REST API GUIDE

Web Help Desk

Version 12.7.10

solarwinds

Last Updated: Thursday, October 20, 2022

Table of Contents

Technician and Administrator interface

Overview

For clarity in the examples, shell commands are split with a backslash character (\) into multiple lines of text. You may omit these backslashes and enter the entire command on a single line. For WebObjects Monitor or hosted deployments, replace `/helpdesk/WebObjects/Helpdesk.woa` with `/cgi-bin/WebObjects/ <app_name>.woa`.

REST support

The Web Help Desk REST Application Programming Interface (API) aims generally to support the Representational State Transfer (REST) architecture for creating, reading, updating, and deleting data. In this architecture, a URL uniquely identifies a resource in Web Help Desk (such as a Ticket or a Ticket list) and an HTTP Request Method identifies an action to be taken on the resource. The content of the HTTP request provides the data used to create and update the resource, which is typically in XML or JSON format.

HTTP defines methods that indicate an action to perform on a targeted resource.

The following table lists the HTTP methods supported by the Web Help Desk REST API.

Method	Action
GET	**Retrieve** the data for the resource identified in the URL. The requested resource can be a single entity (such as a Ticket) or a list of resources (such as a Group Tickets list). For example, the URL to retrieve Ticket number 13924 ends with `/ra/Tickets/13924`. The URL to retrieve Group Tickets ends with `/ra/Tickets/group`.
POST	**Create** a new entity from the data provided by the content portion of the request. This data should match the format obtained by a GET request. Because an identifier for the new entity has not yet been created, POST requests do not include the identifier in the URL, only a resource type identifier. The new resource identifier is returned in the response data.

Method	Action
PUT	**Update** the entity identified by the URL with the data provided by the request. This data should match the format obtained by a GET request. However, update requests need not provide all the data required to create a new entity—only the attributes that are to be changed. Unlike POST requests, PUT requests will include an identifier in the URL that indicates which entity is to be updated.
DELETE	**Delete** the resource identified by the URL.

List paging

When requesting lists, the `limit` and page parameters can be used to batch results. Alternatively, the `batchSize` and `batch` parameters can be used in place of `limit` and `page`. See Paging, limits, and batch size for details.

Data formats

ⓘ In the following examples, `localhost` is the hostname or IP address and port assigned to the Web Help Desk server. For example, `10.10.38.20:8443` where `8443` is the default secure port.

The Web Help Desk REST API supports JSON, XML, and Property List (`.plist`) formats. A special JSON format catered to the Sencha® frameworks is also supported. The format to use is specified by appending a type extension to the requested resource. If no extension is used, `.json` is assumed.

Format	Extension
JSON	`.json (default)`
JSON for Sencha[1]	`.sencha`
XML	`.xml`
Property List	`.plist`

ⓘ [1] JSON for Sencha encapsulates response data inside a single dictionary entry that is keyed by an identifier named `records`. Sencha models can use this identifier as their access point to the data.

In the following example, a session key is requested in JSON format:

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Session.json\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
{"type":"SessionKey","sessionKey":"Tzt9BdAtgo0nButX0lbmj0"}
```

The following example is the same request using XML:

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Session.xml\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
<?xml version="1.0" encoding="UTF-8"?>
    <SessionKeys type="SessionKey">
    <sessionKey>Tzt9BdAtgo0nButX0lbmj0</sessionKey>
    </SessionKeys>
```

Security

To prevent unauthorized third parties from observing your data, send requests using Secure Sockets Layer (SSL), especially when transmitting passwords and API keys.

Referencing clients

Wherever the REST API expects an `id` as a reference to a Client, the client's login ID (`username` or `email`) can be used in place of a numeric identifier, depending on which attribute was configured for Web Help Desk client authentication at Setup > Clients > Options > Client Login Attribute in the Web Help Desk Administrator Console. This allows LDAP clients that are not synchronized with Web Help Desk to be synchronized on-the-fly.

If a client referenced by login ID does not exist in the Web Help Desk database, all enabled LDAP Connections are searched for the client. If found, a new client account is created in the Web Help Desk database and initialized with values from the LDAP record.

If an account for the client does exist in the Web Help Desk database and the account is associated with an LDAP Connection, the account is synchronized with the LDAP Connection only if the LDAP cache timeout for the user has expired. See Setup > Clients > AD / LDAP Connections > [LDAP Connection] > Connection Basics > Cache Time Period in the Web Help Desk Administrator Console.

When used as an `id`, `usernames` and `emails` must be percent encoded (or "URL encoded"). This means that `%40` should be used in place of the "at" sign (@) and `%2E` should be used in place of periods. For example, `demo@mycompany.com` would be encoded as `demo%40mycompany%2Ecom`.

Authentication

Each request to the Web Help Desk REST API requires authentication. You can use the following authentication methods:

- Authenticating with username/password combination
- Authenticating with the user (Tech) API key
- Authenticating with the username and the application API key combination
- Authenticating with a temporary session key

In the following sections, `localhost` is the hostname or IP address and port assigned to the Web Help Desk server. For example, `10.10.38.20:8443`, where 8443 is the default secure port.

Authenticating with a username/password combination

In this authentication method, `username`and `password` parameters must be added to each HTTP request.

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Tickets/1\
> ?username=admin&password=admin"

Tickets/1
```

Authenticating with the user (Tech) API key

In this authentication method, the `apiKey` parameter must added to each HTTP request.

> **API keys**
> Techs can generate their API keys in the Web Help Desk Administrator Console at Setup > Techs > My Account > API Key. API keys should be considered confidential.

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Tickets/1\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

Tickets/1
```

Authenticating with the application API key

In this authentication method, `username` and `apiKey` parameters must be added to each HTTP request. The `apiKey` value differs from API keys generated for specific Techs.

> **Application API keys**
> Administrators can generate application API keys in the Web Help Desk Administrator Console at Setup > General > Authentication > Application API Keys. API keys should be considered confidential.

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Tickets/1\
> ?username=admin&apiKey=55KXyFEmI2lPEvTRDWyvlKboWIqpx2lItYvHRs5y"

Tickets/1
```

Authenticating with a session key

You can obtain a session key by providing a valid username and password to the Session resource. Using a session key can improve performance because it enables the caching of frequently-used resources. Also, session keys bring an extra level of security because they expire after 30 minutes of inactivity. A stolen session key that expired will be rejected.

In the following example, a session key is obtained and then used to authenticate a subsequent request for ticket details.

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Session\
> ?username=admin&password=admin"
{"type":"Session","sessionKey":"xDBRI20YLZtFkhcroHaEGg","instanceId":4}

$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa/4\
> /ra/Tickets/1\
> ?sessionKey=xDBRI20YLZtFkhcroHaEGg"

Tickets/1
```

You can use an API key in place of the username and password to obtain a session key, as shown below:

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Session\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
{
"type": "Session",
"sessionKey": "Krxz6CDcD4KfgXos6uuohM",
"currentTechId": 1,
"instanceId": -1
}
```

Terminating the session

The REST session can be terminated by submitting an HTTP `DELETE` request with the specified `sessionKey`—the key of the active session.

```
$ curl -X DELETE \
> "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Session\
> ?sessionKey=xDBRI20YLZtFkhcroHaEGg"

OK
```

Authenticating to a hosted Web Help Desk account

When authenticating to a hosted Web Help Desk account, the hosted account ID must also be included.

```
curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Session\
> ?username=admin&password=admin&accountId=1429"
```

Because hosted Web Help Desk accounts use a multi-instance WebObjects Monitor deployment, you must provide instance identifiers when using session authentication.

Using session authentication in a WebObjects Monitor deployment

When using a multi-instance configuration, requests that authenticate with a session key must be sent to the same instance from which the session key was originally obtained. In a WebObjects Monitor configuration, the instance to use is specified by including an instance number immediately after `<app_name>.woa`:

```
http://mycompany.com/cgi-bin/WebObjects/Helpdesk.woa/4/ra/Tickets/1
```

The number of the instance providing the session is returned as the `instanceId` attribute of the `/ra/Session` response data. If no instance is associated with the session, `instanceId` will be `-1`. The WebObjects adapter treats instance identifier `-1` the same way it treats requests without an instance identifier: it selects an arbitrary instance based on the configured load-balancing algorithm.

Authentication failures

If a request fails to authenticate due to invalid or missing credentials or to an expired session key, an HTTP 401 ("Unauthorized") status code is returned. Note the inclusion of the curl `-i` option in the following example in order to view the response headers.

```
$ curl -i "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Tickets/1"
HTTP/1.0 401 Apple WebObjects
     x-webobjects-loadaverage: 1
     content-length: 24

     Authentication failure.
```

Permission failures

If a request authenticates successfully but the authenticated Tech is unauthorized to perform the requested action, an HTTP 403 ("Forbidden") status code is returned.

Common request parameters

In the following sections, `localhost` is the hostname or IP address and port assigned to the Web Help Desk server. For example, `10.10.38.20:8443` where `8443` is the default secure port.

Style

You can add the `style` parameter to the HTTP request when entity information or a list of entities is requested. If it contains a value that differs from the `short` value, long and detailed information about the entities is returned. Otherwise, if no `style` parameter is provided or the `style=short` parameter is provided, short information is returned.

The following table lists the REST endpoints that support the `style` request parameter.

REST endpoint	List action	One entity action
Assets	Yes	No (detailed only)
Asset Statuses	Yes	Yes
Asset Types	Yes	Yes
Bulk Actions	Yes	Yes
Clients	Yes	Yes
Companies	Yes	Yes
Departments	Yes	Yes
Locations	Yes	Yes
Manufacturers	Yes	Yes
Models	Yes	Yes
Setup	Yes (short = detailed)	Yes (short = detailed)
Priority Types	Yes	Yes
Request Types	Yes	Yes
Rooms	Yes	Yes
Status Types	Yes	Yes
Techs	Yes	Yes

REST endpoint	List action	One entity action
Tickets	Yes	Yes
Ticket Notes	Yes	Yes

Examples:

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Assets/?style=short\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
```

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Assets/?style=details\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
```

Qualifier

You can add the qualifier parameter to the HTTP request with a value representing a filter to apply to the search results.

Qualifier clauses are in the following format:

`(<attribute> <op> <value>)`

where:

- `<attribute>` is the attribute name.
- `<op>` is one of the following operations: `=`, `!=`, `<`, `>`, `<=`, `>=`, `like`, or `caseInsensitiveLike`. When `like` or `caseInsensitiveLike` is used, `<value>` may contain asterisks (*) as wildcard placeholders.
- `<value>` is the attribute value. It could be a number, string, date, and so on. String and date values must be enclosed with single quotes (for example, 'some string value').

Qualifier clauses can be joined and nested with the keywords `and`, `or`, and `not`. Additionally, left and right brackets - '(' and ')' - can be used to create more complex qualifiers.

To reference nested attributes, use a period to join the components of the attribute path (for example, `location.locationName`).

When providing the qualifier as a GET query parameter, remember to percent-encode it.

The following table lists the REST endpoints that support the qualifier request parameter.

REST endpoint	Note
Assets	Full support
Asset Types	Full support
Clients	Limited support (predefined qualified is already used)
Companies	Full support
Locations	Full support
Manufacturers	Full support
Models	Full support
Tickets	Full support (limited support when the `list` parameter is used)

Example: All deleted Tickets.

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Tickets?qualifier=(deleted %3D 1)\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
```

Example: All Tickets including deleted.

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Tickets?qualifier=((deleted %3D null) or (deleted %3D 0)
  or (deleted %3D 1))\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
```

Paging, limits, batch size

When requesting lists, the `limit` and `page` parameters can be used to batch results.

Parameter	Description
`limit`	The maximum number of items to retrieve. Defaults to Setup > General > Options > Max Items Per Page in the Web Help Desk Administrator Console. An upper limit for this value is set by Setup > General > Options > Max Search Results in the Web Help Desk Administrator Console. If Max Search Results is empty, the default limit is used. See the table below.
`page`	Page of results to retrieve. Returns `limit` number of items, starting with item (`page * limit`) of the search results.

There are various `limit` default values due to historical reasons. The following table lists the REST endpoints and their `limit` default values.

Rest endpoint	Limit	Max limit	Uses general settings *
Assets	100	1000	Yes
Asset Statuses	100	1000	Yes
Asset Types	100	1000	Yes
Billing Rates	-	-	No
Bulk Actions	100	1000	Yes
Clients	25	100	No
Companies	100	1000	Yes
Departments	100	1000	Yes
Locations	100	1000	Yes
Manufacturers	100	1000	Yes
Models	100	1000	Yes
Setup	-	-	No
Priority Types	100	1000	Yes
Request Types	100	1000	Yes
Rooms	100	1000	Yes
Status Types	100	1000	Yes
Techs	100	1000	Yes
Tickets	25	100	No
Ticket Notes	100	1000	Yes

* Setup > General > Options: Max Search Results, Max Items Per Page

Example: First three Request Types.

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/RequestTypes?limit=3\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
```

Example: Two Request Types from page 3 (Fifth and sixth Request Type).

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/RequestTypes?page=3&limit=2\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
```

Mandatory fields

The following table lists the values in REST endpoints that are mandatory for POST/PUT (create/update) operations. They cannot be `null`, they must be present when creating an object, and they cannot be nullified when updating an object.

REST endpoint	Field name	Data type	Note
Assets			
	`assetNumber`	string	unique value
	`model`	object	
Asset Types			
	`assetType`	string	
Tickets			
	`problemtype`	object	
Companies			
	`companyName`	string	
Locations			
	`locationName`	string	
Manufacturers			
	`fullName`	string	
Models			
	`modelName`	string	
	`assettype`	object	
	`manufacturer`	object	

Tickets

In the following sections, `localhost` is the hostname or IP address and port assigned to the Web Help Desk server. For example, `10.10.38.20:8443` where `8443` is the default secure port.

Modifiable ticket fields

The following table lists all Ticket fields that can be modified with create and update operations. The fields with darker background are mandatory.

Field name	Data type	Note
`assets`	array of Asset objects	
`assignToCreatingTech`	boolean	
`bccAddress`	string	255 characters
`ccAddressForTech`	string	255 characters
`clientReporter`	Client object	
`clientTech`	Tech object	
`department`	Department object	
`detail`	string	long text
`emailBcc`	boolean	
`emailCc`	boolean	
`emailClient`	boolean	
`emailGroupManager`	boolean	
`emailTech`	boolean	
`emailTehGroupLevel`	boolean	
`location`	Location object	
`prioritytype`	Priority Type object	
`problemtype`	Request Type object	mandatory
`reportDateUtc`	date	
`room`	string	80 characters

Field name	Data type	Note
`statustype`	Status Type object	
`subject`	string	255 characters

HTTP GET request parameters

The following parameters can be added to the HTTP request.

Parameter	Required	Description
`withTUC=true`	optional	The following properties will contain the date in UTC format: • `closeDateUtc` • `lastUpdatedUtc` • `displayDueDateUtc` • `notes.dateUtc`

Getting a list of Tickets for a Tech

The following resources can be used to retrieve a list of tickets.

List	Resource
My Tickets	`/ra/Tickets/mine`
Group Tickets	`/ra/Tickets/group`
Flagged Tickets	`/ra/Tickets/flagged`
Recent Tickets	`/ra/Tickets/recent`

Alternatively, the query parameter list can be used with `/ra/Tickets` to identify the list to retrieve:

Parameter	Description
`list`	Ticket list to retrieve (either `mine`, `group`, `flagged`, or `recent`).

Lists reflect the access permissions of the authenticated tech. Tickets are returned in reverse order of the last update date (`lastUpdated`).

In the following example, we request the first page of the authenticated user's Group Tickets using a page size (`limit`) of three tickets:

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Tickets/group?page=1&limit=3\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

Tickets/group
```

The following example uses the list query parameter to retrieve the same list:

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Tickets?list=group&page=1&limit=3\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
```

Searching for Tickets

Tickets can be searched for by including a qualifier query parameter. When the qualifier parameter is included, the list type is optional.

Example: My tickets with location 'ATL'.

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Tickets?list=mine&qualifier=(location.locationName%3D'ATL')\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

Tickets
```

Example: The same as the previous example, using the qualifier parameter only.

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Tickets?qualifier=((clientTech.clientId %3D 1) and
  (statustype.listFilterType %3D 1) and (location.locationName %3D 'ATL'))\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

Tickets
```

Example: All tickets that are in the status that has id 1.

See Status Types for details.

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Tickets?qualifier=(statusTypeId %3D 1)&limit=3\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

Tickets
```

Example: All tickets that are in the status have the name Open (same as previous).

See Status Types for details.

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Tickets?qualifier=(statustype.statusTypeName %3D 'Open')&limit=3\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
Tickets
```

Searching for deleted Tickets

You can search for Tickets by including a qualifier query parameter with a value containing the word "deleted". In this case, the list type must not be provided.

Example: My deleted Tickets.

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Tickets?qualifier=((clientTech.clientId %3D 1) and
(statustype.listFilterType %3D 1) and (deleted %3D 1))&limit=3\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

Tickets
```

Example: All my Tickets including deleted.

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Tickets?qualifier=((clientTech.clientId %3D 1) and
(statustype.listFilterType %3D 1) and ((deleted %3D null) or (deleted %3D 0)
or (deleted %3D 1)))&limit=3\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

Tickets
```

Getting Ticket details

To retrieve details for a specific ticket, request the `/ra/Tickets/<ticket_number>` resource.

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Tickets/1\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

Tickets/1
```

Creating a Ticket

Tickets are created by submitting an HTTP POST request containing ticket data in the specified format. The HTTP response provides the ticket `id`, as well as the assigned `statusTypeId`, `locationId`, and `lastUpdated` date.

If `sendEmail` is true, a confirmation email is sent to parties assigned to receive email (such as `emailClient`, `emailTech`, `emailCc/ccAddressesForTech`, and so on).

Custom Field Values
You can include custom fields in the request data. See Custom Field Definitions for details.

Example: Create a request data file `ticket.json` with the following content.

```
{"reportDateUtc":"2013-12-11T14:15:16Z","room":"","emailClient":true,
"emailTech":true,"emailTechGroupLevel":false,"emailGroupManager":false,
"emailCc":false,"ccAddressesForTech":"","emailBcc":false,"bccAddresses":
"","subject":"Need more memory","detail":"My computer is running really slow.
I think it's because I need more memory.","assignToCreatingTech":false,
"problemtype":
{"type":"ProblemType","id":6},"sendEmail":false,"location"
{"type":"Location","id":1},"department":{"type":"Department","id":1},
"clientReporter"
{"type":"Client","id":1},"customField_1":"Version9.1.7","statustype":{"type":
"StatusType","id":1},"prioritytype":
{"type":"PriorityType","id":1},"assets"
[{"id":1,"type":"Asset"},{"id":2,"type":"Asset"}]}
```

When you submit the HTTP `POST` request, you will receive a response similar to the following:

```
$ curl -X POST -d @ticket.json \
> "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Tickets\
```

```
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
{"type":"Ticket","id":4,"subject":"Need more memory","statusTypeId":1,
"locationId":1,"lastUpdated":"2013-12-11T14:51:11Z"}
```

When you create a ticket with an archived request type, the response is:

```
400 Bad Request error. Cannot create ticket with an archived request type.
```

Updating a Ticket

The syntax for updating a ticket is identical to creating a ticket (above), except that the request should use the HTTP PUT method and the ticket number (`id`) must be provided:

```
$ curl -X PUT -d @ticket.json \
> "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Tickets/4\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
```

Custom Field values
You can include Custom Fields in the request data. See Custom Field Definitions for details.

Referencing Clients
Wherever the REST API expects an `id` to reference a Client, the client's login ID (username or email) can be used in place of a numeric identifier. See Referencing Clients for details.

When you update a ticket with an archived request type using the API, the response is:

```
200 OK
```

When you update a ticket by changing the request type to an archived request type, the response is the same, but the ticket will not be updated.

Changing a Ticket Status

Ticket status can be changed by submitting an HTTP `PUT` request. It is basically an update ticket request with the Status Type change.

Example: To close the Ticket, create a request data file `ticket.json` with the following content:

```
{"statustype":{"type":"StatusType","id":3}}
```

When you submit the HTTP `PUT` request, you will receive a response like the following:

```
$ curl -X PUT -d @ticket.json \
> "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Tickets/4\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
{"id":4,"type":"Ticket","lastUpdated":"2014-01-23T13:50:30Z","locationId":7,
"statusTypeId":1,"subject":"Can I..."}
```

Escalating and de-escalating a Ticket

The syntax for escalating and deescalating a ticket is identical to updating a ticket , except that the request should use the HTTP `GET` method and the ticket number (`id`) must be provided:

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Tickets/4/escalate\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Tickets/4/deescalate\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
```

Deleting a Ticket

Tickets are deleted by submitting an HTTP `DELETE` request. The HTTP response will provide short information about the deleted ticket.

```
$ curl -X DELETE \
> "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Tickets/39\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
{"id":39,"type":"Ticket","lastUpdated":"2014-01-22T15:26:40Z","locationId":1,
"statusTypeId":1,"subject":"Need more memory"}
```

Getting a list of notes on a Ticket

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/TicketNotes?jobTicketId=1\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
```

```
TicketNotes
```

Adding a note to a Ticket

The following table lists all Ticket Note fields that can be set with a create operation.

Field name	Data type	Note
bccAddresses	string	
billingRate	Billing Rates object	
ccAddressesForTech	string	
emailBcc	boolean	
emailCc	boolean	
emailClient	boolean	
emailGroupManager	boolean	
emailTech	boolean	
emailTechGroupLevel	boolean	
isHidden	boolean	
isSolution	boolean	
jobticket	Ticket object	
noteText	string	Long text
statusTypeId	integer	
workTime	string	Number of minutes

Example: Create a request data file `tech_note.json` with the following content:

```
{"noteText":"Reinstalled the printer driver. Seems to be working fine now.",
"jobticket":
{"type":"JobTicket","id":1},"workTime":"20","isHidden":false,"isSolution":
false,"billingRate":
{"type":"BillingRate","id":1},"statusTypeId":5,"emailClient":true,"emailTech"
:true,"emailTechGroupLevel":
```

When you submit the HTTP `POST` request, you will receive the following response:

```
$ curl -X POST -d @tech_note.json \
> "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/TechNotes\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
{"id":16,"type":"TechNote","date":"2013-12-
16T11:19:27Z","noteText":"Reinstalled the printer driver. Seems to be working
fine now.","jobticket":{"id":1,"type":"Ticket"}}
```

Getting a Ticket attachment

```
$ curl -I -X GET "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/TicketAttachments/1\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
HTTP/1.1 200 Apple WebObjects
cache-control: private
content-disposition: inline;filename="BugReport.txt"
expires: Thu, 13-Oct-2011 23:12:13 GMT
date: Tue, 13-Sep-2011 23:07:52 GMT
x-webobjects-loadaverage: 1
content-type: application/octet-stream; charset=ISO-8859-1; name=
"BugReport.txt"
content-length: 18854
```

The binary contents of the attachment are returned in the HTTP response body as an octet stream. The previous example uses the curl `-I` option to show only the HTTP headers of the response.

Adding an attachment to a Ticket or Ticket Note

If the user is logged in to the Web Help Desk (GUI) and has a valid REST session, the application accepts "/attachment/upload**" REST calls.

The following table lists all HTTP request parameters that should be provided.

Parameter name	Data type	Parameter value	Note
`type`	string	One of the following: • `jobTicket` • `techNote` • `clientNote` • `faq` • `client`	The entity type. The attachment will be assigned to this type of entity.
`entityId`	integer	Entity ID	The attachment will be assigned to the entity with specified ID
`returnFields`	string	For example: `"id.uploadDate"`	Optional parameter

The following steps describe how to add an attachment to the Ticket or the Ticket Note.

1. Create a REST session for a user. See Authenticating with a session key for details.
2. Call the "/attachment/upload**" REST with the Cookie HTTP header that contains the Java session ID and WebObjects session ID from the previous call. See the example below.
3. Terminate the session. See Terminating the session for details.

Example: Create an HTTP request with the following content (`add_attachment.raw`).

```
POST
/helpdesk/attachment/upload?type=jobTicket&entityId=40&returnFields=id,
uploadDate HTTP/1.1
Cookie: JSESSIONID=BA1B63AA2DD9EBBB62B7A20E37377DED;
wosid=EoPFUriceH4t4jn5HiXiqg
Content-Type: multipart/form-data; boundary====1400591996857===
Cache-Control: no-cache
Pragma: no-cache
User-Agent: Java/1.7.0_55
Host: 127.0.0.1:8081
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive
Content-Length: 2143
------=1400591996857===
Content-Disposition: form-data; name="fileUpload"; filename="my.ini"
Content-Type: null
Content-Transfer-Encoding: binary
# MySQL Server Instance Configuration File
# ----------------------------------------------------------------------
```

```
... the rest of the file...
--====1400591996857===--
```

When you submit the HTTP `POST` request, you will receive the response as follows:

```
$ curl -X POST -d @add_attachment.raw \
> "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra//helpdesk/attachment/upload?
type=jobTicket&entityId=40&returnFields=id,uploadDate"
{"id":11,"uploadDate":"2014-04-15 14:27:51.432+02"}
```

Sending email for a Ticket

```
$ curl -X POST -d '{"ticketId":1}'\
> "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Email\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
{"ticketId":"1", "techNoteId":null, "fullDetails":false}
```

Sending email for a Tech Note

```
$ curl -X POST -d '{"techNoteId":1, "fullDetails":false}'\
> "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Email\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
{"ticketId":null, "techNoteId":1, "fullDetails":false}
```

Billing rates

Getting a list of Billing Rates

ⓘ In the following examples, `localhost` is the hostname or IP address and port assigned to the Web Help Desk server. For example, `10.10.38.20:8443` where `8443` is the default secure port.

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/BillingRates\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
```

```
BillingRates
```

If the Tech has no permission to view the billing rate hourly rate (located at Setup > Techs > Tech Permissions > View Billing Rate Hourly Rate in the Web Help Desk Administrator Console), the list of billing rates will not contain the hourly rate values.

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/BillingRates\
> ?username=tech&password=tech"

BillingRates
```

Techs

ⓘ In the following sections, `localhost` is the hostname or IP address and port assigned to the Web Help Desk server. For example, `10.10.38.20:8443` where `8443` is the default secure port.

Getting a list of Techs

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Techs?limit=3\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

Techs
```

Getting a specific Tech

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Techs/1\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

Techs/1
```

Getting the currently authenticated Tech

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Techs/currentTech\
```

```
> ?sessionKey=Tzt9BdAtgo0nButX0lbmj0"

Techs/currentTech
```

Clients

ⓘ In the following sections, `localhost` is the hostname or IP address and port assigned to the Web Help Desk server. For example, `10.10.38.20:8443` where `8443` is the default secure port.

Modifiable client fields

Field name	Data type	Note
address	string	255 characters
authenticationDate	date	
authenticationKey	string	50 characters
authenticatedEmail	string	255 characters
baseDN	string	255 characters
city	string	100 characters
color	string	Seven characters (for example, #F99D1C)
creationDate	date	
dashboardLayout	string	long text
email	string	255 characters
employmentStatus	string	50 characters
firstName	string	80 characters
lastActivity	date	
lastIpAddress	string	80 characters
lastName	string	80 characters
latestChangeToInactive	date	
localeString	string	10 characters

Field name	Data type	Note
location	Location object	
notes	string	long text
password	string	80 characters
passwordCacheDate	date	
phone	string	50 characters
phone2	string	50 characters
secondaryEmail	string	255 characters
state	string	50 characters
ticketCC	string	255 characters
username	string	50 characters
zip	string	25 characters

Getting a list of Clients

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Clients?limit=3\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

Clients
```

Searching for Clients

Clients can be searched by including a qualifier query parameter.

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Clients?qualifier=(firstName%20like%20'Rich*')%20and%20
(location.locationName%3D'PES')\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

Clients
```

Getting unsynchronized LDAP Clients

By default, records are returned for only those LDAP Clients that are synchronized with the Web Help Desk Administrator Console. To include LDAP Clients that are not synchronized, include `searchLdap=true` as a query parameter.

Parameter	Description
`searchLdap`	If set to `true`, up to 25 unsynchronized clients associated with LDAP Connections will be included in the search results if the number of matching synchronized clients does not exceed the Max Search Results value in Setup > General > Options.

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Clients?searchLdap=true\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
```

In the search results, LDAP Client records that have never been synchronized will include two additional attributes (`isUnsyncedLdapClient` and `dn`) and their `id` attribute will be set to the client's login ID (username or password).

Attribute	Description
`isUnsyncedLdapClient`	Set to `true` to indicate that the Client record is in a configured LDAP Connection (Setup > Clients > LDAP Connections) but has not synchronized with the Web Help Desk database. Synchronization is performed automatically when: • The client record is referenced as an argument to associate with an entity (for example, the `clientReporter` of a Ticket). • The client authenticates with Web Help Desk or submits an email ticket. • A Bulk Synchronization occurs (Setup > Clients > LDAP Connections > [LDAP Connection] > Connection Basics > Bulk Synchronization). Unsynchronized LDAP clients do not appear in client search results unless `searchLdap=true` is included as a query parameter.
`dn`	The value of the LDAP record's `distinguishedName` property. This is provided so the record can be referenced directly through the LDAP directory service.

Attribute	Description
`id`	Because the client record has never been synchronized with the Web Help Desk database and does not yet have a numeric ID, the `id` parameter will be returned as the client's login ID attribute (either username or email, depending on the Setup > Clients > Options > Client Login Attribute setting). The client login attribute can be used wherever a client ID is expected. This is true of non-LDAP clients as well.

Getting Client details

When referencing clients, the `id` can be a numeric identifier or the client's `username` or `email address`— depending on which attribute is configured for Web Help Desk client authentication (Setup > Clients > Options > Client Login Attribute).

Encoding usernames and emails as client IDs
When used as an `id`, `usernames` and `emails` must be percent encoded (or "URL encoded"). This means that `%40` should be used in place of the "at" sign (@) and `%2E` should be used in place of periods. For example, `demo@mycompany.com` would be encoded as `demo%40mycompany%2Ecom`.

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Clients/1\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

Clients/1
```

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Clients/client\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

Clients/client
```

Creating a Client

Clients are created by submitting an HTTP `POST` request containing client data in the specified format. The HTTP response will provide the assigned client `id`.

Example: Create a request data file `client.json` with the following content:

```
{"type":"Client","email":"client@helpdesk.com","firstName":"Demo","lastName":
"Client","phone":"1-510-279-2251","phone2":"1-877-943-0008","location":
{"id":3},"username":"client"}
```

When you submit the HTTP `POST` request, you will receive the response as follows:

```
$ curl -X POST -d @client.json \
> "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Clients\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

{"id":4,"type":"Client","email":"client@helpdesk.com","firstName":"Demo",
"lastName":"Client","notes":null,"phone":"1-510-279-2251","phone2":
"1-877-943-0008","department":null,"location":{"id":3,"type":"Location",
"address":null,"city":null,"locationName":"Marshall High School","postalCode"
:null,"state":null,"priorityTypes":[],"useSpecificPriorities":false,
"defaultPriorityTypeId":null},"room":null,"companyName"
```

LDAP Client account creation
It is not necessary to create accounts for clients that are contained in LDAP Connections but not synchronized with Web Help Desk. These client accounts are created automatically. They can be referenced by using their login ID in place of a numeric id value. See Referencing Clients for details.

Updating a Client

The syntax for updating a client is identical to creating a client, except that the request should use the HTTP `PUT` method and you provide the client `id`, as shown below:

```
$ curl -X PUT -d @client.json \
> "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Clients/4\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
```

Referencing Clients
Wherever the REST API expects an `id` to reference a Client, the client's login ID (username or email) can be used in place of a numeric identifier. See Referencing Clients for details.

Priority types

The `dueTimeMinutes`, `alertReminderMinutes`, and `clientReminderMinutes` properties can contain a positive number that represents a number of minutes or the value -1 that represents the value "Never" on the GUI.

ⓘ In the following sections, `localhost` is the hostname or IP address and port assigned to the Web Help Desk server. For example, `10.10.38.20:8443` where `8443` is the default secure port.

Getting a list of Priority Types

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/PriorityTypes\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

PriorityTypes
```

Searching for Priority Types

You can search for Priority Types by including the following optional parameter:

Parameter	Description
`locationId`	If set to a Location `id`, Priority Types will be filtered to those associated with the Location with the specified `locationId`. If the Location has no specific Priority Type, all Priority Types are included in the result.

Example: All Priority Types of the specified Location.

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/PriorityTypes?locationId=2\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

PriorityTypes
```

Getting a specific priority type

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/PriorityTypes/1\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

PriorityTypes/1
```

Request types

In the following sections, `localhost` is the hostname or IP address and port assigned to the Web Help Desk server. For example, `10.10.38.20:8443` where `8443` is the default secure port.

Getting a list of Request Types

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/RequestTypes?limit=3\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

RequestTypes
```

Searching for Request Types

You can search for Request Types by including one of the following optional parameters listed below.

Parameter	Description
`locationId`	If set to a Location `id`, Request Types will be filtered to those associated with the Location with the specified `locationId`. If a Request Type is not linked to any specific Location, it is included in the result.
`departmentId`	If set to some Department `id`, Request Types will be filtered to those associated with the Department with the specified `departmentId`. If a Request Type is not linked to any specific Department, it is included in the result.
`clientId`	If set to a Client `id`, Request Types will be filtered to those associated with the Client's Locations and Departments.
`parentId`	If set to a Request Type `id`, Request Types will be filtered to those associated with the parent Request Type with the specified ID.

Parameter	Description
`list`	`all`: all Request Types will be returned. `leavesOnly`: Request Types will be filtered to leave Request Types only. By default, if the list parameter is not specified or if it contains a value that differs from the values above, Request Types will be filtered to parent Request Types only.

If more than one parameter is added to the request, the logical operator 'AND' is used when Request Types are filtered.

Example: All Request Types with no specific Location or with Location with id=1.

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/RequestTypes?locationId=1&limit=3\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

RequestTypes
```

Getting a specific Request Type

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/RequestTypes/1\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

RequestTypes/1
```

Status types

In the following sections, `localhost` is the hostname or IP address and port assigned to the Web Help Desk server. For example, `10.10.38.20:8443` where `8443` is the default secure port.

Getting a list of Status Types

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/StatusTypes\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
```

```
StatusTypes
```

Getting a specific Status Type

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/StatusTypes/1\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

StatusTypes/1
```

Departments

In the following sections, `localhost` is the hostname or IP address and port assigned to the Web Help Desk server. For example, `10.10.38.20:8443` where `8443` is the default secure port.

Getting a list of Departments

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Departments\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

Departments
```

Getting a specific Department

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Departments/1\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

Departments/1
```

Locations

In the following sections, `localhost` is the hostname or IP address and port assigned to the Web Help Desk server. For example, `10.10.38.20:8443` where `8443` is the default secure port.

Modifiable Location fields

The following table lists all Location fields that can be modified with create and update operations. The fields with darker background are mandatory.

Field name	Data type	Note
address	string	80 characters
billingRate	Billing Rate object	
businessZone	Business Zone object	No REST endpoint for Business Zones
city	string	40 characters
color	string	Seven characters
country	string	80 characters
domainName	string	1024 characters
fax	string	20 characters
hasPartsBoolean	boolean	
isClientAssetSearchEnabled	boolean	
isClientLoginEnabled	boolean	
isCreateEmailTicketsEnabled	boolean	
isCreateWebTicketsEnabled	boolean	
isInactive	boolean	
isWorkTimeVisible	boolean	
locationName	string	255 characters (mandatory)
note	string	Long text
outgoingEmailAccount	SMTP Server object	No REST endpoint for SMTP servers
phone	string	20 characters
phone2	string	20 characters
postalCode	string	20 characters
priorityType	Priority Type object	Default Priority Type

Field name	Data type	Note
`priorityTypes`	Array of Priority Type objects	
`state`	string	20 characters
`taxRates`	Array of Tax Rate objects	No REST endpoint for Tax Rates
`useSpecificPriorities`	boolean	

Getting a list of Locations

Parameter	Description
`filter`	If true, Locations will be filtered to those associated with the authenticated Tech. The default is `false`.

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Locations?limit=3\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

Locations
```

Searching for Locations

You can search for Locations by including a qualifier query parameter.

Example: All Locations with a name like 'Sample'.

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Locations?qualifier=(locationName like '*Sample*')\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

Locations
```

Searching for deleted Locations

You can search for Locations by including a qualifier query parameter.

Example: All deleted Locations

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Locations?qualifier=(deleted %3D 1)\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

Locations
```

Example: All Locations including deleted

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Locations?qualifier=((deleted %3D null) or (deleted %3D 0) or
(deleted %3D 1))&limit=3\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

Locations
```

Getting a specific Location

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Locations/1\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

Locations/1
```

Creating a Location

Locations are created by submitting an HTTP `POST` request containing Location data in the specified format. The HTTP response will provide the Location id, as well as the `LocationName`, the assigned `priorityTypes`, and the default Priority Type `id`.

Example: Create a request data file `location.json` with the following content:

```
{"address":"The Address","billingRate":
{"type":"BillingRate","id":1},"businessZone":
{"type":"BusinessZone","id":1},"city":"The City","color":"#FE2517","country":
"The Country","domainName":"domain.com","fax":"123456789",
"hasPartsBoolean":false,"isClientAssetSearchEnabled":false,
"isClientLoginEnabled":false,"isCreateEmailTicketsEnabled":false,
"isCreateWebTicketsEnabled"false,"isInactive":false,
"isWorkTimeVisible":true,"locationName":"The Location",
"note":"This is my Location","outgoingEmailAccount"
{"type":"SmtpServer","id":1},"phone":"123456789","phone2":"123456788",
```

```
"postalCode":"12345","priorityType"
{"type":"PriorityType","id":1},"state":"The state",
"useSpecificPriorities":true}
```

When you submit the HTTP `POST` request, you will receive a response as shown below.

```
$ curl -X POST -d @location.json \
> "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Locations\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
{"id":14,"type":"Location","locationName":"The
Location","priorityTypes":[],"defaultPriorityTypeId":1}
```

Updating a Location

The syntax for updating a Location is identical to creating a location. except that the request should use the HTTP `PUT` method and provide the Location (`id`).

```
$ curl -X PUT -d @location.json \
> "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Locations/14\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
```

Deleting a Location

You can delete Locations by submitting an HTTP `DELETE` request. The HTTP response provides information about the deleted Location.

```
$ curl -X DELETE \
> "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Locations/14\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
{"id":14,"type":"Location","locationName":"The Location","priorityTypes":[],
"defaultPriorityTypeId":1}
```

Rooms

In the following sections, `localhost` is the hostname or IP address and port assigned to the Web Help Desk server. For example, `10.10.38.20:8443` where `8443` is the default secure port.

Getting a list of Rooms

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Rooms?limit=3\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

Rooms
```

Getting a specific Room

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Rooms/1\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

Rooms/1
```

Bulk actions

In the following sections, `localhost` is the hostname or IP address and port assigned to the Web Help Desk server. For example, `10.10.38.20:8443` where `8443` is the default secure port.

Getting a list of Bulk Actions

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/TicketBulkActions\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

TicketBulkActions
```

Custom field definitions

In the following sections, `localhost` is the hostname or IP address and port assigned to the Web Help Desk server. For example, `10.10.38.20:8443` where `8443` is the default secure port.

The following table lists the types of custom field definitions. Only the fields with darker background can be listed/acquired:

Type	Value
Asset	1
Client	2
Location	3
Ticket	4
Department	5
Purchase orders	6
Task Element	7
Part	8

Getting a list of Ticket Custom Field Definitions

You can search for Ticket Custom Field Definitions by including one of the optional parameters listed below.

Parameter	Description
`requestTypeId`	If set to a Request Type `id`, Custom Field Definitions will be filtered to those associated with the Request Type with the specified `requestTypeId`.
`statusTypeId`	If set to some Status Type `id`, Custom Field Definitions will be filtered to those associated with the Status Type with the specified `statusTypeId`. If no `requestTypeId` parameter is provided, then the `statusTypeId` parameter is ignored.

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/CustomFieldDefinitions?limit=3\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
```

```
CustomFieldDefinitions
```

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/CustomFieldDefinitions?requestTypeId=54&limit=3\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

CustomFieldDefinitions
```

Getting a list of Asset Custom Field Definitions

Asset Custom Field Definitions can be searched for by including the following optional parameter:

Parameter	Description
`assetTypeId`	If set to an Asset Type `id`, Custom Field Definitions will be filtered to those associated with the Asset Type with the specified `assetTypeId`.

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/CustomFieldDefinitions/Asset?limit=3\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

CustomFieldDefinitions/Asset
```

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/CustomFieldDefinitions/Asset?assetTypeId=10&limit=3\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
CustomFieldDefinitions/Asset
```

Getting a list of Location Custom Field Definitions

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/CustomFieldDefinitions/Location?limit=3\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

CustomFieldDefinitions/Location
```

Setting Custom Field values to entities

The following table lists all REST endpoints that support the changing of Custom Field values. If a Custom Field is required, it has to be included in the create and update requests.

REST Endpoint
Assets
Tickets
Locations

Syntax

You can change the Custom Field values of an entity using one of the following options:

- `customFields` represents an array of Custom Fields objects with the `definitionId` and `restValue` properties. Here, the `definitionId` value corresponds to the `id` property of the custom field's corresponding `CustomFieldDefinition`, and the value of the `restValue` property represents the actual Custom Field value.
- `customField_<id>`, where `<id>` corresponds to the id property of the custom field's corresponding `CustomFieldDefinition`.

Example: Set Custom Field values (array) of Ticket number 1.

```
$ curl -X PUT \
> "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Tickets/1\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
HTTP PUT data:
{
     ...
     "customFields": [
     {
     "definitionId": 1,
     "restValue": "Version 9.1.7"
     },
     {
     "definitionId": 2,
     "restValue": 12345
     }
     ],
     ...
     }
```

Example: Set Custom Field values (by id) of Ticket number 1.

```
$ curl -X PUT \
> "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Tickets/1\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

HTTP PUT data:
{
     ...
     "customField_1": "Version 9.1.7",
     "customField_2": "12345",
     ...
     }
```

Setup

In the following section, `localhost` is the hostname or IP address and port assigned to the Web Help Desk server. For example, `10.10.38.20:8443` where `8443` is the default secure port.

Getting system configuration settings

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Preferences\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

Preferences
```

Assets

In the following sections, `localhost` is the hostname or IP address and port assigned to the Web Help Desk server. For example, `10.10.38.20:8443` where `8443` is the default secure port.

Modifiable Asset Fields

The following table lists all Asset fields that can be modified with create or update operations. The fields with darker background are mandatory.

Field name	data type	Note
assetNumber	string	60 characters (mandatory)
assetstatus	Asset Status object	
billingRate	Billing Rate object	
clients	Array of Client objects	
contractExpiration	date	
isNotesVisibleToClients	boolean	
isReservable	boolean	
isSynchronizationDisabled	boolean	
location	Location object	
macAddress	string	40 characters
model	Model object	mandatory
networkAddress	string	255 characters
networkName	string	255 characters
notes	string	Long text
purchaseDate	date	
serialNumber	string	255 characters
version	string	100 characters
warrantyType	Warranty Type object	No REST endpoint for Warranty Types

Getting a list of Assets

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Assets?limit=3\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
```

```
Assets
```

Searching for Assets

You can search for Assets by including a qualifier query parameter.

Example: All Assets with the asset number '171'.

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Assets?assetNumber=171\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

Assets
```

Example: The same as the previous example, when using the qualifier parameter only.

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Assets?qualifier=(assetNumber %3D '171')\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

Assets
```

Searching for deleted Assets

You can search for Assets by including a qualifier query parameter.

Example: All deleted Assets.

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Assets?qualifier=(deleted %3D 1)&limit=3\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

Assets
```

Example: All Assets including deleted.

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Assets?qualifier=((deleted %3D null) or (deleted %3D 0) or (deleted %3D
1))&limit=3\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
```

```
Assets
```

Getting Asset details

To retrieve details of a specific Asset, request the `/ra/Assets/<asset_id>` resource, as shown below:

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Assets/4\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

Assets/4
```

Creating an Asset

Create an Asset by submitting an HTTP `POST` request containing Asset data in the specified format. The HTTP response provides:

- The Asset `id`, as well as the `assetNumber`.
- The assigned `assetstatus`.
- The `location`.
- The `model`, along with the `assettype` and the `manufacturer`.

To create an Asset, make sure that all relevant entities referenced by this Asset currently exist. These entities are mainly Asset Type, Manufacturer and Model. As a result, the Asset creation process should consist of the following steps:

1. Create or find all relevant entities that you want to assign to a new Asset, for example, Asset Status, Billing Rate, Location, Clients and so on. See Getting a list of Asset Statuses, Getting a list of Billing Rates, Getting a list of Locations, Getting a list of Clients, and so on.)
2. Create the Asset Type. See Creating an Asset Type for details.
3. Create the Manufacturer. See Creating a Manufacturer for details.
4. Create the Model. See Creating a Model for details.
5. Create the Asset.

> **Custom Field values**
> Custom Fields can be included in the request data. See Custom Field Definitions for details.

Example: Create a request data file `asset.json` with the following content:

```
{"assetNumber":"A666","contractExpiration":"2013-07-
29T11:16:55Z","macAddress":

"macAddress","networkAddress":"networkAddress","networkName":"networkName","n
otes":
"notes","purchaseDate":"2013-07-29T00:00:00Z","serialNumber":"serialNumber",
"version":"version","assetstatus":
{"id":1},"billingRate":{"id":1},"location":{"id":1},"model":
{"id":1},"warrantyType":{"id":1},"clients":[{"id":1,"type":"Client"},

{"id":2,"type":"Client"}],"isSynchronizationDisabled":false,"isReservable":fa
lse,
"isNotesVisibleToClients":false,"customFields":
[{"definitionId":14,"restValue":"Some string value"},
{"definitionId":15,"restValue":12345},
{"definitionId":16,"restValue":12.34}]}
```

When you submit the HTTP `POST` request, you will receive a response as shown below:

```
$ curl -X POST -d @asset.json \
> "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Assets\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
{"id":9,"type":"Asset","assetNumber":"A666","contractExpiration":
"2013-07-
29T11:16:55Z","macAddress":"macAddress","networkAddress":"networkAddress",
"networkName":"networkName","notes":"notes","purchaseDate":"2013-07-
29T00:00:00Z",
"serialNumber":"serialNumber","version":"version","assetstatus":
{"id":1,"type":"AssetStatus"},"billingRate":
{"id":1,"type":"BillingRate"},"location":
{"id":1,"type":"Location","address":null,"city":null,"locationName":"Sample
Location","postalCode":null,"state":null,"defaultPriorityTypeId":null},"mode
l":

{"id":1,"type":"Model","assetTypeId":1,"manufacturerId":1,"modelId":1,"modelN
ame":
"Sample Model","warrantyTypeId":1,"assettype":
{"id":1,"type":"AssetType","assetType":"Sample Type"},"manufacturer":

{"id":1,"type":"Manufacturer","address":null,"city":null,"country":null,"fax"
:null,
"fullName":"Sample
Manufacturer","manufacturerId":1,"name":"Sample","phone":null,
"postalCode":null,"state":null,"url":null}},"warrantyType":
{"id":1,"type": "WarrantyType"},"clients":[
{"id":1,"type":"Client"},
{"id":2,"type":"Client"}],"isSynchronizationDisabled"

:false,"isReservable":false,"leaseExpirationDate":null,"warrantyExpirationDat
e":
"2014-07-29T00:00:00Z","isNotesVisibleToClients":false,"isDeleted":false,
"assetCustomFields":[{"id":5,"type":"AssetCustomField","definitionId":14,
"restValue":"Somestringvalue"},
{"id":4,"type":"AssetCustomField","definitionId":15,"restValue":12345},
{"id":6,"type":"AssetCustomField","definitionId":16,"restValue":12.34}]}
```

Updating an Asset

The syntax for updating an Asset is similar to creating an Asset, except that the request should use the HTTP `PUT` method, and the Asset (`id`) is provided.

To update an Asset, make sure that all relevant entities referenced by this Asset currently exist. These entities are mainly Asset Type, Manufacturer and Model. As a result, the update process should consist of the following steps:

1. Create or find all relevant entities that you want to assign to a new Asset, for example, Asset Status, Billing Rate, Location, Clients and so on. (See Getting a list of Asset Statuses, Getting a list of Billing Rates, Getting a list of Locations, Getting a list of Clients, and so on.)
2. Find an existing Asset Type or create one if it does not exist. See Getting a list of Asset Types or Creating an Asset Type for details.
3. Find an existing Manufacturer or create one if it does not exist. See Getting a list of Manufacturers or Creating a Manufacturer for details.
4. Find an existing Model or create one if it does not exist. See Getting a list of Models or Creating a Model for details.
5. Update the Asset.

```
$ curl -X PUT -d @asset.json \
> "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Assets/1\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
```

Deleting an Asset

You can delete an Asset by submitting an HTTP `DELETE` request. If the Asset is deleted, the HTTP status code 200 OK is returned. Otherwise, an error message is returned—for example, Entity Asset with ID29 not found.

```
$ curl -X DELETE \
> "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Assets/29\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
```

Asset statuses

In the following sections, `localhost` is the hostname or IP address and port assigned to the Web Help Desk server. For example, `10.10.38.20:8443` where `8443` is the default secure port.

Getting a list of Asset Statuses

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/AssetStatuses?limit=3\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
AssetStatuses
```

Getting a specific Asset Status

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/AssetStatuses/1\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
AssetStatuses/1
```

Asset types

In the following sections, `localhost` is the hostname or IP address and port assigned to the Web Help Desk server. For example, `10.10.38.20:8443` where `8443` is the default secure port.

Modifiable Asset Type Fields

The following table lists all Asset Type fields that can be modified with create or update operations. The field listed below is mandatory.

Field name	Data type	Note
`assetType`	string	60 characters (mandatory)

Getting a list of Asset Types

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/AssetTypes?limit=3\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
```

```
AssetTypes
```

Searching for Asset Types

Asset Types, including deleted Asset Types, can be searched for by including a qualifier query parameter.

Example: All Asset Types that contain the word 'top'.

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/AssetTypes?qualifier=(assetType like '*top*')\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

AssetType
```

Searching for deleted Asset Types

Asset Types can be searched for by including a qualifier query parameter.

Example: All deleted Asset Types.

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/AssetTypes?qualifier=(deleted %3D 1)\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

AssetTypes
```

Getting a specific Asset Type

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/AssetTypes/1\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
AssetTypes/1
```

Creating an Asset Type

Asset Types are created by submitting an HTTP `POST` request containing Asset Type data in the specified format. The HTTP response will provide the Asset Type `id`, as well as the `assetType`.

Example: Create a request data file `assettype.json` with the following content:

```
{"assetType":"AssetType Test"}
```

When you submit the HTTP POST request you will receive the response as follows:

```
$ curl -X POST -d @assettype.json \
> "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/AssetTypes\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
{"id":6,"type":"AssetType","assetType":"AssetTypeTest","hasSpecificCustomFiel
ds"
:false}
```

Updating an Asset Type

The syntax for updating an Asset Type is identical to creating an Asset Type, except that the request should use the HTTP `PUT` method, and the Asset Type (`id`) must be provided:

```
$ curl -X PUT -d @assettype.json \
> "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/AssetTypes/1\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
```

Deleting an Asset Type

You can delete Asset Types by submitting an HTTP `DELETE` request. If the Asset Type is deleted, the HTTP status code 200 OK is returned. Otherwise, an error message is returned—for example, Entity AssetType with ID12 not found.

```
$ curl -X DELETE \
> "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/AssetTypes/12\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
```

Manufacturers

In the following sections, `localhost` is the hostname or IP address and port assigned to the Web Help Desk server. For example, `10.10.38.20:8443` where `8443` is the default secure port.

Modifiable Manufacturer Fields

The following table lists all Manufacturer fields that can be modified with create and update operations. The fields with darker background are mandatory:

Field name	Data type	Note
`address`	string	80 characters
`city`	string	40 characters
`country`	string	80 characters
`fax`	string	30 characters
`fullName`	string	60 characters (mandatory)
`name`	string	60 characters
`phone`	string	30 characters
`postalCode`	string	30 characters
`state`	string	30 characters
`url`	string	255 characters

Getting a list of Manufacturers

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Manufacturers?limit=3\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

Manufacturers
```

Searching for Manufacturers

You can search for Manufacturers by including a qualifier query parameter.

Example: All Manufacturers with a name like 'Dell'.

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Manufacturers?qualifier=(name like '*Dell*')\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

Manufacturers
```

Searching for deleted Manufacturers

You can search for deleted Manufacturers by including a qualifier query parameter

Example: All deleted Manufacturers.

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Manufacturers?qualifier=(deleted %3D 1)\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

Manufacturers
```

Example: All Manufacturers including deleted.

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Manufacturers?qualifier=((deleted %3D null) or (deleted %3D 0) or
(deleted %3D 1))&limit=3\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
Manufacturers
```

Getting a specific Manufacturer

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Manufacturers/1\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

Manufacturers/1
```

Creating a Manufacturer

You can create a Manufacturer by submitting an HTTP `POST` request containing Manufacturer data in the specified format. The HTTP response will provide the Manufacturer `id`, as well as the `name` and `fullName`.

Example: Create a request data file manufacturer.json with the following content.

```
{"address":null,"city":null,"country":null,"fax":null,"fullName":"Dummy full
name","name":"Dummy name","phone":null,"postalCode":null,"state":null,"url"
:null}
```

When you submit the HTTP `POST` request, you will receive the response as follows:

```
$ curl -X POST -d @manufacturer.json \
> "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Manufacturers\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
{"id":6,"type":"Manufacturer","address":null,"city":null,"country":null,"fax"
:null,
"fullName":"Dummy full name","manufacturerId":6,"name":"Dummy
name","phone":null,
"postalCode":null,"state":null,"url":null}
```

Updating a Manufacturer

The syntax for updating a Manufacturer is identical to creating a Manufacturer, except that the request should use the HTTP `PUT` method, and the Manufacturer (`id`) must be provided:

```
$ curl -X PUT -d @manufacturer.json \
> "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Manufacturers/1\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
```

Deleting a Manufacturer

You can delete a Manufacturer by submitting an HTTP `DELETE` request. If the Manufacturer is deleted, the HTTP status code 200 OK is returned. Otherwise, an error message is returned—for example, Entity Manufacturer with ID10 not found.

```
$ curl -X DELETE \
> "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Manufacturers/10\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
```

Models

In the following sections, `localhost` is the hostname or IP address and port assigned to the Web Help Desk server. For example, `10.10.38.20:8443` where `8443` is the default secure port.

Modifiable Model Fields

The following table lists all Model fields that can be modified with create or update operations. The fields with darker background are mandatory:

Field name	Data type	Note
`assetType`	Asset Type object	mandatory
`manufacturer`	Manufacturer object	mandatory
`modelName`	string	255 characters (mandatory)
`warrantyType`	Warranty Type object	No REST endpoint for Warranty Types

Getting a list of Models

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Models?limit=3\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

Models
```

Searching for Models

You can search for Models by including a qualifier query parameter.

Example: All Models with a name like 'Mac'.

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Models?qualifier=(modelName like '*Mac*')\
```

```
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
Models
```

Searching for deleted Models

You can search for deleted Models by including a qualifier query parameter.

Example: All deleted Models.

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Models?qualifier=(deleted %3D 1)\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

Models
```

Example: All Models including deleted

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Models?qualifier=((deleted %3D null) or (deleted %3D 0) or (deleted %3D
1))&limit=3\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

Models
```

Getting a specific Model

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Models/1\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"

Models/1
```

Creating a Model

You can create a Model by submitting an HTTP `POST` request containing Model data in the specified format. The HTTP response will provide:

- Model `id` and `modelName`
- Assigned `assettype`

- `manufacturer`
- `warrantyType`

To create a Model, make sure that all relevant entities referenced by this Model currently exist. These entities are mainly Asset Type and Manufacturer. As a result, the Model creation process should consist of the following steps:

1. Create the Asset Type.
2. Create the Manufacturer.
3. Create the Model.

Example: Create a request data file model.json with the following content:

```
{"modelName":"ModelName","assettype":
{"id":1,"type":"AssetType"},"manufacturer":
{"id":1,"type":"Manufacturer"},"warrantyType":
{"id":1,"type":"WarrantyType"}}
```

When you submit the HTTP `POST` request, you will receive the response as follows:

```
$ curl -X POST -d @model.json \
> "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Models\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
{"id":7,"type":"Model","modelName":"ModelName","assettype":
{"id":1,"type":"AssetType"},"manufacturer":
{"id":1,"type":"Manufacturer"},"warrantyType":
{"id":1,"type":"WarrantyType"}}
```

Updating a Model

The syntax for updating a Model is identical to creating a model, except that the request should use the HTTP `PUT` method and provide the Model (`id`).

To update a Model, make sure that all relevant entities that are referenced by this Model currently exist. These entities are mainly Asset Type and Manufacturer. As a result, the update process should consist of the following steps:

1. Locate an existing Asset Type or create one if it does not exist. See Getting a list of Asset Types or Creating an Asset Type for details.

2. Locate an existing Manufacturer or create one if it does not exist. See Getting a list of Manufacturers or Creating a Manufacturer for details.
3. Update the Model.

```
$ curl -X PUT -d @model.json \
> "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Models/1\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
```

Deleting a Model

You can delete a Model by submitting an HTTP `DELETE` request. If the Model is deleted, the HTTP status code 200 OK is returned. Otherwise, an error message is returned—for example, Entity Model with ID11 not found.

```
$ curl -X DELETE \
> "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Models/11\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
```

Companies

ⓘ In the following sections, `localhost` is the hostname or IP address and port assigned to the Web Help Desk server. For example, `10.10.38.20:8443` where `8443` is the default secure port.

Modifiable Company Fields

The following table lists all Company fields that can be modified with create and update operations. The fields with darker background are mandatory:

Field name	Data type	Note
companyName	string	Mandatory
address1	string	80 characters
address2	string	80 characters
city	string	40 characters
state	string	30 characters

Field name	Data type	Note
postalCode	string	30 characters
country	string	80 characters
phone	string	30 characters
phone2	string	30 characters
fax	string	30 characters
domainName	string	1024 characters
color	string	Seven characters (for example, #F99D1C)
locations	Location object	
techs	Tech object	

Getting a list of Companies

```
$ curl "https://localhost:8081/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Companies?limit=3\
> &apiKey=vfdRQo1nXyxMf9Spj6VbtvqNEuYJYdahuC6lP1h0"

Companies
```

Searching for Companies

You can search for Companies by including a qualifier query parameter.

Example: All Companies with a name like 'SolarWinds'.

```
$ curl "https://localhost:8081/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Companies?qualifier=(companyName like "*Solarwinds*")\
> apiKey=IrYl9j3ozjdr5FZoE0gyn6x6W6pWue0IiRrMsqF3"

Companies
```

Searching for deleted Companies

You can search for deleted Companies by including a qualifier query parameter

Example: All deleted Companies.

```
$ curl "localhost:8081/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Companies?qualifier=(deleted %3D 1)\
> &apiKey=IrYl9j3ozjdr5FZoE0gyn6x6W6pWue0IiRrMsqF3"

Companies
```

Example: All Companies including deleted.

```
$ curl "https://localhost:8081/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Companies?qualifier=(deleted %3D 1)\
> &apiKey=IrYl9j3ozjdr5FZoE0gyn6x6W6pWue0IiRrMsqF3"

Companies
```

Getting a specific Company

```
$ curl "https://localhost:8081/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Companies/1\
> ?apiKey=IrYl9j3ozjdr5FZoE0gyn6x6W6pWue0IiRrMsqF3"

Companies/1
```

Creating a Company

You can create a Company by submitting an HTTP `POST` request containing Company data in the specified format. The HTTP response will provide the Company `id`, as well as the `companyName` and `address1`.

Example: Create a request data file company.json with the following content.

```
{
"companyName":"SolarWinds",
"address1":null,
"address2":null,
"city":null,
"state":null,
"postalCode":null,
"county":null,
"phone":null,
"fax":null,
"domainName":null,
"color":"#ffffff",
"locations":null,
"techs":null
}
```

When you submit the HTTP `POST` request, you will receive the response as follows:

```
$ curl -X POST -d @company.json \
> "https://localhost:8081/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Company\
> ?apiKey=vfdRQo1nXyxMf9Spj6VbtvqNEuYJYdahuC6lP1h0"
```

Updating a Company

The syntax for updating a Manufacturer is identical to creating a Company, except that the request should use the HTTP `PUT` method, and the Company (`id`) must be provided:

```
$ curl -X PUT -d @company.json \
> "https://localhost:8081/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Company/1\
> ?apiKey=vfdRQo1nXyxMf9Spj6VbtvqNEuYJYdahuC6lP1h0"
```

Deleting a Company

You can delete a Company by submitting an HTTP `DELETE` request. If the Company is deleted, the HTTP status code 200 OK is returned. Otherwise, an error message is returned—for example, Entity Company with ID10 not found.

```
$ curl -X DELETE \
> "https://localhost:8081/helpdesk/WebObjects/Helpdesk.woa\
> /ra/Company/12\
> ?apiKey=vfdRQo1nXyxMf9Spj6VbtvqNEuYJYdahuC6lP1h0"
```

Client interface

Client authentication

Each Client request to the Web Help Desk REST API requires authentication. You can use the following authentication methods:

- Authenticating with username/email/password combination
- Authenticating with a temporary session key

ⓘ In the following sections, `localhost` is the hostname or IP address and port assigned to the Web Help Desk server. For example, `10.10.38.20:8443` where `8443` is the default secure port.

Authenticating with username/email/password combination

Using this authentication method, you must add the `login` and `password` parameters to each HTTP request. The `login` parameter value contains the username or email based on the Client Login Attribute (Setup > Clients > Options > Client Login Attribute).

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/clientInterface/Tickets\
> ?login=client&password=client"
[{"id":38,"type":"Ticket","lastUpdated":"2013-02-26T10:44:39Z",...
```

Authenticating with a session key

You can obtain a session key by providing a valid login and password to the Session resource. Using a session key can improve performance because it enables the caching of frequently-used resources. Also, session keys bring an extra level of security because they expire after 30 minutes of inactivity—for example, a stolen session key that expired will be rejected.

In the following example, a session key is obtained and used to authenticate a subsequent request for ticket details.

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/clientInterface/Session\
> ?login=client&password=client"
{"type":"Session","sessionKey":"xDBRI20YLZtFkhcroHaEGg","instanceId":4}
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa/4\
```

```
> /ra/clientInterface/Tickets\
> ?sessionKey=xDBRI20YLZtFkhcroHaEGg"
[{"id":38,"type":"Ticket","lastUpdated":"2013-02-26T10:44:39Z",...
```

Terminating the session

You can terminate the REST session by submitting an HTTP `DELETE` request with the specified `sessionKey`—the key of the active session.

```
$ curl -X DELETE \
> "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/clientInterface/Session\
> ?sessionKey=xDBRI20YLZtFkhcroHaEGg"
OK
```

Authenticating to a hosted Web Help Desk account

When authenticating to a hosted Web Help Desk account, include the hosted account ID.

```
curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/clientInterface/Session\
> ?login=client&password=client&accountId=1429"
```

Authentication failures

If a request fails to authenticate due to invalid credentials, missing credentials, or an expired session key, an HTTP 401 ("Unauthorized") status code will be returned.

Note the inclusion of the curl `-i` option in the following example in order to view the response headers.

```
$ curl -i "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/clientInterface/Tickets"
HTTP/1.0 401 Apple WebObjects
     x-webobjects-loadaverage: 1
     content-length: 24

     Authentication failure.
```

Permission failures

If a request authenticates successfully but the authenticated Client is unauthorized to perform the requested action, an HTTP 403 ("Forbidden") status code will be returned.

Mandatory fields

The following table lists the values in REST endpoints that are mandatory for `POST/PUT` (create/update) operations. They cannot be `null`, must be present when creating an object, and cannot be nullified when updating an object.

REST Endpoint	Field name	Data type
Tickets		
	`problemtype`	object

Tickets

In the following sections, `localhost` is the hostname or IP address and port assigned to the Web Help Desk server. For example, `10.10.38.20:8443` where `8443` is the default secure port.

Modifiable Ticket Fields

The following table lists all Ticket fields that can be modified with the create operation. The fields with a dark background are mandatory.

Field name	Data type	Note
`detail`	string	Long text
`problemtype`	Request Type object	mandatory
`subject`	string	255 characters

HTTP GET request parameters

The following parameters can be added to the HTTP request:

Parameter	Required	Description
`withUTC-true`	optional	The following properties will contain the date in UTC format: • `closeDateUtc` • `lastUpdatedUtc`, • `displayDueDateUtc` • `notes.dateUtc`

Getting a list of Tickets for a Client

The list reflects the access permissions of the authenticated Client. Tickets are returned in reverse order of the last update date (`lastUpdated`).

In the following example, we request the first page of the authenticated user's Tickets using a page size (`limit`) of three Tickets:

```
$ curl "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/clientInterface/Tickets?page=1&limit=3\
> &apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
[{"id":38,"type":"Ticket","lastUpdated":"2013-02-
26T10:44:39Z","shortSubject":
"test","shortDetail":"test","displayClient":"DemoClient","prettyLastUpdated":
"1 yearago","latestNote":null},
{"id":37,"type":"Ticket","lastUpdated":"2013-02-26T10:41:23Z","shortSubject":
"Login not working","shortDetail":"please provide the login id for susan
smith.",
"displayClient":"DemoClient","prettyLastUpdated":"1 year
ago","latestNote":null},
{"id":36,"type":"Ticket","lastUpdated":"2013-02-26T10:16:05Z","shortSubject":
"Test","shortDetail":"Test","displayClient":"DemoClient","prettyLastUpdated":
"1 year ago","latestNote":null}]
```

Searching for Tickets

You can search for Tickets by including a qualifier query parameter.

For more information, see Searching for Tickets.

Creating a Ticket

You can create a Ticket by submitting an HTTP `POST` request containing ticket data in the specified format. The HTTP response provides the ticket `id`, as well as the `shortSubject`, `shortDetail`, and `lastUpdated` date.

If `sendEmail` is true, a confirmation email will be sent to the parties assigned to receive email.

> **Custom Field values**
> Custom Fields can be included in the request data. See Custom Field Definitions for details.

Example: Create a request data file ticket.json with the following content.

```
{"subject": "Need more memory","detail":"My computer is running really slow.
I think it's because I need more memory.","problemtype":
{"type":"ProblemType",
"id":6},"customField_12":"Version12.2.0",
"sendEmail":true}
```

When you submit the HTTP POST request you will get a response like the following:

```
$ curl -X POST -d @ticket.json \
> "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/clientInterface/Tickets\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
{"id":39,"type":"Ticket","lastUpdated":"2014-04-25T11:34:07Z","shortSubject":
"Need more memory","shortDetail":"My computer is running really slow. I think
it's because I need more memory.","displayClient":"Approver One",
"prettyLastUpdated":"moments ago","latestNote":null}
```

Adding a Client Note to a Ticket

The following table lists all Ticket Client Note fields that can be set with a create operation:

Field name	Data type	Note
`jobticket`	Ticket object	
`noteText`	string	Long text

Example: Create a request data file `client_note.json` with the following content:

```
{"noteText":"Reinstalled the printer driver. Seems to be working fine now.",
"jobticket":{"type":"JobTicket","id":39}}
```

When you submit the HTTP `POST` request you will get the response as follows:

```
$ curl -X POST -d @client_note.json \
> "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra/clientInterface/ClientNotes\
> ?apiKey=v32lXMFAi7dl3zGrtETArXqKVF8svfAfXZpIwC0P"
{"id":26,"type":"ClientNote","date":"2014-04-28T14:48:35Z","noteText":
"Reinstalled the printer driver. Seems to be working fine now.","jobticket":
{"id":39,"type":"Ticket"}}
```

Adding an attachment to a Ticket or a Ticket Client Note

If the user is logged in to the Web Help Desk (GUI) or has a valid REST session, the application accepts "/attachment/upload**" REST calls.

The following table lists all HTTP request parameters that should be provided:

Parameter name	Data type	Parameter value	Note

`type`	string	One of the following:
		`jobTicket`
		`clientNote`
			The entity type.
			The attachment will be assigned to this type of entity.

`entityId`	integer	The entity ID	The attachment will be assigned to the entity with the specified ID
`returnFields`	string	For example, "id,uploadDate"	Optional parameter

The following steps show how to add an attachment to the Ticket or the Ticket Client Note:

1. Create a REST session for a user. See Authenticating with a session key for details.
2. Call the "/attachment/upload**" REST with the Cookie HTTP header that contains: Java session ID and WebObjects session ID from the previous call. (See the example below.)
3. Terminate the session. See Terminating the session for details.

Example: Create a HTTP request with the following content (add_attachment.raw):

```
POST
/helpdesk/attachment/upload?type=jobTicket&entityId=40&returnFields=id,
uploadDate HTTP/1.1
Cookie: JSESSIONID=BA1B63AA2DD9EBBB62B7A20E37377DED;
wosid=EoPFUriceH4t4jn5HiXiqg
Content-Type: multipart/form-data; boundary====1400591996857===
Cache-Control: no-cache
Pragma: no-cache
User-Agent: Java/1.7.0_55
Host: 127.0.0.1:8081
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive
Content-Length: 2143
--====1400591996857===
Content-Disposition: form-data; name="fileUpload"; filename="my.ini"
Content-Type: null
Content-Transfer-Encoding: binary
# MySQL Server Instance Configuration File
# ------------------------------------------------------------------
... the rest of the file...
--====1400591996857===--
```

When you submit the HTTP `POST` request, you will receive the following response:

```
$ curl -X POST -d @add_attachment.raw \
> "https://localhost/helpdesk/WebObjects/Helpdesk.woa\
> /ra//helpdesk/attachment/upload?type=jobTicket&entityId=40&returnFields=id,
uploadDate"
{"id":11,"uploadDate":"2014-04-15 14:27:51.432+02"}
```