Go To Online Help

RESTful API

The Rent Manager 12 Web API(WAPI12) is designed to follow RESTful principles.

Short glossary of RESTful terms

Resource	An access point for WAPI12. Can represent data interaction or a way to execute work.
Entity	The objects that resources allow access to. "/Tenants/" is a resource that allows access to "Tenant" entities.
URL	Defines where the resource is located.
Http Method or Http Verb	Indicates the desired method of interaction with a resource.

HTTP Methods

GET	Retrieves data from the specified resource.
POST	Sends data to the specified resource. Depending on the resource, this data can be saved or used to perform work.
DELETE	Deletes data at the specified resource.

Resources

WAPI12 name resources for the entity that they access, e.g. GET /Tenants retrieves the collection of tenants. The resources are grouped into three categories:

Collection Resource

When making a request to a Collection Resource you are interacting with a collection of entities.

GET Requests

```
GET https://sampleco.api.rentmanager.com/Tenants
```

Return a collection of all entities of a resource. May use features on the query string including: fields, embed, Filters, pagenumber, pagesize, nocontent and orderby to control results returned. All GET requests on a collection resource must also return a X-Total-Results header. Pagination may be enabled automatically by the system if the result set contains more than 1000 records.

POST Requests

```
POST https://sampleco.api.rentmanager.com/Tenants
```

Creates or Updates entities of the specified resource. All POST requests on a collection resource should include a List of DataModel objects of the correct type for the entities being modified. The response will contain the updated entities.

DELETE Requests

```
DELETE https://sampleco.api.rentmanager.com/Tenants
```

Deletes entities of the specified resource. All DELETE requests on a collection resource must include a list of

Integers which are the Id's of the entities being deleted.

Instance Resource

When making a request to an instance resource you are interacting with a single entity of that resource.

GET Requests

```
GET https://sampleco.api.rentmanager.com/Tenants/{id}
```

Return a specific entity from a resource. The URL must include the ID of the entity to retrieve. May use features on the query string including: fields, embed, and nocontent to control results returned.

POST Requests

```
POST https://sampleco.api.rentmanager.com/Tenants/{id}
```

Updates a specific entity of the specified resource. The URL must include the ID of the entity to modify. All POST requests on an instance resource should include a DataModel object of the correct type for the entity being modified. The response will contain the updated entity.

DELETE Requests

```
DELETE https://sampleco.api.rentmanager.com/Tenants/{id}
```

Deletes a specific entity of the specified resource. The URL must include the ID of the entity to delete.

Action Resource

When making a request to an action resource you are performing work related to that resource. Action resources do not concern themselves with specific entities, but with work related to those entities. Action resources only allow the POST method.

POST Requests

```
POST https://sampleco.api.rentmanager.com/RecurringCharges/PostRecurringCharges
```

Performs a unit of work. The POST body may include a DataModel that is used to provide some information for the completion of the work. These requests should return an HTTP Status code describing the result of the work.

Experimental Resources

Some resources may be denoted as experimental. These resources are not guaranteed to remain in the WAPI and may be altered at anytime. As the name implies, these are endpoints that are likely to undergo several iterations before the experimental classification is removed.

Request Structure

A request consists of three major parts: the url, the header, and the body. The body is only included for POST requests.
WAPI12 only accepts requests over an SSL encrypted session.

HTTP Request Example

```
GET https://sampleco.api.rentmanager.com/Tenants
```

Request Part	Description
GET	HTTP Request Method
https	Secured, SSL Encrypted
sampleco	Company identifier
api	Product identifier
rentmanager.com	RentManager base URL
Tenants	Resource identifier

HTTP Header Example

```
GET /Tenants HTTP/1.1
Accept: application/json
Content-Type: application/json
Host: sampleco.api.rentmanager.com
X-RM12Api-ApiToken: 2e995f2073a64bb1ac3e649a746e62fb
```

Header Part	
GET	Request Method. One of GET, POST or DELETE
/Tenants	Resource to interact with
HTTP/1.1	Protocol
Accept	Format the response should be provided in
Content-Type	Format the request body is provided in
Host	Host and domain information
X-RM12Api-ApiToken	The Authentication Token required for all API requests.

Data Models

GET Requests return Data Models, and POST requests may include a data model as part of the request body.

Data Model Example

Field	Data Type	Features
ApiUri	String	
Id	Int	
FirstName	String	
LastName	String	
ColorId	Int	
Color	Resource	Embed
DateCreated	DateTime	
CreateUserId	Int	
CreateUser	Resource	Embed
DateUpdated	DateTime	
UpdateUserId	Int	
UpdateUser	Resource	Embed
PrimaryContact	Resource	Embed, Atomic Create
Contacts	Resource	Embed

All Data Models include the "APIURI" field. This field specifies the Resource used to retrieve the data.

Data Model JSON Example

```
{
   "ApiUri": "https://sampleco.api.rentmanager.com/Tenants/1",
   "Id": 1,
   "FirstName": "Joe",
   "LastName": "Rocker",
   "ColorId": 1,
   "Color": {
      "ApiUri": "https://sampleco.api.rentmanager.com/Colors/1"
   },
   "DateCreated": "2014-01-01 14:32:25",
   "CreateUserId": 1,
   "CreateUser": {
      "ApiUri": "https://sampleco.api.rentmanager.com/Users/1"
   },
   "DateUpdated": "2014-01-01 14:32:25",
   "UpdateUserId": 1,
   "UpdateUser": {
      "ApiUri": "https://sampleco.api.rentmanager.com/Users/1"
   },
   "PrimaryContact": {
      "ApiUri": "https://sampleco.api.rentmanager.com/Tenants/1/PrimaryContact"
   },
   "Contacts": [
      {
      "ApiUri": "https://sampleco.api.rentmanager.com/Tenants/1/Contacts"
      }
   ]
}
```

Note that all Resource fields include at least the "APIURI" field. This follows the draft W3C recommendation for Linked Data (www.w3.org/TR/json-ld/) for the @id. @id is not used because it does not translate well to .net language data models.

Linked Data

Data models may include information about other resources as they relate to the current resource. This linked

data is included with a minimum of the "APIURI" field within the data model. Linked data may have its details embedded in-line by use of the embed query string parameter. With the exception of linked data marked as "atomic create", linked data fields are ignored for POST and DELETE requests.

ApiUri always contains a link that is the most direct way to retrieve the information linked.

Attributes

Fields within an object model may have attribute(s) associated with that field.

Primary Key

Is the unique identifier for that given object.

Required

This field is required to be passed in a POST request when creating a new record of a given resource.

Required (create)

The field may be passed in a POST request when creating a new record of a given resource. The linked data will then be created at the same time as the parent data.

Read Only

The field will not be updated when being passed in a POST request for the record of a given resource. The data may only be viewed and not modified.

Concurrency Key

Concurrency may be enabled or disabled during authentication. The default is enabled. When concurrency is enabled, request to update a record must include the UpdateDate field

Calculated Field

The field is not being read directly out of the database. Rather it is using multiple fields in the model to calculate its displayed value.

Requires Embed '{Embed Option}'

A field needs an embed to be processed with the request, in order to gather all of the required information for the field.

Query String Parameters

A request may include one or more query string parameters that affect how the request is processed and how the data is returned.

Optional Parameter	Use
fields	Return only the fields listed with the result.
embed	Include a related resource in-line with the result.
filter	Filter the result based on the criteria given. Uses a variant of the Resource Query Language (RQL) specification. (https://github.com/persvr/rql).
pagenumber	Return a page of results of a resource. Used in conjunction with pagesize
pagesize	Number of records to return in a page. Used in conjunction with pagenumber.
nocontent	Do not return any content. Headers will be returned normally. Useful for Create and Update requests.
orderby	Sort the results prior to returning.

Query String Example

```
GET https://sampleco.api.rentmanager.com/Tenants?
fields=Id,FirstName,LastName,Color&embeds=Color&filters=LastName,eq,Rockler&pagenumber=1&pagesize=10&orderby
```

Fields

The fields query string parameter instructs the API to include only the fields specified in the response rather than the entire data model.

Request

```
GET https://sampleco.api.rentmanager.com/Tenants?fields=Id,FirstName,LastName
```

Response

Here we see the result of the field parameter.

```
{
   "Id": 1,
   "FirstName": "Joe",
   "LastName": "Rockler"
}
```

Embed

The embed query string parameter instructs the API to include linked data in-line with the response rather than simply including the link identifier.

Request

The following request embeds Color within the response.

```
GET https://sampleco.api.rentmanager.com/Tenants?embed=Color
```

Response

Here we see the result of the embed parameter.

```
{
   "ApiUri": "https://sampleco.api.rentmanager.com/Tenants/1",
   "Id": 1,
   "FirstName": "Joe",
   "LastName": "Rocker",
   "ColorId": 1,
   "Color": {
      "ApiUri": "https://sampleco.api.rentmanager.com/Colors/1",
      "Id": 1,
      "Name": "Black",
      "IsDisplayColor": true,
      "HexValue": "000000"
   },
   "DateCreated": "2014-01-01 14:32:25",
   "CreateUserId": 1,
   "CreateUser": {
      "ApiUri": "https://sampleco.api.rentmanager.com/Users/1"
   },
   "DateUpdated": "2014-01-01 14:32:25",
   "UpdateUserId": 1,
   "UpdateUser": {
      "ApiUri": "https://sampleco.api.rentmanager.com/Users/1"
   },
   "PrimaryContact": {
      "ApiUri": "https://sampleco.api.rentmanager.com/Tenants/1/PrimaryContact"
   }
}
```


When used in conjunction with the "fields" query string parameter to control the returned data model fields any field that is to be embedded must also appear in the "fields" parameter or it will not be included in the response.

Filter

The filter query string parameter instructs the API to include only results that meet the criteria given in the response. The filter format implements a subset of the Resource Query Language (RQL) specification which can be found at https://github.com/persvr/rql. The filter query string parameter is in the format of field=operator=value. Multiple filters can be used together by placing a semicolon between each filter.

RQL Operators

Operation	RQL Operator	Descriptions	Example
Between	bt	Between two values **(Inclusive)**	/Tenants?filters=TenantID,bt,(2,10)
Contains	ct	Object contains value	/Tenants?filters=Name,ct,John
Ends With	ew	Ends with the value	/Tenants?filters=Name,ew,Hudson
Equal To	eq	=	/Tenants?filters=TenantID,eq,10
Greater Than	gt	>	/Tenants?filters=TenantDisplayID,gt,8
Greater Than Or Equal To	ge	>=	/Tenants?filters=TenantDisplayID,ge,8
Greater Than Or Equal To Or Null	gen	>= OR null	/ServiceManagerIssues?filters=AssignedOpenDate,gen,2018-06-06
Greater Than Or Null	gtn	> OR null	/ServiceManagerIssues?filters=AssignedOpenDate,gtn,2018-06-06
Has Value	hv	Object has a value	/ServiceManagerIssues?filters=AssignedOpenDate,hv,2018-06-06
In	in	Has any of a list of values	/Tenants?filters=TenantID,in,(4,5,6)
Less Than	lt	<	/Tenants?filters=AccountGroupID,lt,10;
Less Than Or Equal To	le	<=	/Tenants?filters=AccountGroupID,le,10;
Less Than Or Equal To Or Null	len	<= OR null	/ServiceManagerIssues?filters=AssignedOpenDate,len,2018-06-06
Less Than Or Null	ltn	< OR null	/ServiceManagerIssues?filters=AssignedOpenDate,ltn,2018-06-06
Not Equal To	ne	<>	/Tenants?filters=TenantID,ne,10
Not In	ni	Does not have any of a list of values	/Tenants?filters=TenantID,ni,(4,5,6)
Starts With	sw	Starts with the value	/Tenants?filters=Name,sw,J

Request

```
GET https://sampleco.api.rentmanager.com/Tenants?filter=FirstName,eq,Joe;DateCreated,ge,2014-01-01
```

Response

The response would only contain records from the Tenants resource that had "Joe" for the FirstName field and which was created on or after 2014-01-01.

Deprecated Filters

Some filters may be denoted as deprecated. After the deprecation date, they can no longer be guaranteed to function properly and will no longer be supported. This typically means the filter has been renamed or replaced.

SaveOptions

The SaveOption query string parameter passes instructions to the API for how certain models should be saved.

Request

The following request posts a Charge to a tenant and ignores HardCloseDate.

```
Post https://sampleco.api.rentmanager.com/Tenants/{id}/Charges?saveOptions=IgnoreHardClose
```

Response

The response would contain the new charge model.

Pagination

Pagination functionality is provided through the use of the pagenumber and pagesize query string parameters. Any resource that returns more than 1000 records will automatically convert to using pagination. If pagenumber is specified and pagesize is greater than 1000 or is not provided pagesize will be set to 1000.

When using pagination the HTTP response headers will include additional information about the pagination.

Pagination Response Headers

Header Name	Purpose
Link	The Link header will contain links to one or more of First, Previous, Next and Last. It is not necessary to include the first, previous, next or last links if they are already on that page of results.
X-Total-Results	The X-Total-Results header will contain the total number of results that were found from the request.

Request

```
GET https://sampleco.api.rentmanager.com/Tenants?pagenumber=4&pagesize=25
```

Response Headers

```
HTTP/1.1 200 OK
Content-Type: application/json
Link: <https://sampleco.api.rentmanager.com/Tenants?pagenumber=1&pagesize=25>; rel="first",
<https://sampleco.api.rentmanager.com/Tenants?pagenumber=3&pagesize=25>; rel="previous",
<https://sampleco.api.rentmanager.com/Tenants?pagenumber=5&pagesize=25>; rel="next",
<https://sampleco.api.rentmanager.com/Tenants?pagenumber=12&pagesize=25>; rel="last"
X-Total-Results: 296
```

NoContent

The nocontent query string parameter is used when the response would normally include data, however you do not wish to receive any data. This can be useful when creating or updating records if you are not interested in the response data.

OrderBy **Partial Support**

The orderby query string parameter is used when you wish to sort the resulting data by a specified field or fields. The fields to order by must be within the resource. You cannot sort by linked data even if it has been embedded. Multiple fields are sorted in the order given, separated by commas.

Request

The following orders the results by LastName.

```
GET https://sampleco.api.rentmanager.com/Tenants?orderby=LastName
```

Ascending is the default sort order. To sort descending specify the DESC keyword after the field.

The following orders the results by LastName descending.

```
GET https://sampleco.api.rentmanager.com/Tenants?orderby=LastName%20DESC
```

The following orders the results by LastName then by FirstName.

```
GET https://sampleco.api.rentmanager.com/Tenants?orderby=LastName,FirstName
```

Concurrency **Partial Support**

Concurrency may be enabled or disabled during authentication. The default is enabled. When concurrency is enabled request to update a record must include the UpdateDate field.

Concurrency cannot currently be disabled

Creating a Record**Partial Support**

When creating a record the http response should include a Location header to the newly created record. This is included even if the nocontent query string parameter is specified.

Supporting Partial Updates

The WAPI supports partial updates. The user can provide only the fields they wish to change to the API in a POST to an instance resource and the API should handle the process of updating the fields that were passed. Fields that are not passed are not updated.

To accomplish this the API must always perform a fetch of the object internally before updating the properties for the fields that were provided and then performing the update.

As noted above, if concurrency is enabled the POST must include the UpdateDate field in the posted data.

Example Model

Field	Data Type	Features
ApiUri	String	
Id	Int	
FirstName	String	
LastName	String	
ColorId	Int	
Color	Resource	Embed
DateCreated	DateTime	
CreateUserId	Int	
CreateUser	Resource	Embed
DateUpdated	DateTime	
UpdateUserId	Int	
UpdateUser	Resource	Embed
PrimaryContact	Resource	Embed, Atomic Create
Contacts	Resource	Embed

All Data Models include the "APIURI" field. This field specifies the Resource used to retrieve the data.

Data Model JSON Example

Note in the example below only ApiUri, Id, FirstName and LastName were passed. This allows the user to update specific fields instead of passing all fields in the model.

```
{
   "ApiUri": "https://sampleco.api.rentmanager.com/Tenants/1",
   "Id": 1,
   "FirstName": "Joe",
   "LastName": "Rocker"
}
```

Note that all Resource fields include at least the "APIURI" field. This follows the draft W3C recommendation for Linked Data (www.w3.org/TR/json-ld/) for the @id. @id is not used because it does not translate well to .net language data models.

Rate Limiting

The WAPI enforces rate limiting. This is implemented on a per company basis. Rate limiting is calculated on an hourly basis (number of request per hour) and all responses include rate limiting information including:

X-RateLimit: 60 (Number of requests allowed in the current timespan)
X-RateRemaining: 59 (Number of requests remaining in the current timespan)
X-RateTimeLeft: 4250 (Number of seconds remaining until the rate resets)

Finite vs Dynamic Results

Some collection resources have a finite collection of possible results that are defined by the resource. For example, the collection of Addresses associated with a given Tenant is a finite list defined by the AddressTypes associated with Tenants. When requesting addresses for a tenant the result will contain a record for each AddressType even if the value of that address type is blank or NULL.

Because of this Finite Resources can be viewed as having only Read and Update interactions available to them. Update implies the ability to set an address of a type that is already associated to the entity and the ability to clear an address of a type that is already associated to the entity. To Create a new address you must first associate an AddressType to an entity which may involve creating a new AddressType. To Delete an address you must disassociate the type from the entity. This will delete the addresses of that type for all entities it was associated with.

Request

```
GET https://sampleco.api.rentmanager.com/Tenants/115/Addresses
```

Response

```
{
   "Addresses": [
      {
         "APIURI": "https://sampleco.api.rentmanager.com/Addresses/12",
         "AddressTypeId": 4,
         "AddressType": {
            "APIURI": "https://sampleco.api.rentmanager.com/AddressTypes/4"
         },
         "Street1": "123 South Street",
         "Street2": "",
         "City": "Milford",
         "State": "Ohio",
         "PostalCode": "45150",
         "IsPrimary": true,
         "IsBilling": true
      },
      {
         "APIURI": "https://sampleco.api.rentmanager.com/Addresses/16",
      " A d d r e s s T y p e I d " :   5 ,
       " A d d r e s s T y p e " :   {
            "APIURI": "https://sampleco.api.rentmanager.com/AddressTypes/5"
         },
         "Street1": "548 Commerce Place",
         "Street2": "Suite 4",
         "City": "Loveland",
         "State": "Ohio",
         "PostalCode": "45140",
         "IsPrimary": false,
         "IsBilling": false
      },
      {
      "APIURI": null,
         "AddressTypeId": 6,
         "AddressType": {
            "APIURI": "https://sampleco.api.rentmanager.com/AddressTypes/6"
         },
         "Street1": null,
         "Street2": null,
         "City": null,
         "State": null,
         "PostalCode": null,
         "IsPrimary": null,
         "IsBilling": null
      },
   ]
}
```


This allows the client to update the associated addresses quickly.

Backwards Compatibility

At any time, LCS may make the following modifications to the WAPI without affecting usage:

- Create a new resource
- Create a new data model
- Add a field to an existing data model

The following cannot be done without breaking compatibility

- Changing a resource name – Redirects could be used if the resource was renamed but the functionality remained the same.
- Removing or renaming a field in a data model

Versioning

Versioning issues should be rare. If a resource is scheduled to be removed, It will respond with a custom header for a period of time before it is removed. Once the functionality has been removed, Requests for that resource will respond with 410 - Gone

Responding to Requests for Feedback

There are requests that might result in an exception because feedback is required. Some examples include: Hard Close Override, Owner Overdraft Override, Syncing Primary Contact and Account Names (when preference is set to Ask), and setting User Defined Field default values.

In cases where the user has sufficient privileges to allow the operation and an override has not been set the system should return an exception. The HTTP Response Code of 412 (Precondition Failed) should be returned along with the ErrorModel.

To override the precondition you should post again with the following Save Option provided.

Cross Join Tables **Partial Support**

There are many instances in the Rent Manager database where cross join tables are used to define the relationship between two types of data. Some examples are:

- Users to Roles
- Users to Privileges
- Roles to Privileges
- Properties to Floors
- Units to Amenities
- Users to Properties
- Users to Bank/CC Accounts

There are two ways to retrieve linked items:

Method	Result
GET https://sampleco.api.rentmanager.com/Users/{id}/Roles	The list of Roles the user is in
GET https://sampleco.api.rentmanager.com/Users/{id}/RoleSelectList	List of UserRoleSelectListItem Models

The Select List Item models allows for a more lightweight response that models both selected and available options in one call. All of the models follow a similar pattern:

Field	Type	Embed	Read Only	Atomic Create	Notes
Model1ID	Int		X		
Model2ID	Int		X		
Selected	Boolean				
Model1	DataModel	X	X		
Model2	DataModel	X	X		

Model1ID, Model1, Model2ID, and Model2 are Read-Only values. In this example, they would contain the UserID, UserModel, RoleID, and Role, repsectively. The user can change the Selected field and POST it back through the Unit to update the roles a user has been associated with. To interact with Users or Roles directly (add, view, edit, delete items) you would use the Users or Roles resource.

HTTP Response Codes

200 Codes indicate a successful request.

400 Codes indicate an issue with the request from the client side (something the client did or did not do). All 400 codes should return an ErrorModel object that includes details about the nature of the error.

500 Codes indicate an issue with the request from the server side. All 500 codes should return an ErrorModel

object that includes details about the nature of the error.

200 – OK

The request completed successfully.

201 – Created

The record(s) were created.

204 – No Content

The request was successful and there is no content to return. Used in response to DELETE requests.

206 – Partial Content

The response does not include all records. Additional requests are required to retrieve additional records. Indicates that pagination has been activated. There should be a Link: HTTP Header containing URL's for one or more of First, Previous, Next, or Last. There should also be a X-Total-Results: HTTP Header containing the total number of results that the request resulted in.

400 – Bad Request

The response failed because the request was invalid. Typically this is due to malformed input data.

401 – Unauthorized

The request failed because the client is not authenticated.

403 – Forbidden

The request failed because the client lacks sufficient permissions to perform the requested task.

404 – Not Found

The request failed because it did not find any matching data.

GET /Tenants/5 – Record 5 was not found to retrieve.

POST /Tenants/5 – Record 5 was not found for update.

DELETE /Tenants/5 – Record 5 was not found for delete.

In the case of collection resource requests, if the input content contains one or more specific records to retrieve then one or more of those records were not found. Remember that all collection resource requests are atomic. The returned Error model should include information on which records were not found.

405 – Method Not Allowed

The request was made to a valid resource, however the specified method (GET, POST, DELETE) is not allowed on this resource.

409 – Conflict

The request resulted in a conflict.

For Create requests this indicates that a record with the same parameters already exists and duplicates are not allowed.

For Update requests this indicates that a record was found but the concurrency information has changed, indicating that the underlying data has changed. In this case the client should request the latest version of the data, and attempt their update again.

412 – Precondition Failed

The request resulted in an exception because a precondition was not met. For example if the request would violate hard close and the hard close override was not sent with the request they will receive this response.

429 – Too Many Requests

The request failed because there have been too many requests from your client. Rate limiting has been enforced.

500 – Internal Server Error

An unhandled, unexpected error has occurred.

Resources and Permissions

Access to a resource may respond with one or more codes depending on the nature of the request and the privileges granted to the user making the request.

```
GET /Resource
```

When retrieving a collection of values from a resource the user, depending on filters, can be retrieving all records, or a subset of records. The following responses should be used.

If the user has access to at least one record in the resource after the filter is applied.

200 – OK

If the user does not have access to any records in the resource, regardless of the filter.

403 – Unauthorized

No records exist in the resource after the filter is applied, even if the reason no records exist is because of permissions.

404 – Not Found

GET /Resource/{Id}

If the user has access to the record specified by {Id}

200 – OK

If the {Id} specified is not found, or does not have access to the record:

404 – Not Found

Authentication Process

During authentication the client will POST a UserAuthorizationModel object to /Authentication/AuthorizeUser and if authentication is successful the server will respond with an APIToken. The APIToken must then be passed in the X-RM12Api-ApiToken header for all other requests.

1. Request is made to POST /Authentication/AuthorizeUser with a UserAuthorizationModel containing a minimum of the username and password.
2. WAPI12 looks up the company based on the information in the URL. This will retrieve the Company connection information.
3. WAPI12 connects to the Company database.
4. WAPI12 performs a login.
    1. Determine if the credentials are valid.
    2. Determine if the user already has a login token.
    3. Return or generate a login token as needed.

5. Create a APIToken.
6. Return the APIToken.
7. Subsequent requests include the APIToken which is used to connect and validate authentication for all other resources.

A token is always invalidated if the pertinent information about the user changes, such as permissions or password.