

EWSearchTableWithQueryPaginated

eExecutes a preconfigured named Saved Search against the specified table and/or an *ad hoc* query and returns and array of record data for the records that match the criteria. This operation variant allows pagination - iteration through results page by page.

Syntax

```
EWWSBaseUserObject[] os = ew.EWSearchTableWithQueryPaginated(String sessionId,
String tableName, String[] fieldNames, String searchName, String query, int page,
int limit);
```

Usage

Use EWSearchTableWithQueryPaginated call to search for records in the specified table based on a Saved Search pre-configured in the GUI and/or to filter the records with an ad-hoc query. Specify page and limit parameters to retrieve data page by page.

Rules and Guidelines

When querying records, consider the following rules and guidelines:

- The username that was used to obtain the specified session token must have sufficient access rights to read individual records within the specified table. Please verify specific permissions via **Setup > Access > Manage Groups > [Edit Group] > Table > [Select Table] > Permissions**.
- Agiloft allows specifying fine-grained access permissions on the field level. The username that was used to obtain the specified session token must have sufficient access rights to be able to read field content. Please verify specific permissions via **Setup > Access > Manage Groups > [Select Group to Edit] > Table > [Select Table] > Field Permissions**.
- This call does not return records that have been deleted.
- This method never returns null. In the case when no records are found an empty array is returned.

- Special note on memory management: As WS integration implies pass-by-value semantics, the memory allocated for the resulting array will be released once the data is sent to the client and the server-side JVM's garbage collector considers it eligible for discarding.
- If the client-side environment is the one with a garbage collector, the memory used by client-side array will be cleared once the client-side garbage collector considers it eligible.
- When the client environment uses explicit memory management, the client is responsible for freeing up the used memory explicitly.
- When using EWSearchTableWithQueryPaginated method, only the fields explicitly listed in the call are read. Within the values returned for the fields requested explicitly, a null value - nillable="true" - means the actual null value was retrieved. However, the rest of the fields - those not listed in the fieldNames array - will also appear on the wire as nillable="true" elements due to limitations of the underlying Web Services stack.
- To read all fields, use "*" string constant as the only element in the fieldNames array.
- The main difference from using the EWSelectAndRead method, which uses an SQL where clause, is that ad-hoc queries operate on a higher level, can use logical field names, are capable of recognizing choice values and high-level relationships between table fields, and can use advanced and time-based criteria.
- If the query doesn't parse according to the grammar, an attempt is made to parse the parameter value as a sequence of identifiers using a different grammar. If both fail, the parameter value is treated as a Full-Text Search query.
- The ad-hoc query grammar is described at the end of this section.
- Page numbers start with 0 (zero). A limit (page size) 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 (no records).
- When a page is not found, 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 - 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.

Unsupported Types of Fields

Embedded search results are not supported by the SOAP interface.

Steps for Searching Records with a Saved Search and/or Ad-hoc Query

1. Optionally create a Saved Search in the GUI.
2. Perform the call using the name of the search and additionally filter the results with an ad-hoc query or use the ad-hoc query without the search.
3. Handle the results, specifically the situations where there are no elements, one element, or more than one element in the returned array.
4. Iterate to the next page as required.

Example Task

In MyKB knowledgebase, as user A, find all cases assigned to the user used to login with low priority. Return the first 40 summaries as two String arrays, up to 20 elements each.

Completion of the task is performed by the following steps:

1. Login to MyKB with "A" and "password" and English as the local language.
2. Search for cases using My Assigned search, additionally filtering by low priority.
3. Logout

Sample Code - Java

```
public String[] search() throws Exception {
    EWServiceAPI binding = new EWServiceAPIServiceLocator().getDemo();
    try {
        String sessionId = binding.EWLogin("MyKB", "A", "password", "en");
        String[][] result = new String[2][];
        EWWSBaseUserObject[] records1 = binding.EWSearchTableWithQueryPaginated
```

```

(sessionId, "case",
    new String[] {"summary"}, "My Assigned", "Priority=Low", 0, 20);
result[0] = new String[records1.length];
for (int i=0; i<records1.length; i++) {
    result[0][i] = records1[i].getSummary();
}
EWSBaseUserObject[] records2 = binding.EWSearchTableWithQueryPaginated
(sessionId, "case",
    new String[] {"summary"}, "My Assigned", "priority=Low", 1, 20);
result[1] = new String[records2.length];
for (int j=0; j<records2.length; j++) {
    result[0][j] = records2[j].getSummary();
}
return result;
} finally {
    binding.EWLogout(sessionId);
}
}

```

You can generate sample Web Services code for any table by selecting **Setup > Table > [Select Table to Edit] > API > Download Sample**.

Arguments

Name	Type	Description
sessionId	String	Session token
tableName	String	The name of the table where the query has to be performed.
fieldNames	String array	The list of fields to read
searchName	String	The optional name of the Saved Search to run
query	String	The ad-hoc query
page	int	The page number
limit	int	The records per page limit (page size)

Response

An array of the records as descendants of EWSBaseUserObject - a complex structure described in WSDL.

Faults

- EWSessionException** - client not logged in or session has expired; client should re-login.
- EWPermissionException** - user used to create the session lacks sufficient privileges to run the query.
- EWWrongDataException** - client has supplied wrong data.
- EWOperationException** - the operation has been blocked by an Agiloft function, e.g. table-level lock.
- EWIntegrityException** - specified table cannot be found or its primary key cannot be identified.
- EWUnexpectedException** - an unexpected exception has happened; the admin user should report this for investigation.

Informal Grammar Description for Ad-hoc Queries

Field names are usually column labels as seen in the UI. However, DB and User column names are accepted too. Both field names and values may be surrounded by single quotes ('). If they contain spaces or some weird characters then quoting is mandatory. For example:

Example	Result
Priority=Low	OK
'Priority'='Low'	OK
Bug Priority=Low	Invalid
'Bug Priority'=Low	OK
'Bug Priority'=Very Low	Invalid
'Bug Priority'='Very Low'	OK

Simple criteria

Simple criteria has the form of

```
<field name><operator><value>
```

where operator is one of:

Operator	Definition
=	equals
!=	not equals
~=	contains
!~=	doesn't contain
>=	greater or equals
<=	lesser or equals
>	greater
<	lesser
<<	included by
!<<	not included by

The (not) included by operator expects a comma-separated list of values without spaces at the right-hand side and checks if field value is included into this list. In other words `Priority << High,Low` is a short-hand for `Priority=High || Priority=Low`, where `||` is OR, as described below.

Logical criteria

Allows to combine other criterias using AND and OR operators. '&&' is AND, '||' is OR.

Operator precedence

Expression is evaluated from left to right, braces may be used for grouping. For example '`A && B || C`' means '`(A && B) || C`'.

Time-based criteria

Allows to set relative date constraints. The form is `<field name><operator><mode><value>`, where operator is one of `=,!=,<,>,<=,>=`, mode is either `'-'`, which means 'old', `'+'`, which means 'in the future' or `'#'`, which means 'absolute'. 'value' is an integer followed by a single character:

--	--

m	minute
h	hour
w	week
M	month
y	year

Examples:

Date<-1y	'Date' is less than one year old
Date>=+10m	'Date' is greater or equal than 10 minutes in the future
Duration=#2h	'Duration' is exactly two hours

Currently, more complex expressions like 'two years, one month and three hours' are not supported.

Advanced criteria

Advanced criteria has the form of <field name>:<from>-><to> and means 'field field name has changed from *from* to *to*'. Either from or to but not both at the same time may hold '?' meaning 'any value'.

This criteria searches through record history, and thus the history column must exist and must track changes to the specified field. All simple and time-based criterias have implicit 'now' flag set, which means that they will match the current record state, not the state when advanced criteria has been satisfied. In the other words, if we have a record with the following modification history, with the bottom state being the most recent:

State=Open, Priority=Low State=Closed, Priority=Low State=Closed, Priority=High

Then 'State:Open->Closed && Priority=Low' will not find it, but 'State:Open->Closed && Priority=High' will.

More examples

- Status=Open && ('Assigned To' = john || 'Assigned To' = jane)
- (Priority>High || Summary ~= Urgent) && State:Closed->Reopened
- OS=Windows,Linux && 'Modification Date' < -1y