RoomWizard™

Application Programming Interface

Part number 370-0181-00

Table of Contents

Overview

RoomWizard is a web-based scheduling system for meeting rooms and other shared spaces. RoomWizard incorporates the use of touch-screen appliances that are situated outside of the room, confirming the schedule and location of meetings.

This document describes an Application Programmer Interface (API) which devices must adhere to in order to host or manipulate RoomWizard reservations. It defines the behavior of a “connector” which is a proxy for target room reservations systems. Depending on RoomWizard’s deployment, the API either acts as a connector and can be manipulated using this API, or it acts as a client of an external connector and expects to use this API to manipulate room bookings on the server-based corporate calendaring system (e.g. Microsoft Exchange or Lotus Domino).

Introduction

This document describes the behavior that is expected from a software module whose purpose is to make room reservations that are available for retrieval and manipulation by RoomWizard systems, as performed by a target room reservation system. We call such a software module a “connector.”

The connector is expected to offer an HTTP-based interface, irrespective of the capabilities of the target room reservation system.

This API is documented for the following audiences:

- Connector writers
- Client writers

The API may contain some comments or examples about the behavior of a particular client (RoomWizard itself) or a particular connector (RoomWizard’s internal database).

Notational Conventions

Any Backus-Naur Form (BNF) notations used in this document are exactly the same as described in section 2.1 of RFC2068.

The key words "MUST," "MUST NOT," "REQUIRED," "SHALL," "SHALL NOT,", "SHOULD," "SHOULD NOT," "RECOMMENDED," "MAY," and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

- **MUST** This word, or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the specification.
- **MUST NOT** This phrase, or the phrase "SHALL NOT", mean that the definition is an absolute prohibition of the specification.
- **SHOULD** This word, or the adjective "RECOMMENDED", mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
- **SHOULD NOT** This phrase, or the phrase "NOT RECOMMENDED" mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full

implications should be understood and the case carefully weighed before implementing any behavior described with this label.

- **MAY** This word, or the adjective "**OPTIONAL**," mean that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation which does not include a particular option MUST be prepared to interoperate with another implementation which does include the option, though perhaps with reduced functionality. In the same vein an implementation which does include a particular option MUST be prepared to interoperate with another implementation which does not include the option (except, of course, for the feature the option provides.)

Terminology

URI/URL: A Uniform Resource Identifier and Uniform Resource Locator, respectively. These terms (and the distinction between them) are defined in the RFC2396.

Property: A name/value pair that contains descriptive information about a resource.

Connector: The software module implementing this API

Client: A software module or entity that uses this API, e.g.

RoomWizard: Single board computer based display device that displays information about room bookings, see www.polyvision.com.

Target System: The Room Booking system for which the connector acts as a proxy, e.g. Microsoft Exchange or Lotus Domino.

Request: HTTP GET or POST request as defined in RFC2068.

Response: HTTP response, within the context of this API, containing an XML structure.

Room id: The identifier of a room.

Booking id: The identifier of a booking.

Historic booking: A booking that is wholly in the past, its end time is prior to the target system's current time.

Active booking: A booking that is currently going on. Its start time is prior to or corresponds with the target system's current time and its end time is after the target system's current time.

Future booking: A booking whose start time is after the target system's current time.

Guidelines

PolyVision adheres to the following guidelines:

- PolyVision will not change the existence or meaning of existing attributes when a new version of this API is introduced. This should insure backwards compatibility between API revisions.

Client writers must adhere to the following guidelines:

- The client must ignore any XML return data that it does not understand. This should ensure that clients suffer no ill effect of the introduction of newer connector software that may offer a richer return dataset.

Connector writers must adhere to the following guidelines:

- The connector must ignore any HTTP request arguments that it does not understand. This should ensure that connectors suffer no ill effect of the introduction of newer or different client software.
- The connector must not violate any integrity constraints of the target system. This includes both integrity constraints that are actively enforced by the target system and integrity constraints that the target system expects connecting software to uphold.

  Example: a given target system may have an SQL database and a custom client. It may rely on the clients to ensure that no reservation clashes occur; a connector for that system must then also make sure that it does not create reservations that clash.
- Some connectors may only have to proxy for only one room on the target system. In this case the room_id parameter should be ignored and the connector must apply all operations to that particular room.

Assumptions

This API is based on a number of assumptions about the behavior of a target system. It will be very hard to create a connector for a target system that does not satisfy these assumptions. The assumptions are the following:

- The **Room id** must be unique for every room per connector.
- The **Booking id** must be unique for every booking for a given room.
- **Booking ids** for historic and active bookings must not change.
- **Booking ids** for future bookings should not change.
- **Booking ids** should not be reused for a given room.
- **Bookings** never end before they start.

General

This document details mechanisms for extracting information from the connector software, such as information about room reservations and the means of manipulating these room reservations. All of these operations are assumed to take place in the form of a request and response process.

This chapter contains the definitions common to these operations. The individual operations are further detailed in the following chapters.

Transport mechanism

The transport mechanism chosen for communication between a client and a connector is HTTP.

The connector must interpret and respond to HTTP GET requests, it may interpret and respond to HTTP POST requests.

The connector must be able to interpret requests in the HTTP default character set of "ASCII. "[1] The connector must encode its responses in the ISO-8859-1 characters set.

Request syntax

A common URL is used for all requests. All operations use this common URL as well as all clients that use a specific connector.

The type of request is determined by the "command" parameter.

```
Request =     Request-URL ? "command=" command
Request-URL = <URL on the server on which the connector is
              listening>
Command =     ("about_connector")                    |
              ("get_bookings" "&" get_bookings_args)|
              ("add_booking" "&" add_booking_args)  |
              ("edit_booking" "&" edit_booking_args)|
              ("delete_booking" "&" delete_booking_args)
```

1. It is highly likely that a future version of the API will specify that a connector must be able to interpret HTTP GET and POST requests that contain the request in the ISO-8859-1 character set, see http://www.w3.org/TR/1999/REC-html401-19991224/html40.txt (section 17.13.4) and http://www.ietf.org/rfc/rfc1738.txt.

The location of the Request-URL is specified on the RoomWizard client using its setup pages. RoomWizard then prepends the appropriate command to this string and uses this to contact the connector.

For information about RoomWizard setup pages, refer to the RoomWizard *System Manager Guide* available on-line at:

www.polyvision.com/support/downloads-roomwizard.asp#guides

Example

In RoomWizard, the Request-URL is defined as:

```
http://exchangeserver1/cgi-bin/rwconnector
```

If the RoomWizard wanted to perform a "delete_booking" request (which uses parameters “room_id” and “booking_id ,” with values of "Room 1" and "2002/03/19 12:00-13:00" respectively) it would construction the following URL:

```
http://exchangeserver1/cgi-bin/
rwconnector?command=delete_booking&room_id=Room+1&booking_id
=2002%2F03%2F19+12%3A00-13%3A00
```

and use this to request the connector information.

Response syntax

Responses to the defined requests are carried in the bodies of the HTTP responses. These responses carry an XML structure that contains all response information.

Structure

All responses have the following general XML structure:

```
<?xml version="1.0"?>
<kwe:result
   xmlns:kwe=”http://www.appliancestudio.com/kwe/1.0/”
   xmlns:rb=”http://www.appliancestudio.com/rb/1.0/”>
      <kwe:date>date</kwe:date>
      <kwe:time>time</kwe:time>
      <kwe:result_code>numeric result code</kwe:result_code>

      [<rb:result_code>numeric result code</rb:result_code>]

      individual command tags go here …

      [<kwe:debug>]
        debug text goes here …
      [</kwe:debug>]
</kwe:result>
```

Namespaces

The response structure uses 2 XML namespaces:

- **kwe**: short for “Knowledge Workspace Environment”, this is the space in which information common to all requests is contained, this namespace is intended to be used for many more and varied devices and connectors.
- **rb**: short for “Room Booking”, this is the space in which result information particular to the booking of meeting rooms should be contained.

A connector writer may add other namespaces to the result for development or debugging purposes.

Properties

The result XML structure for every command always has the following properties defined:

- **kwe:date**: The date portion of the time on the targetsystem (required)
- **kwe:time**: The time portion of the time on the target system (required)
- **kwe:result_code**: The general result code for the operation (required)

The following optional tag is defined:

- **kwe:debug**: This section is optional and may contain debugging information.
- **rb:result_code**: The Room Booking result code for the operation (optional, must not appear when the kwe:result_code is “0”)

Any/all commands may provide debug information to allow the functionality to be inspected via a standard HTTP request (using a web browser, for example).

KWE Result codes

The following result codes are common to all operations that operate in the KWE namespace:

0	The operation completed successfully
1	Unknown Error
2	Application error, a more detailed error description may be found in an application specific result_code.
3	Unsupported character set

Several general failures, such as server failure, URL not found and authentication failures are reported by the HTTP transport mechanism itself.

RB Result codes

The following result codes are common to all operations that operate in the RB namespace:

100 Command not found

101 Missing parameter(s)

102 Incorrect parameter(s)

103 Incorrect room password

Individual commands may add more result codes for their individual responses.

Date/Time formats

Date properties must be in the format yyyyMMdd (year, month, day).

Time properties must be in the format hhmmss (hours, minutes, seconds) using a 24 hour clock (military time).

Date parameters for the **get_bookings** request (not response) may also be "today;" the connector must map this onto the current day. Time parameters for the **get_bookings** request (not response) may also be "now": the connector must map this onto the current time.

Example

The following is a valid request:

```
http://exchangeserver1/cgi-bin/rwconnector?com-
mand=get_bookings&room_id=43&range_start_date=today&range_st
art_time=now&range_end_date=20020405&range_end_time=220000
```

This requests the bookings from the current time until 22:00:00 on April 5th 2002 for the room with id 43.

Security

This section describes the API's approach to authentication and encryption.

Authentication

The connector may provide for the following three means of authentication:

- Access to the connector may be restricted by using HTTP "basic" authentication [RFC2068 section 11.1]. If a connector supports basic authentication there must be exactly one username and password that allows this authentication to be successfully completed.
- Access to the individual rooms may be restricted by requiring a correctly filled in **room_password** request argument.
- Access to individual bookings may be restricted by requiring a correctly filled in **booking_password** request argument.

Basic Authentication

If a connector supports "basic" authentication and this authentication is enabled, the connector must:

- authenticate requests that contain the correct authentication headers already
- challenge and authenticate unauthorized requests

When the connector supports basic authentication it must not allow any request to succeed unless the client has successfully authenticated.

How basic authentication is enabled and configured is a matter of connector configuration and/or target system configuration.

Room Authentication

If a connector supports "per room" authentication it must act appropriately for every room that is configured to be "protected". This is defined as allowing room booking inspection, but denying room booking manipulation to non-authorized parties.

If a room is configured to be protected the connector must:

- allow any **get_bookings** request to be services

- use the **room_id** and **room_password** parameters to get access to the target system's bookings for that room for the **add_booking**, **edit_booking** and **delete_booking** commands. These commands should be denied if the **room_id** or the password is invalid.

Example

In some cases a target system may only offer blanket protection of rooms: either all operations are permitted or none. In that case the connector could be configured with passwords for all rooms.

For **add_booking**, **edit_booking** and **delete_booking** operations it should use the **room_password** from the request, for the **get_bookings** operation it should fill in the appropriate password, request the bookings from the target system and return the results to the client.

Booking Authentication

If a connector supports "per booking" authentication it must deny requests for **edit_booking** and **delete_booking**. If the client has not provided the correct password in the **booking_password** parameter the client should be denied.

Encryption

This version of the API specification does not mandate a connector writer to support encrypted HTTP traffic. Future versions of the API are likely to add support for Secure Socket Layer (SSL) encryption.

COMMANDS

This chapter describes the commands that the connector must support.

ABOUT_CONNECTOR

Name: about_connector

Description: This command returns information about the connector itself and the server date and time.

Request syntax:

Requires no further specification.

Result syntax:

```
<?xml version="1.0"?>
   <kwe:result
      xmlns:kwe=”http://www.appliancestudio.com/kwe/1.0/”
      xmlns:rb=”http://www.appliancestudio.com/rb/1.0/”>
      <kwe:date>date</kwe:date>
      <kwe:time>time</kwe:time>
      <kwe:result_code>numeric result code</kwe:result_code>
      [<rb:result_code>numeric result code</rb:result_code>]
      <kwe:connector>
      <kwe:name>Connector description</kwe:name>
      <kwe:version>Connector version</kwe:version>
      <kwe:short>Short name for connector</kwe:short>
         {<kwe:api name=”API name” version="API version"/>}
      </kwe:connector>
</kwe:result>
```

Result properties:

The result XML structure for every command always has a **kwe:connector** block with the following properties defined:

kwe:name: description of the connector (required)

kwe:version: version of the connector software(required)

kwe:short: short name (maximum 16 characters) for the connector e.g. Exch55_beta2

kwe:api: API supported by the connector (required, at least one entry, may be multiple entries)

Example

Request:

```
http://exchangeserver1/cgi-bin/
rwconnector?command=about_connector
```

Response:

```
<?xml version="1.0"?>
   <kwe:result
      xmlns:kwe=http://www.appliancestudio.com/kwe/1.0/
      xmlns:rb="http://www.appliancestudio.com/rb/1.0/">
       <kwe:date>20021231</kwe:date>
       <kwe:time>125930</kwe:time>
       <kwe:result_code>0</kwe:result_code>
       <kwe:connector>
           <kwe:name>Exchange RoomWizard Connector (c)
            Appliance Studio</kwe:name>
          <kwe:version>1.0.1b</kwe:version>
          <kwe:short>Exch55_beta2</kwe:short>
          <kwe:api name="Room Booking API" version="1.0"/>
          <kwe:api name="Room Booking API" version="1.1"/>
       </kwe:connector>
</kwe:result>
```

In this example the “Exchange RoomWizard Connector (c) Appliance Studio” connector supports both the 1.0 and 1.1 versions of this Room Booking API.

GET_BOOKINGS

Name: get_bookings

Description: This command returns room bookings for a given room that fall within a time period that is specified in the request.

Request syntax:

```
get_bookings_args = "room_id" "=" room_id "&"
                    "range_start_date" "=" range_start_date "&"
                    "range_start_time" "=" range_start_time "&"
                    "range_end_date" "=" range_end_date "&"
                    "range_end_time" "=" range_end_time
room_id = <The unique identifier for a room>
range_start_date = ( 4DIGIT 2DIGIT 2DIGIT ) | "today"
range_start_time = ( 2DIGIT 2DIGIT 2DIGIT ) | "now"
range_end_date = ( 4DIGIT 2DIGIT 2DIGIT ) | "today"
range_end_time = ( 2DIGIT 2DIGIT 2DIGIT ) | "now"
```

Request arguments:

Room_id: This identifier uniquely identifies a room within the connector (required)

Range_start_date: Start day of the request time period (required)

Range_start_time: Start time of the request time period (required)

Range_end_date: End day of the request time period (required)

Range_end_time: End time of the request time period (required)

Result syntax:

```
<?xml version="1.0"?>
  <kwe:result
    xmlns:kwe="http://www.appliancestudio.com/kwe/1.0/"
    xmlns:rb="http://www.appliancestudio.com/rb/1.0/">
    <kwe:date>date</kwe:date>
    <kwe:time>time</kwe:time>
    <kwe:result_code>numeric result code</kwe:result_code>
    [<rb:result_code>numeric result code</rb:result_code>]
    <rb:bookings room_id="room identifier">
```

```
[<rb:booking booking_id="booking identifier"
    confidential=("yes"|"no")
    password_protected=("yes|"no")>
<rb:start_date>start day</rb:start_date>
<rb:end_date>start time</rb:end_date>
<rb:start_time>end day</rb:start_time>
<rb:end_time>end time</rb:end_time>
[<rb:purpose>booking purpose</rb:purpose>]
[<rb:repetition></rb:repetition>]
[<rb:attendees>
        {<rb:attendee
           role=("host"|"attendee")
           status=("accepted" | "tentative" |
           "declined" | "noresponse" | "unknown")>
            [<rb:name>attendee name</rb:name>]
            [<rb:email>attendee email address<
            rb:email>]
            [<rb:phone>attendee phone number</
            rb:phone>]
        </rb:attendee>}
    </rb:attendees>]
[<rb:booking_creation_date>booking creation day</
rb:booking_creation_day>]
[<rb:booking_creation_time>booking creation time</
rb:booking_creation_time>]
[<rb:booking_modification_date>booking last
modification day</rb:booking_modification_day>]
[<rb:booking_modification_time>booking last
modification time</rb:booking_modification_time>]
  [<rb:notes>booking notes</rb:notes>]
 </rb:booking>]
</rb:bookings>
</kwe:result>
```

Result properties:

The result XML structure for every command always has a **rb:bookings** block with the following properties defined:

- **room_id**: unique identifier of the room (required)

  The block has zero or more **rb_booking** blocks which identify individual room bookings and can have the following properties defined:

- **booking_id**: unique identifier of the booking(required)
- **confidential**: indicates whether a booking should be regarded as confidential. The definition of confidential depends on the target system and the interpretation of confidential is left to the client. (required)

  Example: The RoomWizard client interprets confidential by not showing confidential booking details on the device itself, these details are available, however, to anybody browsing the booking details using its web server.

- **password_protected**: Indicates whether a password is needed to manipulate the booking (required)
- **range_start_date, range_start_time, range_end_date, range_end_time**: define the time period for which the request searches for room bookings. (required)
- **purpose**: purpose or subject of the booking. (optional)

  The connector writer needs to make an appropriate mapping of the "purpose" property to a field on the target system's booking. A client is likely to prominently display this field.

  If the purpose property is absent and the target system insists on the mapped field the connector is allowed to fill in an empty string or a dummy value.

- **repetition**: this tag must appear if the booking is part of or represents an repeated booking (optional). booking_creation_date, booking_creation_time – date and time at which the booking was first created (optional)
- **booking_modification_date**, **booking_modification_time**: date and time at which the booking was last modified (optional)
- **notes**: booking notes (optional)

The **rb:booking** block has an optional **rb:attendees** block that contains one or more **rb:attendee** blocks. These blocks contain information about the attendees of a meeting and can have the following properties:

- **role**: defines whether the attendee is the host of the meeting or a regular attendee (required)
- **status**: defines the attendee's response to the booking request (required):

  accepted: invitee will attend

  tentative: invitee may attend

  declined: invitee will not attend

  noresponse: invitee has not responded
- **name**: name of the attendee (optional)
- **email**: email address of the attendee (optional)
- **phone**: phone number of the attendee (optional)

Result codes:

The following result codes are specific to the get_bookings request:

1100 incorrect date format

Selection criteria

This request must return all bookings that are active at any point during the period indicated by the request arguments except bookings starting on exactly the end date and time of the period specified.

Formally specified:

Given a request with:

```
range_start_date=rsd
range_start_time=rst
range_end_date=red
range_end_time=ret
```

and a set of bookings on a target system with

```
booking_start_date=bsd
booking_start_time=bst
booking_end_date=bed
booking_end_time=bet
```

return all bookings for which the following is true:

```
(((bsd + bst) < (red + ret)) AND
  ((bed + bet) > (rsd + rst))) OR
 ((bsd + bst) = (rsd + rst) = (red+ ret))
```

assuming that the following is always true (meetings don't end before they start)

```
  (bsd + bst) <=  (bed + bet)
```

Comments

The connector does not need to support requests with multiple Room ids.

The connector should interpret the "today" parameter as the current day on the target system and should interpret the "now" parameter as the current time on the target system.

The connector may return the bookings within the **rb:bookings** block in any order. Clients must not rely on the bookings being ordered in any way in the result section.

The "confidential" property must not affect the processing behavior of the connector in any other way than retrieving the appropriate information from the server and passing it on to the client.

The connector should not include "deleted" bookings in the result set.

The connector should only collect "confirmed" bookings. If the target system can have "unconfirmed" or "not yet accepted" room bookings than these should not be retrieved.

Example

Request:

```
http://exchangeserver1/cgi-bin/
rwconnector?command=get_bookings&room_id=room+1&range_start_
date=20021231&range_start_time=000000&range_end_date=2003010
1&range_end_time=000000
```

This command requests all bookings for "room 1" on 31 December 2002.

Response:

```
<?xml version="1.0"?>
<kwe:result xmlns:kwe="<http://www.appliancestudio.com/kwe/
1.0/>"
xmlns:rb="<http://www.appliancestudio.com/rb/1.0/>">
    <kwe:date>20021231</kwe:date>
    <kwe:time>125932</kwe:time>
    <kwe:result_code>0</kwe:result_code>
    <rb:result_code>0</rb:result_code>
    <rb:bookings room_id="Room 1">
        <rb:booking booking_id="1" private="no"
           password_protected="no">
            <rb:start_date>20020312</rb:start_date>
            <rb:end_date>20020312</rb:end_date>
            <rb:start_time>140000</rb:start_time>
            <rb:end_time>150000</rb:end_time>
            <rb:purpose>Important Sales Meeting</rb:purpose>
            <rb:attendees>
                <rb:attendee role="host" status="accepted">
                    <rb:name>alice</rb:name>
                    <rb:email>alice@host.com</rb:email>
                    <rb:phone>0123 4567890</rb:phone>
                </rb:attendee>
                <rb:attendee role="attendee" status =
                "accepted">
                    <rb:name>bob</rb:name>
                    <rb:email>bob@host.com</rb:email>
                </rb:attendee>
                <rb:attendee role="host" status="tentative">
                    <rb:name>charlie</rb:name>
                    <rb:email>charlie@host.com</rb:email>
                </rb:attendee>
            </rb:attendees>
```

```
            <rb:notes>I may be 5 minutes late starting</
             rb:notes>
        </rb:booking>
    </rb:bookings>
</kwe:result>
```

ADD_BOOKING

Name: add_booking

Description: This command requests that a very simple room booking be added to the target system. A connector should try to add this booking to the target system and report the result back to the client.

Request syntax:

```
add_booking_args = "room_id" "=" room_id
                   "&" "start_date" = start_date
                   "&" "start_time" = start_time
                   "&" "end_date" = end_date
                   "&" "end_time" = end_time
                   [ "&" "host" = host ]
                   [ "&" "purpose" = purpose ]
                   [ "&" "email" = email ]
                   [ "&" "phone" = phone ]
                   [ "&" "notes" = notes ]
                   [ "&" "booking_password" =
                   booking_password ]
                   [ "&" "room_password" = room_password ]
room_id = <The unique identifier for a room>
start_date = ( 4DIGIT 2DIGIT 2DIGIT ) | "today"
start_time = ( 2DIGIT 2DIGIT 2DIGIT ) | "now"
end_date = ( 4DIGIT 2DIGIT 2DIGIT ) | "today"
end_time = ( 2DIGIT 2DIGIT 2DIGIT ) | "now"
host = <meeting host>
purpose = <meeting purpose>
email = <email contact details>
phone = <phone contact details>
```

```
notes = <additional information>
booking_password = <password for protecting the booking>
room_password = <password for getting access to the room
bookings>
```

Request arguments:

room_id	This identifier uniquely identifies a room within the connector (required)
room_password	This password needs to be used if the room is password protected (optional)
start_date	Start day of the booking (required)
start_time	Start time of the booking (required)
end_date	End day of the booking (required)
end_time	End time of the booking (required)
host	Name of the booking organiser (optional)
purpose	Meeting purpose (optional)
email	Email contact details for booking host (optional)
phone	Phone contact details for booking host (optional)
notes	Additional booking information (optional)
booking_password	If present, it makes the booking password protected with this password (optional)

Result syntax:

The resulting XML has the same format as the **get_bookings** result XML. In case of a successful booking it includes a **rb:bookings** block with exactly one **rb:booking** sub-block which contains the information about the newly created room booking. There may be a **rb:attendees** section containing the host that was specified in the request.

Result codes

The following result codes are specific to the **get_bookings** request:

1200	Booking clash
1201	Unbookable time
1202	Not authorized to book

Comments

The connector may use the host, email, and phone details from the request to create a “host” attendee.

The connector should only try to prohibit booking clashes if a target system requires this of any “client” connecting to it, otherwise the connector must accept new bookings that overlap with existing bookings.

The **add_booking** command creates very simple bookings only; more complicated bookings such as repeating bookings or bookings with invitees are not covered by this version of the API.

Example

Request:

```
http://exchangeserver1/cgi-bin/
rwconnector?command=add_bookings&room_id=room+1&start_date=2
0021231&start_time=140000&end_date=20021231&end_time=150000&
host=alice&purpose= Important+Sales+Meeting&email=
alice%40host.com&phone=0123+4567890&notes=I+
may+be+5+minutes+late+starting
```

This command requests the connector to add a booking which takes place in “room 1” on 31 December 2002 between 14:00 and 15:00.

Response:

```
<?xml version="1.0"?>
<kwe:result xmlns:kwe="<http://www.appliancestudio.com/kwe/
1.0/>"
xmlns:rb="<http://www.appliancestudio.com/rb/1.0/>">
    <kwe:date>20021231</kwe:date>
    <kwe:time>125932</kwe:time>
    <kwe:result_code>0</kwe:result_code>
    <rb:result_code>0</rb:result_code>
    <rb:bookings room_id="Room 1">
        <rb:booking booking_id="1" private="no"
            password_protected="no">
             <rb:range_start_date>20020312</rb:
                  range_start_date>
            <rb: range_end_date>20020312</rb: range_end_date>
```

```
            <rb: range_start_time>140000</rb:
                 range_start_time>
            <rb: range_end_time>150000</rb: range_end_time>
            <rb:purpose>Important Sales Meeting</rb:purpose>
            <rb:attendees>
                <rb:attendee role="host" status="accepted">
                    <rb:name>alice</rb:name>
                    <rb:email>alice@host.com</rb:email>
                    <rb:phone>0123 4567890</rb:phone>
                </rb:attendee>
            </rb:attendees>
            <rb:notes>I may be 5 minutes late starting</
             rb:notes>
        </rb:booking>
    </rb:bookings>
</kwe:result>
```

EDIT_BOOKING

Name: edit_booking

Description: This command requests a change to the properties of a room booking.

Request syntax:

```
edit_booking_args = "room_id" "=" room_id
                   "&" "booking_id" "=" booking_id
                    ["&" "new_start_date" = new_start_date]
                    ["&" "new_start_time" = new_start_time]
                    ["&" "new_end_date" = new_end_date]
                    [ "&" "new_end_time" = new_time_date]
                    [ "&" "new_host" = new_host ]
                    [ "&" "new_purpose" = new_purpose ]
                    [ "&" "new_email" = new_email ]
                    [ "&" "new_phone" = new_phone ]
                    [ "&" "new_notes" = new_notes ]
```

```
                    [ "&" "new_password" = new_password ]
                    [ "&" "booking_password" =
                    booking_password ]
                    [ "&" "room_password" = room_password ]
room_id = <The unique identifier for a room>
booking_id = <The unique identifier for a booking>
new_start_date = ( 4DIGIT 2DIGIT 2DIGIT ) | "today"
new_start_time = ( 2DIGIT 2DIGIT 2DIGIT ) | "now"
new_end_date = ( 4DIGIT 2DIGIT 2DIGIT )   | "today"
new_end_time = ( 2DIGIT 2DIGIT 2DIGIT )
new_host = <meeting host>
new_purpose = <booking purpose>
new_email = <email contact details>
new_phone = <phone contact details>
new_notes = <additional booking information>
add_booking_password = <password for protecting the booking>
room_password = <password for getting access to the room
bookings>
booking_password = <password for booking manipulation
operations>
```

Request arguments:

room_id	This identifier uniquely identifies a room within the connector (required)
room_password	This password needs to be used if the room is password protected (optional)
new_start_date	New start day of the booking (optional)
new_start_time	New start time of the booking (optional)
new_end_date	New end day of the booking (optional)
new_end_time	New end time of the booking (optional)
new_host	New name of the meeting organiser (optional)
new_purpose	New meeting purpose (optional)
new_email	New email contact details for booking host (optional)

new_phone	New phone contact details for booking host (optional)
new_notes	New additional booking information (optional)
add_booking_password	New booking password. If the booking_password was absent, this makes the meeting password protected. (optional)
room_password	This password needs to be used if the room is password protected
booking_password	This password needs to be used if the booking is password protected

Result syntax:

The resulting XML has the same format as the **add_bookings** result XML. In case of a successful change it includes a **rb:bookings** block with exactly one **rb:booking** sub-block which contains the information about the modified room booking. There may be a **rb:attendees** section containing the host that was specified in the request.

Result codes

The following result codes are specific to the get_bookings request:

1200	Booking clash
1201	Unbookable time
1202	Not authorized to book
1300	Unknown booking
1301	Incorrect booking password
1302	Not authorized to change booking

Comments

See comments on the **add_booking** command.

It is conceivable that the **booking id** has changed as a result of the **edit_booking** operation. A connector should try to avoid this happening as much as possible.

Example

Request:

```
http://somehost.com/
roomwizard_connector?command=edit_booking&booking_id=1&end_t
ime=153000
```

Edit performed on the above booking, changing end time to 3.30pm

Response:

```
<?xml version="1.0"?>
<kwe:result
xmlns:kwe="<http://www.appliancestudio.com/kwe/1.0/>"
xmlns:rb="<http://www.appliancestudio.com/rb/1.0/>">
    <kwe:date>20021231</kwe:date>
    <kwe:time>125932</kwe:time>
    <kwe:result_code>0</kwe:result_code>
    <rb:result_code>0</rb:result_code>
    <rb:bookings room_id="Room 1">
        <rb:booking booking_id="1" private="no"
            password_protected="no">
            <rb:range_start_date>20020312</rb:
                range_start_date>
           <rb: range_end_date>20020312</rb: range_end_date>
            <rb: range_start_time>140000</rb:
                range_start_time>
            <rb: range_end_time>153000</rb: range_end_time>
            <rb:purpose>Important Sales Meeting</rb:purpose>
            <rb:attendees>
                <rb:attendee role="host" status="accepted">
                    <rb:name>alice</rb:name>
                    <rb:email>alice@host.com</rb:email>
                    <rb:phone>0123 4567890</rb:phone>
                </rb:attendee>
            </rb:attendees>
            <rb:notes>I may be 5 minutes late starting</
            rb:notes>
```

```
        </rb:booking>
    </rb:bookings>
</kwe:result>
```

DELETE_BOOKING

Name: delete_booking

Description: This command requests that a booking should be removed from the target system.

Request syntax:

```
delete_booking_args = "room_id" "=" room_id
                "&" "booking_id" = booking_id
                ["&" "room_password" = room_password]
                ["&" "booking_password" = booking_password]
room_id = <The unique identifier for a room>
booking_id = <The unique identifier for a booking>
room_password = <password for getting access to the room
bookings>
booking_password = <password for booking manipulation
operations>
```

Request arguments:

room_id	This identifier uniquely identifies a room within the connector (required)
booking_id	This identifier uniquely identifies a booking with a room (required)
room_password	This password needs to be used if the room is password protected
booking_password	This password needs to be used if the booking is password protected

Result syntax:

```
<?xml version="1.0"?>
    <kwe:result
      xmlns:kwe="http://www.appliancestudio.com/kwe/1.0/"
      xmlns:rb="http://www.appliancestudio.com/rb/1.0/">
```

```
        <kwe:date>date</kwe:date>
        <kwe:time>time</kwe:time>
        <kwe:result_code>numeric result code</
        kwe:result_code>
        [<rb:result_code>numeric result code</
        rb:result_code>]
</kwe:result>
```

Result codes

The following result codes are specific to the **delete_bookings** request:

1300	Unknown booking
1301	Incorrect booking password
1302	Not authorised to change booking

Example

Request:

```
http://somehost.com/
roomwizard_connector?command=delete_booking&booking_id=1
```

Response:

```
<?xml version="1.0"?>
<kwe:result xmlns:kwe="<http://www.appliancestudio.com/kwe/
1.0/>"
xmlns:rb="<http://www.appliancestudio.com/rb/1.0/>">
    <kwe:date>20021231</kwe:date>
    <kwe:time>130112</kwe:time>
    <kwe:result_code>0</kwe:result_code>
    <rb:result_code>0</rb:result_code>
</kwe:result>
```

INDEX