RoomCloud XML API

Version 2.3.4

16/06/2017

REVISION HISTORY

Version	Description of the revision
2.3.4	Corrected attribute values and descriptions in elements reservation and room
2.3.3	Added new "applyBusinessRule" attributes in modify requests Added new "getBusinessRules" request
2.3.2	Added TEST ENVIRONMENT URL endpoint: https://apitest.roomcloud.net/be/search/xml.jsp
2.3.1	Added new "shop" attributes in Query Inventory requests
2.3.0	Added new "occupancy management" attributes (occupancy, adult and child) in getRooms, view and modify requests
2.2.8	Added prepaid attribute in reservation
2.2.7	Changed reservations section
2.2.6	Added extraAdults and extraAdultsPrice in reservationDetail Added extraAdultPrice in dayPrice
2.2.5	Removed support for HTTP connections Queue mechanism description Added shop attribute inside reservation element
2.2.4	End Point URL changed to https://api.roomcloud.net/be/search/xml.jsp
2.2.3	API KEY
2.2.2	Portal error
2.2.1	Correct useDLM attribute in reservations
2.2.0	API for booking engine
2.1.0	Correct DLM management in reservations
2.0.6	Added attribute isScheduled in modify
2.0.5	Changed ccCode description
2.0.4	Added firstName and lastName fields in Reservation Added firstName and secondName in guest Added rates_id field in DayPrices
2.0.1	First version

Table of contents

1 INTRODUCTION

1.1. API goal

The goal of these API XML is to give PMS/CRS the possibility to use RoomCloud.com(RC) main functions inside their software.

These functions are:
1. Download reservations: RoomCloud periodically retrieves new or modified reservations from the channels; retrieved reservations are then stored in a local database; API functions let you retrieve these reservations.
2. Read RoomCloud Inventory : the local inventory is where it is stored daily room's data (availability, rate etc)
3. Update RoomCloud Inventory and linked channels : when the local inventory is updated, updated data are then forwarded to the channels.

1.2. Data exchange mechanism

Every call to the API is an Https POST to the address:

TEST ENVIRONMENT
https://apitest.roomcloud.net/be/search/xml.jsp

PROD ENVIRONMENT
https://api.roomcloud.net/be/search/xml.jsp

The HTTP request headers must contains ContentType = "text/xml".

Every request is added to an internal QUEUE that is composed by a Waiting Queue and an Active Queue.
For every property the Waiting queue accepts up to 8 requests: it means that an external software can send simultaneously up to 8 requests for the same property.
For every property the Active queue accepts up to 3 requests of different type: it means that Roomcloud can simultaneously elaborate, for the same property, up to 3 different requests queued in the Waiting Queue. As a consequence update requests are always elaborated one by one. Roomcloud never processes an update request until the previous one is completed.

Roomcloud accepts up to 15.000 availability rows per property every 180 minutes.
This limit can be temporarily changed if needed.

1.3. Request message structure and access credentials

Every call must be enclosed in a "Request" element that contains the access credentials (userName and password attributes):

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<Request userName="MyLogin" password="MyPassword">
        …
</Request>
```

Username and password are the same used to access the RoomCloud extranet.

For security reasons, access to the XML interface has a second level authorization control:the caller IP address must be in the Roomcloud whitelist or the Request element must contain an authorized API KEY.
Since the API KEY can be also used to identify the system caller, we strongly suggest to adopt this second method to access the XML interface.

```
<?xml version="1.0" encoding="UTF-8"?>
<Request apikey="Myapikey" userName="MyLogin" password="MyPassword">
        …
</Request>
```

To ask for the API KEY or to whitelist an IP send a request to to the contact specified at the end of this document.

1.4. Response Message structure

Following the basic response message structure is displayed:

```
<?xml version="1.0" encoding="UTF-8"?>
<Response>
        …
</Response>
```

1.5. Error Message

Error messages are returned as attribute of an error element.

```
<error message="Authentication Failed - Wrong Password"/>
```

2 Configuration methods

2.1 getHotels

This method return the list of all hotels associated to an username/password. For each listed hotel it provides an ID and a description. ID must be used to identify the hotel in all other messages.

The ***hotel*** element has the following attributes

id *: Integer → hotel code to identify the hotel inside the channel*
description *: String →hotel name/ description*

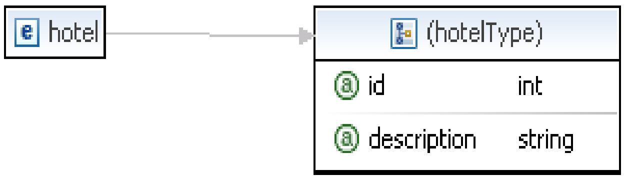


Fig 1: hotel element representation

Ex:

```
<Request userName="MyLogin" password="MyPassword">
  <getHotels/>
</Request>

<Response>
  <hotel id="324" description="Hotel RoomCloud Milano" />
  <hotel id="456" description="Hotel RoomCloud Roma" />
</Response>
```

2.2 getRates

It gives the list of rates for a selected hotel. Each element is characterized by two attributes, rateId and description both required.

The ***rate*** element has the following attributes:

***rateId** : Integer (required) ➔ rate code to identify the rate*
***description** : String (required) ➔ rate description*

Ex:

```
<Request userName="MyLogin" password="MyPassword">
  <getRates hotelId="324"/>
</Request>

<Response>

    <rate rateId="934" description="std" />
    <rate rateId="1452" description="pkg" />

</Response>
```

2.3 getRooms

It gives the list of rooms for a selected hotel. Each room is represented by a room element characterized by two attributes , id and description both required.

The ***room*** element has the following attributes:

***id** : Integer (required) ➔ room code to identify the room*
***description** : String (required) ➔ room description*
***beds :** beds in the room*
***additionalBeds :** max additional beds available for the room. They can be used both as adults and children price, please refer to the children elements just above.*

The room element can contain for children elements:

- rate defines if that base rate can be present under that room

- occupancy: defines the price for each further occupancy available for that room. It's always less than the room base occupancy and defines a room price (not a supplement price).
- adults: defines all the additional adults available for that room. This is always a supplement price (not an entire room price)
- child: defines all the additional children available for that room. This is always a supplement price (not an entire room price)

rate (see above)

occupancy (occupancy price)
id : *Integer (required) → identifier of this occupancy price*
rateId : *Integer (required) → identifier of the related rate*
beds : *Integer (required) → identifier of the occupancy value*

adult (adults price)
id : *Integer (required) → identifier of this children price*
rateId : *Integer (required) → identifier of the related rate*
additionalBeds : *Integer (required) → identifier of the number of additional beds for which is valid this supplement price.*

child (children price)
id : *Integer (required) → identifier of this children price*
rateId : *Integer (required) → identifier of the related rate*
from : *Integer (required) → starting age of child*
to : *Integer (required) → ending age of child*
additionalBeds : *Integer (required) → identifier of the number of additional beds for which is valid this supplement price.*

Ex:

```
<Request userName="MyLogin" password="MyPassword">
        <getRooms  hotelId="324"/>
</Request>

<Response>
<room id="2092" beds="2" additionalBeds="0" description="Double">
        <rate rateId="934" />
        <rate rateId="1452" />
</room>
<room id="3384" beds="2" additionalBeds="2" description="Suite">
        <rate rates_id="934" />
        <occupancy rateId="934" id="1" beds="1" />
        <adult rateId="934" id="3" additionalBeds="1" />
        <adult rateId="934" id="4" additionalBeds="2" />
        <child rateId="934" id="0" from="2" to="7" additionalBeds="1" />
        <child rateId="934" id="1" from="2" to="7" additionalBeds="2" />
        <rate rateId="935" />
        <occupancy rateId="1452" id="1" beds="1" />
        <adult rateId="1452" id="3" additionalBeds="1" />
        <child rateId="1452" id="0" from="2" to="7" additionalBeds="1" />
</room>
</Response>
```

2.4 getPortals

It gives the list of channels managed by RC for a selected hotel. Each channel is represented by a portal element. The ***portal*** element is a complex type element characterized by several attributes and by other optional sub-elements.

The ***portal*** element has the following attributes:

id *: Integer (required)* → code to identify the channel
description *: String (required)* → channel description
portalUserName: *String* → the username used to access the portal extranet
portalPassword : *String*→ the password used to access the portal extranet

Ex:

```
<Request userName="MyLogin" password="MyPassword">
  <getPortals  hotelId="324"/>
</Request>

<Response>
   <portal id="3" description="Expedia" portalUserName="xxx" portalPassword="yyy" />
   <portal id="124" description="Booking"  portalUserName="ccc" portalPassword="bPwd"/>
</Response>
```

3 Portals Password management

When a portal password changes, it's necessary to tell RC in order to make it work. This can be done using the *changePassword* function, supplying to it the obvious arguments*:*

hotelId=the hotel RC is managing
portalId=the portal subject to password change
portalUserName=the portal username
portalPassword=the portal new password

Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<Request userName="MyLogin" password="MyPassword">
<changePassword hotelId=".." portalId=".." portalUserName="MyPortalLogin"
portalPassword="MyPortalPassword"/>
</Request>
```

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<Response>
<ok/>
</Response>
```

4 Reservations

The **reservations** function let the client retrieve reservations from the RoomCloud local database.

The ***reservations*** element has the following attributes:

hotelId *: Integer (required) → RoomCloud hotel code*

useDLM *: true/false (optional)*
startDate *: Date (yyyy-mm-dd)(optional)*
endDate *: Date (yyyy-mm-dd)(optional)*

Different combinations of attributes correspond to different search criteria:

1. To retrieve all the reservations from all portals, **created or modified after the last reservations call** use:
   - useDLM = "true"
   - omit startDate
   - omit endDate

Ex:

```
<Request userName="MyLogin" password="MyPassword">
<reservations hotelId="3284" useDLM="true"/>
</Request>
```

2. To retrieve all the reservations created or modified between two dates use:
   - useDLM = "true"
   - ***startDate*** *= …*
   - ***endDate= …***

Ex:

```
<Request userName="MyLogin" password="MyPassword">
  <reservations  hotelId="3284" useDLM="true" startDate="2012-10-10"
                 endDate="2012-10-12"/>
</Request>
```

3. To retrieve all the reservations with check-in date between two dates use:
   - ***useDLM*** *=* ***"false"*** *or omit parameter useDLM*
   - ***startDate*** = …
   - ***endDate*** = …

Ex:

```
<Request userName="MyLogin" password="MyPassword">
  <reservations  hotelId="3284" useDLM="false" startDate="2012-10-10"
                 endDate="2012-10-12"/>
</Request>
```

Returned informations are a list on reservation elements.
The reservation element has a very complex structure and many of the attributes and sub-elements are **not mandatory** but portal-dependent. Therefore they could be absent (they're classified as **optional**)

reservation element has the following attributes and child elements

id ***:*** *String (required)* → the channel reservation code
portalId ***:*** *Integer (required)* → the channel identification code (0=booking engine)

dlm ***:*** *DateTime (yyyy-mm-dd HH:MM:ss)(optional)* → last modification date
checkin : *Date (yyyy-mm-dd)(required)* → the check-in date
checkout : *Date (yyyy-mm-dd)(required)* → the check-out date

creation_date ***:*** *DateTime (required)* → date and time when reservation was made (yyyy-MM-ddThh:mm:ss)

firstName ***:*** *String (optional)* → the first name of the customer that has completed this reservation
lastName ***:*** *String (optional)* → the second name of the customer that has completed this reservation

address ***:*** *String (optional)* → the customer address
city ***:*** *String (optional)* → the customer city
zipcode ***:*** *String (optional)* → the customer zip code
country ***:*** *String (optional)* → the customer country description

telephone ***:*** *String (optional)* → the customer telephone number
email ***:*** *String (optional)* → email address of the client***.***

rooms ***:*** *Integer (optional)* → the total number of rooms booked
persons ***:*** *Integer (optional) → the total number of adults hosted*
children ***:*** *Integer (optional) → the total number of children hosted*

price *: Double (optional)* → the total price of the reservation
prepaid *: Double (optional)* → prepaid amount of the reservation
currencycode *: String (required) → currency of the price. The channel must use the* ISO 4217 Currency Code List

paymentType *: String (optional)* → code of the type of payment used by the client (1 = by transfer , 2 = by postal order , 3 = by paypal, 4 = by credit card)
shop *: Integer (optional)* → identifies the booking source when the booking is made through the Roomcloud BookingEngine; when shop=4 and shop=6 the booking source is Tripadvisor, when shop=5 the booking source is Google
notes : *String (optional)*
offer ***:*** *String (optional)* → if there is any offer applied here the channel can report the description.

status ***:*** *String (required)* → the state of the reservation (see states table).
cancancel ***:*** *String (optional)* → parameter which indicates that the hotelier can cancel the reservation; can assume following values: 1 true,0 false
cancelbydetail ***:*** *String (optional)* → parameter which indicates that the hotelier can cancel the details of the reservation; can assume following values: 1 true,0 false

ccData : tag used in case of payment by credit card which contains credit card details. It's Optional and so are all his fields :

ccCode *String (optional)* → CVC of the credit card. The channel can use its own codes.
ccNumber *String (optional)* → credit card number
ccExpireDate *String (optional)* → expiration date. The formats accepeted are MM/yy and MM/yyyy.
ccHolder *String (optional)* → holder of the credit card

room : contains details for every room reserved .

id : *String (required)* → code of the reserved room.
description: *String (required)* → it's the name of the room
portalRoomDescription: *String (optional)* → the name of the room on the channel

checkin : *Date (yyyy-mm-dd)(optional)* → the check-in date for this room, can be different from the reservation checkin
checkout : *Date (yyyy-mm-dd)(optional)* → the check-out date for this room, can be different from the reservation checkout

rateId *:* *String (optional)* → rate plan code applied to this room
rateDescription : String (optional) → rate plan description applied to this room
portalRateDescription : String (optional) → rate plan description on the channel

quantity : I*nteger (required)* → it's the room quantity reserved
currency *:* *String (optional)* → currency of the price (ISO 4217 Currency Code List)
price : *Double (required)* → it's the total price for the reservation of this room.
adults *:* *Integer (optional)* → number of adults hosted in this room
children : *Integer (optional)* → number of children hosted in this room.
childrenPrice : *Double (optional)* → total price paid for the children hosted in this room. It must be included in the price field

extraAdults : *Integer (optional)* → number of additional adults hosted in this room.
extraAdultsPrice : *Double (optional)* → total price paid for additional adults hosted in this room. It must be included in the price field

commission : *Double (optional)* → amount of the commission applied by the channel for this room

status *:* *String (required)* → the state of the reservation detail (see states table).

dayPrice : tag used to specify for every room for every day the price charged to the client.**Optional**

roomId *: String (required)* → it's the code of the room
day *: Date (yyyy-mm-dd)(required)* → the day in which this price is charged
price *: Double (required)* → it's the total price for the reservation of this room for this day

childrenPrice : *Double (optional)* → price payed for the children hosted in this room for this day.

extraAdultsPrice : *Double (optional)* → total price paid for additional adults hosted in this room.
roomDescription: *String (optional)* → it's the name of the room
portalRoomDescription: *String (optional)* → the name of the room on the channel
rateId *:* *String (optional)* → rate plan code applied for this price
rateDescription*:* String (optional) → rate plan description for this price
portalRateDescription*:* String (optional) → rate plan channel description for this price

supplement : tag used to send informations (for every room) about supplements charged to the client for services he bought. **Optional**

roomId *: String (required)* → it's the code of the room to which this supplement applies
description : *String (required)* → description of this service type
roomsQuantity : *Integer (optional)* → it's the quantity reserved of this room
price *: Double (required)* → price of this supplement
number : *Integer (required)* → quantity of this service bought by the client (=0 if it can be quantified)
type *: Integer (optional)* → type of this supplement (0=daily, 1=by person, 2= by number)
persons : *Integer (optional)* → number of persons for which this service is bought.

guest :tag used to specify the guests information. **Optional**

roomId *:String (optional)* → it's the code of the room
firstName : *String (required)* → the first name of the guest
secondName : *String (required)* → the second name of the guest
email : *String (optional)* → email of the guest
phone : *String (optional)* → phone of the guest
address : *String (optional)* → address of the guest
zipCode : *String (optional)* → zip code of the guest
city : *String (optional)* → city of the guest
country : *String (optional)* → country of the guest

States table

Code	**Description**
2	*SUBMITTED*
3	*CANCELLED (no longer used)*
4	*CONFIRMED*
5	*REJECTED*
6	*NO SHOW*
7	*DELETED/CANCELLED*
8	*MODIFIED*

REMARK: RoomCloud tries to decode the room present in a reservation retrieved from a channel, with one of those defined in RoomCloud (room mapping). If it occurs, as usual, the roomId returned in the reservation message is the RoomCloud room identification code (I.e one of those you retrieve with the getRooms message).

Ex:

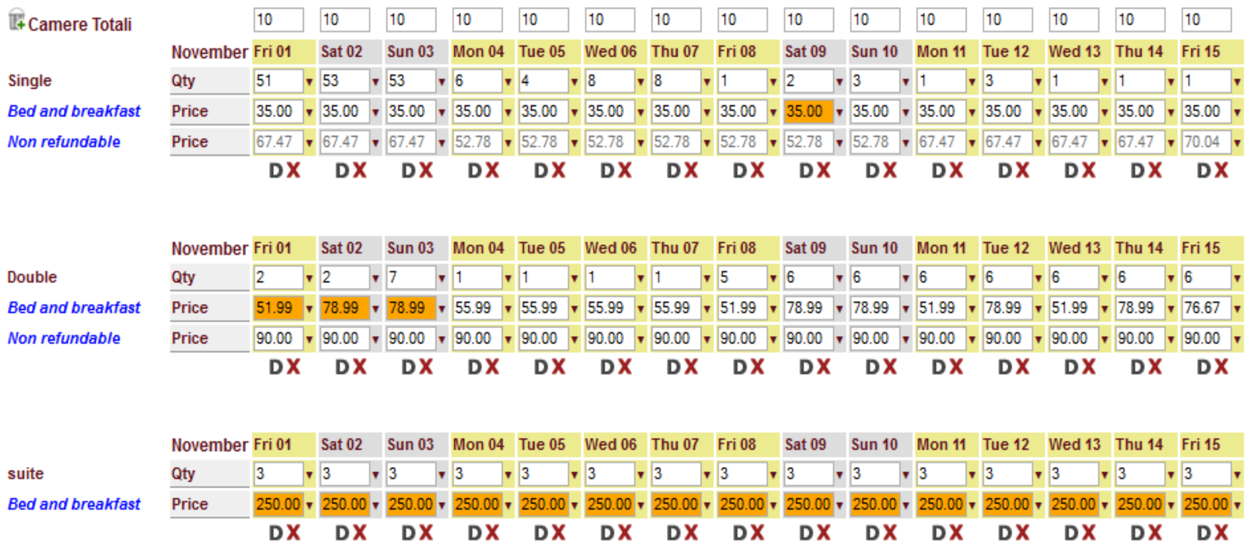
```
<Response>
   <reservation id="2" dlm="2012-10-18 00:05:53" portalId="124" checkin="2012-10-20" checkout="2012-10-21"
firstName="Mario"  lastName="Rossi"   telephone="341534562356" email="test.tecnes@tecnes.com" rooms="1"
price="110.0" prepaid="0.0"  currencycode="EUR" reservation_date="2012-10-15 00:05:50" status="2"
persons="3" children="1" zipcode="26013" address="Via delle bande nere 17" city="Crema" country="Italia"
paymentType="4" offer="" notes="Test reservation by Tecnes Milano" cancelbydetail="0" cancancel="0">
     <ccData  ccCode="123"  ccNumber="4408041234567893"  ccExpireDate="12/2012"  ccHolder="Test
holder"  />
     <room id="1"  checkin="2012-10-20" checkout="2012-10-21" rates_id="p" description="Single" quantity="1"
price="110.0" commission="0.0" adults="1" children="0" childrenPrice="0.0" status="2"/>
     <supplement roomId="1" description="Pizza in camera" roomsQuantity="1" price="10.0" type="2" number="1"
persons="0" />
     <dayPrice  roomId="1" rates_id="p" day="2012-10-20" price="100.0" childrenPrice="0.0" />
     <guest roomId="1"  name="Leopoldo Bindi" email="tecnes@tecnes.com" phone="8528965725" address="Via
Piranesi 26" zipCode="20100" city="Milan" country="Italy" />
  </reservation>
  <reservation id="1" dlm="2012-10-18 00:05:53" portalId="124" checkin="2012-10-19" checkout="2012-10-20"
client="Test Customer 1" telephone="341534562356" email="test.tecnes@tecnes.com" rooms="1" price="110.0"
currencycode="EUR" reservation_date="2012-10-15 00:05:50" status="4" persons="3" children="1"
zipcode="26013" address="Via delle bande nere 17" city="Crema" country="Italia" paymentType="4" offer=""
notes="Test reservation by Tecnes Milano" cancelbydetail="0" cancancel="0">
     <ccData  ccCode="123"  ccNumber="4408041234567893"  ccExpireDate="12/2012"  ccHolder="Test
holder"  />
```

```
    <room id="1" currency="EUR" checkin="2012-10-19" checkout="2012-10-20" roomId="1"  rates_id="p"
name="Single" quantity="1" price="110.0" commission="0.0" adults="1" children="0" childrenPrice="0.0"
status="4"/>
    <supplement  roomId="1"  description="Pizza in camera" roomsQuantity="1" price="10.0" type="2"
number="1" persons="0" />
    <dayPrice roomId="1"  rates_id="p" day="2012-10-19" price="100.0" childrenPrice="0.0" />
    <guest roomId="1"  firstName="Leopoldo" secondName="Bindi" email="test.tecnes@tecnes.com"
phone="8528965725" address="Via Piranesi 26" zipCode="20100" city="Milan" country="Italy" />
  </reservation>
</Response>
```

5 Read and Update Inventory

5.1 The RoomCloud Inventory

The RoomCloud Inventory is a calendar of room availabilities and rates stored on the local RoomCloud database.



Camere Totali		10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
	November	Fri 01	Sat 02	Sun 03	Mon 04	Tue 05	Wed 06	Thu 07	Fri 08	Sat 09	Sun 10	Mon 11	Tue 12	Wed 13	Thu 14	Fri 15
Single	Qty	51	53	53	6	4	8	8	1	2	3	1	3	1	1	1
Bed and breakfast	Price	35.00	35.00	35.00	35.00	35.00	35.00	35.00	35.00	35.00	35.00	35.00	35.00	35.00	35.00	35.00
Non refundable	Price	67.47	67.47	67.47	52.78	52.78	52.78	52.78	52.78	52.78	52.78	67.47	67.47	67.47	67.47	70.04
		DX	DX	DX	DX	DX	DX	DX	DX	DX	DX	DX	DX	DX	DX	DX

	November	Fri 01	Sat 02	Sun 03	Mon 04	Tue 05	Wed 06	Thu 07	Fri 08	Sat 09	Sun 10	Mon 11	Tue 12	Wed 13	Thu 14	Fri 15
Double	Qty	2	2	7	1	1	1	1	5	6	6	6	6	6	6	6
Bed and breakfast	Price	51.99	78.99	78.99	55.99	55.99	55.99	55.99	51.99	78.99	78.99	51.99	78.99	51.99	78.99	76.67
Non refundable	Price	90.00	90.00	90.00	90.00	90.00	90.00	90.00	90.00	90.00	90.00	90.00	90.00	90.00	90.00	90.00
		DX	DX	DX	DX	DX	DX	DX	DX	DX	DX	DX	DX	DX	DX	DX

	November	Fri 01	Sat 02	Sun 03	Mon 04	Tue 05	Wed 06	Thu 07	Fri 08	Sat 09	Sun 10	Mon 11	Tue 12	Wed 13	Thu 14	Fri 15
suite	Qty	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Bed and breakfast	Price	250.00	250.00	250.00	250.00	250.00	250.00	250.00	250.00	250.00	250.00	250.00	250.00	250.00	250.00	250.00
		DX	DX	DX	DX	DX	DX	DX	DX	DX	DX	DX	DX	DX	DX	DX

Fig 2: inventory

For each room type and each day it is possible to define:

1. the type of rates with relative prices
2. the amount of available rooms
3. the minimum stay limitation
4. the close on arrival limitation
5. the close on departure limitation

Each RoomCloud room type and rate type can be mapped to one or more "room type / rate plan" on different channels.

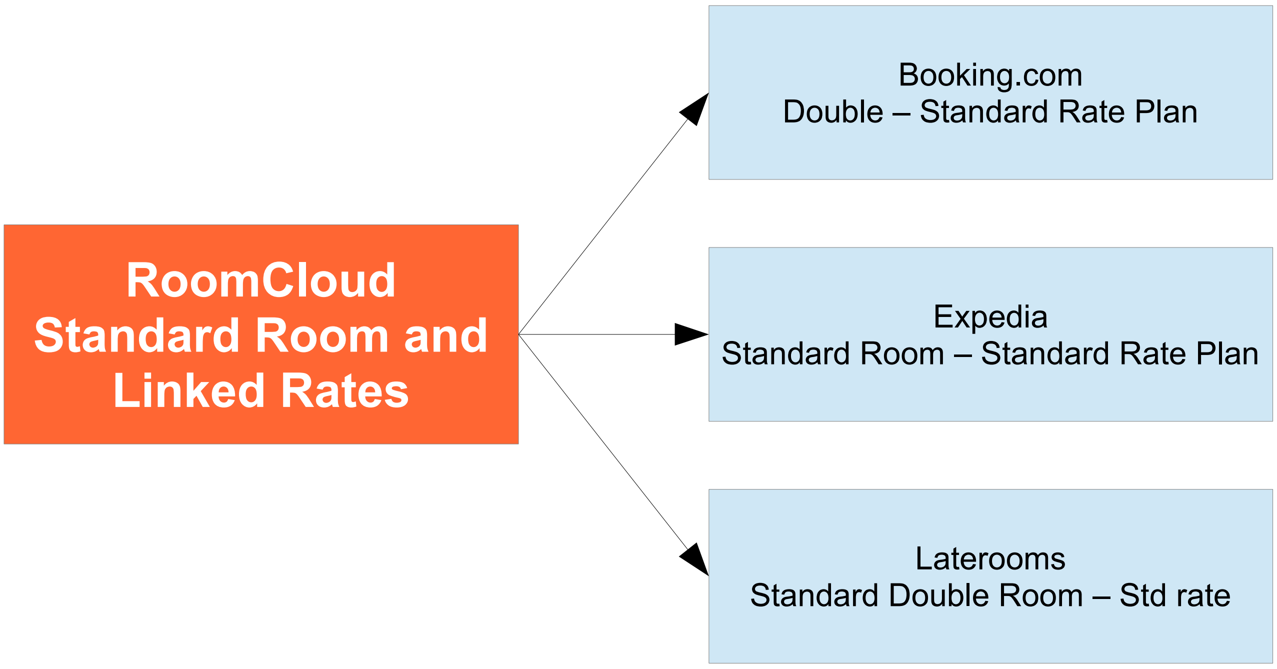

RoomCloud Standard Room and Linked Rates
Booking.com
Double – Standard Rate Plan
Expedia
Standard Room – Standard Rate Plan
Laterooms
Standard Double Room – Std rate

Fig 3: room type mapping

Every time the calendar of a room type and linked rate types are updated, the mapped "room types / rate plans" calendars on the channels are updated as well.

Each channel is represented by a portal element. The ***portal*** element is a complex type element characterized by several attributes and by other optional sub-elements.

RoomCloud XML API provides two methods to respectively read and update inventory data. These two method are:

1. view : get inventory data
2. modify : update inventory data

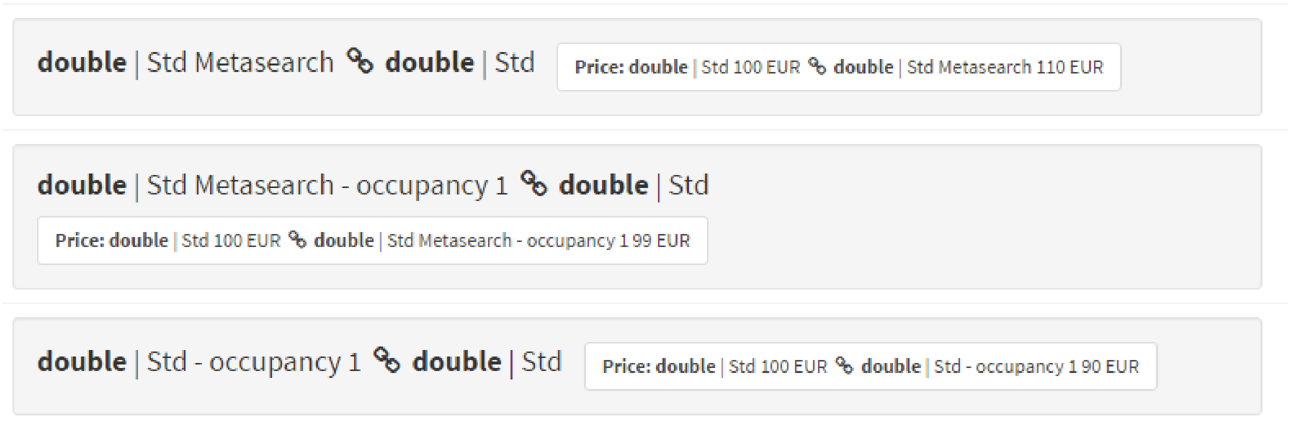
Structure and the usage of these two so important methods are showed in the next paragraphs.

5.2 Business Rules

Hotels can define one or more business rules to establish relationships between different room types and rate types (Room Rules) or between a portal/portalRate and the RoomCloud inventory (Website Rule).

With a Room Rule it is possible to link a room/rate to another room/rate, i.e. it is possible to calculate the availability, and/or the rate and/or the min stay of a room/rate combination starting from a "linked" room/rate combination.
Through a set of Room Rules it is possible to derivate all room/rates combinations from a base room/rate.

Fig 4: room rule example



double | Std Metasearch **double** | Std

Price: **double** | Std 100 EUR **double** | Std Metasearch 110 EUR

double | Std Metasearch - occupancy 1 **double** | Std

Price: **double** | Std 100 EUR **double** | Std Metasearch - occupancy 1 99 EUR

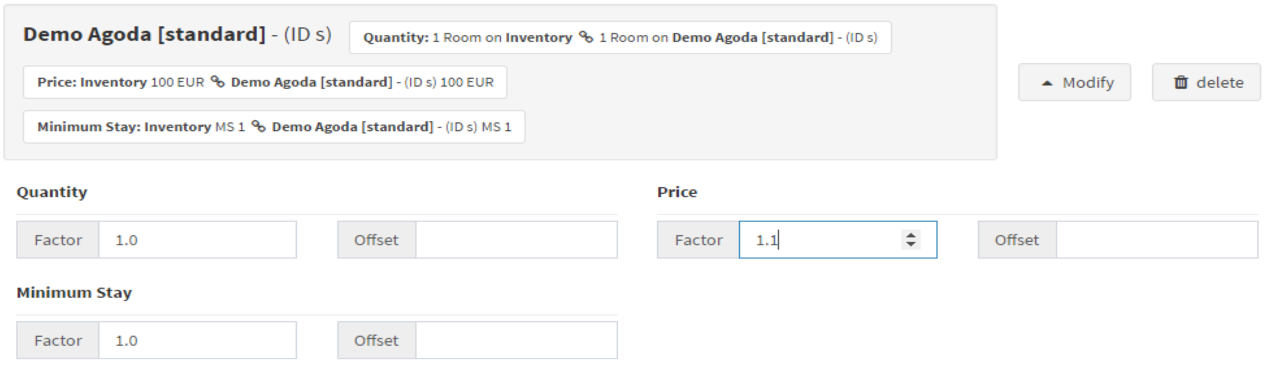
double | Std - occupancy 1 **double** | Std

Price: **double** | Std 100 EUR **double** | Std - occupancy 1 90 EUR

By default the ratio between the RoomCloud inventory and the mapped rooms/rates channels is 1 to 1.
For instance, looking at the example above, if we put in the RoomCloud inventory 100 Euro's on the Single Room and Standered Rate, 100 Euro's will be then sent to booking.com for the Double - Standard Rate Plan, to Expedia for the Standard room - Standard rate Plan and so on.

A website rule let to define different ratio and/or offsets

Fig 5: website rule example



Demo Agoda [standard] - (ID s)

Quantity: 1 Room on **Inventory** 1 Room on **Demo Agoda [standard]** - (ID s)

Price: Inventory 100 EUR **Demo Agoda [standard]** - (ID s) 100 EUR

Minimum Stay: Inventory MS 1 **Demo Agoda [standard]** - (ID s) MS 1

Quantity

Factor	1.0	Offset	

Price

Factor	1.1	Offset	

Minimum Stay

Factor	1.0	Offset	

The message to retrieve the Business Rules for an hotel is getBusinessHotel

The ***getBusinessRule*** element has the following attribute:

hotelId *: Integer (required) → RoomCloud hotel code*

The response message is a list of Business Rules.
The structure is:

businessRule *a*ttributes list:
id *String (required)* → the Business Rules ID
name *String* → the business rule name

active *true/false (optional, default value false)* → true only if the business rule is active

roomRules : element containing room rules

roomRule attribute list:
linkedRoomId *:String (required)* → it's the code of the linked room
linkedRateId *:String (required)* → it's the code of the linked rate
linkedOccupancy *: Integer (optional)* → it's the linked occupancy rate
linkedAdult *: Integer (optional)* → it's the linked adult rate
linkedChild *: Integer (optional)* → it's the linked child rate code
roomId *:String (optional)* → it's the room code
rateId *:String (required)* → it's the rate code
occupancy *: Integer (optional)* → it's the linked occupancy rate
adult *: Integer (optional)* → it's the linked adult rate
child *: String (optional)* → it's the child rate code
qtyFactor : *Double (optional)* → availability multiplication factor
qtyOffset : *Integer (optional)* → value to add to the original availability (it's applied after the qtyFactor)
priceFactor : *Double (optional)* → price multiplication factor
priceOffset : *Double (optional)* → value to add to the original price (it's applied after the priceFactor)
msFactor : *Double (optional)* → minimum stay multiplication factor
msOffset : *Integer (optional)* → value to add to the original min stay (it's applied after the msFactor)
useCoa **:** *true/false (optional, default value false)* → if true the room/rate takes the same coa as the linked room/rate
useCod **:** *true/false (optional, default value false)* → if true the room/rate takes the same cod as the linked room/rate

portalRules : element containing portal rules

portalRule attribute list:
portalId *:String (optional)* → it's the room code
portalRateId *:String (required)* → it's the rate code
qtyFactor : *Double (optional – default value=1)* → availability multiplication factor
qtyOffset : *Integer (optional)* → value to add to the original availability (it's applied after the qtyFactor)
priceFactor : *Double (optional – default value=1)* → price multiplication factor
priceOffset : *Double (optional)* → value to add to the original price (it's applied after the priceFactor)
msFactor : *Double (optional – default value=1)* → minimum stay multiplication factor
msOffset : *Integer (optional)* → value to add to the original min stay (it's applied after the msFactor)

5.3 The Availability element

Availability element represents the basic block of information for a defined room on a defined day. Basically speaking, the Inventory is a sum of availability objects. An availability contains the number of rooms of that type on sale and the list of rates to be applied.

The structure is:
```
<availability day="2009-12-02"  roomId="1" currency="EUR">
    <rate rateId="123" price="300" minimumStay=""/>
    <rate/>
</availability>
```

availability *a*ttributes list:
day *Date (yyyy-mm-dd)(requested)* → the day which the availability refers to
roomId *String (requested)* → the RoomCloud room code
currency *String* → it's the currency used for the price (possible values are EUR, USD etc)
quantity Integer → represents the number of available rooms

rate *a*ttributes list**:**
price *Double* → represents the price of the room
currency *String* → it's the currency used for the price (possible values are EUR, USD etc)
minimumStay Integer → represents the minumum stay
coa *true/false* → Close on arrival
cod *true/false* → Close on departure
closed *true/false* → rate closed to sales

occupancy *a*ttributes list**:**
id : *Integer (required) → identifier of this occupancy price*
rateId : *Integer (required) → identifier of the related rate*
currency *String* → it's the currency used for the price (possible values are EUR, USD etc)
price *Double* → represents the price of the room with this occupancy

adult *a*ttributes list**:**
id : *Integer (required) → identifier of this children price*
rateId : *Integer (required) → identifier of the related rate*
currency *String* → it's the currency used for the price (possible values are EUR, USD etc)
price Double → represents the price of the adult supplement

child *a*ttributes list**:**
id *String (requested)* → the child code
rateId : *Integer (required) → identifier of the related rate*
currency *String* → it's the currency used for the price (possible values are EUR, USD etc)
price Double → represents the price of the child supplement
from : *Integer (required) → starting age of child*
to : *Integer (required) → ending age of child*

5.4 View

The ***view*** element has the following attributes:

hotelId *: Integer (required) → RoomCloud hotel code*
startDate *: Date (yyyy-mm-dd)(required)*
endDate *: Date (yyyy-mm-dd)(required)*

The response to a view call is a list of availability elements

Ex:
```
<Request userName="MyLogin" password="MyPassword">
  <view  hotelId="1111" startDate="2012-10-12" endDate="2012-10-12"/>
</Request>
<Response>
  <availability day="2014-10-10"  roomId="2091"  quantity="5"  >
    <rate rateId="7420" currency="USD" price="80.0" minimumStay="1" coa="false" cod="false"  />
    <occupancy rateId="7420" id="1" currency="USD" price="70.0" />
    <adult rateId="7420" id="3" currency="USD" price="20.0" />
    <adult rateId="7420" id="4" currency="USD" price="15.0" />
    <child rateId="7420" id="0" from="2" to="7" currency="USD" price="5.0" />
    <child rateId="7420" id="1" from="2" to="7" currency="USD" price="3.0" />
    <rate rateId="7421" currency="USD" price="70.0" minimumStay="1" coa="false" cod="false"  />
    <occupancy rateId="7421" id="1" currency="USD" price="50.0" />
  </availability>
</Response>
```

5.5 Modify

The ***modify*** element has the following attributes:

hotelId *: Integer (required) → RoomCloud hotel code*
startDate *: Date (yyyy-mm-dd)(required)*
endDate *: Date (yyyy-mm-dd)(required)*

isScheduled : boolean (true/false) (optional, default=false) → set to true if you want that updates run in background

businessRuleId : String (optional) → the business rule id to apply; if omitted, or empty, the Active Business Rule will be applied at Portals Level only.

and contains a list of availability elements.
Every attribute of each Availability element is optional except for day and roomId.

Ex:
```
<Request userName="MyLogin" password="MyPassword">
  <modify  hotelId="1111" startDate="2012-10-12" endDate="2012-10-13">
    <availability day="2012-10-12" roomId="1" quantity="5">
      <rate rateId="2782" currency="EUR" price="50.0" minimumStay="1" closed="true"  />
      <occupancy rateId="7420" id="1" currency="USD" price="70.0" />
      <adult rateId="7420" id="3" currency="USD" price="20.0" />
      <child rateId="7420" id="1" from="2" to="7" currency="USD" price="3.0" />
      <rate rateId="7421" currency="USD" price="70.0" minimumStay="1" coa="false" cod="false"  />
      <occupancy rateId="7421" id="1" currency="USD" price="50.0" />
    </availability>
  </modify>
</Request>
```

The result of a **Modify** call can be:

a) completely wrong: an error has occurred during the Inventory update procedure; no data has been saved and forwarded to any channel.
b) partially successful: Inventory has been successfully updated, but an error has occurred while sending data to one or more channel.
c) completely successful: Inventory and channels have been successfully updated.

In the first and second case an error element has returned as child of the Response element

Ex a):
```
<Response>
  <error message="Problem have been encountered in the modification of the inventory"/>
</Response>
```

In this case the error element contains the list of portals with update errors.

Ex b):
```
<Response>
        <error message="Problem on portals update">
        <portal id="124" message="Login Failed" code=”2101”>
        </portal>
        <portal id="3" message="error on update of room single" code=”2301”>
        </portal>
```

```
                ...
        </error>
</Response>
```

Ex c):

```
<Response>
        <ok/>
</Response>
```

REMARKS
Update process speed
When you send a modify request, RoomCloud updates the inventory and immediately forwards data to the channels. The process can take up to 10 minutes or more, depending on several factor like how many channels the hotel is connected to, the number of rooms to update etc..
Using the attribute isScheduled="true", RoomCloud put the update request into a queue and immediately releases the connection. The communication process between the caller and RoomCloud is thus speeded up, but the caller is not informed about eventual errors. Inside the RoomCloud website, it is possible to activate the "Alert" feature that let the system send an email in case of modification error.

6 Portal Error Handling

6.1 Error codes

Error codes are 4 digits integer.
The error codes starting with 1 are referred to Roomcloud subsystem errors

Code	Description	Explanation
1000	Generic error	Unclassified error.
1001	Internal Timeout Error	These are internal system error at channel level. Partner can try to resend the message again in a few minutes. An incremental retry strategy can be put in place.
1002	Internal Error	
1003	Internal Server Error	
1004	Database error	
1010	Malformed XML	The message sent by the partner to Roomcloud is not compliant with the Roomcloud specifications.
1011	Invalid API error	
1012	Configuration Error	This error happens whenever one of the mapping codes used in the message (room type id, rat plan id etc) does not belong to the hotel.
1100	Auth denied	These are authorization errors. Partner should avoid to retry to send the message as is since an operator action is required to remove the cause of the error
1101	Wrong Credentials	
1103	Insufficient access rights	
1110	Too Many requests	Hotel has been temporarily blocked since Roomcloud has received tomany request in the last hour. Hotel is automatically unblocked every hour
1111	Blocked account	Hotel has been blocked. To unblock it the partner should contact our support team

The error codes starting with 2 are referred to channel subsystem errors.

Code	Description	Explanation

2001	Internal Error	These are internal system error at channel level. Partner can try ro resend the message again in a few minutes. An incremental retry strategy can be put in place.
2003	Internal Server Error	
2002	Internal Timeout Error	
2004	Database error	
2010	Malformed XML	The message sent by Roomcloud to the channel is not compliant with the channel specifications. The Roomcloud team must operate to solve the related issue.
2011	Invalid API Error	
2100	Auth denied	These are authorization errors. Partner should avoid to retry to send the message as is since an operator action is required to remove the cause of the error
2101	Wrong Credentials	
2103	Insufficient access rights	
2202	Hotel not found	This error happens whenever one of the mapping codes used in the message (room type id, rat plan id etc) does not belong to the account. An operator action is required to remove the cause of the error Message example: `You either specified an invalid hotelID or your account is not linked to this hotel`
2203	Mapping Error	
2300	Allotment restrictions	It means that there is a close out request for one or more room type with a base allotment.
2301	Rate restrictions	It happens when there exist some business restrictions like for instance: • lower or upper threshold for rates • upper threshold for minimum stay
2302	Min Stay Restrictions	
2303	Availability restrictions	
2304	Business restrictions	

7 Query Inventory (booking engine API)

This paraghraph explains the functions to query the public data of hotel rooms, in a similar way to interrogating a hotel booking engine.

7.1 Get available hotels

It shows the list of hotels thst use roomcloud booking engine.
The ***getAvailableHotels*** element has the following attributes:

lang *: String (en,it,de,fr...)(optional) → two letter language of the hotel texts*
shop *: String (optional)* → the shop code (please contact us in order to obtain your personal shop code)

Ex:

```
<Request userName="MyLogin" password="MyPassword">
        <getAvailableHotels lang="en" shop="7" />
</Request>
```

The result of a ***getAvailableHotels*** call can be a list of properties:

property
id *String (required)* → the code of the hotel
name *String (required)* → the name of the hotel
address *String (required)* → the address of the hotel
postcode *String (required)* → the postcode of the hotel
city *String (required)* → the city of the hotel
country *String (required)* → the country code of the hotel
latitude *Decimal (optional)* → the latitude of the hotel
longitude *Decimal (optional)* → the longitude of the hotel
phone *String (required)* → the phone of the hotel

category *String (required)* → the category of the hotel
shop *: String (optional)* → the shop code (please contact us in order to obtain your personal shop code)

Ex:

```
<Response>
        <properties>

        <property id="688">
        <name>SoLoMoKi  Apartments</name>
        <address>Corso Buenos Aires 18</address>
        <postcode>20100</postcode>
        <city>Milano</city>
        <country>IT</country>
        <latitude>45.477541</latitude>
        <longitude>9.208127</longitude>
        <phone>+390267101036</phone>
        <category>Hotel</category>
        <shop>7</shop>
         </property>

        <property id="689">
        <name>Hotel Splendor</name>
        <address>Via Piranesi 26</address>
        <city>Milano</city>
        <country>IT</country>
        <latitude>45.467541</latitude>
        <longitude>9.218127</longitude>
        <phone>+390267101036</phone>
        <category>Hotel</category>
        <shop>7</shop>
         </property>

        </properties>
</Response>
```

7.2 Get available rooms by checkin and nights

It queries a list of hotels for available rooms of roomcloud booking engine.
The ***getAvailableRooms*** element has the following attributes:

property
 id *String (required)* → the code of the hotel
checkin *Date (yyyy-mm-dd)(required) → the arrival date*
nights *Integer (required) → the number of nights stay*
adults *Integer (optional) → the number of adults*
children *Integer (optional) → the number of children*
lang *String (optional) → the language used*
shop *: String (optional)* → the shop code (please contact us in order to obtain your personal shop code)

Ex:

```
<Request userName="MyLogin" password="MyPassword">
  <getAvailableRooms  checkin="2012-10-12" nights="2" adults="2" children="1"  lang="en" shop="7">
    <properties>
        <property id="144" />
    </properties>
  </getAvailableRooms>
</Request>
```

The result of a ***getAvailableRooms*** call can be a list of properties each containing the following attributes:

baseRate *rate that contains the best net price (less tax and fees)*
  id *String (required)* → the code of the best rate among all room bundles (see rateId in roomBundle)
  currency *String (required)* → the currency code
tax *Double* → amount of taxes
otherFees *Double* → additional taxes
pointOfSale
  id *String (required)* → the string identifier the booking engine
  url *String (required)* → the url of the query on the booking engine

roomBundle *list of rooms available for the query*
  roomId *String (required) the room code*
  rateId *String (required) the rate code*
  name *String (required) the room description*
  description *String (required) the room long description*
  occupancy Integer *(required) → the number of beds*
  baserate *Double (required)* → represents the net price of the room (less tax and fees)
  currency *String (required)* → the currency code
  tax *Double (required)* → room taxes
  otherFees *Double (required)*→ room additional taxes

Ex:

```
<Response>
        <properties>
        <property id="2782">
        <checkin>2014-04-19</checkin>
        <nights>2</nights>
        <adults>2</adults>
        <children>1</children>
        <baseRate id="165" currency="EUR">100.00</baseRate>
        <tax>0</tax>
        <otherFees>0</otherFees>
        <shop>7</shop>
        <pointOfSale id="roomcloud.net" url="http://www.roomcloud.net/be/se1/hotel.jsp?
shop=7&amp;hotel=144&amp;adults=2&amp;children=1&amp;start_day=19&amp;start_month=04&amp;start_ye
ar=2014&amp;end_day=21&amp;end_month=04&amp;end_year=2014&amp;lang=en"/>
        <roomBundle>
        <roomId>497</roomId>
        <rateId>165</rateId>
        <name>Single room[Overnight stay]</name>
        <description>SINGLE ROOM[Overnight stay]</description>
        <occupancy>1</occupancy>
        <baserate currency="EUR">100.00</baserate>
        <tax>0</tax>
        <otherFees>0</otherFees>
        </roomBundle>

        <roomBundle>
        <roomId>497</roomId>
        <rateId>166</rateId>
        <name>Single room[Bed and breakfast]</name>
        <description>SINGLE ROOM[Bed and breakfast]</description>
        <occupancy>1</occupancy>
        <baserate currency="EUR">110.00</baserate>
        <tax>0</tax><otherFees>0</otherFees>
        </roomBundle>

        </property>
        </properties>
</Response>
```

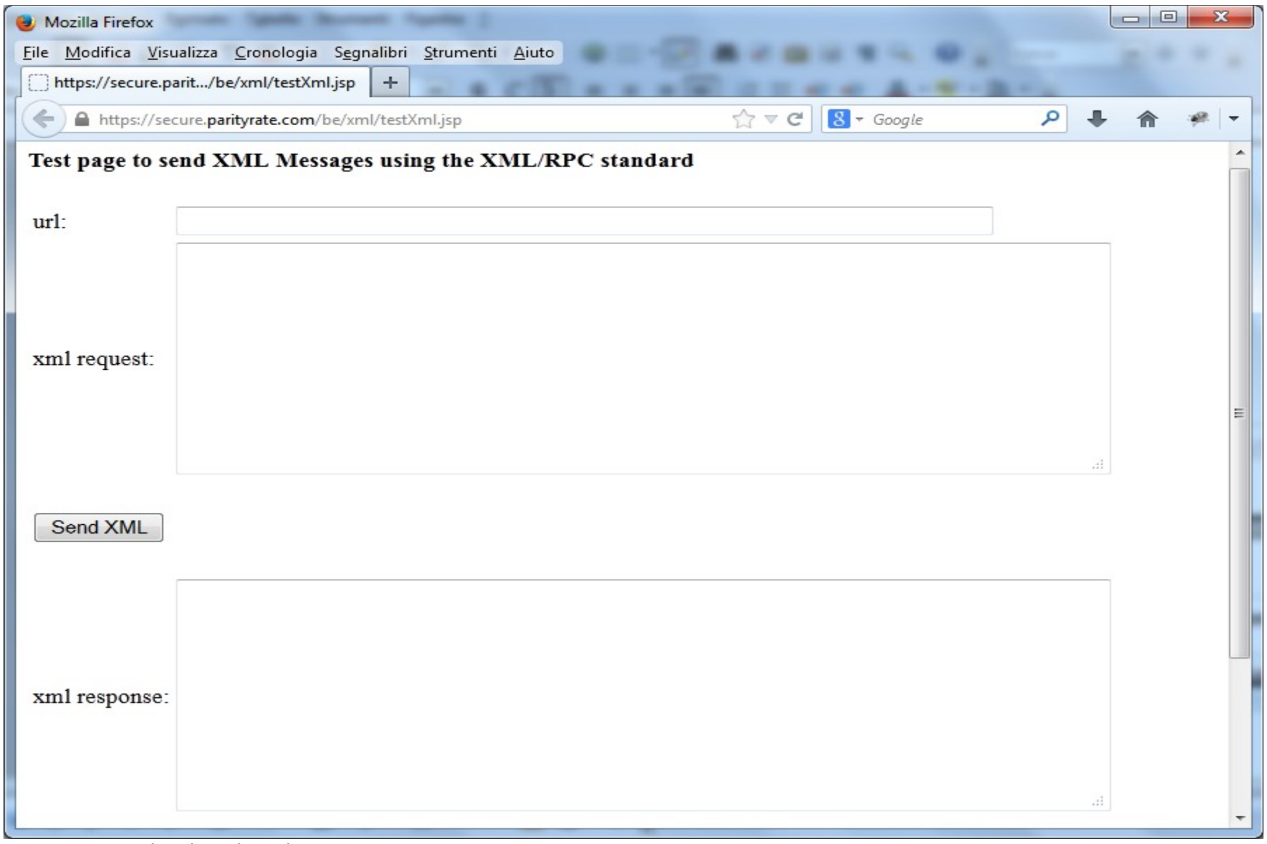
8 Useful things

8.1 Test Page

In order to test the XML messages you can use our test page:

https://apitest.roomcloud.net/be/xml/testXml.jsp

See example below.


Mozilla Firefox
File Modifica Visualizza Cronologia Segnalibri Strumenti Aiuto
https://secure.parit.../be/xml/testXml.jsp
https://secure.parityrate.com/be/xml/testXml.jsp
Google
Send XML

Test page to send XML Messages using the XML/RPC standard

url:

xml request:

xml response:

Fig 6: RoomCloud xml exchange message test page

Put the url https://apitest.roomcloud.net/be/search/xml.jsp in the "url" field.
Put the message you want to try in the "xml request" text area and click on Send XML.
The response will be displayed in the "xml response" text area.

8.2 Contacts

Email to: xmlhelp@tecnes.com