StarRez REST 2.0 Web Services API Guide FEATURED

Welcome to the StarRez REST 2.0 Web Services API Guide. Please see the information below.

Terms of Service

Download

- For any questions on licensing, please contact your account manager.
- Please post all questions about REST Web Services and other technical topics on the Tech Corner Community Forum on StarCare Online.
- **Note:** We do not recommend the use of the ValidateCredentials REST Endpoint as this may be deprecated in the future.

StarRez REST 2.0 Services User Guide v17
Table of Contents

Introduction

Customers who want to interact with StarRez using their own applications can use the StarRez REST web services. Connecting to these web services can be achieved by using any client technology that provides a connection using HTTP. This flexibility enables the web service to be an extremely powerful option for extending StarRez.

The StarRez REST Web Service has the following key benefits for a StarRez customer:

- Allows control of data within
- Shortens the Development
- Lowers the cost and fees paid to StarRez for custom

All examples within this document use Microsoft's C# programming language but any programming language that supports HTTP can be used.

Best Practices

The following are recommended best practices for working with the Web Services:

- XML fragments is the best method for transferring data to our Web Easier to read and understand and work with.
- The other option is GET and POST but these are not as easy to work with.

Version 2.0 Enhancements

Web Service	Feature
RestService	SetPhoto function.
	GetPhoto function.
Reports	One function to select reports with report name or ID.

Terminology

Term	Description
REST	Representational State Transfer. Architectural principle used to create web services that are stateless and expose directory structure Uniform Resource Indicators (URI's).
HTTP	Hypertext Transfer Protocol
IIS	Internet Information Services. Microsoft server hosting application.

How to read this document

This document implements some basic conventions.

Curly Braces – '{}'

Any text that contains curly braces is to be replaced by what is described between the curly braces: https://{Portal REST URL}/services

In the example above, the value '{Portal REST URL}' needs to be replaced with a valid path to a StarRez

Portal REST instance. **Brackets – '[]'**

Any text that contains brackets is optional. Multiple optional values are separated by the ','.

Pipes – '|'

Pipe characters that appear between Curly Braces or Brackets indicate a selection of possible values.

Web Services

The following REST web services are available from StarRez:

Web Service	Description
RestService	Provides access to all StarRez data. Data from Tables within StarRez can be selected and updated. Records can also be created and existing records can be deleted.
Reporting	Provides access to existing reports on StarRez. Reports can be downloaded in a variety of formats (XML, CSV, etc.).
Accounts	Provides the ability to obtain an Entries balance and create transactions and payments.
DatabaseInfo	Provides access to the Database Tables, Columns and Functions.
StudentAuth	Provides the ability to check credentials and authenticate via REST.

Please check with StarRez if you are unsure which web services you currently have available.

Requests

The REST web service is flexible. You can send request data to the web service using any of the following methods:

Within the URL (GET)

This is the same as examples shown previously. Any request data is simply appended as parameters to the web service URL. This allows data to be selected easily using just a URL.

Within the request body (POST)

The request data is placed within the body of any HTTP request (POST) in the following format:

field=value&field=value

Within the request body (POST) as XML

The request data is placed within the body of any HTTP request (POST) in XML format. This is the most powerful and expressive way of interacting with the web services, and is the recommended method for most use cases.

Within the request body (POST) as JSON

The request data is placed within the body of any HTTP request (POST) in JSON format. This is as powerful as the XML format, and is the best way to interact with the web services from languages that have good JSON support (e.g., JavaScript).

Functions

Each web service provides functions which can be used to perform a variety of tasks on StarRez. This section describes each function for each web service.

RestService

Query

Allows the user to select ad-hoc data from the database using StarRez Query Language (StarQL). Supports POST or GET data (using the q parameter). The data will be returned in a format appropriate for desired content accept type.

URL:

/query

Example Query

The following example selects an Entry's First Name and Last Name for the Entry with an ID of '31'.

URL:

/query

POST Data:

SELECT NameFirst, NameLast FROM Entry WHERE EntryID = 31

GET:

/query?q=select namefirst from Entry where EntryID = 31

For more information on StarQL please see https://support.starrez.com/hc/en-us/articles/205268046- StarQL-Guide.

Select

Selects data from a table and optionally a related table.

Request Format URL:

/select/{table}[{.format}][/{id}]

XML:

<table>

<field>**value**</{field}>

</table>

GET:

/select/{table}{.format}/{id}

Example select

The following example selects the Entry with an ID of '31'.

URL:

/select/entry

XML:

<Entry>

<EntryID>**31**</EntryID>

</Entry>

GET:

/select/entry/31

JSON:

{
"EntryID": "31"
}

Example selection of a custom field

The following will select an Entry custom field with the ID of '2'.

URL:

/select/entrycustomfield

XML:

<EntryCustomField>

<EntryCustomFieldID>**2**</EntryCustomFieldID>

</EntryCustomField>

GET:

/select/entrycustomfield/2

JSON:

{
"CustomFieldDefinitionID" :"2"
}

Example selection of a custom field without specifying the custom field ID.

The following will select an Entry custom field for the Entry with an ID of '31' and with a CustomFieldDefinitionID of '2'.

URL:

/select/entrycustomfield

XML:

<EntryCustomField>

<EntryID>**31**</EntryID>

<CustomFieldDefinitionID>**2**</CustomFieldDefinitionID>

</EntryCustomField>

GET:

/select/entrycustomfield?entryid=31&customfielddefinitionid[_operator%3Din]

=2

Example selection of Entries using data in related table as a filter.

The following will select an Entry which has the StaffID of '1' which is stored in the EntryDetail table.

- **NB.** Using related tables to filter is only supported by XML queries.

URL:

/select/entry

XML:

<Entry>

<EntryDetail>

<StaffID>**1**</StaffID>

</EntryDetail>

</Entry>

JSON:

```
{
"EntryDetail": {
"StaffID" : "1" }
}
```

Example selection of Entries using a sub query of data in related table as a filter.

Criteria can be nested in any sub query. By default queries are joined using an 'AND', however sub queries can also use the 'or' operator.

The following will select Entries whose first name is 'John' and who have the StaffID of '1' or '2' which is stored in the EntryDetail table.

URL:

/select/entry

XML:

<NameFirst>**John**</NameFirst>

<EntryDetail>

<_relationship>**or**</_relationship>

<StaffID>**1**</StaffID>

<StaffID>**2**</StaffID>

</EntryDetail>

</Entry>

JSON:

```
{ "NameFirst": "John",
"_criteria": {
"EntryDetail": {
"StaffID" : "1",
"StaffID": "2",
"_relationship": "OR"}
}
}
```

Example selection of Entries using a nested sub query and empty table nodes.

Related Table sub queries can be nested more than one layer. Empty table nodes will use the "exists" operator.

The following will select Entries who have an EntryApplicationRoomPreference record and also have an EntryApplicationNote with NoteType ‘A note’.

- **NB.** Using related tables to filter is only supported by XML queries.

URL:

/select/entry

XML:

```
<Entry>
<EntryApplication>
<EntryApplicationRoomPreference />
<EntryApplicationNote>
<NoteType>A note</NoteType>
</EntryApplicationNote>
</EntryApplication>
</Entry>
```

Example specifying multiple values to select entries

URL:

/select/entry

JSON:

```
{
"NameLast": { "_operator": "in", "value": "Smith,Jones,Truman" }
}
```

Special Attributes

Attributes allow additional filtering of results. A full list of Special Attributes can be found in Appendix C – Special Attributes.

Example select with Special Attributes

The following example selects all Entries where the first name is ‘John’ and the date of birth is after January 1st 1980 and before January 1st 1990.

URL:

/select/entry

XML:

```
<NameFirst _operator="in">John</NameFirst>
<DOB _operator="gt">1980-01-01</DOB>
<DOB _operator="lt">1990-01-01</DOB>
</Entry>
```

GET:

/select/Entry?NameLast[_operator%3Din]=John&DOB[_operator%3Dgt]=1980-01- 01&DOB[_operator%3Dlt]=1990-01-01

JSON:

```
{
"NameFirst": {"_operator": "in", "value":"John"},
"DOB": {"_operator": "gt", "value": "1980-01-01"},
"DOB": {"_operator": "lt", "value": "1990-01-01"},
}
```

Special Fields

Special fields allow the additional filtering of results. A full list of Special fields can be found in Appendix B

– Special Fields.

Example select with Special Fields

The following example selects an Entry with an ID of '31'. In addition, it also selects all fields from the 'EntryDetail' related table.

URL:

/select/entry

XML:

<Entry>

<EntryID>**31**</EntryID>

<_relatedTables>**EntryDetail**</_relatedTables>

</Entry>

GET:

/select/Entry/31?_relatedtables=EntryDetail

Result Formats

Results are sent to the client as ATOM feeds by default. Results can also be returned in XML format. When the StarRez API returns a result as XML, it includes just the content of what would have been returned by the same result in an ATOM feed. An XML result can be specified by specifying the '.xml' extension within the request URL.

Example select URL that specifies that the result should be in XML format

The following will select an Entry with the ID of '31' and return the results in XML format.

XML:

<Entry>

<EntryID>**31**</EntryID>

<_relatedTables>**EntryDetail**</_relatedTables>

</Entry>

GET:

/select/Entry/31?_relatedtables=EntryDetail

Error Messages

The web service will return a HTTP Status Code specifying what type of error has occurred (refer to Appendix A – HTTP Status Codes). In addition an XML fragment will be returned providing a description of the error.

Example error message

The following example shows the error message returned for a request for Entries with a first name of 'Jane'.

<errors>

<error>

<description>**No record exists in table Entry which match the desired criteria.**</description>

</error> </errors>

The above error specifies that no records exist that have a first name of 'Jane'. The HTTP Status Code within this result was '404'.

Update

Updates a table record.

Request Format

URL:

/update/{table}/{id}

XML:

<table>

<field>**value**</field>

</table>

GET:

/update/{table}/{id}?

{field}={value}&{field}={value}..

Example update

The following example updates the 'NamePreferred' field for the Entry with an ID of '31'.

URL:

/update/entry/31

XML:

<Entry>

<NamePreferred>**Mr Smith**</NamePreferred>

</Entry>

GET:

/update/entry/31?namepreferred=Mr%20Smith

Example update of a custom field

The following will update a boolean Entry custom field with the ID of '21' to the value 'false'. For a full list of custom field types refer to Appendix E – Custom Field Types.

URL:

/update/entrycustomfield/21

XML:

<EntryCustomField>

<ValueBoolean>**false**</ValueBoolean>

</EntryCustomField>

GET:

/update/entrycustomfield/21?valueboolean=false

Related Tables

Related tables can also be updated. Record to update within the related table can be specified and optionally created if they do not exist.

Example updating a related table record

The following example will update the 'Occupation' field within the related table 'EntryDetail' for the Entry with an ID of '31':

URL:

/update/entry/31

XML:

<Entry>

<EntryDetail>

<Occupation>**Lawyer**</Occupation>

</EntryDetail>

</Entry>

Example updating a related table where related record exists

The following example will update the related table 'Booking' record field 'NumberOfChildren' within the where the 'TermSessionID' is '16':

URL:

/update/entry/31

XML:

<Entry>

<Booking TermSessionID="7">

<NumberOfChildren>1</NumberOfChildren>

</Booking>

</Entry>

GET:

/update/entry/31?Booking[TermSessionID%3D7].NumberOfChildren=1

Example updating a related table where related record does not exist

The following example will update the related table 'Booking' record field 'NumberOfChildren' within the where the 'TermSessionID' is '7'. If a booking with this criteria does not exist then it will be created:

URL:

/update/entry/31

XML:

<Entry>

<Booking TermSessionID="16" _create="true">

<NumberOfChildren>1</NumberOfChildren>

</Booking>

</Entry>

GET:

/update/entry/31?Booking[TermSessionID%3D7%26_create% 3Dtrue].NumberOfChildren=1

Create

Creates a record for a table. This will also create any required records in related tables.

Request URL:

/create/{table}

XML:

<table>

<field>**value**</{field}>

</table>

GET:

/create/{table}?{field}={value}&{field}={value}..

Example creating a record

The following will create a new Entry record for 'John Smith'.

URL:

/create/entry

XML:

```
<Entry>

<NameLast>Smith</NameLast>

<NameFirst>John</NameFirst>

</Entry>
```

GET:

/create/Entry?NameLast=Smith&NameFirst=John

Example creating a record and a related record

The following will create a new Entry record for ‘John Smith’ and create a new Booking for TermSessionID ‘15’.

URL:

/create/entry

XML:

```
<Entry>

<NameLast>Smith</NameLast>

<NameFirst>John</NameFirst>

<Booking>

<TermSessionID>15</TermSessionID>

</Booking>

</Entry>
```

Example updating a custom field

The following updates a custom date field for an Entry custom field with an ID of ‘2’.

URL:

/update/{customfieldtable}/{id}

XML:

```
<EntryCustomField>

<value>2015-01-01</value>

</EntryCustomField>
```

GET:

/update/entrycustomfield/2?value=2015-01-01

Example creating a group and updating the group custom field records

URL:

/create/group

JSON:

```
{
"Description": "Soccer Camp 2022",
"CheckInDate": "2022-12-01T08:00:00",
"CheckOutDate": "2022-12-31T17:00:00",
"GroupCustomField": [
{
"_filter":{
CustomFieldDefinitionID:"12"
},
"ValueString": "0002"
},
{
"_filter":{
CustomFieldDefinitionID:"13"
},
"ValueBoolean": true
}
]
}
```

Delete

Deletes a record from a table. Records from related tables may also be deleted. The Delete function requires no data to be sent within the request.

Request URL:

/delete/{table}/{id}

Example deleting a record

The following will delete the Entry with an ID of ‘45’. URL:

/delete/entry/45

SetPhoto

Allows the user to set a photo that represents a record, e.g. Entry Photo. This function can only be used with HTTP POST. Attempting to upload an image using HTTP GET will result in an error. If the table requested does not provide the ability to upload a photo then an error will be returned. Images can be sent within the body of the requesting HTTP POST message in either raw binary or Base64 format.

Request

URL:

/setphoto/{table}/{id}

Example setting Entry Photo

The following will set the photo for an Entry with an ID of ‘45’. URL:

/setphoto/entry/45

Supported Image Formats

Name	Description
Bmp	Bitmap (BMP) image format.

Gif	Graphics Interchange Format (GIF) image format.
Jpeg	Joint Photographic Experts Group (JPEG) image format.
Png	W3C Portable Network Graphics (PNG) image format.

GetPhoto

Retrieves a photo for a given table in JPEG format. An optional size parameter can be sent to specify the maximum size of the photo in pixels. If either the width or height of the photo extends beyond the given maximum size then the photo will be resized proportionally before being returned.

Request

URL:

/photo/{table}/{id}/[{size}]

Example getting Entry Photo

The following will get the photo for an Entry with an ID of '45'.

URL:

/photo/entry/45

GetNextMessage

Retrieves the next Data Subscriptions message that needs to be processed by the specified message subscriber. Subscribers are set up in the Data Subscriptions setup area of StarRez. The result of this web service method includes all of the details of a message, including existing and new data if the message action supports it. For XML requests (recommended) the existing and new data is returned as part of the message, rather than being encoded into it. For JSON requests the existing and new data is encoded as XML data into a JSON string.

The request will return a 404 if there are no messages to process. Once a message has been processed it should be deleted (using /delete/message/{message_id}). Until that message is deleted the GetNextMessage function will continue to return the same message.

Request

URL:

/getnextmessage/{subscriber_id}

Example getting a message

The following will get the next message to be processed for Message Subscriber 3. URL:

/getnextmessage/3

GetAttachment

Allows the user to retrieve an attachment from the database.

URL: https://globalsuppstarrezrest.starrez.cloud/StarRezREST/services/attachment/{id}

Parameter	Optional	Default Value	Description
id	No		

ListAttachments

Allows the user to retrieve a list of attachments from the database for the specified record.

URL: https://globalsuppstarrezrest.starrez.cloud/StarRezREST/services/attachment/{tableName}/{id}

Parameter	Optional	Default Value	Description
tableName	No		
id	No		

***TableName refers to the RecordAttachment.TableName value in the StarRez Database, so if the attachment is on the Entry then you would use Entry as the tableName.

RoomListService

GetRoomList: Gets the free room list for the specified dates.

URL: https://globalsuppstarrezrest.starrez.cloud/StarRezREST/services/roomlist/{dateStart}/{dateEnd}

Parameter	Description
ListType	Possible values: FreeRooms, FreeAndCurrent, Reserved, InRoom, AllRooms, AllOpenRooms Specifies the type of list to return. Defaults to 'FreeRooms' if not specified.
MandatoryProfilesToApply	Possible values: Student, Staff, Both Sets a value indicating which mandatory profiles to apply if the for the MustMatchMandatoryProfiles and EntryIDToMatchSpecialProfiles options are used

IncludeOriginalGender	True/False Sets a value indicating whether to include a rooms original gender in the results
RoomHoldEntryID	Number Sets the EntryID for which to include held rooms for, and exclude rooms held for other entries
IncludeRoomRate	True/False Sets a value indicating whether to include room rate details in the search results
RoomRateID	Number Specifies a specific room rate to use
LowerRoomRateValue	Number The lower value for Room Rate filtering
UpperRoomRateValue	Number The upper value for Room Rate filtering
RoomHold_PortalUserTokenID	Number Sets the PortalUserTokenID for which to include held rooms for, and exclude rooms held for other entries
IncludeFloorWebComments	True/False Sets value indicating whether to include the Web Comments from the floor
ShowOnlyBeds	True/False Sets a value indicating whether to show only bed type room spaces
IncludeRoomLocationAllocateSortOrder	True/False Sets a value indicating whether to include room location allocate sort order
IncludeRoomAllocateSortOrder	True/False Sets a value indicating whether to include room allocate sort order
IncludeRoomSpaceRoomRateID	True/False Sets a value indicating whether to include room rate from the room rate saved against the roomspace
IncludeRoomLocationAreaID	True/False Sets a value indicating whether to include room location allocate sort order

IncludeRoomLocationSectionID	True/False Sets a value indicating whether to include room location section ID
IncludeClosedRoomSpacesInTotal	True/False Sets a value indicating whether to include any closed room in the Total Room Spaces count. Dynamic Gender and IncludeTotalRoomSpaces must be set for this to work.
IncludeTotalRoomSpaces	True/False Sets a value indicating whether to select the Total Room spaces. Dynamic Gender must be turned on for this to work
IncludeActualRoomSpaceTotal	True/False Sets a value indicating whether to select the actual room space total, potentially excluding closed or booked rooms, depending on the room list type of the search
RoomTypeFromRoomTypeCapacity	True/False Sets a value indicating whether the room type description should come from the room type capacity table
IncludeRoomTypeCapacityField	True/False Sets a value indicating whether to include the RoomTypeCapacity.Capacity field
MinimumFreeBeds	Number Sets the minimum free beds. This setting can only be used in conjuction with SelectRoomsNotRoomSpaces
ExactNumberOfFreeBeds	Number Sets the exact number of free beds. As per MinimumFreeBeds but an exact match
RoomListSearchLevel	Possible values: RoomType, RoomLocationArea, RoomLocation, RoomLocationSection, RoomLocationFloorSuite, Room, RoomSpace Sets a value indicating whether the room list will contain Rooms instead of RoomSpaces. This is not valid with AllRooms list type

ViewOnWebOnly	True/False Sets whether to only show rooms that are viewable on the web
ViewOnWebAtAllLevels	True/False Sets whether to only show rooms that are viewable on the web at all room levels
EntryIDToMatchSpecialProfiles	Number Sets the entry ID to match special profiles against. If set, any special profiles this entry has, must also exist on the room, or the room must have no special profiles
MustMatchMandatoryProfiles	True/False Sets a value indicating whether the room mandatory profiles must match the entry mandatory profiles
MatchMandatoryProfilesOption	Possible values: Ignored, MustMatchAtLeastOne, MustMatchAll Sets a value indicating whether the entry mandatory profiles don't have to or must match one/all room mandatory profiles
IncludeRoomsWithoutMandatoryProfiles	True/False Sets a value indicating whether the room without mandatory profiles should be included in the results when the entry has mandatory profiles
IncludeEntryIDFields	True/False
IncludeRoomSpaceExtensionFields	True/False
RoomLocationFloorSuiteIDFilter	Number. Multiple values can be separated by commas. Sets the RoomLocationFloorSuite IDs to filter by
RoomLocationSectionIDFilter	Number. Multiple values can be separated by commas. Sets the RoomLocationSection IDs to filter by
ClassificationIDFilter	Number. Multiple values can be separated by commas. Sets the Classification IDs to filter by

RoomSpaceIDFilter	Number. Multiple values can be separated by commas. Sets the RoomSpace IDs to filter by
RoomTypeIDFilter	Number. Multiple values can be separated by commas. Sets the RoomType IDs to filter by
RoomBaseIDFilter	Number. Multiple values can be separated by commas. Sets the RoomBase IDs to filter by
RoomLocationAreaIDFilter	Number. Multiple values can be separated by commas. Sets the RoomLocationArea IDs to filter by
RoomLocationIDFilter	Number. Multiple values can be separated by commas. Sets the RoomLocation IDs to filter by
ProfileTypeIDFilter	Number. Multiple values can be separated by commas. Sets the ProfileType IDs to filter by
ProfileItemIDFilter	Number. Multiple values can be separated by commas. Sets the ProfileItem IDs to filter by
TermTypeIDFilter	Number. Multiple values can be separated by commas. Sets the TermType IDs to filter by
RoomGenderFilter	Possible values: DynamicGender, Male, Female, Neutral, CoEd Sets the room gender type to filter by
UseContractDates	True/False Sets a value indicating whether to use contract dates instead of CheckIn and CheckOut dates
CurrentRoomSpaceID	Number Sets the current room space ID, used to exclude that room space from the list, in a room change or swap scenario

IncludeFloorIsSuite	True/False Sets a value indicating whether to include the Floor's IsSuite property
IncludeFloorIsOverrideGender	True/False Sets a value indicating whether to include the Floor's OverrideGender property
IncludeCoEdRooms	True/False Sets a value indicating whether to include CoEd rooms in the results
RoomService1Include	True/False Sets a value indicating whether to include room service 1
RoomService2Include	True/False Sets a value indicating whether to include room service 2
RoomService3Include	True/False Sets a value indicating whether to include room service 3
RoomService1Filter	True/False Sets a value to filter the room service 1
RoomService2Filter	True/False Sets a value to filter the room service 2
RoomService3Filter	True/False Sets a value to filter the room service 3
IncludeReportFields	True/False Sets a value indicating whether to include fields that are useful for reporting the results
MaxRecords	Number Sets the maximum records to return
PageStartIndex	Number Sets the start index of the page of data to return
PageSize	Number Sets the size of the page of data to return

UseWebDescriptions	True/False Setting this property to true will return the WebDescription instead of Description where the table has a WebDescription field. If the web description is blank the Description is returned.
EmptyRoomsOnly	True/False Setting this property to true will ensure only empty rooms are included in the search results, not any partially filled rooms.
ForcePaginationForAllRooms	True/False Sets whether to always use pagination for the All Rooms Query. This will fix the ordering issues but will only work on real db columns and will be slower.
LocationRelationship	Possible values: And, Or
SingleResultPerRoomTypeAndLocation	True/False
MergeRoomSpacesWithSameRoomBaseButMismatchingRoomRateIDs	True/False
AllowOverbookingOfRoomTypeAndLocation	True/False
GroupID	Number
IgnoreHistoryForAvailability	True/False
RoommateGroupFilterID	Number
DontUseRoomSummaryView	True/False

***TableName refers to the RecordAttachment.TableName value in the StarRez Database, so if the attachment is on the Entry then you would use Entry as the tableName.

Reporting

Reports on StarRez can be obtained from the web service in a variety of formats. For a full list of formats refer to Appendix F – Report Formats.

GetReport

Selects a StarRez report using either the report name or ID.

Request URL:

/getreport/{ReportName|ID}.{format}

XML:

<Parameters>

<ParameterName>**value**</ParameterName>

</Parameters>

GET:

/getreport/{reportname|ID}.{format}?[{field}={value}]

Example retrieving a Report by name

The following will retrieve a report with the name 'MyEntries' in Comma Separated Value (CSV) format. This report has one parameter called 'NameFirst'. Alternatively, the report ID could be sent instead of the report name with identical results.

URL:

/getreport/MyEntries.csv

XML:

<Parameters>

<NameFirst _operator=**"in"**>**John**</NameFirst>

</Parameters>

GET:

/getreport/MyEntries.csv?NameFirst=John

Accounting

This web service exposes key StarRez accounting functions to clients. This function requires no additional data.

GetBalance

Gets the current balance of an Entry. A Charge Group can optionally be specified.

Request

GET:

/accounts/getbalance/{entryID}[/{chargegroupid}]

Example balance request

The following will obtain the balance for the Entry with an ID of '31'.

GET:

/accounts/getbalance/31

Example balance request for a specific Charge Group

The following will obtain the balance of all Charge Groups with and ID of '12' for the Entry with an ID of '31'.

GET:

/accounts/getbalance/31/12

CreateTransaction

Creates a transaction for an Entry.

The accounts web service will automatically perform tax calculations as required based on the charge item settings that have been defined, so only one Amount needs to be set, even though multiple amounts may be returned as a result. This is similar to creating a transaction through StarRez, where only one amount needs to be supplied.

To ensure that duplicate transactions are not created, clients can optionally specify the ExternalID field, as a unique identifier for a particular financial transaction that a record should be created for. When this field is used, the API will check to see if there are any existing transactions in the database with a matching ExternalID, and if so, will return a HTTP 409 Conflict response, to indicate that duplicate data is being submitted. No data will be saved to the database in this case.

Request:

URL:

/accounts/createtransaction/{entryID}

XML:

<Transaction>

<Amount>**value**</Amount>

<ChargeItemID>**value**</ChargeItemID>

<field>**value**</field>

</Transaction>

GET:

/accounts/createtransaction/{entryid}?{field}={value}..

Example creating a transaction

The following will create a transaction for an Entry with the ID of '31'. This transaction will have a description of 'Deposit', an amount of four dollars and charge item ID of '1'.

URL:

/accounts/createtransaction/31

XML:

<Transaction>

<Amount>**4.00**</Amount>

<ChargeItem>**1**</ChargeItem>

<Description>**Deposit**</Description> </Transaction>

GET:

/accounts/createtransaction/{entryid}?Description=Deposit&Amount=4&ChargeIt emID=1

Create Payment

Creates a payment for an Entry.

The accounts web service will automatically perform tax calculations as required based on the charge item settings that have been defined, so only one Amount needs to be set, even though multiple amounts may be returned as a result. This is similar to creating a payment through StarRez, where only one amount needs to be supplied.

To ensure that duplicate transactions are not created, clients can optionally specify the TransactionExternalID field, as a unique identifier for a particular financial transaction that a record should be created for. When this field is used, the API will check to see if there are any existing transactions in the database with a matching ExternalID, and if so, will return a HTTP 409 Conflict response, to indicate that duplicate data is being submitted. No data will be saved to the database in this case.

Request:

URL:

/accounts/createpayment/{entryID}}

XML:

```
<Payment>

<Description>value</Description>

<Amount>value</Amount>

<field>value</field>

<BreakUp ChargeGroupID=”value”>

<Amount>value</Amount>

<field>value</field>

</BreakUp>

</Payment>
```

Example creating a payment

The following will create a payment for an Entry with the ID of ‘31’. This payment will be for the amount of ten dollars.

URL:

/accounts/createpayment/31

XML:

```
<Payment>

<TransactionTypeEnum>Payment</ TransactionTypeEnum>

<Description>Fee paid</Description>

<Amount>10.00</Amount>

<BreakUp ChargeGroupID="8">

<Amount>10.00</Amount>

</BreakUp>

</Payment>
```

DatabaseInfo

TableList

Lists tables that can be used by the web services.

URL:

/databaseinfo/tablelist.{format}

Parameter	Optional	Default Value	Description
format	Yes	atom	

ColumnList

Lists the columns that are present in the specified table.

URL:

/databaseinfo/columnlist/{tablename}.{format}

Parameter	Optional	Default Value	Description
tableName	No		The table to list columns for
format	Yes	atom	

RelatedTableList

Lists the tables that are related to the specified table.

URL:

/databaseinfo/relatedtablelist/{tablename}.{format}

Parameter	Optional	Default Value	Description
tableName	No		The table to list related tables for

format	Yes	atom	

FunctionList

Lists the functions that are available via web services on the specified table.

URL:

/databaseinfo/functionlist/{tablename}.{format}

Parameter	Optional	Default Value	Description
tableName	No		The table to list related tables for
format	Yes	atom	

StudentAuth

GenerateMagicLink

Allows the user to generate a PortalX Magic Link that can be used to direct a student to a PortalX page and log them in directly.

URL: http://localhost/StarRezRest/services/studentauth/generatemagiclink/{entryID}

Parameter	Optional	Default Value	Description
entryID	No		
returnURL	No		

ValidateCredentials

Allows the user to validate a student's credentials in the secure password system.

Note: We do not recommend the use of the ValidateCredentials REST Endpoint as this may be deprecated in the future. These endpoints do not accept GET requests. Use a POST request to the "validate credentials" endpoint and send the credentials in the body.

URL: http://localhost/StarRezRest/services/studentauth/validatecredentials URL: http://localhost/StarRezRest/services/studentauth/{username}/{password}

Note: these endpoints do not accept GET requests. Use a POST request to the "validatecredentials" endpoint and send the credentials in the body.

Parameter	Optional	Default Value	Description
username	No		The username to validate
password	No		The password to validate

Managing Errors

Inserting or updating data can produce errors that can be fixed. It is important to understand that not all errors can be ignored. By default the web service will automatically ignore all ignorable errors and fix all fixable errors.

If it is necessary to ensure that all errors are either ignored or fixed then this can be achieved by implementing the '_error' field. This field can contain child elements which specify how errors should be handled. For a full list of elements that can be specified refer to Appendix D – Error Elements.

Example forcing of errors

The following will turn off the default ignoring and fixing of errors for the creation of an Entry Note.

URL:

/create/entrynote

XML:

<EntryNote>

<EntryID>**31**</EntryID>

<_error>

<_autoIgnore>**false**</_autoIgnore>

<_autoFix>**false**</_autoFix>

</_error>

</EntryNote>

GET:

/create/entrynote?entryID=31&_error._autoignore=false&_error._autofix=false

Security

Security is implemented through the standard StarRez security model with each level of functionality being securable - the web services as a whole, the REST service itself, and each individual function of the service. Security is setup per group, as with the rest of StarRez.

Security credentials are passed in the header of the HTTP request, and need to be supplied to each request. There are two header fields, 'StarRezUsername' and 'StarRezPassword', which need to be supplied with appropriate credentials. The particular request will then "run as" that user. Security credentials and access rights are cached within the service for one hour, so that regular use of the service does not affect performance.

IIS may have also been configured to handle security for incoming requests with Basic Authentication or Integrated Authentication. If you are unsure what is setup contact StarRez support.

REST API Authentication Changes for v10.2 and Greater

As part our ongoing commitment to security and improving the features we offer, we are making an important change to how authentication is handled in our Web Services (REST API). Previously HTTP Basic, or custom StarRezUsername/StarRezPassword headers could be used to send through credentials. Those credentials could either be a username and API Token, or a username and password.

Beginning with v10.2 and moving forward authentication will use HTTP Basic authentication and send through a username and API Token. Any other method used will simply be treated as an anonymous request. Some of the benefits of using an API Token:

- More secure method of authentication
- API Tokens automatically expire
- No need to hardcodepasswords
- No transmission of passwords

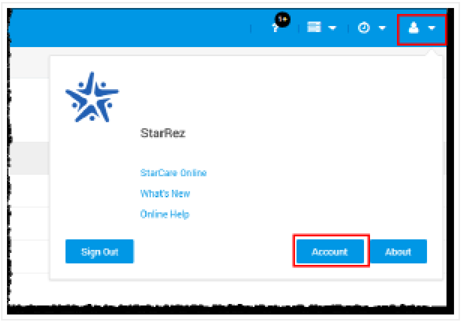
REST API Authentication Changes for v10.2 and Greater

Create a REST API Token

Token based authentication is now StarRez's preferred method of authenticating against REST web services. Tokens are a modern approach to authentication, and have a number of benefits over sending username/passwords for each REST call including improved security, ongoing reliability, and performance. Tokens are also useful for customers who would like to make REST calls via a browser without having to be prompted for credentials each time. Follow the steps below to generate an API key token through the StarRez Web interface.

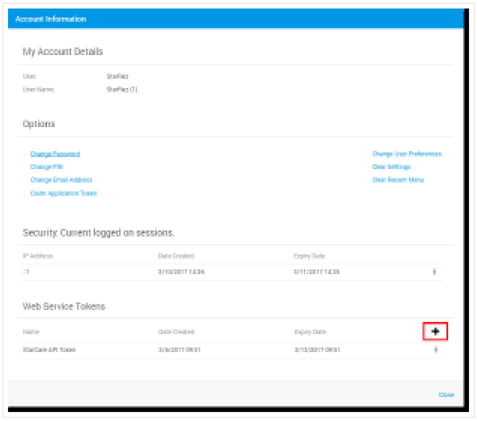
- Information on creating a REST API Token in StarRez Web is mirrored [in this article as well](#).

Step 1: Go to **Account** in StarRez Web.



Step 2: At the bottom of the Account Information screen in the Web Service Tokens section, click on the

plus sign to add a New Token.


Account Information

My Account Details

User: StarRez

User Name: StarRez (1)

Options

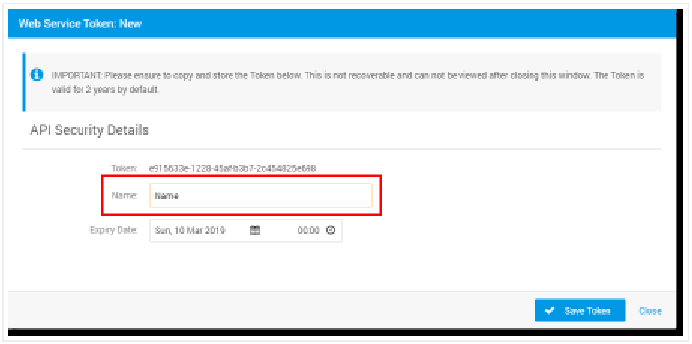
Security: Current logged on sessions.

IP Address	Date Created	Expiry Date
::1	3/10/2017 14:36	3/11/2017 14:36

Web Service Tokens

Name	Date Created	Expiry Date
StarCare API Token	3/6/2017 09:31	3/13/2017 09:31

Step 3: Enter a **Name**/description for the Token and set the **Expiry Date.** Make sure you copy the token from this screen. It cannot be viewed after you close this window and is not recoverable.


Web Service Token: New

IMPORTANT: Please ensure to copy and store the Token below. This is not recoverable and can not be viewed after closing this window. The Token is valid for 2 years by default.

API Security Details

Token: e915633e-1228-45afb3b7-2c454825e698

Name: Name

Expiry Date: Sun, 10 Mar 2019 00:00

Step 4: Test your Token (Optional)

Using your REST URL, you can perform a simple test of your token. Please note, REST may not be available via a browser at your institution.

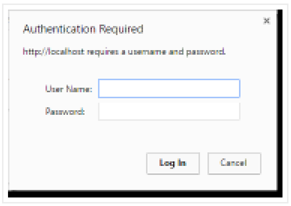
Execute a simple REST call. The formatting of the first part of the URL will vary depending on the DNS name of your web server and the name of the REST virtual directory. It will always end with the following: services/select/Entry/1 (You can use any web service call to test) Here is the formatting outlined in greater detail:

{yourpublicfacingurl.edu}/{nameofrestdirectory}/services/select/entry/1

Enter your credentials:

User Name = *your actual username*

Password = *Token*


Authentication Required

http://localhost requires a username and password.

User Name:

Password:

The credentials should now be saved for the created session.

Note: Managing API tokens is available in the Windows Desktop Client Security application beginning with v10.2.

Appendices

Appendix A - HTTP Status Codes

HTTP Response Code	Description
200	200 OK: everything went fine.
400	400 Bad Request: your request is invalid, or badly formed, and we'll return an error message that tells you why.
403	403 Forbidden: your request is valid, but you do not have permission to select data from the specified table, or update the specified field
404	404 Not Found: your request is valid, but no data was found, or the table you are trying to use does not exist.

Appendix B - Special Fields

Field	Description

_top	Restrict records to the top n results. If not specified, 100 will be used.
	
	If you experience a timeout with a /Query REST call you can use the TOP and START keywords to specify how many records to pull for a page of data.
	
	For example, to select entries from the database in 10 record batches you could issue the following queries:
	SELECT TOP 10 EntryID FROM Entry
	SELECT TOP 10 START AT 11 EntryID FROM Entry
	SELECT TOP 10 START AT 21 EntryID FROM Entry SELECT TOP 10 START AT 31 EntryID FROM Entry

_pageIndex **_pageSize**	Format the data into a series of pages before returning the records. The _pageIndex field specifies the 0-based record index that should be at the top of the page, and the _pageSize specifies the number of records in the page. Cannot be used in conjunction with _top. Paging will take precedence. The _pageIndex and _pageSize error elements are only available to the SELECT method in REST API. These error elements are not supported by the /Query or /GetReport methods.
_orderby	Specify the ordering of the fields. Syntax is _orderby=fieldname1,fieldname2. Using fieldname.desc will order by that field descending
_fields	Specify which fields should be returned. If not specified, all fields will be returned. Certain important fields will always been returned even if not specified in _fields. This is by design, and its behaviour cannot be changed. We recommend not relying on these automatic fields, in case they change in the future.
_relatedtables	Specify which related tables of the main record should be expanded and provided in the result. Specifying one or more tables here can save multiple selects to get data

_includeLookupCaptions	
_loadAll	
_relationship	Specify the relationship between criteria. If not specified 'and' will be used. Supported values are 'and' and 'or'.
_loadDeletedAndHiddenRecords **_deletedAndHidden**	By default, records that are hidden or deleted will not be returned in API calls. This can be overridden by including these records in the URL or in the body of the post. For example, the URL below would select Charge Group ID 34, even if the record was hidden. select/chargegroup/34?_loadDeletedAndHiddenRecords=true These can also be used in the XML of the call, similar to _autoIgnore. <_deletedAndHidden>**true**</_deletedAndHidden>

Appendix C - Special Attributes

Attribute	Abbreviation	Description
NotEquals	ne	Specifies the field should not equal the value
GreaterThan	gt	Specifies the field should be greater than the value
LessThan	lt	Specifies the field should be less than the value
GreaterThanOrEqualTo	gte	Specifies the field should be greater than or equal to the value
LessThanOrEqualTo	lte	Specifies the field should be less than or equal to the value
Contains	c	Specifies the field should contain the value
NotContains	nc	Specifies the field should not contain the value
StartsWith	sw	Specifies the field should start with the value
NotStartsWith	nsw	Specifies the field should not start with the value
EndsWith	ew	Specifies the field should end with the value
NotEndsWith	new	Specifies the field should not end with the value

In	in	Specifies the field should equal any of the values **NOTE:** Currently only supports comma separated numbers (eg. 1,2,3,4)
NotIn	notin	Specifies the field should not equal any of the values **NOTE:** Currently only supports comma separated numbers (eg. 1,2,3,4)

Appendix D - Error Elements

Child Element	Optional (Y/N)	Possible Values	Description
_autoignore	Y	true | false	Specifies if the web service should automatically resolve ignorable errors for the incoming request.
_autofix	Y	true | false	Specifies if the web service should automatically resolve fixable errors for the incoming request.
ignore	Y	String	Ignores a specific error. This element is to contain the rule name.
fix	Y	String	Fixes a specific error. This element is to contain the rule name.
dontignore	Y	String	Do not ignores a specific error. This element is to contain the rule name.

dontfix	Y	String	Do not fix a specific error. This element is to contain the rule name.

Appendix E - Custom Field Types

Type	Web Service Field Name
String	valuestring
Boolean	valueboolean
Date	valuedate
Integer	valueinteger
Money	valuemoney

Appendix F - Report Formats

Format	Format Specifier
XML	.xml
CSV	.csv
HTM	.htm
HTML	.html
HTML-XML	.html-xml
ATOM (default)	.atom or nothing

JSON	.json

Appendix G - Meal Plan Examples

Create a new EntryMeal record

The following example creates a new EntryMeal record, passing in the EntryID, MealPlanID, TermSessionID and start and end dates. These would not be hard-coded in production code, and it is assumed they would have been previously looked up.

The EntryMealID of the record that is created is returned in the content of an entry in the ATOM feed. The id field of that entry contains the URL that can be used to retrieve the full record, by a simple GET request.

There are more fields in the EntryMeal table than are specified here. Please see the database documentation for more information.

// The url of the service

string url = "http://testserver/services/create/EntryMeal";

// assemble the criteria to create an EntryMeal record string data = @"<EntryMeal>

<EntryID>" + entryID + @"</EntryID>

<MealPlanID>1</MealPlanID>

<TermSessionID>1</TermSessionID>

<DateStart>1 Jan 2010</DateStart>

<DateEnd>23 Apr 2010</DateEnd>

</EntryMeal>"; string result;

// Perform the service call

var status = PerformRequest(url, "POST", data, **out** result);

// If status is 403, then the request was wrong. If status is 404, no records were found

if (status == HttpStatusCode.OK)

{

// Use the System.Xml.XmlDocument class to parse and query the XML returned XmlDocument doc = **new** XmlDocument();

doc.LoadXml(result);

// The XML returned is an ATOM feed, and is namespaced accordingly, so we need to setup a NamespaceManager

XmlNamespaceManager mgr = **new** XmlNamespaceManager(doc.NameTable); mgr.AddNamespace("atom", "http://www.w3.org/2005/Atom");

// The EntryMealID is returned in the content of an atom entry, so we'll grab that in case we need it var contentNode = doc.SelectSingleNode("//atom:content", mgr); int entryMealID = Convert.ToInt32(contentNode.InnerText);

// The to select out the record that was created, you can use the URL in the id field of the atom entry

// or http://testserver/services/select/EntryMeal/<entryMealID>

}

Update an EntryMeal Record

The following example updates an EntryMeal record, cancelling it. To cancel a meal plan, we set the Cancelled flag, we set the DateEnd to today, and we set some comments so people know what is going on. In this case, we don't care about the returned data.

We specify the record to update in the URL of the call, and the data to update in the posted XML fragment.

// The url of the service

string url = "http://testserver/services/update/EntryMeal/" + entryMealID;

// assemble the criteria to update a meal plan, cancelling it and updating the comments to reflect that string data = @"<EntryMeal>

<Cancelled>true</Cancelled>

<DateEnd>" + DateTime.Today.ToString("d MMM yyyy") + @"

<Comments>This meal plan has been cancelled</Comments>

</EntryMeal>"; string result;

// Perform the service call
var status = PerformRequest(url, "POST", data, out result);

// If status is 403, then the request was wrong. If status is 404, no records were found ***if***

(status == HttpStatusCode.OK)

{

  // Data is returned from this request as normal, but in this case we don't care about it

}

Appendix H - Accounting Examples

Get the current balance for an Entry

// The url of the service

string url = "http://testserver/services/accounts/GetBalance/" + entryID; string result;

// Perform the service call

var status = PerformRequest(url, "GET", "", **out** result);

// If status is 403, then the request was wrong. If status is 404, no records were found **if (**status **==** HttpStatusCode**.**OK**)**

{

// Use the System.Xml.XmlDocument class to parse and query the XML returned XmlDocument doc **= new** XmlDocument**();**

doc**.**LoadXml**(**result**);**

// The XML returned is an ATOM feed, and is namespaced accordingly, so we need to setup a NamespaceManager

XmlNamespaceManager mgr **= new** XmlNamespaceManager**(**doc**.**NameTable**);** mgr**.**AddNamespace**(**"atom"**,** "http://www.w3.org/2005/Atom"**);**

// Use XPath to find the EntryID element. Notice we prefix with the namespace we setup above var balanceNode **=** doc**.**SelectSingleNode**(**"//atom:Balance/atom:TotalAmount"**,** mgr**);**

// Convert the value to a decimal

decimal balance **=** Convert**.**ToDecimal**(**balanceNode**.**InnerText**);**

// Do something with balance

}

Create a Payment

The following example creates a $10 payment **for** a user**,** broken up over two charge groups**.** An example usage might be to import a payment from a live payment system**,** or perhaps automatically apply a payment **if** the balance **(**from the previous example**)** reaches a certain threshold**.**

// The url of the service

string url **=** "http://testserver/services/accounts/CreatePayment/" **+** entryID**;**

// assemble the criteria to create a Payment record string data **=** @"<Payment>

<TransactionTypeEnum>Payment</TransactionTypeEnum>

<Description>Payment imported from merchant</Description>

<Amount>10</Amount>

<BreakUp ChargeGroupID="1">

<TransactionComments**>**This **is** a transaction comment**,** on a $6 transaction **for** charge group 1**</**TransactionComments**>**

<Amount**>**6**</**Amount**>**

<TransactionTag**>**FF**</**TransactionTag**>**

</BreakUp**>**

<BreakUp ChargeGroup**=**"Data Account"**>**

<Amount**>**4**</**Amount**>

<TransactionComments>This **is** another transaction comment**,** on a $4 transaction **for** the Data Account charge group**</**TransactionComments**>**

</BreakUp**>**

</Payment**>**"; string result**;**

// Perform the service call

var status **=** PerformRequest**(**url**,** "POST"**,** data**, out** result**);**

// If status is 403, then the request was wrong. If status is 404, no records were found **if (**status **==** HttpStatusCode**.**OK**)**

{

// Use the System.Xml.XmlDocument class to parse and query the XML returned XmlDocument doc **= new** XmlDocument**();**

doc**.**LoadXml**(**result**);**

// The XML returned is an ATOM feed, and is namespaced accordingly, so we need to setup a NamespaceManager

XmlNamespaceManager mgr **= new** XmlNamespaceManager**(**doc**.**NameTable**);** mgr**.**AddNamespace**(**"atom"**,** "http://www.w3.org/2005/Atom"**);**

// The full Payment record is returned in the content of an atom entry, so we'll grab the PaymentID in case we need it

var paymentIDNode **=** doc**.**SelectSingleNode**(**"//atom:Payment/atom:PaymentID"**,** mgr**);** int paymentID **=** Convert**.**ToInt32**(**paymentIDNode**.**InnerText**);**

}

Appendix I - Demographic Data Example

When using REST for your Demographic Import, we suggest including the following fields at minimum:

1. Entry.NameFirst
2. Entry.NameLast
3. Entry.ID1
4. Entry.DOB
5. Entry.Birth_GenderEnum
6. Entry.GenderEnum (On record insert only)
7. Entry.EntryStatus* (On record insert only)

*EntryStatus is suggested because the default status for this is Reserved and best practice is to have all new records with an Entry Status of Application.

The following example will create an Entry record on the database.

// assemble the criteria to create the Entry string data **=** @"<Entry>

<NameLast>Peterson</NameLast>

<NameFirst>John</NameFirst>

</Entry>"; string result;

// Perform the service call

var status = PerformRequest(url, "POST", data, out result);

// If status is 403, then the request was wrong. If status is 404, no records were found

if (status == HttpStatusCode.OK)

{

  // Data is returned from this request as normal

}

// The url of the service

string url = "http://testserver/services/create/entry";

Appendix J - Check In / Check Out Examples

The following two examples display how to check-in and check-out an Entry with an EntryID value of '31' using a request to the web service.

Check In:

Using GET:

/function/entry/31/CheckInOut?operation=CheckIn

Using POST (C#):

// The url of the service
string url = "http://testserver/services/function/entry/31/CheckInOut";

// assemble the criteria to check-in the Entry	
string data =	@"<Parameters>

<operation>CheckIn</operation>

</Parameters>"; string result;

// Perform the service call
var status **=** PerformRequest**(**url**,** "POST"**,** data**,** out result**);**

// If status is 403, then the request was wrong. If status is 404, no records were found

if **(**status **==** HttpStatusCode**.**OK**)**

{

  // Data is returned from this request as normal

}

Check Out:

Using GET:

/function/entry/31/CheckInOut?operation=CheckOut

Using POST (C#):

// The url of the service
string url **=** "http://testserver/services/function/entry/31/CheckInOut”**;**

// assemble the criteria to check-out the Entry	
string data **=**	@"<Parameters>

<operation>CheckOut</operation>

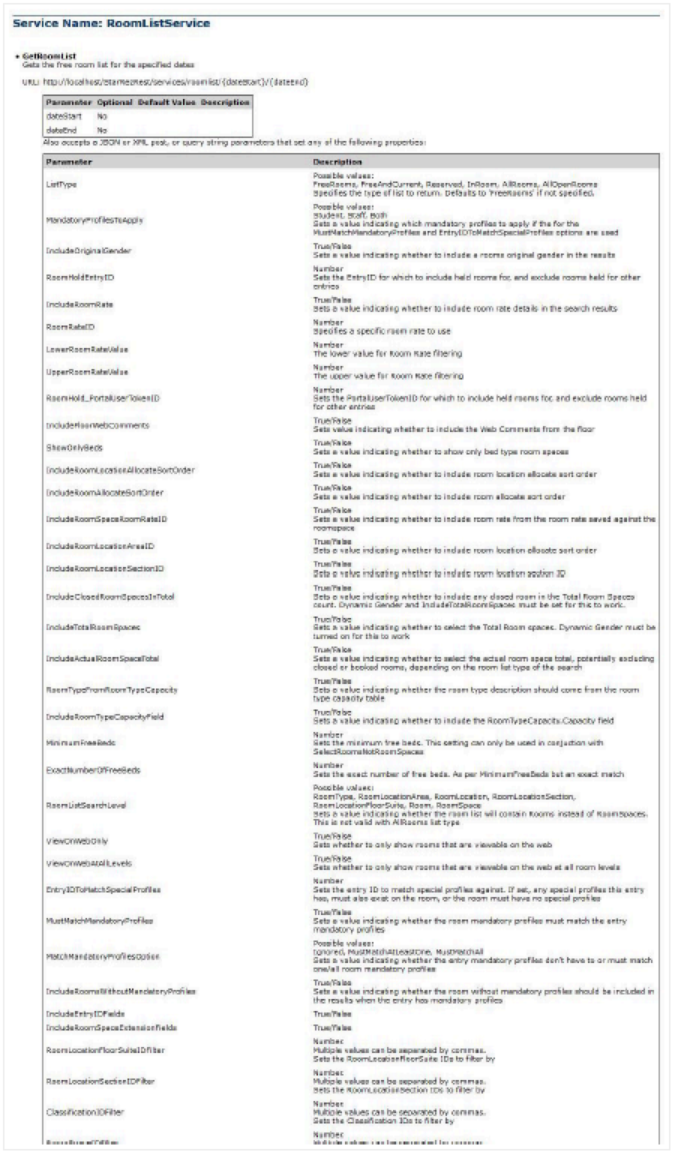
</Parameters>"**;** string result**;**

// Perform the service call
var status **=** PerformRequest**(**url**,** "POST"**,** data**,** out result**);**

// If status is 403, then the request was wrong. If status is 404, no records were found ***if***

(status **==** HttpStatusCode**.**OK**)**

```
{
  // Data is returned from this request as normal
```



Service Name: RoomListService

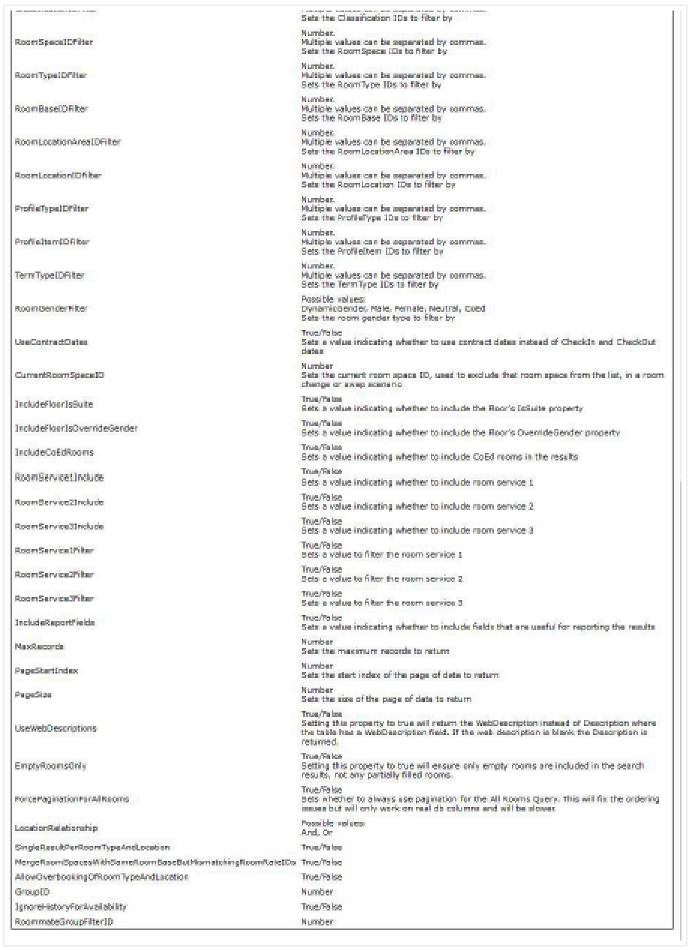
- **GetRoomList**
  Gets the free room list for the specified dates

URL: http://localhost/starrezrest/services/roomlist/{dateStart}/{dateEnd}

Parameter	Optional	Default Value	Description
dateStart	No		
dateEnd	No		

Also accepts a JSON or XML post, or query string parameters that set any of the following properties:

Parameter	Description
ListType	Possible values: FreeRooms, FreeAndCurrent, Reserved, InRoom, AllRooms, AllOpenRooms Specifies the type of list to return. Defaults to 'FreeRooms' if not specified.
MandatoryProfilesToApply	Possible values: Student, Staff, Both Sets a value indicating which mandatory profiles to apply if the for the MustMatchMandatoryProfiles and EntryIDToMatchSpecialProfiles options are used
IncludeOriginalGender	True/False Sets a value indicating whether to include a rooms original gender in the results
RoomHoldEntryID	Number Sets the EntryID for which to include held rooms for, and exclude rooms held for other entries
IncludeRoomRate	True/False Sets a value indicating whether to include room rate details in the search results
RoomRateID	Number Specifies a specific room rate to use
LowerRoomRateValue	Number The lower value for Room Rate filtering
UpperRoomRateValue	Number The upper value for Room Rate filtering
RoomHold_PortalUserTokenID	Number Sets the PortalUserTokenID for which to include held rooms for, and exclude rooms held for other entries
IncludeFloorWebComments	True/False Sets value indicating whether to include the Web Comments from the floor
ShowOnlyBeds	True/False Sets a value indicating whether to show only bed type room spaces
IncludeRoomLocationAllocateSortOrder	True/False Sets a value indicating whether to include room location allocate sort order
IncludeRoomAllocateSortOrder	True/False Sets a value indicating whether to include room allocate sort order
IncludeRoomSpaceRoomRateID	True/False Sets a value indicating whether to include room rate from the room rate saved against the roomspace
IncludeRoomLocationAreaID	True/False Sets a value indicating whether to include room location allocate sort order
IncludeRoomLocationSectionID	True/False Sets a value indicating whether to include room location section ID
IncludeClosedRoomSpacesInTotal	True/False Sets a value indicating whether to include any closed room in the Total Room Spaces count. Dynamic Gender and IncludeTotalRoomSpaces must be set for this to work.
IncludeTotalRoomSpaces	True/False Sets a value indicating whether to select the Total Room spaces. Dynamic Gender must be turned on for this to work
IncludeActualRoomSpaceTotal	True/False Sets a value indicating whether to select the actual room space total, potentially excluding closed or booked rooms, depending on the room list type of the search
RoomTypeFromRoomTypeCapacity	True/False Sets a value indicating whether the room type description should come from the room type capacity table
IncludeRoomTypeCapacityField	True/False Sets a value indicating whether to include the RoomTypeCapacity.Capacity field
MinimumFreeBeds	Number Sets the minimum free beds. This setting can only be used in conjuction with SelectRoomsNotRoomSpaces
ExactNumberOfFreeBeds	Number Sets the exact number of free beds. As per MinimumFreeBeds but an exact match
RoomListSearchLevel	Possible values: RoomType, RoomLocationArea, RoomLocation, RoomLocationSection, RoomLocationFloorSuite, Room, RoomSpace Sets a value indicating whether the room list will contain Rooms instead of RoomSpaces. This is not valid with AllRooms list type
ViewOnWebOnly	True/False Sets whether to only show rooms that are viewable on the web
ViewOnWebAtAllLevels	True/False Sets whether to only show rooms that are viewable on the web at all room levels
EntryIDToMatchSpecialProfiles	Number Sets the entry ID to match special profiles against. If set, any special profiles this entry has, must also exist on the room, or the room must have no special profiles
MustMatchMandatoryProfiles	True/False Sets a value indicating whether the room mandatory profiles must match the entry mandatory profiles
MatchMandatoryProfilesOption	Possible values: Ignored, MustMatchAtLeastOne, MustMatchAll Sets a value indicating whether the entry mandatory profiles don't have to or must match one/all room mandatory profiles
IncludeRoomsWithoutMandatoryProfiles	True/False Sets a value indicating whether the room without mandatory profiles should be included in the results when the entry has mandatory profiles
IncludeEntryIDFields	True/False
IncludeRoomSpaceExtensionFields	True/False
RoomLocationFloorSuiteIDFilter	Number Multiple values can be separated by commas. Sets the RoomLocationFloorSuite IDs to filter by
RoomLocationSectionIDFilter	Number Multiple values can be separated by commas. Sets the RoomLocationSection IDs to filter by
ClassificationIDFilter	Number Multiple values can be separated by commas. Sets the Classification IDs to filter by
[illegible]	Number [illegible]



[illegible]	[illegible] Sets the Classification IDs to filter by
RoomSpaceIDFilter	Number. Multiple values can be separated by commas. Sets the RoomSpace IDs to filter by
RoomTypeIDFilter	Number. Multiple values can be separated by commas. Sets the RoomType IDs to filter by
RoomBaseIDFilter	Number. Multiple values can be separated by commas. Sets the RoomBase IDs to filter by
RoomLocationAreaIDFilter	Number. Multiple values can be separated by commas. Sets the RoomLocationArea IDs to filter by
RoomLocationIDFilter	Number. Multiple values can be separated by commas. Sets the RoomLocation IDs to filter by
ProfileTypeIDFilter	Number. Multiple values can be separated by commas. Sets the ProfileType IDs to filter by
ProfileItemIDFilter	Number. Multiple values can be separated by commas. Sets the ProfileItem IDs to filter by
TermTypeIDFilter	Number. Multiple values can be separated by commas. Sets the TermType IDs to filter by
RoomGenderFilter	Possible values: DynamicGender, Male, Female, Neutral, CoEd Sets the room gender type to filter by
UseContractDates	True/False Sets a value indicating whether to use contract dates instead of CheckIn and CheckOut dates
CurrentRoomSpaceID	Number Sets the current room space ID, used to exclude that room space from the list, in a room change or swap scenario
IncludeFloorIsSuite	True/False Sets a value indicating whether to include the Floor's IsSuite property
IncludeFloorIsOverrideGender	True/False Sets a value indicating whether to include the Floor's OverrideGender property
IncludeCoEdRooms	True/False Sets a value indicating whether to include CoEd rooms in the results
RoomService1Include	True/False Sets a value indicating whether to include room service 1
RoomService2Include	True/False Sets a value indicating whether to include room service 2
RoomService3Include	True/False Sets a value indicating whether to include room service 3
RoomService1Filter	True/False Sets a value to filter the room service 1
RoomService2Filter	True/False Sets a value to filter the room service 2
RoomService3Filter	True/False Sets a value to filter the room service 3
IncludeReportFields	True/False Sets a value indicating whether to include fields that are useful for reporting the results
MaxRecords	Number Sets the maximum records to return
PageStartIndex	Number Sets the start index of the page of data to return
PageSize	Number Sets the size of the page of data to return
UseWebDescriptions	True/False Setting this property to true will return the WebDescription instead of Description where the table has a WebDescription field. If the web description is blank the Description is returned.
EmptyRoomsOnly	True/False Setting this property to true will ensure only empty rooms are included in the search results, not any partially filled rooms.
ForcePaginationForAllRooms	True/False Sets whether to always use pagination for the All Rooms Query. This will fix the ordering issues but will only work on real db columns and will be slower
LocationRelationship	Possible values: And, Or
SingleResultPerRoomTypeAndLocation	True/False
MergeRoomSpacesWithSameRoomBaseButMismatchingRoomRateIDs	True/False
AllowOverbookingOfRoomTypeAndLocation	True/False
GroupID	Number
IgnoreHistoryForAvailability	True/False
RoommateGroupFilterID	Number