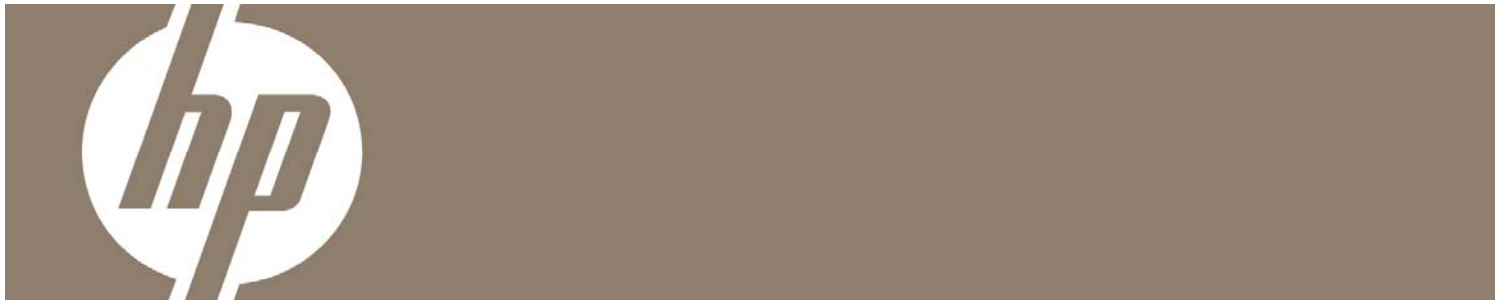


HP Virtual Rooms: Application Programming Interfaces

Integrating HP Virtual Rooms into customer applications



Introduction	3
Integration points	3
Choosing an integration point	3
1. Forms integration	3
Target audience	3
Integration required	3
Relationship to other HP Virtual Rooms components	3
Resources available	3
2. Portal integration	4
Target audience	4
Integration required	4
Relationship to other HP Virtual Rooms components	4
Resources available	4
3. Hosting integration	4
Target audience	4
Integration required	4
Relationship to other HP Virtual Rooms components	4
Resources available	4
Forms integration	5
Calling mechanism	5
Authentication	5
Information	5
Sample form	7
Portal integration	8
Calling mechanism	8
Authentication	8
Information	9
User Information	10
Room Information	11
Key Generation	14
Room Attendance	15
Event Scheduling	17

Room Activity	24
Exceptions.....	26
Hosting integration	27
Calling mechanism.....	27
Partner identification.....	27
Resource allocation	27
Authentication	28
Rooms	28
Events.....	31
Attend event.....	34
Attendance records	37
Message passing	40
Account operations	42
Exceptions.....	45
For more information.....	46

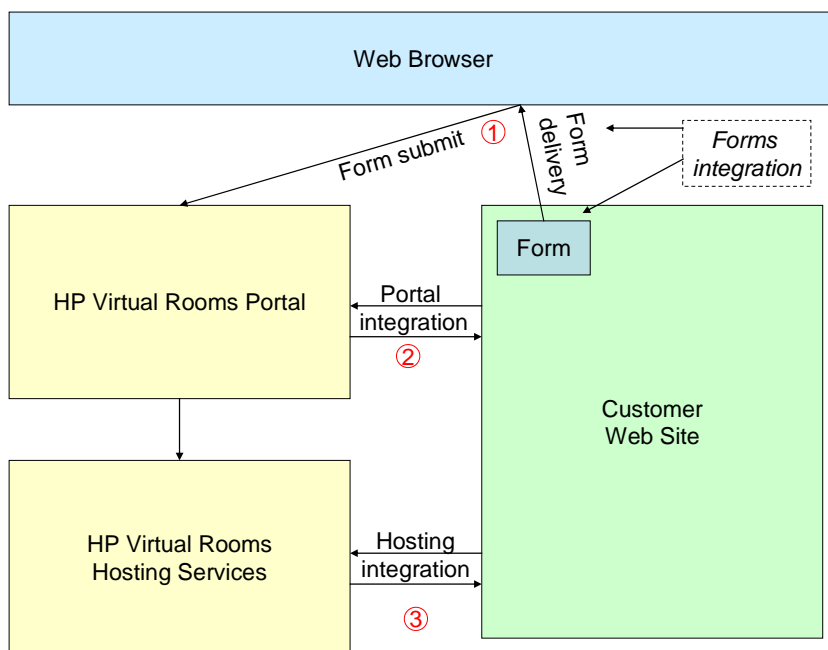
Introduction

HP Virtual Rooms provides the capabilities for groups to participate in virtual meetings using a wide range of interactive tools, with the benefit of persistent data. In addition to the interactive rooms, the HP Virtual Rooms solution provides support for creation, maintenance and scheduling of rooms and events through a web portal.

Many customers wish to integrate HP Virtual Rooms into their own applications. This may be achieved at three levels, each with their own benefits. This document outlines these various interfaces and provides information to help customers decide on the best integration mechanism for them. It then describes each of the interfaces in detail.

Integration points

The three integration points are shown in the following diagram.



Choosing an integration point

1. Forms integration

Target audience

Customers who want a simple form on their own web page to allow their users to attend an event.

Integration required

Customization of a web form, and integration of that form into the customer's own web

Relationship to other HP Virtual Rooms components

All room and event management continues to be accessed through the HP Virtual Rooms web portal.

Resources available

Sample HTML code is available to give customers a guide to customization.

2. Portal integration

Target audience

Customers who want easily to provide event scheduling and room attendance capabilities to their users through their own portal. Customers may also access room and event attendance information.

Customers may optionally provide a mechanism to authenticate their own users and supply this information to HP Virtual Rooms in a secure manner.

Integration required

Integration uses a web service to access HP Virtual Rooms services. Users may be authenticated using username and password, or through an external authentication mechanism.

Relationship to other HP Virtual Rooms components

Room and event management may also be accessed through the regular HP Virtual Rooms portal. User information is also stored in the HP Virtual Rooms portal. If using the username/password form of authentication, user accounts must be created in the HP Virtual Rooms portal prior to use of any tool built using portal-level integration.

Resources available

Access to the WSDL definition of the web service is provided through the web service itself. Assemblies to simplify integration are under development and will be available to run in both the Microsoft .NET 1.1 and 2.0 frameworks

3. Hosting integration

Target audience

Customers who want full control of all use of their HP Virtual Rooms resources. This is typically a reseller model. Customers may subdivide their available seats and provide their own mechanisms for creating rooms and events, and for attending those events. The ability to use “keys” to attend events is not available through this integration point – customers are responsible for creating their own mechanism. Customers also assume responsibility for managing their own end-user community. Mechanisms are provided for customers to reliably track their users as they attend events.

Integration required

Integration uses a web service to access HP Virtual Rooms services. Customers may make use of this service in whatever way suits their business.

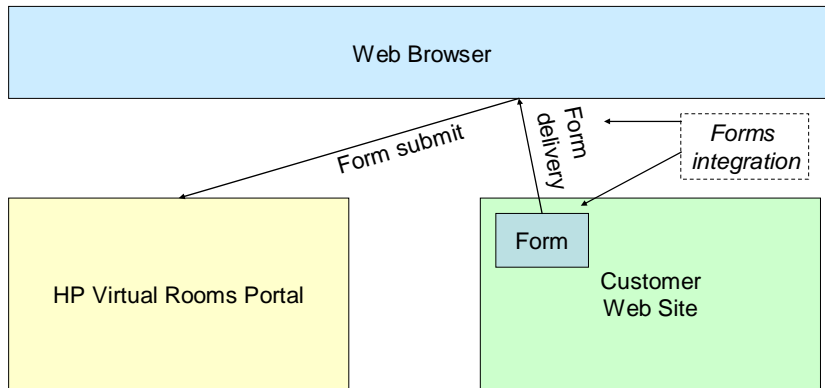
Relationship to other HP Virtual Rooms components

This integration point provides a mechanism to essentially build a replacement for the HP Virtual Rooms portal. Users of the customer’s replacement portal can *not* access those resources through the regular HP Virtual Rooms portal.

Resources available

Access to the WSDL definition of the web service is provided through the web service itself.

Forms integration



Calling mechanism

Web-level integration is achieved simply by the customer placing a form on their web site with a defined POST target and a set of well-known fields.

The exact formatting of the form may be customized by the customer to match the look and feel guidelines of their own web site. The HTML fragment shown below is simply an example.

Authentication

No authentication is required by HP Virtual Rooms when using this integration method as it simply replaces the normal methods for attending events which are provided to all users on the HP Virtual Rooms portal.

Customers are advised, however, to consider implementing appropriate authentication mechanisms for the pages they provide which include these forms – especially if the event key is embedded within the form.

Information

The following form attributes must be specified:

Attribute name	Value
Method	GET or POST
Action	http://www.rooms.hp.com/partner/keyaccess.aspx

The following form fields may be specified. Mandatory fields are indicated in **bold type**.

Field name	Use	Default value or action
Name	The name to be used in the room	No default (error)
Key	The key to be used for access to the room	No default (error)
Language	The language to be used for room display. See example for possible values	No default (error) – use the string “en” for English

Field name	Use	Default value or action
Privacy	False to request anonymous access to the room	True
ReturnURL	A URL to which the user web browser will be returned after successful room entry	User will land on an HP Virtual Rooms page
FailureURL	A URL to which the user web browser will be returned after unsuccessful room entry. The query string variable ErrNo will be set to an appropriate error number in this case (see below)	User will land on an HP Virtual Rooms page with an appropriate error message

Note that any or all of the form fields may be specified as hidden form fields in order to pre-fill responses for the user.

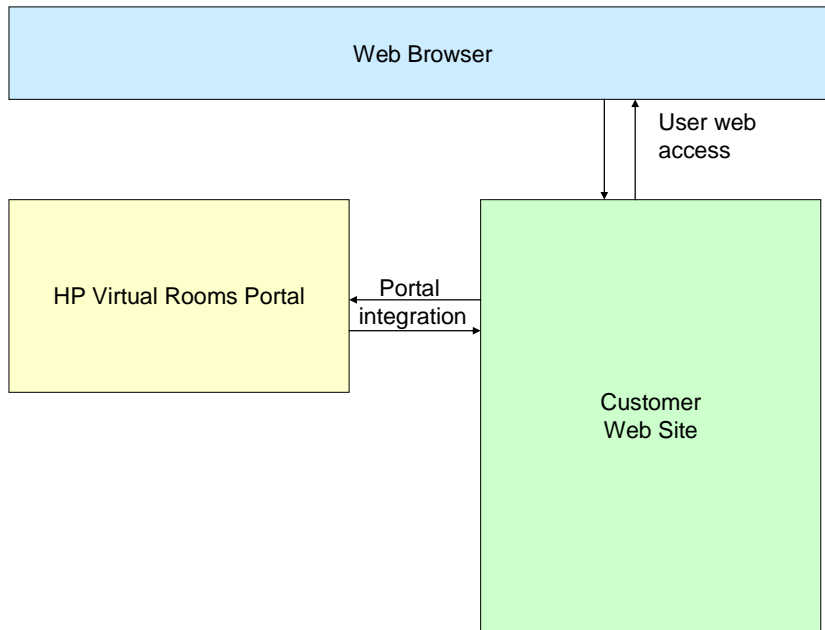
If the attempt to enter a room is unsuccessful and the FailureURL form field is specified, the following error numbers may be returned on the query string of the resulting page:

Error number	Meaning
1	Missing name
2	Missing key
3	Missing or unsupported language
5	Invalid key specified

Sample form

```
<table cellpadding=5>
<form method="POST"
      action="http://www.rooms.hp.com/partner/keyaccess.aspx">
<input type="hidden" name="ReturnURL"
      value="http://mywebsite/home.htm">
<input type="hidden" name="FailureURL"
      value="http://mywebsite/fail.htm">
<input type="hidden" name="Privacy" value="true">
  <tr>
    <td>Name:</td>
    <td><input type="text" name="name" size="20"></td>
  </tr>
  <tr>
    <td>Key:</td>
    <td><input type="password" Name="key" size="20"></td>
  </tr>
  <tr>
    <td>Select language:</td>
    <td>
      <select name="language" size="1">
        <option value="">Please select</option>
        <option value="en">English</option>
        <option value="zh">Chinese</option>
        <option value="cs">Czech</option>
        <option value="fr">French</option>
        <option value="de">German</option>
        <option value="el">Greek</option>
        <option value="hu">Hungarian</option>
        <option value="ja">Japanese</option>
        <option value="ko">Korean</option>
        <option value="pl">Polish</option>
        <option value="ru">Russian</option>
        <option value="es">Spanish</option>
      </select>
    </td>
  </tr>
  <tr>
    <td colspan="2" align="center">
      <input type="submit" value="Submit" name="B1">
      <input type="reset" value="Reset" name="B2">
    </td>
  </tr>
</form>
</table>
```

Portal integration



Calling mechanism

The portal API is implemented as a web service which is accessible to applications using the HTTP protocol. Its methods are defined using Web Services Description Language. For security purposes, all transactions with the portal API are carried as SSL transactions. User identification is carried in the encrypted SOAP header.

Authentication

All requests to the portal API should normally be accompanied by authentication information. This is carried in the SOAP header of the request, which is encrypted as the requests are carried across a secure link.

The authentication information may be expressed in one of two ways. For users who access their HP Virtual Rooms using a simple username and password, these same credentials may be used to access the portal API. For some corporate users, an alternate authentication mechanism may be defined which results in an opaque, time-limited token. This token may be passed to the authentication mechanism. Note that only one of these mechanisms may be used in a request.

In general, requests relating to a particular room or event must use the authentication information for the owner or scheduler of the room or event in question.

The authentication information definition is:

```
Class HPVCUserAuth : Inherits SoapHeader
    Dim AccountName as String,
    Dim Pwd as String,
    Dim Opaque as String
End Class
```

Information

A number of general methods are provided to access information about the HP Virtual Rooms service.

PortalLocation

- Gets the web location of the current HP Virtual Rooms portal.

PortalLocation() As String

On success returns a URL for the portal.

On error throws a SOAP exception (see page 26).

Example:

```
Dim api as New HPVCScheduler
' No authentication required
Dim location as String
Try
    location = api.PortalLocation()
Catch ex As SoapException
    ' Handle failure
End Try
```

ClientVersion

- Returns the current version of the client software for HP Virtual Rooms

ClientVersion() As String

On success returns a string representing the client version.

On error throws a SOAP exception (see page 26).

Example:

```
Dim api as New HPVCScheduler
' No authentication required
Dim version as String
Try
    version = api.ClientVersion()
Catch ex as SoapException
    ' Handle failure
End Try
```

User Information

The methods in this section return information about the capabilities of the user whose credentials are supplied.

VerifyCredentials

- Takes no action, but causes the supplied user credentials to be validated

VerifyCredentials() As Boolean

On success returns true if the credentials are valid, otherwise false.

On error throws a SOAP exception (see page 26).

Example:

```
Dim api as New HPVCScheduler
Dim auth as New HPVCTestUserAuth
auth.AccountName = "myusername"
auth.Pwd = "mypassword"
api.HPVCTestUserAuthValue = auth
Dim result as Boolean
Try
    result = api.VerifyCredentials()
    ' Success
Catch ex As SoapException
    ' Failure
    result = False
End Try
```

HasBackpack

- Tests whether the current user has, or may have, a backpack.

HasBackpack() As Boolean

On success returns true if the user has, or may have, a backpack, otherwise false

On error throws a SOAP exception (see page 26).

Example:

```
Dim api as New HPVCTestScheduler
Dim auth as New HPVCTestUserAuth
auth.AccountName = "myusername"
auth.Pwd = "mypassword"
api.HPVCTestUserAuthValue = auth
Dim result as Boolean
Try
    result = api.HasBackpack()
    ' Success
Catch ex As SoapException
    ' Failure
    result = False
End Try
```

Room Information

The methods in this section return information about rooms owned by or accessible to the current user.

GetRooms

- Returns a list of all rooms in all license pools in which the current user may schedule events.

GetRooms() As ArrayList

On success returns an array of strings describing the requested rooms. Each string contains two semi-colon(;) separated values. The first value is the number of the room, the second is its name. If a user has access to rooms in more than one license pool, license pool separator strings will be inserted in the array. A license pool separator string contains two semi-colon separated values. The first value is a negative integer representing the license pool id; the second value is the name of the license pool, preceded by the fixed string "-- LP: ".

On error throws a SOAP exception (see page 26).

Example:

```
Dim api as New HPVCScheduler
api.HPVCUserAuthValue = auth ' see previous example
Dim roomList as ArrayList
Try
    roomList = api.GetRooms()
    'Success
    For Each roomEntry As String In RoomList
        Dim thisRoom() As String = roomEntry.Split(";")
        Dim roomNumber as Integer = Integer.Parse(thisRoom(0))
        If roomNumber > 0
            ' This is a room
            Console.WriteLine("Room " & roomNumber & ": " & _
                thisRoom(1))
        Else
            ' This is a license pool prefix
            ' Skips the prefix on the pool name
            Console.WriteLine("Pool " & -roomNumber & ": " & _
                thisRoom(1).SubString(7))
        End If
    Next
Catch ex As SoapException
    ' Failure
    Console.WriteLine("Could not get room list")
End Try
```

GetMyRooms

- Returns a list of all rooms in all license pools which are owned by the current user.

GetMyRooms() As ArrayList

On success returns an array of strings describing the requested rooms. Each string contains two semi-colon(;) separated values. The first value is the number of the room, the second is its name. If a user owns rooms in more than one license pool, license pool separator strings will be inserted in the array. A license pool separator string contains two semi-colon separated values. The first value is a negative integer representing the license pool id; the second value is the name of the license pool, preceded by the fixed string "-- LP: ".

On error throws a SOAP exception (see page 26).

Example:

```
Dim api as New HPVCScheduler
api.HPVCUserAuthValue = auth ' see previous example
Dim roomList as ArrayList
Try
    roomList = api.GetMyRooms()
    'Success
    For Each roomEntry As String In RoomList
        Dim thisRoom() As String = roomEntry.Split(";")
        Dim roomNumber as Integer = Integer.Parse(thisRoom(0))
        If roomNumber > 0
            ' This is a room
            Console.WriteLine("Room " & roomNumber & ": " & _
                thisRoom(1))
        Else
            ' This is a license pool prefix
            ' Skips the prefix on the pool name
            Console.WriteLine("Pool " & -roomNumber & ": " & _
                thisRoom(1).SubString(7))
        End If
    Next
Catch ex As SoapException
    ' Failure
    Console.WriteLine("Could not get room list")
End Try
```

GetOtherRooms

- Returns a list of all rooms in all license pools to which the current user has access either for scheduling or presenting (apart from rooms owned by the user).

GetOtherRooms() As ArrayList

On success returns an array of strings describing the requested rooms. Each string contains three semi-colon(;) separated values. The first value is the number of the room, the second is its name and the third is 1 if the user has scheduling access, or 0 if the user only has presenter access. If a user has access to rooms in more than one license pool, license pool separator strings will be inserted in the array. A license pool separator string contains two semi-colon separated values. The first value is a negative integer representing the license pool id; the second value is the name of the license pool, preceded by the fixed string "-- LP: ".

On error throws a SOAP exception (see page 26).

Example:

```
Dim api as New HPVCScheduler
api.HPVCUserAuthValue = auth ' see previous example
Dim roomList as ArrayList
Try
    roomList = api.GetOtherRooms()
    'Success
    For Each roomEntry As String In RoomList
        Dim thisRoom() As String = roomEntry.Split(";")
        Dim roomNumber as Integer = Integer.Parse(thisRoom(0))
        If roomNumber > 0
            ' This is a room
            Dim access As String
            Select Case Integer.Parse(thisRoom(2))
                Case 1
                    access = "scheduler"
                Case 0
                    access = "presenter"
                Case Else
                    access = "unknown"
            End Select
            Console.WriteLine("Room " & roomNumber & ": " & _
                thisRoom(1) & " - " & access)
        Else
            ' This is a license pool prefix
            ' Skips the prefix on the pool name
            Console.WriteLine("Pool " & -roomNumber & ": " & _
                thisRoom(1).SubString(7))
        End If
    Next
Catch ex As SoapException
    ' Failure
    Console.WriteLine("Could not get room list")
End Try
```

Key Generation

Apart from rooms which are owned by or assigned to a user as a scheduler or presenter, the normal mechanism used to enter a room is through a “key”. These are often generated as a side-effect of scheduling an event, but there are other forms. The methods in this section permit the creation of such keys

There are two principal roles used to enter a room – presenter and participant. The following enumeration is used to identify these roles.

```
Enum HPVCRoleType
    Participant = 1
    Presenter = 2
End Enum
```

GetRoomQuickKey

- Returns a key which can be used for immediate access to a room. At the time of writing, the keys generated by this method are valid for 5 minutes after issue.

GetRoomQuickKey(roomID as Integer, role as HPVCRoleType) As Integer

On success returns a quick key, expressed as a number, which grants the requested level of access to the room whose room number is given.

On error throws a SOAP exception (see page 26).

Example:

```
Dim roomNumber as Integer = 1234

Dim api as New HPVCScheduler
api.HPVCUserAuthValue = auth ' see previous example
Try
    Dim result as Integer
    result = api.GetRoomQuickKey(roomNumber, HPVCRoleType.Participant)
    ' Success
    Console.WriteLine("Key is " & result.ToString())
Catch ex As SoapException
    ' Failure
    Console.WriteLine("Could not obtain key")
End Try
```

GetRoomMeetNowKey

- Returns a key which can be used for immediate access to a room. At the time of writing, the keys generated by this method are valid for 60 minutes after issue.

GetRoomMeetNowKey(roomID as Integer, role as HPVCRoleType) As Integer

On success returns a meet now key, expressed as a number, which grants the requested level of access to the room whose room number is given.

On error throws a SOAP exception (see page 26).

Example:

```
Dim roomNumber as Integer = 1234

Dim api as New HPVCScheduler
api.HPVCUserAuthValue = auth ' see previous example
Try
    Dim result as Integer
    result = api.GetRoomMeetNowKey(roomNumber, HPVCRoleType.Presenter)
    ' Success
    Console.WriteLine("Key is " & result.ToString())
Catch ex As SoapException
    ' Failure
    Console.WriteLine("Could not obtain key")
End Try
```

Room Attendance

The methods in this section return access tokens (expressed as a URL) for entry into a particular room. Note that it is the responsibility of the caller to use the returned URL to actually enter the room.

GetRoomEntryURL

- Returns a URL which, when given to a web browser, will enter the user into the given room as a presenter. The room specified must be one to which the user has access otherwise an exception results.

GetRoomEntryURL(roomID as Integer) As String

On success returns a URL

On error throws a SOAP exception (see page 26).

Example:

```
Dim roomNumber as Integer = 1234

Dim api as New HPVCScheduler
api.HPVCUserAuthValue = auth ' see previous example
Try
    Dim result as String
    result = api.GetRoomEntryURL(roomNumber)
    ' Success
    Console.WriteLine("URL is " & result)
Catch ex As SoapException
    ' Failure
    Console.WriteLine("Could not obtain entry URL")
End Try
```

GetKeyEntryURL

- Returns a URL which, when given to a web browser, will enter the user into the room implied by the key and with the appropriate level of access.

GetKeyEntryURL(key as String) As String

On success returns a URL

On error throws a SOAP exception (see page 26).

Example:

```
Dim entryKey as String = "RR19T05GAZ"

Dim api as New HPVCScheduler
api.HPVCUserAuthValue = auth ' see previous example
Try
    Dim result as String
    result = api.GetKeyEntryURL(entryKey)
    ' Success
    Console.WriteLine("URL is " & result)
Catch ex As SoapException
    ' Failure
    Console.WriteLine("Could not obtain entry URL")
End Try
```

GetBackpackURL

- Returns a URL which, when given to a web browser, will enter the user into their own backpack. If the user is not permitted a backpack (see "Example:

- Dim api as New HPVCScheduler

' No authentication required

Dim version as String

Try

version = api.ClientVersion()

Catch ex as SoapException

' Handle failure

End Try

- User Information" on page 9), an exception results.

GetBackpackURL() As String

On success returns a URL

On error throws a SOAP exception (see page 26).

Example:

```
Dim api as New HPVCScheduler
api.HPVCUserAuthValue = auth ' see previous example
Try
    Dim result as String
    result = api.GetBackpackURL(entryKey)
    ' Success
    Console.WriteLine("URL is " & result)
Catch ex As SoapException
    ' Failure
    Console.WriteLine("Could not obtain backpack URL")
End Try
```

Event Scheduling

The methods in this section permit the scheduling, rescheduling and cancelling of scheduled events. Both single events and recurring events are supported. Recurring events use the following structure to express the recurrence being requested.

```
Structure HPVCRecurrenceDescription
    Dim lOccurrences as Integer ' number of occurrences
    Dim dRecurrenceType as Integer ' see below
    Dim lInterval as Integer ' skip count between occurrences
    Dim dDayOfWeekMask as Integer ' bit mask for the included days of the week
    Dim dDayOfMonth as Integer
    Dim dInstance as Integer ' 1 = first, 2 = second, ... 5 = last
    Dim dMonthOfYear as Integer
End Structure
```

When no recurrence is intended, as in the case of a single event, a recurrence description structure indicating lOccurrences = 1 should be used. Other fields are ignored in this case.

Example: no recurrence

```
Dim recurrence as HPVCRecurrenceDescription
recurrence.lOccurrences = 1
```

The following recurrence types are supported:

- Daily (dRecurrenceType = 0)
 - lInterval specifies the time between successive occurrences (e.g. 2 means "every 2 days")

Example: daily recurrence (every day for 5 days)

```
Dim recurrence as HPVCRecurrenceDescription
recurrence.dRecurrenceType = 0
recurrence.lOccurrences = 5
```

Example: daily recurrence (every 3 days for 10 occurrences)

```
Dim recurrence as HPVCRecurrenceDescription
recurrence.dRecurrenceType = 0
recurrence.lOccurrences = 10
recurrence.lInterval = 3
```

- Weekly (dRecurrenceType = 1)
 - dDayOfWeekMask is used to indicate which days of the week should be considered for occurrences. The least-significant bit (value 1) represents Sunday, the next bit (value 2) represents Monday and so forth. lInterval specifies the interval in weeks between successive occurrences

Example: weekly recurrence (every Monday for 5 weeks)

```
Dim recurrence as HPVCRecurrenceDescription
recurrence.dRecurrenceType = 1
recurrence.lOccurrences = 5
recurrence.dDayOfWeekMask = 2 ' Monday
```

Example: weekly recurrence (every other Monday and Thursday for 10 occurrences)

```
Dim recurrence as HPVCRecurrenceDescription
recurrence.dRecurrenceType = 1
recurrence.lOccurrences = 10
recurrence.lInterval = 2
recurrence.dDayOfWeekMask = 2 + 16
```

- Monthly by date (dRecurrenceType = 2)
 - dDayOfMonth indicates the day of the month. lInterval specifies the interval in months between successive occurrences

Example: monthly recurrence (10th day of each month for 5 months)

```
Dim recurrence as HPVCRecurrenceDescription
recurrence.dRecurrenceType = 2
recurrence.lOccurrences = 5
recurrence.dDayOfMonth = 10
```

Example: monthly recurrence (15th day of every third month for 10 occurrences)

```
(e.g. January, April, July, ...)
Dim recurrence as HPVCRecurrenceDescription
recurrence.dRecurrenceType = 2
recurrence.lOccurrences = 10
recurrence.lInterval = 3
recurrence.dDayOfMonth = 15
```

- Monthly by day (dRecurrenceType = 3)
 - dDayOfWeekMask indicates the day of the week (only one bit may be set), dInstance indicates the ordinal number of that day in the month, lInterval specifies the interval in months between occurrences.

Example: monthly recurrence (2nd Monday and Tuesday of each month for 5 months)

```
Dim recurrence as HPVCRecurrenceDescription
recurrence.dRecurrenceType = 3
recurrence.lOccurrences = 10      ' 5 months, 2 each month
recurrence.dDayOfWeekMask = 2 + 4 ' Monday + Tuesday
recurrence.dInstance = 2
```

Example: monthly recurrence (Last Friday of every third month for 10 occurrences)

```
(e.g. January, April, July, ...)
Dim recurrence as HPVCRecurrenceDescription
recurrence.dRecurrenceType = 3
recurrence.lOccurrences = 10
recurrence.lInterval = 3
recurrence.dDayOfWeekMask = 32    ' Friday
recurrence.dInstance = 5
```

- Yearly by date (dRecurrenceType = 5)
 - dDayOfMonth indicates the day of the month, dMonthOfYear indicates the month of the year

Example: yearly recurrence (April 30th each year for 5 years)

```
Dim recurrence as HPVCRecurrenceDescription
recurrence.dRecurrenceType = 5
recurrence.lOccurrences = 5
recurrence.dDayOfMonth = 30
recurrence.dMonthOfYear = 4
```

- Yearly by day (dRecurrenceType = 6)
 - dDayOfWeekMask indicates the day of the week (only one bit may be set), dInstance indicates the ordinal number of that day in the month, dMonthOfYear indicates the month of the year

Example: yearly recurrence (April 30th each year for 5 years)

```
Dim recurrence as HPVCRecurrenceDescription
recurrence.dRecurrenceType = 5
recurrence.lOccurrences = 5
recurrence.dDayOfMonth = 30
recurrence.dMonthOfYear = 4
```

The following structure is used to return information about an event which has been successfully booked or modified. The event number is returned, together with the presenter and participant keys.

```
Structure HPVCEventInformation
  Dim ID As Integer
  Dim PresenterString as String
  Dim ParticipantString as String
End Structure
```

When dates and times are used as parameters, they should be passed as strings, using the following 24-hour format, expressed in UTC.

yyyy-mm-dd hh:mm

Examples:

```
"2007-02-08 17:00"    '----- 5 p.m. on February 8, 2007
"2008-12-20 08:30"    '----- 8:30 a.m. on December 20, 2008
```

CheckAvailability2

- Determines if an event could be booked as specified.

```
CheckAvailability2(roomID As Integer,  
    startTime as String,  
    recurrence As HPVCRecurrenceDescription,  
    duration as Integer, ' in minutes  
    Seats as Integer,  
    Record as HPVCRecordingType, ' use None (=0)  
    VOIP As Boolean, ' use False  
    Encrypted as Boolean) ' use False  
As Boolean
```

On success returns true if an event could be scheduled as specified, otherwise false.

On error throws a SOAP exception (see page 26).

Example:

```
Dim roomNumber As Integer = 1234  
Dim startTime As String = "2007-02-08 17:00"  
Dim recurrence as HPVCRecurrenceDescription ' see examples above  
Dim eventDuration as Integer = 90 ' 1 hour 30 minutes  
Dim seatCount As Integer = 50  
  
Dim api as New HPVCScheduler  
api.HPVCUserAuthValue = auth ' see previous example  
Try  
    Dim result as String  
    result = api.CheckAvailability2(roomNumber, _  
        startTime, recurrence, duration, seatCount, _  
        HPVCRecordingType.None, False, False)  
    ' Success  
    Console.WriteLine("Proposed event time is available")  
Catch ex As SoapException  
    ' Failure  
    Console.WriteLine("Proposed event time is not available ")  
End Try
```

BookEvent2

- Attempts to book an event as specified.

```
BookEvent2(roomID As Integer,  
            eventName as String,  
            startTime as String,  
            duration as Integer, ' in minutes  
            recurrence As HPVCCurrenceDescription,  
            Seats as Integer,  
            Record as HPVCCordingType, ' use None (=0)  
            VOIP As Boolean, ' use False  
            Encrypted as Boolean) ' use False  
As HPVCEventInformation
```

On success returns information about the scheduled event.

On error throws a SOAP exception (see page 26).

Example:

```
Dim roomNumber As Integer = 1234  
Dim startTime As String = "2007-02-08 17:00"  
Dim recurrence as HPVCCurrenceDescription ' see examples above  
Dim eventDuration as Integer = 90 ' 1 hour 30 minutes  
Dim seatCount As Integer = 50  
  
Dim api as New HPVCScheduler  
api.HPVCUserAuthValue = auth ' see previous example  
Try  
    Dim result as HPVCEventInformation  
    result = api.BookEvent2(roomNumber, _  
                            startTime, recurrence, duration, seatCount, _  
                            HPVCCordingType.None, False, False)  
    ' Success  
    Console.WriteLine("Event id " & result.ID & " scheduled")  
    Console.WriteLine("Presenter key: " & result.PresenterString)  
    Console.WriteLine("Participant key: " & result.ParticipantString)  
Catch ex As SoapException  
    ' Failure  
    Console.WriteLine("Proposed event time is not available ")  
End Try
```

CheckModification2

- Determines if an existing event could be modified as specified.

```
CheckModification2(eventID as Integer,  
    roomID As Integer,  
    startTime as String,  
    recurrence As HPVCRecurrenceDescription,  
    duration as Integer, ' in minutes  
    Seats as Integer,  
    Record as HPVCRecordingType, ' use None (=0)  
    VOIP As Boolean, ' use False  
    Encrypted as Boolean) ' use False  
As Boolean
```

On success returns true if the specified event could be modified as specified, otherwise false.

On error throws a SOAP exception (see page 26).

Example:

```
Dim eventID As Integer = 54509  
Dim roomNumber As Integer = 1234  
Dim startTime As String = "2007-03-08 17:00"  
Dim recurrence as HPVCRecurrenceDescription ' see examples above  
Dim eventDuration as Integer = 90 ' 1 hour 30 minutes  
Dim seatCount As Integer = 50  
  
Dim api as New HPVCScheduler  
api.HPVCUserAuthValue = auth ' see previous example  
Try  
    Dim result as String  
    result = api.CheckModification2(eventID, roomNumber, _  
        startTime, recurrence, duration, seatCount, _  
        HPVCRecordingType.None, False, False)  
    ' Success  
    If result Then  
        Console.WriteLine("Proposed event time is available")  
    Else  
        Console.WriteLine("Proposed event time is not available ")  
    End If  
Catch ex As SoapException  
    ' Failure  
    Console.WriteLine("Proposed event time is not available ")  
End Try
```

ModifyEvent2

- Attempts to modify an existing event as specified.

```
BookEvent2(eventID As Integer,  
            roomID As Integer,  
            eventName as String,  
            startTime as String,  
            duration as Integer, ' in minutes  
            recurrence As HPVCRecurrenceDescription,  
            Seats as Integer,  
            Record as HPVCRecordingType, ' use None (=0)  
            VOIP As Boolean, ' use False  
            Encrypted as Boolean) ' use False  
            As HPVCEventInformation
```

On success returns information about the newly modified event.

On error throws a SOAP exception (see page 26).

Example:

```
Dim eventID As Integer = 54509  
Dim roomNumber As Integer = 1234  
Dim startTime As String = "2007-03-08 17:00"  
Dim recurrence as HPVCRecurrenceDescription ' see examples above  
Dim eventDuration as Integer = 90 ' 1 hour 30 minutes  
Dim seatCount As Integer = 50  
  
Dim api as New HPVCScheduler  
api.HPVCUserAuthValue = auth ' see previous example  
Try  
    Dim result as HPVCEventInformation  
    result = api.ModifyEvent2(eventID, roomNumber, _  
                             startTime, recurrence, duration, seatCount, _  
                             HPVCRecordingType.None, False, False)  
    ' Success  
    Console.WriteLine("Event id " & result.ID & " re-scheduled")  
    Console.WriteLine("Presenter key: " & result.PresenterString)  
    Console.WriteLine("Participant key: " & result.ParticipantString)  
Catch ex As SoapException  
    ' Failure  
    Console.WriteLine("Proposed event time is not available ")  
End Try
```

DeleteEvent

- Attempts to delete an existing event (including a recurring event).

DeleteEvent(eventID As Integer) As Boolean

On success returns true if the event was successfully deleted, otherwise false.

On error throws a SOAP exception (see page 26).

Example:

```
Dim eventID As Integer = 54509

Dim api as New HPVCScheduler
api.HPVCUserAuthValue = auth ' see previous example
Try
    Dim result as HPVCEventInformation
    result = api.DeleteEvent(eventID)
    ' Success
    If result Then
        Console.WriteLine("Event id " & eventID & " deleted")
    Else
        Console.WriteLine("Could not delete event")
    End If
Catch ex As SoapException
    ' Failure
    Console.WriteLine("Could not delete event")
End Try
```

Room Activity

This methods returns information about room activity.

The following structure is use to return information about a particular room entry.

```
Structure RoomActivityInfo
    Dim IsAnonymous As Boolean ' name information is not available
    Dim ParticipantName As String
    Dim ParticipantID As String ' uniquely identifies an
authenticated user
    Dim InTime As String
    Dim IsOutTimeNull As Boolean ' true if the user is still in the
room
    Dim OutTime As String ' only valid if IsOutTimeNull is false
    Dim Role As Role ' Presenter or Participant
    Dim IsOwner As Boolean ' true if the user is the room owner or a
scheduler
    Dim IsTechSupport As Boolean ' true if the user is tech support
End Structure
```

When dates and times are used as parameters or return values, they should be strings, using the following 24-hour format, expressed in UTC.

yyyy-mm-dd hh:mm

Examples:

```
"2007-02-08 17:00" '----- 5 p.m. on February 8, 2007
"2008-12-20 08:30" '----- 8:30 a.m. on December 20, 2008
```

GetRoomActivity

- Returns an information record for every room entry in the specified period.

```
GetRoomActivity(roomID As Integer, startTime as String, stopTime as String)  
As RoomActivityInfo()
```

On success returns an array of all room entries during the specified time.

On error throws a SOAP exception (see page 26).

Example:

```
Dim roomNumber As Integer = 1234  
Dim startTime As String = "2007-03-08 17:00"  
Dim stopTime As String = "2007-03-08 18:30"  
  
Dim api as New HPVCScheduler  
api.HPVCUserAuthValue = auth ' see previous example  
Try  
    Dim result as RoomActivityInfo()  
    result = api.GetRoomActivity(roomNumber, _  
        startTime, stopTime)  
    ' Success  
    Console.WriteLine(result.Length.ToString & " records")  
    For i As Integer = 0 To result.Length - 1  
        With result(i)  
            If .IsAnonymous Then  
                Console.Write("<anonymous>")  
            Else  
                Console.Write(.ParticipantName)  
            End If  
            If .Role = Role.Presenter Then  
                Console.Write(" (presenter) ")  
            Else  
                Console.Write(" (participant) ")  
            End If  
            ' Could additionally write times here  
            Console.WriteLine("")  
        End With  
    Next  
Catch ex As SoapException  
    ' Failure  
    Console.WriteLine("Room activity not available")  
End Try
```

Exceptions

All operations on the portal service are called as a SOAP request. If the request succeeds, the appropriate value is returned. If the request is unsuccessful, a SOAP exception is thrown, which should be handled by the calling application.

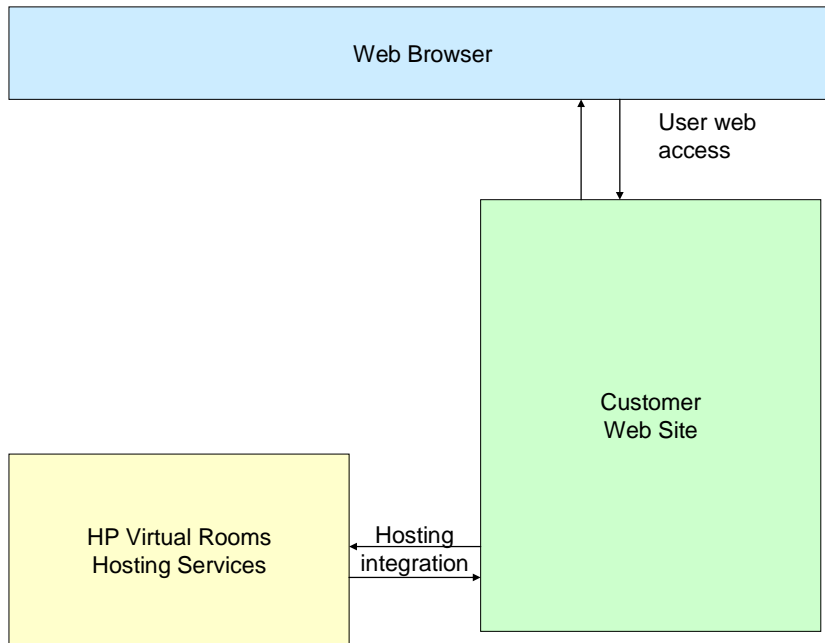
SOAP exceptions may be thrown for any number of reasons. An exception from the portal service is indicated by the presence of a child detail element with the name "HPVRoomError". Within this child element, an attribute of "HP:ErrorNo" contains the error number, "HP:ErrorDesc" may contain an additional description, and "HP:ErrorValue" may contain a value which qualifies the error. The errors currently thrown are listed in the following table.

Error Number	Meaning
10	Server Exception
20	Permission Denied
30	Key Use Error
400	Backpack Not Allowed
21	Unknown Role
300	Recurring Event Error
301	Scheduling Error
302	Delete Error
303	Cannot Delete Event Services
900	License Pool selected, not room
25	Authentication Expired
997	Authentication Data Error
998	Authentication Error
999	Authentication Failure

Example (examining a returned SoapException)

```
Try
    ....
Catch ex as SoapException
    If ex.Detail.HasChildNodes
        Dim childElement As XmlElement
        childElement = ex.Detail.Item("HPVRoomError")
        With childElement
            Console.WriteLine("HPVR Error Number: " & _
                .Attributes("HP:ErrorNo").Value.ToString)
            Console.WriteLine("HPVR Error Description: " & _
                .Attributes("HP:ErrorDesc").Value.ToString)
            Console.WriteLine("HPVR Error Additional Info: " & _
                .Attributes("HP:ErrorValue").Value.ToString)
        End With
    Else
        Console.Write("No additional information available")
    End If
End Try
```

Hosting integration



Calling mechanism

The room hosting API is implemented as a web service which is accessible using the HTTP protocol. Its methods are defined using Web Services Description Language. For security purposes, all transactions with the room hosting API will be carried as SSL transactions. Partner identification is carried in the encrypted SOAP header.

Partner identification

Customers using the hosting integration method will be identified by a unique string, or “Partner ID” which will be assigned by Hewlett-Packard at the time the hosting API account is set up. These customers will also be given an initial partner password which is required for access through the room hosting API.

Some customers may wish to subdivide part of their resource allocation among their own end-users. The room hosting API allows the reseller to create sub-accounts, specifying the resources allocated to the sub-account.

All requests to the room hosting API will include this two-part identifier – partner id and optional sub-account id, together with the partner password. This determines the set of resources used to fulfill the request.

Resource allocation

When a hosting API account is created, various resources will be allocated and certain limit values set. For example, the maximum number of available seats will be allocated, and the maximum size of any single event will be set. Availability of sufficient resources will be checked whenever events are scheduled or re-scheduled using the hosting API

Resellers who wish to create sub-accounts for their end-user customers can choose how to allocate these resources and may also restrict the limits for a particular sub-account. For example, a reseller who has purchased 500 concurrent seats with a maximum event size of 50 may create a sub-account

and allocate it 100 seats with a maximum event size of 25. In this case, only 400 seats will remain for the reseller to schedule through their main hosting API account. Note that limits may be reduced but not increased. In this example, it would not be possible to set the maximum event size for a sub-account greater than 50.

Authentication

All requests to the room hosting API should normally be accompanied by authentication information. This is carried in the SOAP header of the request, which is encrypted as the requests are carried across a secure link.

The authentication information consists of three parts: a partner identifier, which is assigned by HP when an account is created; a partner password, which is similarly assigned when an account is created and may be changed later on request; an optional subaccount identifier, which is used to identify any subdivision of the account resources that may have created (see page 41 for more details of how to create and delete subaccounts).

In general, requests relating to a particular room or event must use the authentication information for the subaccount (if any) where the room or event was created.

The authentication information definition is:

```
Class HPVRoomsPartnerAuth : Inherits SoapHeader
    AccountName as String,
    Pwd as String,
    SubAccountName as String
End Class
```

Example (create authentication structure)

```
Dim authHeader as New HPVRoomsPartnerAuth
With authHeader
    .AccountName = "MyPartnerName"
    .Pwd = "MyPassword"
End With
```

Rooms

To be able to schedule an event and attend it, you first need to create a room. Rooms are created and then tracked by a room Id. Once created, a list of current rooms can be accessed and rooms can be deleted.

CreateRoom

- Creates a new room and returns its identifier.

```
CreateRoom(PartnerRefStr As String) As Integer
```

On success returns RoomId.

On error throws a SOAP exception (see page 45).

Example

```
Dim myRoomName As String = "Conference Room 1"

Dim api as New HPVROOMS
api.HPVRoomsPartnerAuthValue = authHeader ' see above
Try
    Dim result as Integer
    result = api.CreateRoom(myRoomName)
    Console.WriteLine("Room " & result & " created")
Catch ex As SoapException
    Console.WriteLine("Unable to create room")
End Try
```

DeleteRoom

- Deletes a room.

DeleteRoom(RoomId As Integer) As Integer

On success returns 0.

On error throws a SOAP exception (see page 45).

Example

```
Dim roomID As Integer = 1234

Dim api as New HPVROOMS
api.HPVRoomsPartnerAuthValue = authHeader ' see above
Try
    Dim result as Integer
    result = api.DeleteRoom(roomID)
    Console.WriteLine("Room " & roomID & " deleted")
Catch ex As SoapException
    Console.WriteLine("Unable to delete room")
End Try
```

GetRooms

- Returns all rooms owned by the account and subaccount (if used).

GetRooms() As DataSet

On success returns dataset of rooms owned. The columns in the dataset are:

RoomID: Integer

AccountRoomName: String (the name provided on room creation)

On error throws a SOAP exception (see page 45).

Example

```
Dim api as New HPVROOMS
api.HPVRoomsPartnerAuthValue = authHeader ' see above
Try
    Dim result as DataSet
    result = api.GetRooms()
    With result
        Console.WriteLine(.Tables(0).Count.ToString & " rooms")
        For Each room as DataRow In result.Tables(0)
            Console.WriteLine(room("AccountRoomName") & " " & _
                room("RoomID"))
        Next
    End With
Catch ex As SoapException
    Console.WriteLine("Unable to get list of rooms")
End Try
```

GetAvailableCapacity

- Returns the available capacity in a room based on its identifier, time, duration, and requested resources. This may be used to determine possible times for an event prior to actually scheduling the event.
- Note that recording is currently not offered and all room activity is encrypted.

```
GetAvailableCapacity(RoomId As Integer,  
    ByVal StartTime As Date,  
    ByVal Duration As Integer, ' in minutes  
    ByVal Recorded as HostingOptions, ' use NORECORDING  
    ByVal HostingMonths as Integer,  
    ByVal Unused as Boolean,  
    ByVal Encrypted As Boolean) As Integer
```

On success returns available capacity.

On error throws a SOAP exception (see page 45).

Example

```
Dim startTime as DateTime = DateTime.UtcNow()  
Dim duration as Integer = 60  
  
Dim api as New HPVROOMS  
api.HPVRoomsPartnerAuthValue = authHeader ' see above  
Try  
    Dim result as Integer  
    result = api.GetAvailableCapacity(startTime, duration, _  
        HostingOptions.NORECORDING, 0, False, False)  
    Console.WriteLine(result.ToString() & " seats available")  
Catch ex As SoapException  
    Console.WriteLine("Unable to check availability")  
End Try
```

Events

Once a room is created, you can schedule and attend events in that room. A list of events can be accessed and events can be modified to change the date, time and capacity, to request recording or to be deleted. An event is referenced by an event Id.

CreateEvent

- Creates an event and returns the new identifier.
- Note that recording is currently not offered and all room activity is encrypted.

```
CreateEvent(ByVal RoomId As Integer,  
            ByVal StartTime As Date,  
            ByVal Duration As Integer, ' in minutes  
            ByVal Capacity As Integer,  
            ByVal Recorded As HostingOptions,  
            ByVal HostingMonths as Integer,  
            ByVal AutoExtend as Boolean, ' unused  
            ByVal unused as Boolean,  
            ByVal Encrypted as Boolean) As Integer
```

On success returns the identifier for the new event.

On error throws a SOAP exception (see page 45).

Example

```
Dim roomID As Integer = 1234  
Dim startTime as DateTime = DateTime.UtcNow()  
Dim duration as Integer = 60  
Dim capacity As Integer = 50  
  
Dim api as New HPVROOMS  
api.HPVRoomsPartnerAuthValue = authHeader ' see above  
Try  
    Dim result as Integer  
    result = api.CreateEvent(roomID, startTime, duration, capacity, _  
                            HostingOptions.NORECORDING, 0, _  
                            False, False, False)  
    Console.WriteLine("Event id " & result & " created")  
Catch ex As SoapException  
    Console.WriteLine("Unable to create event")  
End Try
```

DeleteEvent

- Deletes an event. If an event has

```
DeleteEvent(EventId As Integer) As Integer
```

On success returns 0.

On error throws a SOAP exception (see page 45).

Example

```
Dim eventID As Integer = 56789  
  
Dim api as NEW HPVROOMS  
api.HPVRoomsPartnerAuthValue = authHeader ' see above  
Try  
    Dim result as Integer  
    result = api.DeleteEvent(eventID)  
    Console.WriteLine("Event id " & result & " deleted")  
Catch ex As SoapException  
    Console.WriteLine("Unable to delete event")  
End Try
```

ModifyEvent

- Modifies an event. Note that some modifications may not be permitted after an event has started.

```
ModifyEvent(EventId As Integer,  
            ByVal StartTime As Date,  
            ByVal Duration As Integer, ' in minutes  
            ByVal Capacity As Integer,  
            ByVal Recorded As HostingOptions,  
            ByVal HostingMonths as Integer,  
            ByVal unused as Boolean,  
            ByVal Encrypted as Boolean) As Integer
```

On success returns EventId.

On error throws error a SOAP exception (see page 45).

Example

```
Dim eventID As Integer = 56789  
Dim startTime as DateTime = DateTime.UtcNow()  
Dim duration as Integer = 60  
Dim capacity As Integer = 50  
  
Dim api as New HPVROOMS  
api.HPVRoomsPartnerAuthValue = authHeader ' see above  
Try  
    Dim result as Integer  
    result = api.ModifyEvent(eventID, startTime, duration, capacity, _  
                            HostingOptions.NORecording, 0, _  
                            False, False)  
    Console.WriteLine("Event id " & result & " modified")  
Catch ex As SoapException  
    Console.WriteLine("Unable to modify event")  
End Try
```

GetEvents

- Returns events in the specified room.

GetEvents(ByVal RoomId As Integer) As DataSet

On success returns dataset of events in RoomId. Dataset contains:

EventID: Integer
RoomID: Integer
StartTime: DateTime
EndTime: DateTime
EventSize: Integer
Encrypted: Boolean
Recorded: RecordHostingOption
HostingMonths: Integer
Unused: Integer
AccessRecordingFiles: Boolean
unused: Boolean

On error throws error a SOAP exception (see page 45).

Example

```
Dim roomID As Integer = 1234

Dim api as New HPVROOMS
api.HPVRoomsPartnerAuthValue = authHeader ' see above
Try
    Dim result as DataSet
    result = api.GetEvents(RoomId)
    With result
        Console.WriteLine(.Tables(0).Count.ToString & " events")
        For Each ev as DataRow In result.Tables(0)
            Console.WriteLine(ev("ID") & ": " & ev("StartTime") & _
                " - " & ev("EventSize"))
        Next
    End With
Catch ex As SoapException
    Console.WriteLine("Unable to get list of events")
End Try
```

GetEventDetails

- Returns details of the specified event.

GetEventDetails(ByVal EventId As Integer) As DataSet

On success returns dataset of the details of the event EventId. The dataset structure is the same as for GetEvents.

On error throws error a SOAP exception (see page 45).

Example

```
Dim eventID As Integer = 56789

Dim api as New HPVROOMS
api.HPVRoomsPartnerAuthValue = authHeader ' see above
Try
    Dim result as DataSet
    result = api.GetEventDetails(eventID)
    With result
        If .Tables(0).Count = 1 Then
            Dim ev As DataRow = .Tables(0).Item(0)
            Console.WriteLine(ev("ID") & ": " & ev("StartTime") & _
                " - " & ev("EventSize"))
        Else
            Console.WriteLine("Event " & eventID & " does not exist")
        End If
    End With
Catch ex As SoapException
    Console.WriteLine("Unable to get event details")
End Try
```

Attend event

After you have created an event, you will want to attend it when it occurs. When you create an event, you are returned an event Id. The participants to this event will access your portal presenting the required credentials. When you decide to allow the participant to enter, you will call an interface method passing in the event Id, your identity of the participant (for attendance records), the name that should appear in the room plus others. The Role should be "Presenter" or "Attendee". In addition, you pass the desired language for rendering the room. If PrivacyOK is false, this attendee will appear as anonymous in attendance records.

You will be returned a complete encoded URL to which you will redirect the participant. At that point, they will enter the room for the event. Their attendance will be tracked by the specified participant identifier. Note that each returned URL is different and particular to the given participant. Each URL should only be used once.

GetEventURL

- Returns an event access URL for eventid.

```
GetEventURL(EventId As Integer,  
            PartnerRefStr As String,  
            ParticipantName As String,  
            Role As String,  
            Language as String,  
            PrivacyOk as Boolean) As String
```

On success returns URL to access event with the given parameters.

On error throws error a SOAP exception (see page 45).

Example

```
Dim eventID As Integer = 56789  
Dim participantName = "Joe Smith"  
Dim participantID = "Q54898"  
Dim role As String = "Presenter"  
Dim language as String = "en"  
Dim privacy As Boolean = True  
  
Dim api as New HPVROOMS  
api.HPVRoomsPartnerAuthValue = authHeader ' see above  
Try  
    Dim result as String  
    result = api.GetEventURL(eventID, _  
                            participantID, participantName, _  
                            role, language, privacy)  
    Console.WriteLine("URL: " & result)  
Catch ex As SoapException  
    Console.WriteLine("Unable to get event URL")  
End Try
```

GetRoomURL

- Returns a URL to enter the specified room as the room owner. Only a limited number of people may enter each room as owner at any one time. Room owners always enter as presenter.

```
GetRoomURL(RoomId As Integer,  
            PartnerRefStr As String,  
            ParticipantName As String,  
            Language as String,  
            PrivacyOk as Boolean) As String
```

On success returns URL to access room with the given parameters.

On error throws error a SOAP exception (see page 45).

Example

```
Dim roomID As Integer = 1234  
Dim participantName = "Joe Smith"  
Dim participantID = "Q54898"  
Dim language as String = "en"  
Dim privacy As Boolean = True  
  
Dim api as New HPVROOMS  
api.HPVRoomsPartnerAuthValue = authHeader ' see above  
Try  
    Dim result as String  
    result = api.GetRoomURL(roomID, _  
                            participantID, participantName, _  
                            language, privacy)  
    Console.WriteLine("URL: " & result)  
Catch ex As SoapException  
    Console.WriteLine("Unable to get room URL")  
End Try
```

Attendance records

You can track event attendance. The participant identity you give to GetEventURL is used to obtain attendance data about an individual. You can also request records by event Id and room Id.

AttendanceForEvent

- Returns the event attendance event id as a dataset. The role list is expressed as series of characters (P=Presenter, A=Attendee, O=Owner, Z=Tech Support, Y=Privacy OK). The OutTime may be null if the user is still in the event. The participant ID is the reference string presented when creating the original event URL.

AttendanceForEvent(EventId As Integer) As Dataset

On success returns dataset of attendance records. Dataset contains:

ParticipantID: String

Role: String

InTime: DateTime

OutTime: DateTime

On error throws a SOAP exception (see page 45).

Example

```
Dim eventID As Integer = 56789

Dim api as New HPVROOMS
api.HPVRoomsPartnerAuthValue = authHeader ' see above
Try
    Dim result as DataSet
    result = api.AttendanceForEvent(eventId)
    With result
        Console.WriteLine(.Tables(0).Count.ToString & " records")
        For Each user as DataRow In result.Tables(0)
            Console.WriteLine(user("ParticipantID") & " " & _
                user("InTime"))
        Next
    End With
Catch ex As SoapException
    Console.WriteLine("Unable to get attendance records")
End Try
```

AttendanceForRoom

- Returns the event attendance room id as a dataset.

```
AttendanceForRoom(ByVal RoomId As Integer,  
    ByVal StartTime as Date,  
    ByVal StopTime as Date) As Dataset
```

On success returns dataset of attendance records. The dataset elements are the same as for AttendanceForEvent.

On error throws a SOAP exception (see page 45).

Example

```
Dim roomID As Integer = 1234  
Dim fromDate As New DateTime(2007,2,1)  
Dim toDate As New DateTime(2007,2,8)  
  
Dim api as New HPVROOMS  
api.HPVRoomsPartnerAuthValue = authHeader ' see above  
Try  
    Dim result as DataSet  
    result = api.AttendanceForRoom(roomID, fromDate, toDate)  
    With result  
        Console.WriteLine(.Tables(0).Count.ToString & " records")  
        For Each user as DataRow In result.Tables(0)  
            Console.WriteLine(user("ParticipantID") & " " & _  
                user("InTime"))  
        Next  
    End With  
Catch ex As SoapException  
    Console.WriteLine("Unable to get attendance records")  
End Try
```

AttendanceForParticipant

- Returns the event attendance for a particular participant Id as a dataset.

AttendanceForParticipant (ByVal ParticipantId As String) As Dataset

On success returns dataset of attendance records. Dataset contains:

EventID: Integer
Role: String
InTime: DateTime
OutTime: DateTime

On error throws a SOAP exception (see page 45).

Example

```
Dim participantID As String = "Q54898"

Dim api as New HPVROOMS
api.HPVRoomsPartnerAuthValue = authHeader ' see above
Try
    Dim result as DataSet
    result = api.AttendanceForParticipant(participantID)
    With result
        Console.WriteLine(.Tables(0).Count.ToString & " records")
        For Each user as DataRow In result.Tables(0)
            Console.WriteLine(user("ParticipantID") & " " & _
                user("InTime"))
        Next
    End With
Catch ex As SoapException
    Console.WriteLine("Unable to get attendance records")
End Try
```

Message passing

Communication between the API application and the hosting service is, by definition, always customer-initiated. No assumption is made about the ability of the hosting service to make callbacks, thus allowing the customer to locate their application behind their own firewall.

In order for the portal to update the API application when certain events happen, a message queue is provided. The API application should query this queue on a regular basis in order to remain updated on the status of its rooms and events.

The methods in this section may only be used when authenticated as the main account (not as a subaccount).

GetAllNewMessages

- Returns a dataset containing all messages which have not been previously received. Calling GetAllNewMessages results in all messages which are returned being marked as read.

GetAllNewMessages() As Dataset

On success returns a dataset containing the following:

MessageID: Integer,
PartnerID: String,
MessageType: Integer
MessageTypeID: Integer
MessageValue: Integer
MessageDesc: String
MessageDate: Date

On error throws a SOAP exception (see page 45).

The types of messages currently generated are:

MessageTypeID	Detail
1	Event details have been changed by the hosting service. Currently this only used when an event is auto-extended. MessageTypeID contains the event identifier.

Example

```
Dim api as New HPVROOMS
api.HPVRoomsPartnerAuthValue = authHeader ' see above
Try
    Dim result as DataSet
    result = api.GetAllNewMessages()
    With result
        Console.WriteLine(.Tables(0).Count.ToString & " records")
        For Each msg as DataRow In result.Tables(0)
            If CType(msg("MessageType"), Integer) = 1 Then
                Console.WriteLine("Event " & msg("MessageTypeID") & _
                    " has been extended")
            End If
        Next
    End With
Catch ex As SoapException
    Console.WriteLine("Unable to get messages")
End Try
```

GetAllMessages

- Returns a dataset containing all messages regardless of whether they have previously been received or not.

GetAllMessages() As Dataset

On success, returns a dataset of messages. The format of the dataset is the same as for GetAllNewMessages.

On error throws a SOAP exception (see page 45).

Example

```
Dim api as New HPVROOMS
api.HPVRoomsPartnerAuthValue = authHeader ' see above
Try
    Dim result as DataSet
    result = api.GetAllMessages()
    With result
        Console.WriteLine(.Tables(0).Count.ToString & " records")
        For Each msg as DataRow In result.Tables(0)
            If CType(msg("MessageType"), Integer) = 1 Then
                Console.WriteLine("Event " & msg("MessageTypeID") & _
                    " has been extended")
            End If
        Next
    End With
Catch ex As SoapException
    Console.WriteLine("Unable to get messages")
End Try
```

DeleteMessageByMessageId

- Deletes a message from the message queue so that it will not be received again (even by GetAllMessages). This should be used after successful processing of a message.

DeleteMessageByMessageId(messageId As Integer) as Integer

On success, returns 0.

On error throws a SOAP exception (see page 45).

Example

```
Dim api as New HPVROOMS
api.HPVRoomsPartnerAuthValue = authHeader ' see above
Try
    Dim result as DataSet
    result = api.GetAllNewMessages()
    With result
        Console.WriteLine(.Tables(0).Count.ToString & " records")
        For Each msg as DataRow In result.Tables(0)
            If CType(msg("MessageType"), Integer) = 1 Then
                Console.WriteLine("Event " & msg("MessageTypeID") & _
                    " has been extended")
                api.DeleteMessageByMessageID( _
                    CType(msg("MessageID"), Integer))
            End If
        Next
    End With
Catch ex As SoapException
    Console.WriteLine("Unable to get messages")
End Try
```

Account operations

You can create subaccounts as a means of controlling access to your resources. For example, you could create two subaccounts and assign half your available seats to each. Now, an event created as part of one subaccount can only use half the seat maximum. This can be useful if you host multiple end-user accounts in a service provider model and wish to have the hosting service control resource allocation for you.

The methods in this section may only be used when authenticated as the main account (not as a subaccount).

CreateSubAccount

- Creates a new subaccount.

CreateSubAccount(name as String) as Integer

On success returns 0.

On error throws a SOAP exception (see page 45).

Example

```
Dim subAccountName as String = "MyClient"

Dim api as New HPVROOMS
api.HPVRoomsPartnerAuthValue = authHeader ' must not set SubAccount
Try
    Dim result as Integer
    result = api.CreateSubAccount(subAccountName)
    Console.WriteLine("Sub account " & subAccountName & " created")
Catch ex As SoapException
    Console.WriteLine("Unable to create sub account")
End Try
```

DeleteSubAccount

- Deletes a subaccount. All rooms and events associated with the subaccount will also be deleted.

DeleteSubAccount(name as String) as Integer

On success returns 0.

On error throws a SOAP exception (see page 45).

Example

```
Dim subAccountName as String = "MyClient"

Dim api as New HPVROOMS
api.HPVRoomsPartnerAuthValue = authHeader ' must not set SubAccount
Try
    Dim result as Integer
    result = api.DeleteSubAccount(subAccountName)
    Console.WriteLine("Sub account " & subAccountName & " deleted")
Catch ex As SoapException
    Console.WriteLine("Unable to delete sub account")
End Try
```

UpdateSubAccountResources

- Allocates some set of resources of the parent account to the subaccount. Note that this must be called after creating a new subaccount to allocate resources.
- You will be given information about your available resources when your room hosting API account is created. It is not possible to give away more resources than you own. In addition, some parameters have system-imposed limits. Exceeding these limits will result in an error containing further detail.

```
UpdateSubAccountResources(name as String,  
    seats as Integer,  
    PrivateRecorders as Integer, ' unused  
    SharedRecorders as Integer, ' unused  
    RecordingHosting as HostingOptions, ' unused  
    voip as Boolean, ' unused  
    maxRooms as Integer,  
    maxEventSize as Integer,  
    maxEventMinutes as Integer,  
    eventLeadTimeMinutes as Integer) as Integer
```

On success returns 0.

On error throws a SOAP exception (see page 45).

Example

```
Dim subAccountName as String = "MyClient"  
Dim seats as Integer = 50  
Dim maxRooms As Integer = 10  
Dim maxEventSize As Integer = 20  
Dim maxEventDuration As Integer = 60  
Dim eventLeadTime As Integer = 15  
  
Dim api as New HPVROOMS  
api.HPVRoomsPartnerAuthValue = authHeader ' must not set SubAccount  
Try  
    Dim result as Integer  
    result = api.UpdateSubAccountResources(subAccountName, _  
        seats, 0, 0, HostingOptions.NORecording, False, _  
        maxRooms, maxEventSize, maxEventDuration, _  
        eventLeadTime)  
    Console.WriteLine("Sub account " & subAccountName & " modified")  
Catch ex As SoapException  
    Console.WriteLine("Unable to modify sub account")  
End Try
```

ChangeRoomSubAccount

- Moves a room from one subaccount to another. The room identified by roomID is moved to the specified subaccount.

```
ChangeRoomSubAccount(SubAccountName as String,  
    roomID as Integer) as Integer
```

On error returns 0.

On error throws a SOAP exception (see page 45).

Example

```
Dim roomID as Integer = 1234  
Dim subAccountName as String = "MyNewClient"  
  
Dim api as New HPVROOMS  
api.HPVRoomsPartnerAuthValue = authHeader ' must not set SubAccount  
Try  
    Dim result as Integer  
    result = api.ChangeRoomSubAccount(subAccountName, roomID)  
    Console.WriteLine("Room " & roomID & " reassigned")  
Catch ex As SoapException  
    Console.WriteLine("Unable to reassign room")  
End Try
```

Exceptions

All operations on the room hosting service are called as a SOAP request. If the request succeeds, the appropriate value is returned. If the request is unsuccessful, a SOAP exception is thrown, which should be handled by the calling application.

SOAP exceptions may be thrown for any number of reasons. An exception from the room hosting service is indicated by the presence of a child detail element with the name "HPVRoomError". Within this child element, an attribute of "HP:ErrorNo" contains the error number, "HP:ErrorDesc" may contain an additional description, and "HP:ErrorValue" may contain a value which qualifies the error. The errors currently thrown are listed in the following table.

Error Number	Meaning
10	Internal Error (description gives details)
15	Error preparing room for user (description gives details)
200	Invalid input data (description gives details)
201	Illegal operation (description gives details)
202	Maximum number of subaccounts has been reached
203	Recorder not available
204	A conflicting event exists
205	Insufficient seats to fulfill request
206	No recorder available
208	Not enough capacity to fulfill request
209	Event has not yet started (value is minutes until event)
210	Event has already finished
211	Event occurs during system blackout
212	Cannot create room
213	Event capacity exceeded
215	Cannot delete room because it is active

Example (examining a returned SoapException)

```
Try
    ....
Catch ex as SoapException
    If ex.Detail.HasChildNodes
        Dim childElement As XmlElement
        childElement = ex.Detail.Item("HPVRoomError")
        With childElement
            Console.WriteLine("HPVR Error Number: " & _
                .Attributes("HP:ErrorNo").Value.ToString)
            Console.WriteLine("HPVR Error Description: " & _
                .Attributes("HP:ErrorDesc").Value.ToString)
            Console.WriteLine("HPVR Error Additional Info: " & _
                .Attributes("HP:ErrorValue").Value.ToString)
        End With
    Else
        Console.Write("No additional information available")
    End If
End Try
```

For more information

www.hp.com/info/rooms

© 2007 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice. The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

January 2007

