REST Interface

Agiloft supports REST-style invocations that correspond to CRUD operations: Create, Read, Update, Delete. Additionally limited SQL select functionality is supported via Select, and saved search and ad hoc queries via Search. Attachments can be managed via via the Attach, RemoveAttached, and RetrieveAttached calls.

This allows you to automate certain actions from external agents such as browser forms, JavaScript/AJAX.

For CRUD operations two invocation styles are supported:

- Pure REST, where CRUD operations map directly to HTTP methods.
- A fallback GET/POST method that can be used instead, as not all user-agents may support all HTTP methods.
- Select is only available via GET/POST.

Operation	/ewws/REST/...	GET/POST	Returns
Create	POST	/ewws/EWCreate	ID of the newly created record
Read	GET	/ewws/EWRead	encoded record information
Update	PUT	/ewws/EWUpdate	encoded record information after update
Delete	DELETE	/ewws/EWDelete	does not return anything
Select		/ewws/EWSelect	a list of record identifiers and a length of that list

URL Conventions

The following general conventions apply to how the URL is constructed:

For pure REST, where the /{id}part is omitted for Create (POST), but used for all others:

```
/ewws/REST/{kbName}/{table}[/{id}]?$login={login}&password={password}&lang={lang}
&...
```

For the fallback Get/Post interface:

```
/ewws/{operation}?$KB={kbName}&$table={table}&$login={login}&password={password}
&lang={lang}&...
```

The parameters of the POST request can be inserted into the body of the request to conceal the user credentials.

The body (for POST and PUT) or the rest of the URL string should contain the parameter name/value pairs, as per operation specification.

Example

```
GET {server name}/ewws/REST/Demo/Company
/123?$login=user&$password=123&$lang=en&id=123
```

Or

```
POST {server name}/ewws
/EWRead?$KB=Demo&$table=Company&$id=123&$login=user&$password=123&$lang=en&id=1
```

Notes

- Both knowledgebase names and table names are case sensitive when using the REST interface.
- To find the correct case for your table, go to **Setup > Tables**, select your table, click Edit, and look for the Logical Table Name. This is the name and capitalization you should use when accessing that particular table.

Return Values

Return values are returned in an encoded form suitable for the JavaScript `eval()` operation to be applied. Extended characters (like ö) are returned in a UTF-encoded format. After this, the values can be accessed from local variables.

To avoid interfering with variables that may already exist in the client script or document, all table column names in the variables that result from the `eval()` call are prefixed with EWREST_. As such, what is returned is escaped using JavaScript rules. The content type of the field is irrelevant for the escaping.

```
EWREST_company_name='Agiloft';
EWREST__1794_full_name=' agiloft.com Admin';
EWREST_website_url=' http://www.agiloft.com'
EWREST_date_updated=' 21 8 06 15:18:43 PM';
EWREST_id=' 21';
```

⚠ You may want to consider using a JSON decorator to receive a JSON formatted stream instead, since JSON has more readily available parsers. Here is its syntax in a REST call:

```
https://<hostname>/ewws/EWRead/.
json?$KB=KB&$table=<table>&$login=admin&$password=<pwd>$lang=en&id=<id>
```

In this case the return result would look like:

```
{"success":true,"message":"","result":{...,"company_name":"Agiloft","
_1794_full_name":"Agiloft System","id":21}}
```

HTTP Status Codes

- 200: successful operation
- 400: wrong data specified in the request
- 403: operation not permitted with specified credentials
- 408: request timeout, user may attempt to retry the request
- 409: operation cannot be performed, usually means one of the functionalities has blocked it
- 500: any other problem

Delays

Each call via the REST interface has a delay inserted after the operation has completed. The delay is set to one second by default and is configurable via the global variable Web Services Delay (WSDelay).

- An operation on a record may invoke rules and other functions, which need to be allowed enough time and resources to complete.
- Additionally, the delay is needed because client applications could mistakenly used web services, resulting in a flood of requests.

Data Encoding

The following conventions are in place to encode aspects of a typical Agiloft knowledgebase:

Simple fields

Simple fields can be filled directly by setting the value for them.

```
... &first_name=John&last_name=Doe&...
```

Choice fields

Choice fields are encoded directly with their text values as seen in the GUI.

```
...&country=USA&...
```

⚠ For ad hoc queries, in the Select call choice values should be addressed via the ID values obtained from GetChoiceLineId.

Multi-choice fields

Multi-choice fields are encoded as multiple key/value pairs.

```
... &contactMethod=phone&contactMethod=email&...
```

Date, date-time and time fields

Date, date-time and time fields can be encoded with any of 3,275 formats currently supported. The system evaluates the possible formats sequentially and stops when parsing in one of the formats succeeds.

Please refer to the following document to see the list of supported date-time formats: datetime.txt

Elapsed time fields

can be encoded as "days:hours:minutes:seconds" e.g "0:1:35:15"6

Linked field relationships

If the linked field allows independent values these can be simply assigned to the columns in the main table:

```
...&company_name=Agiloft&...
```

To create a link based on the values of imported columns, you can use Query By Example, expressed with a colon ':' qualifier.

Example values have to be provided for one or more of the imported columns in the following way:

```
...&company_name=:Agiloft&...
```

Or

```
...&company_name=Company:Agiloft&...
```

where donor table name is required if the link type is "single field from multiple tables", but may be omitted in the case of a single donor table.

If the value contains the colon ':' symbol or the question mark '?' symbols, they have to be escaped with backward-slash "\" in the following way:

```
...&company_url=Company:http\://www.example.com&...
```

For more complex queries, possibly including sub-selects, aggregative functions and columns from the donor table that are not imported into the target, you can use a SQL query, expressed with a question mark '?' qualifier. The "where" clause follows the qualifier for the query that will be run against the donor table. The columns in the query have to be referred with their database names rather then logical ones.

Example

To look for Employee records where the Company currency is EUR, and not in the linked set, use the following search:

```
http://localhost:8080/ewws
/EWSearch?$login=admin&$password=qwerty&$lang=en&$table=contacts.
employees&$KB=Demo&query=_1576_company_name0=Company?currency like 'EUR'
```

Multiple Values for the Linked Field

These are encoded as multiple key/value pairs:

```
...&company_name=Company:Agiloft& company_name=Company:SaaSWizard&...
```

File and Image Fields (Attached Files)

The REST interface accepts files in POST requests when used with **enctype="multipart/form-data"** encoding.

The name of the form field should match the name of the file or image field. Additionally a field *fieldName$overwrite* can be specified with any value to instruct the REST interface to override the current data in the file or image field, rather than add.

Application clients can use REST - Attach operation instead with PUT HTTP method.

Decorators

REST calls allow use of three decorators:

- Asynchronous decorator
    - Can be applied to EWCreate, EWUpdate and EWDelete calls to do "fire-and-forget" type of call.
    - Should be used if the results normally returned by an operation are not important to the caller e.g. a scenario when a lot of records have to be created in the backend.

    Use

    ```
    /ewws/async/EWCreate?...
    or
    /ewws/EWCreate/.async?...
    ```

- Redirect decorator
    - Can be applied to all calls to have a HTTP redirect issued depending on the success of the operation. Useful the scenarios of integration with web sites. Please note the page to which redirect is performed will NOT receive any return data.
    - Calls require two parameters to be specified: *$exiturl* and *$errorurl* - for redirecting in case of successful operation and in case of error respectively. Both parameters should be absolute URLs and URL encoded if necessary.

Use

```
/ewws/redirect/EWCreate?...&$exiturl=http%3A%2F%2Fwww.google.
com&$errorurl=http%3A%2F%2Fwww.google.com%2F404
or
/ewws/EWCreate/.redirect?...&$exiturl=http%3A%2F%2Fwww.google.
com&$errorurl=http%3A%2F%2Fwww.google.com%2F404
```

- JSON decorator produces a JSON formatted stream.

  Use

  ```
  /ewws/EWRead/.json?...
  ```

Please note that decorators can be chained. In the case of chaining they are applied from left to right.

Use

```
/ewws/redirect/EWCreate/.async?...&$exiturl=http%3A%2F%2Fwww.google.
com&$errorurl=http%3A%2F%2Fwww.google.com%2F404
/ewws/async/EWCreate/.redirect?...&$exiturl=http%3A%2F%2Fwww.google.
com&$errorurl=http%3A%2F%2Fwww.google.com%2F404
/ewws/redirect/async/EWCreate?...&$exiturl=http%3A%2F%2Fwww.google.
com&$errorurl=http%3A%2F%2Fwww.google.com%2F404
/ewws/async/redirect/EWCreate?...&$exiturl=http%3A%2F%2Fwww.google.
com&$errorurl=http%3A%2F%2Fwww.google.com%2F404
```