


# REST - Search

---

The EWSearch REST operation:

- Runs a saved search or an ad hoc query on the specified table, implementing a search-like operation of the REST interface.
- Accepts the URL with parameters as per general URL conventions which can be viewed in the [REST Interface](#). The name of the saved search or an ad hoc query may be included in the call. This operation allows retrieval of results page by page (pagination) using "page" and "limit" parameters.
- Supported Content-Type : application/x-www-form-urlencoded
- Returns the number of records found, and encoded sets of field values - one for each record found. If pagination is being used, the number of records is in the context of the current page and the specified page size.
- Each set will include the ID and the fields used by the table and its subtables to define record ownership, such as `requester_login` in the examples below. Additional fields can be added to the list by specifying a series of 'field' parameters with logical field names.

 All parameters and names must be properly URL-encoded.

## Ad Hoc Searches

In opposition to saved searches, ad hoc searches are limited to run-time queries using the 'query' clause. An example of such a REST query is shown below using the conditions `&query=summary~='test' %26%26 priority=3`:

```
http://localhost:8080/ewws  
/EWSearch?$KB=Demo&$table=case&$login=admin&$password=qwerty&$lang=en&query=summary~=  
'test' %26%26 priority=3
```

## Pagination

---

- Page numbers start with 0 (zero). A page size limit value of zero indicates "all records" and so all records are returned on page 0 when limit 0 is specified; otherwise with a non-zero page number an empty result is returned, meaning no records.
- When a page is not found, an empty result is returned.

- A call using pagination always returns a page worth of data. However, to truly take advantage of pagination all other parameters must remain the same. If the table, fields, saved search, query or limit on a subsequent call is different from the previous one, the underlying query is automatically rebuilt and re-run.
- As such only one "open" query is allowed per client session. If the client requires multiple queries to be iterated in parallel, the client code should create multiple sessions using the same login credentials.
- This method doesn't support multi-threading – the client is responsible for restricting access to a single thread; i.e. one client thread = one session = one open query.
- The query will remain "open" until the session is closed, that is an explicit logout is performed by the client or session timeout occurs, or the application server discards the underlying low-level objects as a result of resource management. In this case the query will be rebuilt and re-run automatically on the next call.
- If the query is rebuilt and re-run, the result of the next call may not be fully consistent with the results of the previous call, as the underlying data may have changed – for example, a record was deleted, or the sort order for the search in question was changed. Therefore, the dataset may appear to have "holes" and/or the logical page boundaries may shift when iterating the query page by page.
- At this time the REST interface creates a new session and performs an explicit logout for each call. As such, though pagination is available, the query will always be rebuilt and rerun. The ability to issue multiple REST calls within a single session, similar to the SOAP interface, is in development.

# Examples of REST Searches

## Example 1

Assume an instance of Agiloft is available on localhost, port 8080 and the knowledgebase is called 'Demo'. List the Service Request records that correspond to the saved search 'C: Status is Closed'. Additionally, only return those that have High priority.


The following request is issued:

```
http://localhost:8080/ewws  
/EWSearch?$KB=Demo&$login=admin&$password=qwerty&$table=helpdesk_case&$lang=en&
```

If there are no records found, the following result will be returned:

```
EWREST_id_length = '0';
```

The following result will be returned in the case of four records being found.

 If no return fields are specified in the request, it will return the ID, type fields and requester login field, based on the [record ownership](#) defined for the Service Request table, based on the permissions definition for the Service Request table.

```
EWREST_length = '4';  
EWREST_login_0='ewsystem';  
EWREST_type_0='helpdesk_case';  
EWREST_id_0='318';  
EWREST_login_1='ewsystem';  
EWREST_type_1='helpdesk_case';  
EWREST_id_1='151';  
EWREST_login_2='ewsystem';  
EWREST_type_2='helpdesk_case';  
EWREST_id_2='146';  
EWREST_login_3='internal';  
EWREST_type_3='helpdesk_case';  
EWREST_id_3='145';
```

## Example 2

Assume an instance of Agiloft is available on localhost, port 8080 and Demo KB. Now we want to retrieve particular fields.

The following request is issued:

```
http://localhost:8080/ewws  
/EWSearch?$KB=Demo&$login=admin&$password=qwerty&$table=helpdesk_case&$lang=en&
```

If there are no records found, the following result will be returned:

```
EWREST_id_length = '0';
```

The following result will be returned in the case of four records being found:

```
EWREST_length = '4';  
EWREST_summary_0='Here is a new service request with some tasks';  
EWREST_priority_0='High';  
EWREST_summary_1='New Employee Setup for Patricia Smith';  
EWREST_priority_1='High';  
EWREST_summary_2='Upgrading Our Software';  
EWREST_priority_2='High';  
EWREST_summary_3='Need New Wireless Card for Laptop';  
EWREST_priority_3='High';
```

### Example 3

Assume an instance of Agiloft is available on localhost, port 8080 and Demo KB. Now we want to retrieve particular fields in addition to the default ones as per Example 2.

We are retrieving data in pages 2 records at a time and are interested in the 2nd page only. Page numbers counts from 0.

The following request is issued:

```
http://localhost:8080/ewws  
/EWSearch?$KB=Demo&$login=admin&$password=qwerty&$table=helpdesk_case&$lang=en&
```

If there are no records found, the following result will be returned:

```
EWREST_id_length = '0';
```

The following result is returned in the case of four records found in total, showing last two ones:

```
EWREST_length = '2';  
EWREST_summary_0='Upgrading Our Software';  
EWREST_priority_0='High';  
EWREST_summary_1='Need New Wireless Card for Laptop';  
EWREST_priority_1='High';
```

Here is an example for a JavaScript-based client that invokes the REST interface via AJAX:

```
function xmlhttpGet (strURL) {
    var xmlhttpReq=false;
    var self=this;
    // Mozilla/Safari
    if (window.XMLHttpRequest) {
        try {
            netscape.security.PrivilegeManager.
enablePrivilege("UniversalBrowserRead");
        } catch (e) {
            alert("Permission UniversalBrowserRead denied.");
        }
        self.xmlHttpRequest=new XMLHttpRequest();
    } // IE
    else if (window.ActiveXObject) {
        self.xmlHttpRequest=new ActiveXObject("Microsoft.xmlHTTP");
    }
    self.xmlHttpRequest.open('GET', strURL, true);
    self.xmlHttpRequest.onreadystatechange=requestComplete;
    self.xmlHttpRequest.send(null);
}
function requestComplete() {
    if (xmlHttpRequest.readyState==4||xmlHttpRequest.readyState=="complete") {
        eval (self.xmlHttpRequest.responseText);
        alert ("Id of new ticket"+EWREST_id);
    }
}
function main() {
    xmlhttpGet('
http://localhost:8080/ewws
/EWSearch?$KB=Demo&$login=admin&$password=qwerty&$table=helpdesk_case&$lang=en&search
```