephone1* – primary contact telephone number
8. *telephone2* – secondary contact telephone number
9. *fax1* – primary fax
10. *fax2* – secondary fax
11. *mobile1* – primary mobile phone
12. *mobile2* – secondary mobile phone
13. *vatcode* – VAT code of the company
14. *email* – the e-mail address of the company
15. *web* – the web url of the company
16. *availability* – company availability: 0-realtime, 1-periodic, 2-rare

17. *checkoutnote* – checkout note shown on price quote
18. *rating* – company review rating

1.2. getResources

```
public String getResources(long _userId,String _username,String _password,long
_companyId) throws RemoteException;
```

Retrieves all of the yachts available for booking with current characteristics, prices, extras and discounts.

The parameters for the method are:

1. long *userId* – the userId of the connecting party
2. String *username* – the user name of the connecting party
3. String *password* – the password of the connecting party
4. long *companyId* – the id of the requested company, use -1 to retrieve resources for all companies

The result string returns a list of resource details (Resource Details).

1.3. getResourceDetails

```
public String getResourceDetails(long _userId,String _username,String _password,long
_resourceId) throws RemoteException;
```

The result string returns resource details (Resource Details) for a single resource.

1.4. getAvailabilityInfo

```
public String getAvailabilityInfo(long _userId,String _username,String _password,
long _companyId , int _year, boolean _onlyBusy, Date _lastModified) throws
RemoteException;
```

For the retrieval of the booking sheet, a service call to getAvailabilityInfo method is placed. This is the only relevant method that needs to be used for availability retrieval

The parameters for the method are:

5. long *userId* – the userId of the connecting party
6. String *username* – the user name of the connecting party
7. String *password* – the password of the connecting party
8. long *companyId* – the id of the requested company, use -1 to retrieve availability for all companies
9. int *year* – the year for which the booking sheet is being retrieved, use -1 to retrieve availability for current and next year
10. boolean *onlyBusy* – set this to true if you only wish to retrieve the active terms, if set to false it will also pull all of the canceled reservations. If you are using this service for availability retrieval only you will always want to set this field to true .
11. Date *lastModified* – the last retrieval date (or 1.1.1970 for all reservations) – you can use this field to retrieve only changed reservations after last retrieval. This produces faster synchs but is more complex to implement on the connecting party's side. If you don't wish to use this feature and always want to pull all of the bookings, always set this parameter to “1.1.1970”

The method returns a single String with xml response in the following format:

```
<root company_id="" checktime="">
 <reservation id="n" resourceid="n" status="s" blocksavailability="n" datefrom="d"
dateto="d" basefrom="n" baseto="n" optionexpirydate="d" created="d" lastmodified="d"
mybooking=""/>
</root>
```

Root section contains these information:

1. *company_id* – id of the charter company
2. *checktime* – time when request was made

Each "<reservation />" section contains the information about a specific term. All reservations are returned including canceled reservations and current offers that don't block availability field explanation:

1. *id* - unique reservation id
2. *resourceid* - the unique id of the individual yacht
3. *status* – status of the term (Option/Reservation/Owner week/Service/Canceled/Option expired)
4. *blocksavailability* (1/0) – if set to 1 then reservation blocks availability and is written in the booking sheet. For availability synchronizations only the terms with blocksavailability set to "1" are significant
5. *datefrom* – date of the checkin in format "yyyy-dd-mm"
6. *dateto* – date of the checkout in format "yyyy-dd-mm"
7. *basefrom* – the unique id of the check in base
8. *baseto* – the unique id of the checkout base
9. *optionexpirydate* – the unix timestamp of the expiration date for the option created – the unix timestamp of the creation date of the reservation
10. *lastmodified* – the unix timestamp of the last reservation change time
11. *mybooking* – true if reservation belongs to the company which made the request

1.5. getShortAvailabilityInfo

```
public String getShortAvailabilityInfo(long _userId,String _username,String
_password,long _companyId,int _year,int _resultTypeId) throws RemoteException;
```

Retrieves availability information in short format:

1. long *userId* – the userId of the connecting party
1. String *username* – the user name of the connecting party
2. String *password* – the password of the connecting party
3. long companyId – the id of the requested company. Can be set to "-1" to retrieve availability from all companies but it will take longer and response will be larger in size.
4. int *year* – the year for which the booking sheet is being retrieved
5. int *resultTypeId* – type of result that will be returned
    1. 1 – Binary
    2. 2 – Hex
    3. 3 - Status

Result types

Binary – use "1" to get results in binary format

```
<root year="" resulttypeid="" companyid="">
```

```
<![CDATA[
resourceId:availabilityInfo;resourceId:availabilityInfo; ...
]]>
</root>
```

Response contains availability information for yachts. Availabilities for multiple yachts are separated by ";". Availability information for each yacht contains resourceId and availabilityInfo separated by ":".

resourceId – id of the resource

availabilityInfo – availability information for the resource in binary format. Each availabilityInfo is 365 characters long (or 366 in case of the leap year) and it represents the whole year (single character is one day of the year). First character is January 1st, second character is January 2nd, and so on. If character is equal to "0" it means that yacht is available on that day, otherwise character is equal to "1".

Hex – use "2" to get results in hexadecimal format

```
<root year="" resulttypeid="">
<![CDATA[
resourceId:availabilityInfo;resourceId:availabilityInfo; ...
]]>
</root>
```

Response contains availability information for yachts. Availabilities for multiple yachts are separated by ";". Availability information for each yacht contains resourceId and availabilityInfo separated by ":".

resourceId – id of the resource

availabilityInfo – availability information for the resource in hexadecimal format. Each availabilityInfo is 92 characters long and it represents the whole year. First character is January 1, January 2, January 3 and January 4 , second character is January 5, January 6, January 7 and January 8, and so on. Last characters should be discarded depending on how long the requested year is.

For example, if availabilityInfo is "fe03f0..." in binary it is "0000 0001 1111 1100 0000 1111..." and it means that yacht is available from January 1 until January 8, but it is not available from January 8 until January 15, and so on.

Statuses – use "3" to get results in binary format

```
<root year="" resulttypeid="" companyid="">
<![CDATA[
resourceId:availabilityInfo;resourceId:availabilityInfo; ...
]]>
</root>
```

Response is the same as the Binary response with one difference – booked days are not represented with character "1" for all types of reservations but instead reservation status id is used. For example, if availabilityInfo is "2222 2220 0000..." it means that boat is under status Option from January 1 until January 8, and it is free from January 8 etc.

Status codes are:

0 – Available

1 – Reservation

2 – Option

3 – Option in expiration

4 – Service

B – Sleepaboard (for charters)

1.6. getSearchResultsFilter

```
getSearchResultsFilter(long _userId, String _username, String _password,String
_xmlValues)
```

Performs the availability search on the database. It is possible to retrieve all resources that meet the set parameters with their availability status, price and discount data.

The parameters for the method are:

1. long *userId* – the userId of the connecting party
2. String *username* – the user name of the connecting party
3. String *password* – the password of the connecting party
4. String *xmlValues* – xml string with filter parameters

All filter parameters, if used, are contained in _xmlValues string.

```
<filter>
 <element id="datefrom"></element>
 <element id="dateto"></element>
 <element id="flexibility"></element>
 <element id="companyids"></element>
 <element id="countrycodes"></element>
 <element id="baseids"></element>
 <element id="resourceids"></element>
 <element id="resourcetypeids"></element>
 <element id="product"></element>
 <element id="kind"></element>
</filter>
```

1. Date *datefrom* – start of request period in SOAP format (see Date formats). This parameter is obligatory.
2. Date *dateto* – end of request period in SOAP format (see Date formats). This parameter is obligatory.
3. long flexibility – expand search outside of the request period
    1. 1 – on day (default)
    2. 2 – in week
    3. 3 – one week before or after
    4. 4 – two weeks before or after
    5. 5 – in month
    6. 6 – in year

    If flexibility is used only datefrom is relevant (leave dateto at datefrom+7 days)
4. String companyids – filter one or multiple companies separated by comma (e.g. 123,456)
5. String countrycodes – filter one or multiple countries separated by comma (e.g. HR,IT)
6. String baseids – filter one or multiple bases separated by comma
7. String resourceids – filter one or multiple yachts separated by comma
8. String resourcetypeids – filter one or multiple yacht models separated by comma
9. String currency – sets currency for baseprice
10. String product – sets yacht product for the filter (e.g. Bareboat)
11. String kind – sets yacht kind for the filter (e.g. Sail boat)

```
<root>
<resource resourceid="" companyid="" reservationstatus="" baseprice=""
discountvalue="" price="" baseid="" datefrom="" dateto=""></resource>
…
</root>
```

The results are returned as list of <resource></resource> elements. Attributes are:

1. *resourceid* – the id of the retrieved resource / yacht
2. *companyid* – the id of the company that rents the yacht
3. *reservationstatus* – the status of availability for this yacht. This can be:
    1. 0 – Free – yacht is available
    2. 1 – Fixed reservation – the term is booked
    3. 2 – Option – the term is under option
    4. 3 – Option Expired – the term is under option that is about to expire
    5. 4 – Service – The yacht is under service or not available for some other reason
4. *baseprice* – the price for the term in EUR by default. If currency is set in request filter, price will be in the specified currency (without the discount)
5. *discountvalue* – the discount value for the term (in %)
6. *price* – the price with the discount applied
7. *baseid* – the id of the base where resource is available
8. *basetoid* – the id os the base where resource will end up
9. *datefrom* – start date of the requested period
10. *dateto* – end date of the requested period
11. *commissionpercentage* – agency commission percentage
12. *commissionvalue* – value of agency commission
13. *deposit* – deposit
14. *obligatoryextras* – total value obligatory extras

1.7. getSpecialOffers

```
getSpecialOffers(long _userId, String _username, String _password,String _type, Date
_dateFrom, Date _date, String _xmlValues)
```

Retrieves special offers from the database.

The parameters for the method are:

1. long *userId* – the userId of the connecting party
1. String *username* – the user name of the connecting party
2. String *password* – the password of the connecting party
3. String *type* – type of the requested special offer:
    1. 1 – short term offers
    2. 2 – one way offers
4. Date *date* – retrieve special offers for this date. System will return all the special offers around this date +/- two weeks.
5. String *xmlValues* – additional filtering parameters:
    1. companyid – filter results by company id

2. flexibility – filter by flexibility
    1. 1 – on day
    2. 2 – in week
    3. 3 – one week before or after
    4. 4 – two weeks before or after (default)
    5. 5 – in month
    6. 6 – in year

Example of the xmlValues parameter:

```
<values>
      <element id=”companyid”>123</element>
      …
</values>
```

Response format:

```
<root>
<resource resourceid="n" companyid=”” reservationstatus="" baseprice=""
discountvalue="" price="" baseid=”” datefrom=”” dateto=””></resource>
…
</root>
```

The results are returned as list of <resource></resource> elements. Attributes are:

1. *resourceid* – the id of the retrieved resource / yacht
2. *companyid* – the id of the company that rents the yacht
3. *reservationstatus* – the status of availability for this yacht. This can be:
    1. 0 – Free – yacht is available
    2. 1 – Fixed reservation – the term is booked
    3. 2 – Option – the term is under option
    4. 3 – Option Expired – the term is under option that is about to expire
    5. 4 – Service – The yacht is under service or not available for some other reason
4. *baseprice* – the price for the term in EUR (without the discount)
5. *discountvalue* – the discount value for the term (in %)
6. *price* – the price with the discount applied
7. *baseid* – the id of the base where resource is available
8. *datefrom* – start date of the requested period
9. *dateto* – end date of the requested period

1.8. isResourceAvailable

```
public String isResourceAvailable(long _userId,String _username,String _password,long
_resourceId,Date _dateFrom,Date _dateTo) throws RemoteException;
```

Checks if yacht is available in selected period.

1. long *userId* – the userId of the connecting party
1. String *username* – the user name of the connecting party
2. String *password* – the password of the connecting party
3. long resource*Id* – id of the resource
4. Date *dateFrom* – start date of the requested period

5. Date *dateTo* – end date of the requested period

```
<resource id="" isavailable="" status=""/>
```

Result contains id of the resource and its availability.

1. *id* – id of the resource
1. *isavailable* – availability of the resource, 1 if available and 0 otherwise
2. *status* – status of the booking if yacht is reserved, 0 if it is free

1.9. getResourceDiscount

```
public String getResourceDiscount(long _userId,String _username,String _password,long
_resourceId,Date _dateFrom,Date _dateTo) throws RemoteException;
```

Calculates total discount percentage for a particular resource in given date period.

1. long *userId* – the userId of the connecting party
1. String *username* – the user name of the connecting party
2. String *password* – the password of the connecting party
3. long *resourceId* – id of the resource
4. Date *dateFrom* – start date of the requested period
5. Date *dateTo* – end date of the requested period

```
<resource id="" discount="" price=""/>
```

Result contains id of the resource and discount percentage.

1. *id* – id of the resource
1. *discount* – current discount on the base price in percentages on the base price
2. *price* – final price

1.10. getResourcesDiscount

```
public String getResourcesDiscount(long _userId,String _username,String
_password,String _xmlValues) throws RemoteException;
```

Calculates total discount percentage for requested resources in given date period.

1. long *userId* – the userId of the connecting party
1. String *username* – the user name of the connecting party
2. String *password* – the password of the connecting party
3. String *xmlValues* – xml string with filter parameters

All filter parameters, if used, are contained in _xmlValues string.

```
<filter>
      <element id="resourceids"></element>
      <element id="datefrom"></element>
      <element id="dateto"></element>
</filter>
```

1. String *resourceids* – get discounts for these boats, if it is left empty it will return discounts for all boats
2. Date *datefrom* – start of request period in SOAP format (see Date formats). This parameter is obligatory.
3. Date *dateto* – end of request period in SOAP format (see Date formats). This parameter is obligatory.

```
<root>
<resource id="" discount="" price=""/>
…
</root>
```

Result contains all requested resources with their discount and price.

1. *id* – id of the resource
1. *discount* – current discount on the base price in percentages on the base price
2. *price* – final price

1.11. createReservation

```
public String createReservation(long _userId,String _username,String _password, long
_companyId , Date _dateFrom,Date _dateTo,long _resourceId, String _clientName, int
_passengersOnBoard, long baseFromId, long baseToId) throws RemoteException;
```

createReservation is used to place a new option in the Booking Manager system. The system creates an option with standard discounts and obligatory options applied.

The result string returns a reservation details (Reservation details).

1.12. confirmReservation

```
public String confirmReservation(long _userId,String _username,String _password, long
_reservationId) throws RemoteException;
```

Converts an option into finalized reservation.

The result string returns a reservation details (Reservation details).

1.13. cancelReservation

```
public String cancelReservation(long _userId,String _username,String _password, long
_reservationId) throws RemoteException;
```

Cancels a option. An already confirmed booking is not possible to cancel automatically.

The result string returns a reservation details (Reservation details).

1.14. getReservationDetails

```
public String getReservationDetails(long _userId,String _username,String _password,
long _reservationId) throws RemoteException;
```

The result string returns a reservation details (Reservation details).

1.15. getBases

```
public String getBases(long _userId,String _username,String _password) throws
RemoteException;
```

This method retrieves the available home ports/bases of the resources, with some basic information.

The method receives standard authorization fields:

1. long *userId* – the userId of the connecting party
2. String *username* – the user name of the connecting party
3. String *password* – the password of the connecting party

```
<root>
 <base id="" name="" city="" country="" address="" regionids="" longitude=""
latitude="" countryid="" ></base>
</root>
```

Results are in a form of a <base></base> list with following attributes:

1. *id* – the system id of the base. This is useful to correctly identify the bases of yachts in getResources,getSearchResult and getAvailabilityInfo methods
2. *name* – the name of the base
3. *city* – the city of the base
4. *country* – the country name of the base
5. *address* – the address information for the base
6. *regionids* – identifiers of the regions to which this base belongs to
7. *longitude* – geographical longitude of the destination
8. *latitude* – geographical latitude of the destination
9. *countryid* – the country id

1.16. getRegions

```
public String getRegions(long _userId,String _username,String _password) throws
RemoteException;
```

This method retrieves the available subregions/sailing areas names and id's

The method receives standard authorization fields:

1. long *userId* – the userId of the connecting party
1. String *username* – the user name of the connecting party
2. String *password* – the password of the connecting party

```
<root>
 <region id="" name="" ></region>
</root>
```

Results are in a form of a <region></region> list with following attributes:

1. *id* – the system id of the region
2. *name* – the name of the region

1.17. getCountries

```
public String getCountries(long _userId,String _username,String _password) throws
RemoteException;
```

This method retrieves the available countries with some basic information.

The method receives standard authorization fields:

1. long *userId* – the userId of the connecting party
1. String *username* – the user name of the connecting party
2. String *password* – the password of the connecting party

```
<countries>
  <country id="" name="" shortname="" longshortname="" regionid="">
    <translations>
      <translation lang="" value="" />
    </translations>
  </country>
</countries>
```

Results are in a form of a <country></country> list with following attributes:

1. *id* – the system id of the country
2. *name* – the name of the country
3. *shortname* – representation of the country name with 2 characters
4. *longshortname* – representation of the country name with 3 characters
5. *regionid* – id of the region to which this country belongs

<translations> contains country name translations in all available languages:

1. lang – language
2. value – country name in language

1.18. getEquipmentCategories

```
public String getEquipmentCategories(long _userId,String _username,String _password)
throws RemoteException;
```

Returns a list of all generic equipment.

1. long *userId* – the userId of the connecting party
1. String *username* – the user name of the connecting party
2. String *password* – the password of the connecting party

```
<root>
      <equipmentcategory id="" name=""/>
      ...
</root>
```

The results are returned as list of <equipmentcategory></equipmentcategory> elements. Attributes are:

1. *id* – id of the equipment category
1. *name* – name of the equipment category

getShipyards

```
public String getShipyards(long _userId,String _username,String _password) throws
RemoteException;
```

Returns a list of all shipyards.

1. long *userId* – the userId of the connecting party
1. String *username* – the user name of the connecting party
2. String *password* – the password of the connecting party

```
<root>
      <shipyard id="" name="" shortname=""/>
      …
</root>
```

The results are returned as list of <shipyard></shipyard> elements. Attributes are:

1. *id* – id of the shipyard
1. *name* – name of the shipyard
2. *shortname* – short name of the shipyard

2. Invoice Retrieval Methods

2.1. getInvoices

```
public String getInvoices(long userId,String username,String password,int
filterInvoiceType,Date _dateFrom,Date _dateTo)
```

This method is used to fetch a list of all issued invoices in some period. It is possible to filter by date periods, and types of invoice.

The parameters for the method are:

1. long *userId* – the userId of the connecting party
2. String *username* – the user name of the connecting party
3. String *password* – the password of the connecting party
4. int *filterInvoiceType* – the type of invoice that is being fetched
    1. 0 – all invoices
    2. 1 – final invoices
    3. 2 – invoices for advance payments
    4. 3 – storno invoices
1. *datefrom* – start date of the invoicing period
2. *dateto* – end date of the invoicing period

The method returns a single String with xml response in the following format:

```
<root>
 <invoice type="" number="" reservationnumber="" date="" client="" clientcode=""
clientvatcode="" clientid="" guestname="" currency="" exchangerate="" altcurrency=""
altexchangerate="" resource="" resourcetype="" resourcecode="" totalprice=""
totalpricewithouttax="" rate="" totalaltprice="" totalaltpricewithouttax="-"
basefrom="" baseto="" alreadytransferred="" servicedatefrom="" servicedateto=""
paymentmethodname="" paymentmethodtype="" agencyid="" agencyname="" agencycode=""
agencyvatcode="" relatedreservationid="" reservationid="" relatedinvoicenumber="" >

 <vat><item base="" rate="" total="" basealt="" totalalt="" /></vat>
 <services><service name="" total="" rate="" code=""/></services>
 <bi><item name="" value=""/></bi>
</invoice>
</root>
```

Each “<invoice />” section contains the following information about a specific invoice:

3. *type* – the type of the invoice
4. *number* – the number of the invoice
5. *reservationnumber* – the number of the reservation
6. *date* – date of the invoice in format “yyyy-dd-mm”

7. *client* – the name of the client
8. *clientcode* – the code of the client in the addressbook
9. *clientvatcode* – the VAT code of the client
10. *clientid* – the unique id of the client in the Booking Manager database
11. *guestname* – the name of the guest that is traveling
12. *currency* – the currency that the invoice is being issued in
13. *exchangerate* – the exchange rate of the currency on the invoice comparet to EUR
14. *altcurrency* – the alternative currency (usually used to show also the value in domestic currency if different from invoiceing currency)
15. *altexchangerate* – exchange rate of the alternative currency
16. *resource* – the name of the resource (boat) that is invoiced
17. *resourcetype* – the name of the model
18. *resourcecode* – the code of the boat in the yacht editor
19. *totalprice* – the total price of the invoice
20. *totalaltpricewithouttax* – the base price without tax
21. *rate* – tax rate
22. *totalaltprice* – total price in alternative currency
23. *totalaltpricewithouttax* – total price in alternative currency withour tax
24. *basefrom* – base where boat is boarding (check in base)
25. *baseto* – base where boat is disembarking (checkout base)
26. *alreadytransferred* – Flag to mark if this particular invoice was already pulled.
27. *servicedatefrom* – checkin date for the reservations
28. *servicedateto* – checkout date for the reservation
29. *paymentmethodname* – name of the payment method used on the invoice
30. *paymentmethodtype* – type of the transaction (Other, Cash, Card, Check, Transaction account)
31. *agencyid* – id of the agent attached to the reservation
32. *agencyname* – full name of the agent attached to the reservation
33. *agencycode* – code id of the agent
34. *agencyvatcode* – vat code of the agent
35. *relatedreservationid* – id of the related reservation, this parameter does not exist if there is no related reservation
36. *reservationid* – invoice is issued on the reservation identified by this id
37. relatedinvoicenumber – Shows the invoice number of connected advance invoice for a storno invoice. This item is non mandatory and appears only on storno invoices (type 3) which represent a direct cancelation of a specific advance invoice.

Each <invoice /> section also may contain a <vat /> section where all the items in the invoice are separated by tax rate, which is important for bookkeeping purposes, items are as follows:

1. *base* – the base price without tax
2. *rate* – the tax rate
3. *total* – total price with tax
4. *basealt* – base price in alternative currency

5. *totalalt* – total price in alternative currency

Each <invoice/> section may contain a <services/> section which contains all services regarding the invoice. Each <service/> item has these attributes:

1. *name* – name of the service
2. *total* – service price
3. *rate* – service tax rate
4. *code* – service code

Each<invoice/> contains a <bi/> section with <item/> which contains all useful information for business intelligence calculations. Each <item/> has these attributes:

1. *name* – name of the item
1. *value* – value of the item

Items with these names are sent in the <bi/> section:

1. *commission* – agency commission and corresponding amount
2. *profit* – charter profit based on charged amount and agency commission

2.2. getPayment

```
public String getPayment(long userId,String username,String password,long _paymentId)
throws RemoteException;
```

This method is used to retrieve a details for certain payment.

The parameters for the method are:

1. long *userId* – the userId of the connecting party
1. String *username* – the user name of the connecting party
2. String *password* – the password of the connecting party
3. long *paymentId* – id of the payment that is being retrieved

The method returns a single String with xml response containing payment details (Payment details).

2.3. insertPayment

```
public String insertPayment(long userId,String username,String password,long
_reservationId,String _xmlValues) throws RemoteException;
```

This method is used to insert payment to certain reservation.

The parameters for the method are:

1. long *userId* – the userId of the connecting party
1. String *username* – the user name of the connecting party
2. String *password* – the password of the connecting party
3. long *reservationId* – id of the reservation on which the payment will be added
4. String *xmlValues* - xml string with payment properties

All properties for the payment are contained in _xmlValues string:

```
<payment>
      <element id="value">1000.0</element>
      <element id="currency">1</element>
      <element id="exchangerate">7.405</element>
      …
</payment>
```

For all available properties please read Changeable payment properties in Appendix.

The method returns a single String with xml response containing payment details (Payment details).

2.4. updatePayment

```
public String updatePayment(long userId,String username,String password,long
_paymentId,String _xmlValues) throws RemoteException;
```

This method is used to update payment.

The parameters for the method are:

1. long *userId* – the userId of the connecting party
1. String *username* – the user name of the connecting party
2. String *password* – the password of the connecting party
3. long *paymentId* – id of the payment that will be updated
4. String *xmlValues* - xml string with payment properties

All properties for the payment are contained in _xmlValues string:

```
<payment>
      <element id="value">1000.0</element>
      <element id="currency">1</element>
      <element id="exchangerate">7.405</element>
      ...
</payment>
```

For all available properties please read Changeable payment properties in Appendix.

The method returns a single String with xml response containing payment details (Payment details).

2.5. deletePayment

```
public String deletePayment(long userId,String username,String password,long
_paymentId) throws RemoteException;
```

This method is used to delete payment.

The parameters for the method are:

1. long *userId* – the userId of the connecting party
1. String *username* – the user name of the connecting party
2. String *password* – the password of the connecting party
3. long *paymentId* – id of the payment that will be deleted

The method returns a single String with xml response containing payment details (Payment details).

2.6. getCashRegister

```
public String getCashRegister(long userId,String username,String password,Date
_dateFrom,Date _dateTo,String _xmlValues) throws RemoteException;
```

This method is used to get cash register view.

The parameters for the method are:

1. long *userId* – the userId of the connecting party
1. String *username* – the user name of the connecting party

2. String *password* – the password of the connecting party
3. Date *dateFrom* – get cash register from this date
4. Date *dateTo* – get cash register until this date
5. String *xmlValues* – xml string with filter parameters

All filter parameters, if used, are contained in _xmlValues string. All of these filters are optional:

```
<filter>
      <element id="showincurrency">EUR</element>
      <element id="currency">HRK</element>
      <element id="methodofpayment">Cash,Card</element>
      <element id="paymentmethodid">123456789</element>
      <element id="operator">123/John Doe</element>
      <element id="officeid">123456789</element>
      <element id="deviceid">1999</element>
</filter>
```

1. String *showincurrency* – all values will be recalculated in this currency (use ISO 4217 currency code)
2. String *currency* – filter invoices that are issued in requested currency (use ISO 4217 currency code)
3. String *methodofpayment* – one of: Other, Cash, Card, Check, Transaction account. You can use multiple methods by concatenating them with comma.
4. long *paymentmethodid* – id of the payment method
5. String *operator* – invoice operator identifier that issued the invoice
6. long *officeid* – id of the office that issued the invoice
7. long *deviceid* – id of the device that issued the invoice

The method returns info about the cash register in requested period:

```
<root lastreceiptnumber="" lastexpensenumber="">
      <item id="" slipid="" paymentmethodtype="" date="" name="BALANCE TRANSFER"
amount="" currencycode="" />
      <item id="" slipid="" paymentmethodtype="" date="" name="" amount=""
currencycode="" />
      ...

</root>
```

1. long lastreceiptnumber – last receipt number after requested period
2. long lastexpensenumber – last expense number after requested period

<item> is a single cash register entry. In this case, first item is the balance transfer from the previous period.

1. long id – numerical number of the item in current listing
2. String slipid – slip number of the item in current year. If it starts with "U" it is receipt, if it starts with "I" it is an expense.
3. String paymentmethodtype – one of: Other, Cash, Card, Check, Transaction account
4. Date date – invoice date
5. String name – name or invoice number
6. float amount – amount in currency
7. String currencycode – ISO 4217 currency code

3. Advanced Data Manipulation

3.1. getStatusTypes

```
public String getStatusTypes(long _userId,String _username,String _password) throws
RemoteException;
```

This method retrieves the available reservation statuses.

The method receives standard authorization fields:

1. long *userId* – the userId of the connecting party
1. String *username* – the user name of the connecting party
2. String *password* – the password of the connecting party

```
<statustypes>
 <statustype id="" name=""></statustype>
</statustypes>
```

Results are in a form of a <statustype></statustype> list with following attributes:

1. *id* – the system id of the reservation status
2. *name* – the name of reservation status

3.2. getUsers

```
public String getUsers(long _userId,String _username,String _password,long
_lastSyncPoint) throws RemoteException;
```

Returns a list of all users with details. Completely the same as the second getUsers method described below but with no decryption of the data.

3.3. getUsers (2)

```
public String getUsers(long _userId,String _username,String _password,long
_lastSyncPoint,String _encryptionCode) throws RemoteException;
```

Returns users that were changed since _lastSyncPoint and decrypts the data if it is encrypted and if encryption code is provided.

1. long *userId* – the userId of the connecting party
1. String *username* – the user name of the connecting party
2. String *password* – the password of the connecting party
3. long *lastSyncPoint* – point when last synchronization was performed. Use 0 to get all users (this can be a little slow if there is more than 5000 users in your addressbook)
4. String *encryptionCode* – if your data is encrypted system will use this password to decrypt it and return them in response. If encryptionCode is null then it will return encrypted data and you should decrypt it on your side.

```
<users syncpoint="">
<user … >
</user>
...
</users>
```

The results are returned as list of <users></users> elements. Attributes are:

1. *syncpoint* – point when synchronization was performed

The result string returns a list of user details (User details).

3.4. getUserDetails

```
public String getUserDetails(long _userId,String _username,String _password,long
_id,String _encryptionCode) throws RemoteException;
```

The result string returns user details (User details) for a single user.

3.5. insertUser

```
public String insertUser(long _userId,String _username,String _password,String
_xmlValues) throws RemoteException;
```

Inserts new user to your addressbook.

1. long *userId* – the userId of the connecting party
1. String *username* – the user name of the connecting party
2. String *password* – the password of the connecting party
3. String *xmlValues* – xml string with user properties

All properties for the user are contained in _xmlValues string:

```
<user>
      <element id="codeid">123</element>
      <element id="301">John</element>
      <element id="302">Johnson</element>
      ...
</user>
```

For all available properties please read Changeable user properties in Appendix.

3.6. updateUser

```
public String updateUser(long _userId,String _username,String _password,long
_addressBookUserId,String _xmlValues) throws RemoteException;
```

Updates user from the addressbook.

1. long *userId* – the userId of the connecting party
1. String *username* – the user name of the connecting party
2. String *password* – the password of the connecting party
3. long *addressBookUserId* – id of the user to update
4. String *xmlValues* – xml string with user properties

All properties for the user are contained in _xmlValues string:

```
<user>
      <element id="codeid">123</element>
      <element id="301">John</element>
      <element id="302">Johnson</element>
      ...
</user>
```

For all available properties please read Changeable user properties in Appendix.

3.7. insertCrewMember

```
public String insertCrewMember(long _userId,String _username,String _password,long
_reservationId,long _addressBookUserId,boolean _isSkipper) throws RemoteException;
```

Inserts user from the addressbook to the crew list on specified reservation.

1. long *userId* – the userId of the connecting party
1. String *username* – the user name of the connecting party
2. String *password* – the password of the connecting party
3. long *reservationId* – id of the reservation to which the crew member will be added
4. long *addressBookUserId* – id of the user that will be added as a crew member
5. boolean *isSkipper* – 1 if it is a skipper and 0 if it is a crew member (guest)

The method returns details for newly inserted crew member (Crew member details).

3.8. updateCrewMember

```
public String updateCrewMember(long _userId,String _username,String _password,long
_reservationId,long _crewMemberId,String _xmlValues) throws RemoteException;
```

Updates crew member details on certain reservation.

1. long *userId* – the userId of the connecting party
1. String *username* – the user name of the connecting party
2. String *password* – the password of the connecting party
3. long *reservationId* – id of the reservation on which the crew member will be updated
4. long *crewMemberId* – id of the crew member to update
5. String *xmlValues* – xml string with crew member properties

All properties for the crew member are contained in _xmlValues string:

```
<crewmember>
      <element id="1303">John</element>
      <element id="1304">Johnson</element>
      ...
</crewmember>
```

For all available properties please read Changeable crewmember properties in Appendix.

The method returns details for updated crew member (Crew member details).

3.9. deleteCrewMember

```
public String deleteCrewMember(long _userId,String _username,String _password,long
_reservationId,long _crewMemberId) throws RemoteException;
```

Deletes crew member from certain reservation.

1. long *userId* – the userId of the connecting party
1. String *username* – the user name of the connecting party
2. String *password* – the password of the connecting party
3. long *reservationId* – id of the reservation from which the crew member will be deleted
4. long *crewMemberId* – id of the crew member to delete

3.10. getAllOptionItems

```
public String getAllOptionItems(long _userId,String _username,String _password,long
_companyId) throws RemoteException;
```

Retrieves all available option items for certain company.

1. long *userId* – the userId of the connecting party
1. String *username* – the user name of the connecting party
2. String *password* – the password of the connecting party
3. long *companyId* – id of the company whose option items you would like to retrieve

```
<optionitems>
      <optionitem id="" name="" price=””/>
      ...
</optionitems>
```

The results are returned as list of <optionitem></optionitem> elements. Attributes are:

1. *id* – id of the option item
2. *name* – name of the option item
3. *price* – price of the option item

3.11. getAllDiscountItems

```
public String getAllDiscountItems(long _userId,String _username,String _password,long
_companyId) throws RemoteException;
```

Retrieves all available discount items for certain company.

1. long *userId* – the userId of the connecting party
1. String *username* – the user name of the connecting party
2. String *password* – the password of the connecting party
3. long *companyId* – id of the company whose option items you would like to retrieve

```
<discountitems>
      <discountitem id="" name="" value=””/x`>
      ...
</discountitems>
```

The results are returned as list of <discountitem></discountitem> elements. Attributes are:

1. *id* – id of the discount item
2. *name* – name of the discount item
3. *value* – value of the discount item

3.12. insertOptionItem

```
public String insertOptionItem(long _userId,String _username,String _password,long
_reservationId,long _optionItemId,float _amount) throws RemoteException;
```

Inserts option item with the arbitrary amount to certain reservation.

1. long *userId* – the userId of the connecting party
1. String *username* – the user name of the connecting party

2. String *password* – the password of the connecting party
3. long *reservationId* – id of the reservation to which option item will be added
4. long *optionItemId* – id of the option item to add
5. float *amount* – amount for the option item

The result string returns a reservation details (Reservation details).

3.13. insertDiscountItem

```
public String insertDiscountItem(long _userId,String _username,String _password,long
_reservationId,long _discountItemId,float _amount) throws RemoteException;
```

Inserts discount item with the arbitrary amount to certain reservation.

1. long *userId* – the userId of the connecting party
1. String *username* – the user name of the connecting party
2. String *password* – the password of the connecting party
3. long *reservationId* – id of the reservation to which discount item will be added
4. long discount*ItemId* – id of the discount item to add
5. float *amount* – amount for the discount item

The result string returns a reservation details (Reservation details).

3.14. removeInvoiceItem

```
public String removeInvoiceItem(long _userId,String _username,String _password,long
_reservationId,long invoiceItemid) throws RemoteException;
```

Removes discount or option item from a reservation.

1. long *userId* – the userId of the connecting party
1. String *username* – the user name of the connecting party
2. String *password* – the password of the connecting party
3. long *reservationId* – id of the reservation from which invoice item will be removed
4. long *invoiceItemId* – id of the discount or option to remove

You can only remove items from non-confirmed reservations.

The result string returns a reservation details (Reservation details).

3.15. getReservations

```
public String getReservations(long _userId,String _username,String _password) throws
RemoteException;
```

Returns a list of all reservations with details. Completely the same as the second getReservations method described below with the difference of returning all reservations.

3.16. getReservations (2)

```
public String getReservations(long _userId,String _username,String _password,long
_lastSyncPoint) throws RemoteException;
```

Returns reservations that were changed since _lastSyncPoint.

1. long *userId* – the userId of the connecting party
1. String *username* – the user name of the connecting party

2. String *password* – the password of the connecting party
3. long *lastSyncPoint* – point from which you would like to retrieve only newly changed reservations. Use 0 if you want to retrieve all the reservations.

```
<root company_id="" checktime="" syncpoint="">
<reservation id="" charterreservationid="" charterstatus="" resourceid="" status=""
blocksavailability="" datefrom="" dateto="" basefrom="" baseto="" optionexpirydate=""
created="" lastmodified="" companyid="" userid="" code=""></reservation>
…
</root>
```

The results are returned as list of <resource></resource> elements. Attributes are:

1. *company*_id – provided userId in the request
1. *checktime* – time in miliseconds when request has been performed
2. *syncpoint* – point when synchronization occurred. Can be used to retrieve reservations that were changed since last syncpoint
3. *id* – id of the reservation
4. *charterreservationid* – id of the corresponding charter reservation
5. *charterstatus* – status of the corresponding charter reservation
6. *resourceid* – id of the yacht
7. *status* – status of the reservation
8. *blocksavailability* – 1 if this reservation blocks availability
9. *datefrom* – checkin date
10. *dateto* – checkout date
11. *basefrom* – checkin base
12. *baseto* – checkout base
13. *optionexpirydate* – date when option will expire
14. *created* – date when reservation was created
15. *lastmodified* – time in miliseconds when this reservation was last modified
16. *companyid* – id of the charter company
17. *userid* – id of the client
18. *code* – reservation code

3.17. getActiveReservation

```
public String getActiveReservation(long _userId,String _username,String
_password,long _resourceId,Date _onDate) throws RemoteException;
```

Returns reservation id that is active at requested date.

1. long *userId* – the userId of the connecting party
1. String *username* – the user name of the connecting party
2. String *password* – the password of the connecting party
3. long *resourceId* – id of the boat
4. Date *onDate* – date for which active reservation is retrieved

```
<root>
```

```
      <reservation id="">
      </reservation>
      ...
</root>
```

Result is the id of the reservation that is active on requested date:

1. *id* – id of the active reservation

Multiple reservation nodes can exist if reservations overlap by accident.

3.18. getReservationGuest

```
public String getReservationGuest(long _userId,String _username,String _password,long
_reservationId,String _encryptionCode) throws RemoteException;
```

Returns guest for the reservation.

1. long *userId* – the userId of the connecting party
1. String *username* – the user name of the connecting party
2. String *password* – the password of the connecting party
3. long *reservationId* – id of the reservation
4. String *encryptionCode* – encryption code if data is encrypted

```
<user id="" codeid="">
      <property id="" name=""> </property>
      ...
</user>
```

For all available properties please read Changeable user properties in Appendix.

3.19. updateReservation

```
public String updateReservation(long _userId,String _username,String _password,long
_reservationId,String _xmlValues) throws RemoteException;
```

Updates reservation with provided id.

1. long *userId* – the userId of the connecting party
1. String *username* – the user name of the connecting party
2. String *password* – the password of the connecting party
3. long *reservationId* – id of the reservation to update
4. String *xmlValues* – xml string with reservation properties

All properties for the reservation are contained in _xmlValues string:

```
<reservation>
      <element id="codeid">123</element>
      <element id="301">John</element>
      <element id="302">Johnson</element>
      ...
</reservation>
```

For all available properties please read Changeable reservation properties in Appendix.

3.20. getToDoNotes

```
public String getToDoNotes(long _userId,String _username,String _password,int
_objectType,long _objectId) throws RemoteException;
```

Returns existing to do notes on requested object:

1. long *userId* – the userId of the connecting party
1. String *username* – the user name of the connecting party
2. String *password* – the password of the connecting party
3. int *objectType* – type of the object:
    1. 1-reservation
    2. 2-user
    3. 3-resource
4. long *objectId* – id of the object

```
<root>
      <todonote id="" ...>
      </todonote>
      ...
</root>
```

Result is details of all available to do notes on requested objects. Please read To do note details in Appendix.

3.21. insertToDoNote

```
public String insertToDoNote(long _userId,String _username,String _password,int
_objectType,long _objectId,String _xmlValues) throws RemoteException;
```

Inserts new to do note on the object:

1. long *userId* – the userId of the connecting party
1. String *username* – the user name of the connecting party
2. String *password* – the password of the connecting party
3. int *objectType* – type of the object:
    1. 1-reservation
    2. 2-user
    3. 3-resource
4. long *objectId* – id of the object
5. String *xmlValues* – xml string with to do note properties

All properties for the to do note are contained in _xmlValues string:

```
<todonote>
      <element id=”description”>Test description</element>
      <element id=”iscompleted”>1</element>
      ...
</todonote>
```

For all available properties please read Changeable to do note properties in Appendix.

Result is details of new to do note. Please read To do note details for more information.

3.22. updateToDoNote

```
public String updateToDoNote(long _userId,String _username,String _password,long
_noteId,String _xmlValues) throws RemoteException;
```

Updates the existing to do note:

1. long *userId* – the userId of the connecting party
1. String *username* – the user name of the connecting party
2. String *password* – the password of the connecting party
3. long *noteId* – id of the to do note
4. String *xmlValues* – xml string with to do note properties

All properties for the to do note are contained in _xmlValues string:

```
<todonote>
      <element id="description">Test description</element>
      <element id="iscompleted">1</element>
      ...
</todonote>
```

For all available properties please read Changeable to do note properties in Appendix.

Result is details of the updated to do note. Please read To do note details for more information.

3.23. deleteToDoNote

```
public String deleteToDoNote(long _userId,String _username,String _password,long
_noteId) throws RemoteException;
```

Deletes the existing to do note:

1. long *userId* – the userId of the connecting party
1. String *username* – the user name of the connecting party
2. String *password* – the password of the connecting party
3. long *noteId* – id of the to do note

```
<root>
      <todonote id="">
      </todonote>
</root>
```

Returns id of deleted to do note.

3.24. getTranslations

```
public String getTranslations(long _userId,String _username,String _password,String
_xmlValues) throws RemoteException;
```

Returns translations for requested objects:

1. long *userId* – the userId of the connecting party
1. String *username* – the user name of the connecting party
2. String *password* – the password of the connecting party
3. String *xmlValues* – xml string with list of objects and languages

All parameters are contained in xmlValues string:

```
<root>
      <element id=”items”>object1,object2,object3</element>
      <element id=”languages”>de,hr,it</element>
</root>
```

Items element should contain ids of objects (their translationids, to be more precise) for which translations should be returned. Any object in the system that has an “translationid” attribute can be used in this method. “translationid” attribute should be used when calling getTranslations (e.g. if translationid=12:3456789 then <element id=”items”>12:3456789</element>).

Languages element can be ommited and in that case system will return translations for all available languages. All available languages can be found at Available translation languages. If translation does not exist in requested language response will not contain element for that language.

Result is in the format:

```
<root>
  <item id="">
    <translation lang="" value="" />
      ...
  </item>
  ...
</root>
```

This is an example of one request and corresponding response:

```
<!--REQUEST-->
<root>
      <element id=”items”>24:123456789,24:987654321</element>
      <element id=”languages”>de,hr</element>
</root>

<!--RESPONSE -->
<root>
  <item id="24:123456789">
    <translation lang="de" value="12V Kühlschrank" />
    <translation lang="hr" value="12V kuhalo" />
  </item>
  <item id="24:987654321">
    <translation lang="hr" value="Pomoćni čamac" />
  </item>
</root>
```

3.24 *setWeeklyPrice*

```
public String setWeeklyPrice(long userId,String username,String password, long
resourceId, Date dateFrom,Float price , String currency,String product ) throws
RemoteException;
```

Updates the existing price of the boat:

1. long *userId* – the userId of the connecting party
1. String *username* – the user name of the connecting party
1. String *password* – the password of the connecting party
2. long resource*Id* – id of the resource

3. Date *dateFrom* – start date of the season
4. Float price – price of the boat during the season
5. String currency – currency of the price (EUR, GBP, etc)
6. String product – price of the boat for the product

```
<root>
      success
</root>
```

If beginning of the season is 01.05.2019.-01.06.2019, and you insert 11.05.2019 as season date in Date *dateFrom element*, price will change inside that season where the date belongs to.

3.25 *addDocument*

```
public String addDocument(long userId,String username,String password, long
reservationId,String item ) throws RemoteException;
```

Inserts document to specific item (reservation, yacht, model or user)

1.long *userId* – the userId of the connecting party

1. String *username* – the user name of the connecting party
2. String *password* – the password of the connecting party
3. String item – item where you want to add document to (reservation, yacht, model or user).
4. String name – name of the document that will be uploaded

```
<root>
      <document id>"3092325630000100225"
      <name>"test.xls"
      <description>"test1"
      status="UPLOADED"
</root>
```

4. Discount Calculation

Discount rules in Booking Manager are very powerful and complex feature that allows charter operators to very flexibly implement their pricing and discount policies.

Agencies applying those discount rules from the API should follow some basic guidelines in order to implement them correctly. This chapter gives a general overview of specific rules and features to make it easier.

Note: Since discount rule application is a complex subject and is not critical for most agencies, API is also providing an easy way to retrieve a calculated end discount. For all agencies that don't need to import the discount rules, we are advising to instead use the getResourcesDiscount API method.

For all others here are the general explanations:

4.1. Different Discount Types

Duration Discount

Typically used for 2 or 3 week discounts

Service Period

This discount is tied to the Service period, meaning the dates of the actual sailing. It can be used for special offers for sailings taking place in a certain period and can also be used for Early booking.

Reservation Period

This discount is tied to the reservation period, meaning the dates when user places an order. Typically this type of trigger is used for Early Booking discounts:

Last Minute

“Last minute” discount type defines number of days before the charter that this discount is applied to. For example, typical last minute discount is set be valid from 0 to 28 days before the charter.

Although this type of discount is generally also used for Early booking offers, the users of the API should not assume that all Early booking discounts are set as “last minute”. They can also be placed under different discount types depending on how the charter operator defines them, usually “reservation period” type is also used, so in these cases it is easily detected by “validdatefrom” and “validdateto”.

Charter operators can place a "permanent" early booking by using “last minute” type and setting for example 180 - 500 days in validdaysfrom-validdaysto so in this way all bookings made at least 6 months in advance are always treated as Early booking.

4.2. Types of Discount Values

Percentage

The most usual type of discount value, defines a discount in percentage which is written in “value” attribute of particular <discount> element

Amount

If amount type is selected, discount amount is read from the “amountincurrency” together with “currency” attribute of the <discount> element

Free days start & Free days end

Free days discount value type defines a number of days that are free of charge in any given offer. The number of free days is read from the “freedays” attribute of the <discount> element

4.3. Discount application policy

Multiple discounts can be applied for the same boat and same period, but there are some rules for applying them.

1. Each boat has a „maximumdiscount„ flag and it’s defined value (getResources or getResourceDetails).
2. Each discount also has „affectedbymaximum” flag. These two flags are connected in a way that if a discount has affectedbymaximum=1, you will not apply higher percentage than the one stated in „maximumdiscount” flag for each boat. If you have discount with affectedbymaximum=0, you can go over the defined maximum value.
3. If there are more discounts that are valid for the same boat/period/base and they all have excludesotherdiscounts=1 value, the one with lower ID will be applied.

4.4. Examples

Example 1

Applicable discount no.1:

```
<discount id="1950..." name="Special discount" percentage="5.0"
```

```
validdatefrom="1970-01-01" validdateto="2111-04-07" sailingdatefrom="2005-11-
08" sailingdateto="2019-12-31" includedinbaseprice="0" validdaysfrom=""
validdaysto="" discounttype="0" excludesotherdiscounts="1"
affectedbymaximum="0" availableinbase="-1" discountkind="0"
priceincurrency="0.0" currency="EUR" freedays="0.0"></discount>
```

Applicable discount no.2:

```
<discount id="2020..." name="Special 2019" percentage="21.0"
validdatefrom="1970-01-01" validdateto="2124-06-08" sailingdatefrom="2019-01-
10" sailingdateto="2019-12-31" includedinbaseprice="0" validdaysfrom=""
validdaysto="" discounttype="1" excludesotherdiscounts="1"
affectedbymaximum="0" availableinbase="-1" discountkind="0"
priceincurrency="0.0" currency="EUR" freedays="0.0"></discount>
```

In this case you will apply discount of 5 %, because it excludes all others and it has lower ID than the same applicable discount no.2 (<discount id="2020...").

Example 2

```
<discount id="-1" name="Special week discount" percentage="10"
validdatefrom="1970-01-01" validdateto="2124-06-08" sailingdatefrom="2019-01-
10" sailingdateto="2019-01-17" includedinbaseprice="0" validdaysfrom=""
validdaysto="" discounttype="1" excludesotherdiscounts="1"
affectedbymaximum="0" availableinbase="-1" discountkind="0"
priceincurrency="0.0" currency="EUR" freedays="0.0"></discount>
```

This discount is a special weekly offer and will always be applied before all other discounts because of the id being negative. Please note that these special discounts all have the same ids. They will:

1. Always be named "Special week discount"
2. Always have id = -1

Example 3

Boat has an value of maximum discount of 12 % (" maximumdiscount="12.0").

You have 2 applicable discounts:

```
<discount id="2020..." name="Special 2019" percentage="21.0"
validdatefrom="1970-01-01" validdateto="2124-06-08" sailingdatefrom="2019-01-
10" sailingdateto="2019-12-31" includedinbaseprice="0" validdaysfrom=""
validdaysto="" discounttype="1" excludesotherdiscounts="0"
affectedbymaximum="1" availableinbase="-1" discountkind="0"
priceincurrency="0.0" currency="EUR" freedays="0.0"></discount>
```

and

```
<discount id="1950" name="Early booking" percentage="7" validdatefrom="1970-
01-01" validdateto="2124-06-08" sailingdatefrom="2019-01-10"
sailingdateto="2019-01-17" includedinbaseprice="0" validdaysfrom=""
validdaysto="" discounttype="1" excludesotherdiscounts="0"
affectedbymaximum="0" availableinbase="-1" discountkind="0"
priceincurrency="0.0" currency="EUR" freedays="0.0"></discount>
```

In this case first you will calculate 7 % of an discount with lower ID, and then you will take 5 % from second discount to go to maximum 12 %.

5. Code lists

For a successful synchronisation of the availability a receiving party must either change their own ids of the resources and bases to the system ones or create a translation table to match them to their internal lists. The list of bases and yacht codes are downloadable directly from the web service with getBases and getResources functions.

6. APPENDIX

6.1. Date formats

If you are accessing the web service thru PHP or any other language that does not have a automatic translation of the DateTime data into a SOAP envelope, you need to provide a proper format for the date field, which should also include time and a timezone information.

Example of the correct date time:

Date: 1st of January 2012, 11:35 AM

SOAP format: 2012-01-01T11:35:00

6.2. Resource Details

```
<root>
 <resource id="" gid="" name="" base="" model="" berths="" berthsext="" year=""
length="" cabins="" cabinsext="" heads="" headsext="" watercapacity=""
fuelcapacity="" engine="" deposit="EUR"   commissionpercentage="" kind="" draught=""
beam="" lengthatwaterline="" maximumdiscount="" servicetype=""
discountshavesubtotals="" defaultcheckintime="" defaultcheckouttime=""
calculateagencydiscountwithoutvat="" taxableamount="" taxrate=""
genericresourcetypename="" defaultcheckinday="" codeid="" cgid="" companyid=""
transitlog="" defaultcleaningcost="" shipyardid="" saleprice=""
requiredskipperlicense="" ownerid="" mainsail="" genoa="" "maxPeopleOnBoard"
minimumCharterDuration="" certificate="">
 <prices>
 <price datefrom="d" dateto="d" price="nn" currency="xxx" >
  <currencies>
   <currency code="" value="" />
  </currencies>
 </price>
 </prices> <images>
  <image href="URL" comment="" sortorder=""></image>
 </images>
 <equipment>
  <equipmentitem id="" parentid="" name="" value="" categoryname=""
translationid=""></equipmentitem>
 </equipment>
 <extras>
  <extra id="" name="" price="" timeunit="" customquantity="1/0" validdatefrom=""
validdateto="" sailingdatefrom="" sailingdateto="" obligatory="1/0" perperson=""
insludedinbaseprice="" payableoninvoice="1/0" availableinbase=""
includesdepositwaiver="" includedoptions="" validregions=""></extra>
 </extras>
 <discounts>
  <discount id="" name="" percentage="" validdatefrom="" validdateto=""
sailingdatefrom="" sailingdateto="" validdaysfrom="" validdaysto="" discounttype=""
includedinbaseprice="" excludesotherdiscounts="1/0" affectedbymaximum="1/0"
availableinbase=""></discount>
 </discounts>
 <locations defaultbaseid="">
  <location baseid="" datefrom="" dateto=""></location>
 </locations>
 <products>
  <product name="">
   <prices>...</prices>
   <extras>...</extras>
   <discounts>...</discounts>
  </product>
 </products>
</resource>
</root>
```

Each <resource /> element contains attributes describing the basic characteristics of the boat as well as sub-elements for prices, images, equipment, extras, discounts, products and locations

Prices

Prices subelement defines prices for the default product (type of service) defined in *„servicetype" attribute. Most fleets have only one product (servicetype) per one boat with defined pricelists. If additional products exist they are available via <products> sub element.*

Images

All available images for a particular yacht. There are generally three or more images per yacht available. Specific purpose images such as, plan, main and salon image are noted with a special comment

Equipment

All standard equipment of a yacht. If equipment item belongs to any of about 50 generic equipment items that MMK maintains internally, translations are available in 22 languages.

Extras

All available extras for a yacht. Those can be per booking, week, day or custom quantity (per person). Also sometimes extras are marked as obligatory.

Discounts

Discount rules assigned to the yacht. Discounts rules can be applied on a specific date period or on all period, they can be combinable or can override each other. Find more about application of discounts in 4 Discount Calculation chapter of this manual

Locations

In most situations single yacht is assigned to it's home base ("base" attribute). In practice some yachts also change their home bases or locations in general during time, so this sub element will show you this information. Please note that <locations> element will show alternative home bases, but that location of the boat can also vary depending on the start and end port on each reservation

Products

If more products are available on a particular yacht, this will be visible in the <products> sub element with additional price list, discounts and extras applied.

The explanation of response structure:

<resource /> element attributes are:

1. *id* – unique resource id
2. *gid* – generic id of the yacht model, used to uniquely identify the model type of the yacht
3. *name* – the name of the individual yacht
4. *base* – the unique id of the home base
5. *model* – the name of the model
6. *berths* – number of berths
7. *berthsext* – number of berths with the comment included
8. *year* – the build year of the boat
9. *length* – overall length (m)
10. *cabins* – the number of cabins

11. *cabinsext* – number of cabins with the comment included
12. *heads* – number of bathrooms / toilets
13. headext – comment for heads
14. *watercapacity* – capacity of the water tank (l)
15. *fuelcapacity* – capacity of the fuel tank (l)
16. *engine* – engine type
17. *deposit* – security deposit for the yacht on check-in.
18. *commissionpercentage* – agency commission percentage for the yacht. Usually this number is same for all boats of the same provider.
19. *kind* – The kind of a vessel. This can be:
    1. Sail boat
    2. Motor boat
    3. Catamaran
    4. Power Catamaran
    5. Gulet
    6. Motorsailer
    7. Motoryacht
    8. Wooden boat
    9. Houseboat
    10. Trimaran
    11. Rubber boat
    12. Other
20. *draught* – draught (m)
21. *beam* – beam (m)
22. *lengthatwaterline* – length at waterline (m)
23. *maximumdiscount* – maximum discount allowed for this yacht
24. *servicetype* – type of service (product). This can be:
    1. Bareboat
    2. Crewed
    3. Cabin
    4. Flotilla
    5. Power
    6. Berth
    7. Regatta
    8. AllInclusive
25. *discountshavesubtotals* – 1 if discounts have subtotals, 0 otherwise
26. *defaultcheckintime* – checkin hours time
27. *defaultcheckouttime* – checkout hours time
28. *calculateagencydiscountwithoutvat* – 1 if agency discount is calculated without VAT, 0 otherwise
29. *taxableamount* – taxable amount

30. *taxrate* – tax rate
31. *genericresourcetypename* – name of the generic model assigned to the yacht
32. *defaultcheckinday* – default checkin day for this yachts: Sunday=1, Monday=2, Tuesday=3, Wednesday=4, Thursday=5, Friday=6, Saturday=7, Any= -1
33. *codeid* – user defined code for the yacht
34. *cgid* – concatenated generic yacht id containing generic id and number of cabins (generic id + number of cabins)
35. *companyid* – company owner of the yacht
36. *transitlog* – default end cleaning/transit log value for the boat
37. *defaultcleaningcost* – default cleaning cost for the boat
38. *shipyardid* – id of the shipyard (boat manufacturer)
39. *saleprice* – price if resource is on sale, empty if resource is not on sale
40. *requiredskipperlicense* – 1 if skipper license is required for this resource, 0 if not
41. *ownerid* – id of the yacht owner, -1 if yacht doesn't have an owner
42. *mainsail* – type of mainsail of the boat – none, Furling, Full batten
43. *genoa* – genoa type of the boat - none, Furling, Self tacking jib
44. maxPeopleOnBoard – maximum number of people that is allowed to be on board . If the value is -1, it s not defined for the specific yacht.
45. minimumCharterDuration - minimum allowed duration of the charter for this resource
46. validregions – comma separated list of valid region ids for this resource
47. certificate – yacht certificate

<prices /><price /> sub-element attributes are:

1. *datefrom* – date when the price becomes valid (at 0:00h)
2. *dateto* – date until price is valid (at 0:00h)
3. price – the weekly price of the boat in the given period
4. *currency* – the currency of the price (EUR)

   <price><currency /> sub element attributes are

   1. value
   2. currency

<images /><image /> sub-element attributes are:

1. *href* – direct URL of the boat image on the Booking Manager server
2. *comment* – description of the image
3. sortorder – sorting order for an image – integer number

<equipment /><equipmentitem /> sub-element attributes are:

1. *id* – the unique id of the equipment item
2. *parentid* – id of the generic equipment item
3. *name* – the name of the equipment item
4. *categoryname* – the name of the equipment item category

5. *value* – the additional information about the equipment item (number of items, horse power, etc..)
6. *translationid* – key that should be used when requesting translation for this item

<extras /><extra /> sub-element attributes are:

1. *id* – unique id of the extra item
2. *name* – name of the extra item
3. *price* – price of the extra
4. *timeunit* – the timeunit in miliseconds for the extra item:
   - 0 - per booking
   - 3600000 – per hour
   - 86399999 – per day
   - 86400000 – per night
   - 604800000 – per week
   - 604799999 – per week started
4. *customquantity* – flag to mark if the extra is charged per person (1 if it it per person, 0 if it is per booking)
5. *validdatefrom* – which the offer is valid
6. *validdateto* – date until the offer is valid
7. *sailingdatefrom* – the beginning of the sailing period for which the offer is valid
8. *sailingdateto* – the end of the sailing period for which the offer is valid
9. *obligatory* – 1 if the extra item is an obligatory item, 0 if it is an optional item
10. *perperson (DEPRECATED)* – 1 if the extra is counted per person, 0 if counted per booking
11. *includedinbaseprice* – 1 if this extra is already included in the base price of the yacht
12. *payableoninvoice* – 1 if this extra is payable on invoice and 0 if it is payable at the base
13. *availableinbase* – id of the base where this extra is available or -1 if it is available in all bases
14. *includesdepositwaiver* – 1 if this extra includes security deposit waiver
15. *validdaysfrom* – minimum number of days of charter when extras is valid
16. *validdaysto* - maximum number of days of charter when extras is valid
17. *minNumberOfPersons* - minimum number of persons on boat when extra is valid
18. *maxNumberOfPersons* - maximum number of of persons on boat when extra is valid
19. validforbases from – starting (check in) base id where the extra will be applicable
20. validforbases to – end (check out) id of the base where the extra will be applicable

<discounts /><discount /> sub-element attributes are:

1. *id* – the unique id of the discount item
2. *name* – the name of the discount item
3. *percentage* - the percentage value of the discount item (in case of “Percentage” discountkind)
4. *validdatefrom* – date from which the offer is valid
5. *validdateto* – date until the offer is valid
6. *sailingdatefrom* – the beginning of the sailing period for which the offer is valid
7. *sailingdateto* – the end of the sailing period for which the offer is valid

8. *validdaysfrom* – days before a charter when the offer starts being valid (for example 0 for 14 days last minute) . In case of the “service duration” discount type it marks minimum days for duration discount, (for example 14 for two week discount)
9. *validdaysto* – days before a charter when the offer ends being valid (for example 14 for 14 days last minute). In case of the “service duration” discount type it marks maximum days for duration discount, (for example 20 for two week discount)
10. *discounttype* – the type of the discount item which marks how the discount behaves:
    - 0 – Duration discount (two weeks, three weeks)
    - 1 – Reservation period (date of request)
    - 2 – Service period (date of sailing)
    - 3 – Last minute/early booking discount
1. *includedinbaseprice* – 1 if this discount is included in base price
1. *excludesotherdiscounts* – 1 if this discount excludes other discounts
2. *affectedbymaximum* – 1 if this discount is affected by maximum defined discount value
3. *availableinbase* – id of the base where this extra is available or -1 if it is available in all bases
4. *discountkind* – defines the kind of discount value: Percentage, Amount, Free days start, Free days end
5. *priceincurrency* – if „Amount” discount kind is selected
6. *currency* - currency assigned for the discount (in case of „Amount” discountkind)
7. *freedays* – number of free sailing days depending on charter duration if „Free days.. „ discountkind is selected.

<products /><product /> sub element attributes are:

1. name – the name of the product

   <products /><product /><prices> sub element - same structure as <prices>

   <products /><product /><extras> sub element - same structure as <extras>

   <products /><product /><discounts> sub element - same structure as <discounts>

<locations /><location /> sub-element attributes are:

1. *baseid* – the unique id of the base where boat is located
2. *datefrom* – start date of the period when boat is located in specific base
3. *dateto* – end date of the period when boat is located in specific base

defaultbaseid is default location of the boat for periods that are not defined in specific location items.

6.3. Reservation details

```
<root>
 <reservation id="" charterreservation_id="" resourceid="" baseprice="" comission=""
totalprice="" datefrom="" dateto="" status="" statusid="" userid="" code=""
formattedcode="" firstpaymentpercentage="" firstpaymentdays=""
secondpaymentpercentage="" secondpaymentdays="" termsofpayment="" bankingdetails=""
optionexpirydate="" remarks="">
      <extras>
      <extra id="" parentid="" name="" price="" />
            …
      </extras>
      <discounts>
      <discount id="" parentid="" name="" value="" price="" />
            …
      </discounts>
      <payments>
      <payment id="" />
            …
       </payments>
      <paymentplan>
            <deadline date="2024-08-14" amount="1600.0"/>
            <deadline date="2024-10-03" amount="1600.0"/>
      </paymentplan>
      <crewmembers>
            <crewmember />
            …
      </crewmembers>
</root>
```

Results are in form of one <reservation /> element which contains basic reservation attributes as well as sub-elements for extras, discounts and payments:

<reservation /> element attributes are:

1. *id* – the id of the reservation in the system
2. charterreservation_id – **only shown** when using agency token
3. *resourceid* – the id of the boat
4. *baseprice* – the price of the boat from the pricelist
5. *comission* – the comission applied (in EUR)
6. *totalprice* – the final price after comission and discounts
7. *datefrom* – sailing start date
8. *dateto* – sailing end date
9. *status* – status of a booking (Option/Reservation/Owner week/Service/Canceled/Option expired) - DEPRECATED (use statusid field instead)
10. statusid – booking status id
11. *userid* – the id of the provider company
12. *code* – reservation code
13. *formattedcode* – formatted reservation code
14. *firstpaymentpercentage* – percentage for the first payment
15. *firstpaymentdays* – how many days after the confirmation should the first payment be made
16. *secondpaymentpercentage* – percentage for the second payment
17. *secondpaymentdays* – how many days before the charter should the second payment be made

18. *termsofpayment* – string containing terms of payment text
19. *bankingdetails* – string containing banking details text
20. *optionexpirydate* – date when option expires
21. *remarks* – reservation remarks

<extras /><extra /> sub-element attributes are:

22. *id* – the id of the reservation extra
23. *parentid* – parent id of the extra
24. *name* – name of the extra
25. *price* – defined price

<discounts /><discount /> sub-element attributes are:

26. *id* – the id of the reservation discount
27. *parentid* – parent id of the discount
28. *name* – name of the discount
29. *value* – percentage value of the discount
30. *price* – price after the discount

<payments /><payment /> sub-element attributes are:

31. *id* – the id of the payment

<crewmembers /> contains details for each crew member (Crew member details).

<paymentplan/><deadline> sub-element attributes are:

1. date – date of payment deadline
2. amount – amount to be paid by deadline (in currency of reservation)

6.4. Payment details

```
<payment id="" value="" invoiceid="" paymentmethodid="" date="" currency=""
exchangerate="" comment="" bankcostvalue="" invoicedate="" stornodate=""
altcurrency="" altexchangerate="">
</payment>
```

Each “<payment />” section contains the following information about a specific payment:

1. *id* – id of the payment
2. *value* – payment amount
3. *invoiceid* – id of this advanced invoice
4. *paymentmethodid* – id of the payment method assigned to the payment
5. *date* – date when payment was created
6. *currency* – currency assigned to the payment
7. *exchangerate* – exchange rate of the currency
8. *comment* – additional comment
9. *bankcostvalue* – amount of transaction cost
10. *invoicedate* – date when invoice issued
11. *stornodate* – date when storno invoice issued
12. *altcurrency* – the alternative currency

13. *altexchangerate* – exchange rate of the alternative currency

6.5. User details

```
<user id="" codeid="">
      <property id="" name=""> </property>
      ...
</user>
```

Each "<user />" section contains the following information about a specific user:

1. *id* – id of the user
2. *codeid* – user code

Each user contains one or more properties which are described with id, name and a value.

<user /><property /> sub-element attributes are:

3. *id* – id of the property
4. *name* – name of the property

For changeable user properties please read Changeable user properties in Appendix.

6.6. Crew member details

```
<crewmember id="" isskipper="" statusid="" userid="">
      <property id="" name=""/>
      …
</crewmember>
```

Each "<crewmember />" section contains the following information about a specific crew member:

1. *id* – id of the crew member
1. *isskipper* – 1 if it is a skipper, 0 otherwise (DEPRECATED, use statusid instead)
2. *statusid* – status of the crew member: 1=skipper, 2=crew member, 3=passenger
3. *userid* – id of the user in the addressbook

For all available properties please read Changeable crewmember properties in Appendix.

6.7. To do note details

```
<todonote id="" description="" createddate="" iscompleted="" completeddate=""
      typeid="" scheduleddate="" >
</todonote>
```

Each "< todonote />" section contains the following information about a specific to do note:

4. *id* – id of the to do note
5. *description* – description of the to do note
6. *createddate* – date of creation
7. *iscompleted* – 1 if to do note is completed, 0 otherwise
8. *completeddate* – date of completion
9. *typeid* – type of to do note
    1. 0-todo
    2. 1-event
    3. 2-note
10. *scheduleddate* – date

For all available properties please read Changeable to do note properties in Appendix.

6.8. Changeable user properties

Here is the list of all available user properties that can be used to insert or update user in the address book.

Id	Name	Type	Id	Name	Type
codeid	Code id	Long	**329**	Remarks	String
301	First name	String	**331**	State/Country/Province	String
302	Last name	String	**361**	Option expire	Integer
303	Company name	String	**363**	Login disabled	Boolean
305	Address	String	**364**	Invoice to Guest	Boolean
306	Country	Long	**365**	Only from Saturday	Boolean
307	City	String	**366**	Charge tax	Boolean
308	Zip code	String	**367**	Tax office	String
309	E-mail	String	**368**	Skipper License number	String
310	Telephone 1	String	**369**	Skipper license issued by	String
311	Telephone 2	String	**370**	Title	String
312	Mobile 1	String	**373**	Skype Name	String
313	Mobile 2	String	**374**	Contacted by	String
314	Fax 1	String	**375**	Loyalty card number	String
315	Fax 2	String	**376**	Loyalty card year	String
316	VAT Code	String	**377**	Loyalty card date issued	Date
317	Language	String	**378**	Loyalty card date of expiration	Date
318	Birthday	Date	**379**	Date of creation	Date
319	Address 2	String	**380**	E-mail 2	String
320	User Discount	Float	**381**	Maximum number of concurrent options	Integer
321	Webiste	String	**450**	Affiliate enabled	Boolean
322	Birth Place	String	**451**	Cookie duration	String
323	Country of birth	Long	**453**	Commission on base price	Boolean
324	Citizenship	Long	**454**	Commission fixed amount	Float

325	Passport Number	String	**455**	Commission percentage	Float
326	Visa Id	Long	**456**	Notification email	Boolean
328	Password	String	**457**	Separate commission invoice	Boolean

Examples how you should use these properties to set some user values:

```
<element id="codeid">12345</element>
<element id="301">John</element>
<element id="306">12</element>
<element id="455">10.0</element>
```

6.9. Changeable reservation properties

Here is the list of all available reservation properties that can be used to update reservation.

Id	Name	Type	Id	Name	Type
RESOURCE_ID	Yacht id	Long	**653**	Exchange rate	Float
DATE_FROM	Checkin date	Date	**656**	Invoice to guest	Boolean
DATE_TO	Checkout date	Date	**657**	Internal remarks	String
OPTION_EXPIRY_DATE	Option expiry date	Date	**660**	Dont invoice options	Boolean
USER_ID	Client id	Long	**663**	Free entry reservation	Boolean
BASE_FROM	Checkin base	Long	**665**	Alternative exchange rate	Float
BASE_TO	Checkout base	Long	**666**	Master invocie note	String
CODE	Code	Long	**670**	Default cleaning cost	Float
CONFIRMATION_DATE	Confirmation date	Date	**671**	Actual cleaning cost	Float
90	Remark	String	**674**	Checkout remarks	String
579	Default checkin time	String	**679**	Separate commission invoice	Boolean
580	Default checkout time	String	**680**	Commission invoice received	Boolean
622	Base booking price	Float	**681**	Owner bank cost	Float
636	Tax percentage	Float	**1605**	First payment percentage	Long
646	Terms of payment	String	**1606**	First payment days	Integer
648	Note	String	**1607**	Second payment percentage	Long
651	Invoice code	String	**1608**	Second payment days	Integer

652	Date value	Date	**1613**	Reservation number suffix	String

Examples how you should use these properties to set some reservation values:

```
<element id="code">1202</element>
<element id="579">17:00</element>
<element id="653">7.407</element>
<element id="1606">7</element>
```

6.10. Changeable payment properties

Here is the list of all available payment properties that can be used to update payment.

Id	Name	Type	Id	Name	Type
paymentmethodid	Payment method id	Long	**comment**	Comment	String
invoiceid	Invoice id	String	**bankcostvalue**	Bank cost value	Float
value	Value (amount)	Float	**invoicedate**	Invoice date	Date
date	Date	Date	**stornodate**	Storno date	Date
currency	Currency	Long	**altcurrency**	Alternative currency	Long
exchangerate	Exchange rate	Float	**altexchangerate**	Alternative exchange rate	Float

Examples how you should use these properties to set some reservation values:

```
<element id="invoiceid">132/2012</element>
<element id="value">580.0</element>
<element id="exchangerate">7.407</element>
<element id="bankcostvalue">12.5</element>
```

6.11. Changeable crewmember properties

Here is the list of all available crew member properties that can be used to update crew member.

Id	Name	Type	Id	Name	Type
329	Remarks	String	**1313**	Date of entry	Long
1302	Border crossing ID	String	**1314**	Visa expiry date	Date
1303	First name	String	**1315**	Residence	String
1304	Last name	String	**1316**	Former residence	String
1305	Date of birth	Date	**1317**	Date of application	Date
1306	City of birth	String	**1318**	Date of departure	Date
1307	Country of birth	Long	**1319**	Guest type	String
1309	Citizenship	Long	**1320**	Guest status	String
1310	Type of travel document	String	**1322**	Point of entry	String

1311	*Travel document ID* (“***027***” - *ID card,* (“***002***” - *Personal Passport,* (“***024***” - *Other)*	String	**1323**	Skipper license	String
1312	Visa	Long	**1329**	Country of residence	Long
1330	Sex	String			

Examples how you should use these properties to set some crew member values:

```
<element id="329">Requested satellite phone</element>
<element id="1303">John</element>
<element id="1304">Johnson</element>
<element id="1306">New York</element>
```

6.12. Changeable to do note properties

Here is the list of all available to do note properties that can be used to update to do note.

Id	Name	Type	Id	Name	Type
description	Description	String	**completeddate**	Completed date	Date
createddate	Created date	Date	**typeid**	Type id	int
iscompleted	Is completed	Boolean	**scheduleddate**	Scheduled date	Date

Examples how you should use these properties to set some to do note values:

```
<element id="description">Repair yacht</element>
<element id="typeid">0</element>
```

6.13. Available translation languages

Chinese (zh), Croatian (hr), Czech (cs), Dutch (nl), English (en), Finnish (fi), French (fr), German (de), Greek (el), Hungarian (hu), Italian (it), Lithuanian (lt), Norwegian (no), Polish (pl), Russian (ru), Serbian (sh), Slovak (sk), Slovenian (sl), Spanish (es), Swedish (sv), Turkish (tr).

7. Code Examples

7.1. PHP

```
<?php

/**
 * MMKStruct - object with MMK call parameters,
 * simple Array to in0,in1 etc. Object properties as of definitions
 */
class MMKStruct {
 /**
 * Constructor
 *
 * @param $a - array of parameters
 * @ret void
 */
 public function __construct($a) {
  $i=0;
  foreach($a as $var=>$val) {
   $varName = 'in'.$i;
   $this->$varName = $val;
   $i++;
  }
 }
}

// Specify url
$wsdl = 'https://www.booking-manager.com/cbm_web_service2/services/CBM?wsdl';

// Load the SOAP client with definitions
$soapClient = new SoapClient($wsdl, Array('trace'=>1));


try {
 $struct = new MMKStruct(Array(YOUR_COMPANY_ID,'YOUR_EMAIL_ADDRESS',
'YOUR_PASSWORD','635','2019', 'true',0));

 $result = $soapClient->getAvailabilityInfo($struct);

 if (isset($result->out)) {
  $xml = $result->out;
  echo $xml;
 }
}

// catch exceptions
catch (Exception $e) {
 print_r($soapClient->__getLastRequest());
 print_r($soapClient->__getLastResponse());
 print_r($e->getTrace());
 var_dump($e);
}
?>
```

7.2. PHP with Curl

Users with 32bit PHP can have issues with long ID values which will overflow if number is too big. In case that you can't switch to 64bit PHP you can use Curl to create the request instead of using SoapClient object.

```
<?php
//use this to test iv curl is enabled, if it is not then you should enable it
//echo 'Curl: ', function_exists('curl_version') ? 'Enabled' : 'Disabled';

$soap_request  = "<soapenv:Envelope
xmlns:soapenv=\"http://schemas.xmlsoap.org/soap/envelope/\"
xmlns:cbm=\"http://cbm.mmk.com\">\n";
$soap_request .= "<soapenv:Header/>\n";
$soap_request .= "<soapenv:Body>\n";
$soap_request .= "      <cbm:getBases>\n";
$soap_request .= "            <cbm:in0>YOUR_COMPANY_ID</cbm:in0>\n";
$soap_request .= "            <cbm:in1>YOUR_EMAIL_ADDRESS</cbm:in1>\n";
$soap_request .= "            <cbm:in2>YOUR_PASSWORD</cbm:in2>\n";
$soap_request .= "      </cbm:getBases>\n";
$soap_request .= "</soapenv:Body>\n";
$soap_request .= "</soapenv:Envelope>\n";

$header = array(
      "Content-type: text/xml;charset=\"utf-8\"",
      "Accept: text/xml",
      "Cache-Control: no-cache",
      "Pragma: no-cache",
      "SOAPAction: \"run\"",
      "Content-length: ".strlen($soap_request),
      );

$soap_do = curl_init();
curl_setopt($soap_do, CURLOPT_URL,
"https://www.booking-manager.com/cbm_web_service2/services/CBM");
curl_setopt($soap_do, CURLOPT_CONNECTTIMEOUT, 60);
curl_setopt($soap_do, CURLOPT_TIMEOUT,        60);
curl_setopt($soap_do, CURLOPT_RETURNTRANSFER, true );
curl_setopt($soap_do, CURLOPT_SSL_VERIFYPEER, false);
curl_setopt($soap_do, CURLOPT_SSL_VERIFYHOST, false);
curl_setopt($soap_do, CURLOPT_POST,           true );
curl_setopt($soap_do, CURLOPT_POSTFIELDS,     $soap_request);
curl_setopt($soap_do, CURLOPT_HTTPHEADER,     $header);

$result = curl_exec($soap_do);

if($result === false) {
      $err = 'Curl error: ' . curl_error($soap_do);
      curl_close($soap_do);
      print $err;
} else {
      curl_close($soap_do);
      print $result;
}
?>
```

8. Document Changes

8.1. Version 1.27. (16.08.2024.)

- Reservation details - new sub-element
  - paymentplan – *payment plan of a reservation (a list of deadlines)*
- *Resource Details – new values for timeunit*
  - *added `per week started`*

8.2. Version 1.26. (26.07.2024.)

- Reservation details - new attribute value for agency calls and changes to id attribute value
  - *charterreservation_id* – id of the corresponding charter reservation
  - *id* – contains id of the agency reservation when API is called with agency token

8.3. Version 1.25. (10.07.2024.)

- getResourceDetails/getResources - new resource attribute value
  - *certificate* – yacht certificate

8.4. Version 1.24. (09.06.2023.)

- getShortAvailabilityInfo – added Sleepaboard status
- getCompanies, getCcompaniesExtended – added company rating field
- createReservation - added new parameters
  - *baseFromId* – valid id of the start base (as seen in getSearchResultsFilter)
  - *baseToId* – valid id of the end base (as seen in getSearchResultsFilter)

- Resource Details – replaced non-existing params *validpersonsto*/*validpersonfrom* with minNumberOfPersons/maxNumberOfPersons

8.5. Version 1.23. (26.05.2023.)

- Resource Details – new values
  - *validregions*
  - minimumCharterDuration
  - *headsext*
  - *sortorder*

8.6. Version 1.22. (26.04.2023.)

- removeInvoiceItem – added call for removing invoice items from non-confirmed reservations
- added crewmember property information – **1310** – Type of travel document

8.7. Version 1.21. (01.01.2023.)

- getResources – added parameters
  - *minimumCharterDuration*
- getResourceDetails – added parameters
  - *minimumCharterDuration*
- Resource Details – new values for *timeunit*
  - *added per night*
  - *added per hour*

8.8. Version 1.20. (01.12.2022.)

- getSearchResultsFilter – added filter parameter
  - *currency*
- getSearchResultsFilter – added output attributes
  - *commissionpercentage, commissionvalue, deposit, obligatoryextras*
- updateReservation – updated *Changeable reservation properties* synatx

8.9. Version 1.19. (01.07.2022.)

New parameters on existing methods

- getResources (1.2) – added parameters
  - *maxPeopleOnBoard=””*
- getSearchResultsFilter – added filter parameter
  - *kind*
  - *product*
- updateCrewMember – added new Changeable crewmember properties
  - *1329 – Country of residence*
  - *1330 - Sex*
    - Male
    - Female

8.10. Version 1.18 (31.12.2018.)

Removed support for methods

- getSearchResults – This method still works to keep the compatibility with old integrations, but from 1.18 will not be officially supported or documented. All existing integrations are directed to switch to more flexible getSearchResultsFilterAPI call.

New parameters on existing methods

- getResources - added parameters:
  - *kind*
    - *Power Catamaran*
  - *servicetype*
    - *Flotilla*
    - *Power*
    - *Berth*
    - *Regatta*
    - *Allinclusive*
  - *defaultcheckinday*
    - *Any (-1)*
  - *mainsail =””*

- *genoa =””*

- *<extras />*
  - *validpersonsfrom =””*
  - *validpersonsto =””*
- *<discounts />*
  - discountkind =””
  - priceincurrency =””
  - currency =””
  - *freedays =””*

  <products /> section added

  <price><currency>section added

New Discount Calculation chapter in manual

Chapter 4 of this manual to explain discount rules in more detail (Discount Calculation)

8.11. Version 1.17 (22.05.2017.)

Changed parameters

- getAvailabilityInfo (1.4)
  - companyId
  - year
- getResources(1.2)
  - companyId

New parameters on existing methods

- ResourceDetails (6.2)
  - added <location> information

8.12. Version 1.16 (27.04.2016.)

New methods

- getResourcesDiscount (1.11)
- getSearchResultsFilter (1.7)

New parameters on existing methods

- getCountries (1.18)
  - <translations>

8.13. Version 1.15 (17.03.2016.)

New methods

- getTranslations (3.23)

- getCashRegister (2.6)

New parameters on existing methods

- getInvoices (2.1) - added parameters
  - reservationid=””

8.14. Version 1.14 (08.12.2015.)

New parameters on existing methods

- createReservation (1.12) - added parameters
  - statusid=””
- getResourceDiscount (1.10) - added parameters
  - price=””
- isResourceAvailable(1.9) -added parameters
  - status=””

8.15. Version 1.13 (18.11.2015.)

New methods

- getUserDetails (3.4)

New parameters on existing methods

- Resource details (6.2) - added parameters
  - ownerid=””
- Crew member details (6.6) - added parameters
  - statusid=””, userid=””

New elements on existing methods

- Reservation details (6.3) - added element
  - <crewmembers>

8.16. Version 1.12 (15.06.2015.)

New parameters on existing methods

- getInvoices (2.1) - added parameters
  - relatedreservationid=””

8.17. Version 1.11 (18.03.2015.)

New methods

- getSpecialOffers (1.8)
- getResourceDiscount (1.10)
- getResourceDetails (1.3)
- Resource details (6.2) - extracted list of resource details to single place because it will be used as a return string for both getResourceDetails and getResources methods
- getShipyards (Error: Reference source not found)

New parameters on existing methods

- getSearchResults (1.6) - added parameters
  - datefrom="" dateto=""
  - basetoid=""
- Resource details (6.2) - added parameters
  - shipyardid="", saleprice="", requiredskipperlicense="", extras.includedoptions=""
- getBases (1.16) - added parameters
  - countryid=""

8.18. Version 1.10 (28.03.2014.)

New methods

- getShortAvailabilityInfo (1.5)

New parameters on existing methods

- getResources (1.2) - added parameters
  - cgid="" companyid=""

8.19. Version 1.9 (01.10.2013.)

New Parameters on existing methods

- getBases (1.16)
  - inside <base> tag new attributes "longitude" , “latitude”

8.20. Version 1.8 (27.05.2013.)

New methods

- isResourceAvailable(1.9)
- getEquipmentCategories (1.19)
- getActiveReservation(3.16)
- getReservationGuest (3.17)
- getToDoNotes (3.19)
- insertToDoNote (3.20)
- updateToDoNote (3.21)
- deleteToDoNote (3.22)
- Duration Discount (6.7)
- Changeable to do note properties (6.12)

New parameters on existing methods

- getResources (1.2) – added parameters
    - berthsext=”” discountshavesubtotals=””defaultcheckintime=”” defaultcheckouttime=”” calculateagencydiscountwithoutvat=””taxableamount=”” taxrate=”” genericresourcetypename=””
    - inside <equipment> tag new attributes “parentid” “categoryname”
    - inside <extras> tag new attribute "availableinbase"
    - inside <discount> tag new attribute availableinbase=””
- getCompanies (1.1) – new attribute: “availability=””
- getReservations (2) (3.15) - new attribute: code=""
- getInvoices (2.1)
    - inside <invoice>tag new attribute rate=""
    - added services tags <services><service name=”” total=”” rate=””

/></services>