curl | JavaScript

INTRODUCTION

The AgriSales CRM API provides an easy way to integrate your AgriSales CRM data in Airtable with any external system. The API closely follows REST semantics, uses JSON to encode objects, and relies on standard HTTP codes to signal operation outcomes.

The API documentation below is specifically generated for your base. We recommend that you use the graphical Airtable interface to add a few records of example data for each table. These records will be displayed in the documentation examples generated below.

To view documentation for all available endpoints, as well as documentation that has not been generated specific to your base, please visit here.

The ID of this base is `appPVwvxpSYRf5d0K`.

Please note: if you make changes to a field (column) name or type, the API interface for those fields will change correspondingly. Therefore, please make sure to update your API implementation accordingly whenever you make changes to your Airtable schema from the graphical interface.

Official API client:

- JavaScript: airtable.js (Node.js + browser)

Community-built API clients:

- Ruby: airrecord
- .NET: airtable.net
- Python 3: pyairtable
- Python 2/3: airtable.py

METADATA

This API gives you the ability to list all of your bases, tables, fields, and views. For more, see the API reference documentation for List bases and Get base schema.

RATE LIMITS

The API is limited to 5 requests per second per base. If you exceed this rate, you will receive a 429 status code and will need to wait 30 seconds before subsequent requests will succeed.

The official JavaScript client has built-in retry logic.

If you anticipate a higher read volume, we recommend using a caching proxy. This rate limit is the same for all plans and increased limits are not currently available.

AUTHENTICATION

Airtable uses simple token-based authentication. For personal development, we recommend using personal access tokens, which can be created at /create/tokens. To learn more about other authentication methods like OAuth, please visit our developer documentation.

Using Airtable.js in the browser is strongly discouraged, since that will expose your API key to everyone who has access to that web page.

To configure Airtable.js either store your secret API token (e.g. personal access token) in the `AIRTABLE_API_KEY` environment variable or pass it directly to the client library. Note that environment variables are not available when using

EXAMPLE USING ENVIRONMENT VARIABLE

```
# Shell:
$ export AIRTABLE_API_KEY=YOUR_SECRET_API_TOKEN

# Node:
const base = require('airtable').base('appPVwvxpSYRf5d0K');
```

EXAMPLE USING CUSTOM CONFIGURATION

```
var Airtable = require('airtable');
Airtable.configure({
    endpointUrl: 'https://api.airtable.com',
```

Airtable.js in the browser.

All API requests must be authenticated and made over HTTPS.

```
    apiKey: 'YOUR_SECRET_API_TOKEN'
});
var base = Airtable.base('appPVwvxpSYRf5d0K');
```

USERS TABLE

The id for Users is `tblmK0icaHeN7BPeH`. Table ids and table names can be used interchangeably in API requests. Using table ids means table name changes do not require modifications to your API request.

Fields

Each record in the Users table contains the following fields:

Field names and field ids can be used interchangeably. Using field ids means field name changes do not require modifications to your API request. We recommend using field ids over field names where possible, to reduce modifications to your API request if the user changes the field name later.

FIELD NAME	FIELD ID	TYPE	DESCRIPTION
Name	`fld27vZIuvQAGrzb2`	Text	`string` A single line of text.
User ID	`fldhHHpiefSWmVTYG`	Number	`number` An integer (whole number, e.g. 1, 32, 99). This field allows negative and positive numbers.
Email	`fldsZoOjOyMYs2S1F`	Text	`string` A single line of text.
Role	`fldaBIMPfp5G6Gnki`	Single select	`string` Selected option name. When creating or updating records, if the choice string does not exactly match an existing option, the request will fail with an `INVALID_MULTIPLE_CHOICE_OPTIONS` error unless the `typecast` parameter is enabled. If `typecast` is enabled, a new choice will be created if one does not exactly match.
Status	`fldzaQzlHOdEdtaxL`	Single select	`string` Selected option name. When creating or updating records, if the choice string does not exactly match an existing option, the request will fail with an `INVALID_MULTIPLE_CHOICE_OPTIONS` error unless the `typecast` parameter is enabled. If `typecast` is enabled, a new choice will be created if one does not exactly match.

EXAMPLE VALUES

```
"Alice Mugenzi"
"Jean Uwimana"
"Claudine Mukamana"
"Eric Nshimiyimana"
"Marie Uwase"
```

EXAMPLE VALUES

```
101
102
103
104
105
```

EXAMPLE VALUES

```
"alice.mugenzi@email2201.com"
"jean.uwimana@email42f4.com"
"claudine.mukamana@email4c41.com"
"eric.nshimiyimana@emaile11d.com"
"marie.uwase@email330a.com"
```

POSSIBLE VALUES

```
[
    "Admin",
    "Agent",
    "Customer"
]
```

POSSIBLE VALUES

```
[
    "Active",
    "Inactive"
]
```

Last Login	fldzZncQsi0EiFV7M	Date	string (ISO 8601 formatted date) UTC date, e.g. "2014-09-05".	EXAMPLE VALUES "2025-08-17" "2025-08-18" "2025-08-15" "2025-07-30" "2025-08-16"
Profile	fldrjXrFYOzZJr9Y3	Link to another record	array of record IDs (strings) Array of linked records IDs from the Agent Profiles table.	EXAMPLE VALUE ["rec8116cdd76088af", "rec245db9343f55e8", "rec4f3bade67ff565"]
Customer Profile	fldutswIgSEde7LWV	Link to another record	array of record IDs (strings) Array of linked records IDs from the Customer Profiles table.	EXAMPLE VALUE ["rec8116cdd76088af", "rec245db9343f55e8", "rec4f3bade67ff565"]
Full Name	fld9hubnoYqgNNdBK	Formula	number, string, array of numbers or strings Computed value: Name.	EXAMPLE VALUES "\"Alice Mugenzi\"" "\"Jean Uwimana\"" "\"Claudine Mukamana\"" "\"Eric Nshimiyimana\"" "\"Marie Uwase\""
Days Since Last Login	fldAfyRkWhCZ1RuaV	Formula	number, string, array of numbers or strings Computed value: DATETIME_DIFF(TODAY(), Last Login, 'days').	EXAMPLE VALUES 2 1 4 20 3
Agent Profile Region	fldyNmbHllVso2o1L	Lookup	array of numbers, strings, booleans, or objects Array of Region fields in linked Agent Profiles records.	EXAMPLE VALUES ["North"] ["South"] ["East"] ["West"] ["North"]
Customer District	fldqn071BRlDQ5YxU	Lookup	array of numbers, strings, booleans, or objects Array of District fields in linked Customer Profiles records.	EXAMPLE VALUES ["Kigali"] ["Southern"] ["Eastern"] ["Northern"] ["Western"]
User Summary (AI)	fldiALddU5j7pwzpn			

List Users records

To list records in Users, use the `select` method.

`select` returns a query object. To fetch the records matching that query, use the `eachPage` or `firstPage` method of the query object.

Returned records do not include any fields with "empty" values, e.g. `""`, `[]`, or

CODE

```
var Airtable = require('airtable');
var base = new Airtable({apiKey:
'YOUR_SECRET_API_TOKEN'}).base('appPVwvxpSYRf5d0K');

base('Users').select({
    // Selecting the first 3 records in Grid view:
```

`false`.

Note: Airtable's API only accepts request with a URL shorter than 16,000 characters. Encoded formulas may cause your requests to exceed this limit. To fix this issue you can instead make a POST request to `/v0/{baseId}/{tableIdOrName}/listRecords` while passing the parameters within the body of the request instead of the query parameters. See our support article on this for more information.

You can use the following parameters to filter, sort, and format the results:

fields
`array of strings`
optional

Only data for fields whose names are in this list will be included in the result. If you don't need every field, you can use this parameter to reduce the amount of data transferred.

For example, to only return data from `Name` and `User ID`, pass in:

```
fields: ["Name", "User ID"]
```

You can also perform the same action with field ids (they can be found in the fields section):

```
fields: ["fld27vZIuvQAGrzb2", "fldhHHpiefSWmV7
```

filterByFormula
`string`
optional

A formula used to filter records. The formula will be evaluated for each record, and if the result is not `0`, `false`, `""`, `NaN`, `[]`, or `#Error!` the record will be included in the response. We recommend testing your formula in the Formula field UI before using it in your API request.

If combined with the `view` parameter, only records in that view which satisfy the formula will be returned.

The formula must be encoded first before passing it as a value. You can use this tool to not only encode the formula but also create the entire url you need. For example, to only include records where `Name` isn't empty, pass in `NOT({Name} = '')` as a parameter like this:

```
filterByFormula: "NOT({Name} = '')"
```

Note: Airtable's API only accepts request with a URL shorter than 16,000 characters. Encoded formulas may cause your requests to exceed this limit. To fix this issue you can instead make a POST request to `/v0/{baseId}/{tableIdOrName}/listRecords` while passing the parameters within the body of the request instead of the query parameters. See our support article on this for more information.

maxRecords
`number`
optional

The maximum total number of records that will be returned in your requests. If this value is larger than `pageSize` (which is 100 by default), you may have to load multiple pages to reach this total. See the Pagination section below for more.

pageSize
`number`
optional

The number of records returned in each request. Must be less than or equal to 100. Default is 100. See the Pagination section below for more.

sort
`array of objects`
optional

A list of sort objects that specifies how the records will be ordered. Each sort object must have a `field` key specifying the name of the field to sort on, and an optional `direction` key that is either `"asc"` or `"desc"`. The default direction is `"asc"`.

The `sort` parameter overrides the sorting of the view specified in the `view` parameter. If neither the `sort` nor the `view` parameter is included, the order of records is arbitrary.

For example, to sort records by `Name` in

```
    maxRecords: 3,
    view: "Grid view"
}).eachPage(function page(records, fetchNextPage) {
    // This function (`page`) will get called for each page of
records.

    records.forEach(function(record) {
        console.log('Retrieved', record.get('Name'));
    });

    // To fetch the next page of records, call `fetchNextPage`.
    // If there are more records, `page` will get called again.
    // If there are no more records, `done` will get called.
    fetchNextPage();

}, function done(err) {
    if (err) { console.error(err); return; }
});
```

OUTPUT

```
Retrieved Alice Mugenzi
Retrieved Jean Uwimana
Retrieved Claudine Mukamana
```

FETCH FIRST PAGE

```
// If you only want the first page of records, you can
// use `firstPage` instead of `eachPage`.
base('Users').select({
    view: 'Grid view'
}).firstPage(function(err, records) {
    if (err) { console.error(err); return; }
    records.forEach(function(record) {
        console.log('Retrieved', record.get('Name'));
    });
});
```

FETCH ADDITIONAL RECORD METADATA

```
// If you want to fetch the number of comments for each record,
// include the `recordMetadata` param.
base('Users').select({
    recordMetadata: ['commentCount']
}).firstPage(function(err, records) {
    if (err) { console.error(err); return; }
    records.forEach(function(record) {
        console.log('Retrieved a record with', record.commentCount,
'comments');
    });
});
```

descending order, send these two query parameters:

`sort%5B0%5D%5Bfield%5D=Name`
`sort%5B0%5D%5Bdirection%5D=desc`
For example, to sort records by Name in descending order, pass in:

```
[{field: "Name", direction: "desc"}]
```

view
`string`
optional

The name or ID of a view in the Users table. If set, only the records in that view will be returned. The records will be sorted according to the order of the view unless the `sort` parameter is included, which overrides that order. Fields hidden in this view will be returned in the results. To only return a subset of fields, use the `fields` parameter.

cellFormat
`string`
optional

The format that should be used for cell values. Supported values are:

`json`: cells will be formatted as JSON, depending on the field type.

`string`: cells will be formatted as user-facing strings, regardless of the field type. The `timeZone` and `userLocale` parameters are required when using `string` as the `cellFormat`.

Note: You should not rely on the format of these strings, as it is subject to change.

The default is `json`.

timeZone
`string`
optional

The time zone that should be used to format dates when using `string` as the `cellFormat`. This parameter is required when using `string` as the `cellFormat`.

userLocale
`string`
optional

The user locale that should be used to format dates when using `string` as the `cellFormat`. This parameter is required when using `string` as the `cellFormat`.

returnFieldsByFieldId
`boolean`
optional

An optional boolean value that lets you return field objects where the key is the field id.

This defaults to `false`, which returns field objects where the key is the field name.

recordMetadata
`array of strings`
optional

An optional field that, if includes `commentCount`, adds a `commentCount` read only property on each record returned.

Pagination

The server returns one page of records at a time. Each page will contain `pageSize` records, which is 100 by default.

To fetch the next page of records, call `fetchNextPage`.

Pagination will stop when you've reached the end of your table. If the `maxRecords` parameter is passed, pagination will stop once you've reached this maximum.

Iteration may timeout due to client inactivity or server restarts. In that case, the client will receive a 422 response with error message `LIST_RECORDS_ITERATOR_NOT_AVAILABLE`. It may then restart iteration from the beginning.

Retrieve a Users record

To retrieve an existing record in Users table, use the `find` method.

Any "empty" fields (e.g. `""`, `[]`, or `false`) in the record will not be returned.

CODE

```
var Airtable = require('airtable');
var base = new Airtable({apiKey:
'YOUR_SECRET_API_TOKEN'}).base('appPVwvxpSYRf5d0K');

base('Users').find('recLkv8HyT1Avv8z9', function(err, record) {
    if (err) { console.error(err); return; }
    console.log('Retrieved', record.id);
});
```

OUTPUT

```
Retrieved recLkv8HyT1Avv8z9
```

Create Users records

To create new records, use the `create` method. Note that table names and table ids can be used interchangeably. Using table ids means table name changes do not require modifications to your API request.

The first argument should be an array of up to 10 record objects. Each of these objects should have one key whose value is an inner object containing your record's cell values, keyed by either field name or field id.

Returns an array of record objects created if the call succeeded, including record IDs which will uniquely identify the records within AgriSales CRM.

Values for Full Name, Days Since Last Login, Agent Profile Region and Customer District are automatically computed by Airtable and cannot be directly created.

The Airtable API will perform best-effort automatic data conversion from string values if the `typecast` parameter is passed in (click to show example). Automatic conversion is disabled by default to ensure data integrity, but it may be helpful for integrating with 3rd party data sources.

You can also include a single record object at the top level. Click here to show an example.

CODE

```
var Airtable = require('airtable');
var base = new Airtable({apiKey:
'YOUR_SECRET_API_TOKEN'}).base('appPVwvxpSYRf5d0K');

base('Users').create([
  {
    "fields": {
      "Name": "Alice Mugenzi",
      "User ID": 101,
      "Email": "alice.mugenzi@email2201.com",
      "Role": "Admin",
      "Status": "Active",
      "Last Login": "2025-08-17",
      "Profile": [
        "recOornkxOinHnWQG"
      ],
      "Customer Profile": [
        "recaF22BK3FJUAi6s"
      ],
      "User Summary (AI)": {
        "state": "generated",
        "value": "Alice Mugenzi is an active admin user who last
logged in 1 day ago. She manages system settings and oversees user
roles.",
        "isStale": true
      }
    }
  },
  {
    "fields": {
      "Name": "Jean Uwimana",
      "User ID": 102,
      "Email": "jean.uwimana@email42f4.com",
      "Role": "Agent",
      "Status": "Active",
      "Last Login": "2025-08-18",
      "Profile": [
        "recX8ujF2lUzDBNXh"
      ],
      "Customer Profile": [
        "recZ28mz9NEqopI0E"
      ],
      "User Summary (AI)": {
        "state": "generated",
        "value": "Jean Uwimana is an active sales agent in the North
region. He logged in today and handles customer orders.",
        "isStale": true
      }
    }
  }
], function(err, records) {
  if (err) {
    console.error(err);
    return;
  }
  records.forEach(function (record) {
    console.log(record.getId());
  });
});
```

OUTPUT

```
recLkv8HyT1Avv8z9
```

```
recqAIeDLQE7sp7fS
```

Update Users records

To update Users records, use the `update` or `replace` method. An `update` will only update the fields you include. Fields not included will be unchanged. A `replace` will perform a destructive update and clear all unincluded cell values. A `replace` call on Users records will always fail. The example at the right uses the non-destructive `update` method. Click here to show a destructive `replace` call.

The first argument should be an array of up to 10 record objects. Each of these objects should have an `id` property representing the record ID and a `fields` property which contains all of your record's cell values by field name or field id for all of your record's cell values by field name.

To link to new records in Profile and Customer Profile, add new linked record IDs to the existing array. Be sure to include all existing linked record IDs that you wish to retain. To unlink records, include the existing array of record IDs, excluding any that you wish to unlink.

Values for Full Name, Days Since Last Login, Agent Profile Region and Customer District are automatically computed by Airtable and cannot be directly updated. You cannot clear these, even with a `replace` call.

Automatic data conversion for update actions can be enabled via `typecast` parameter. See create record for details.

You can also include a single record object at the top level. Click here to show an example.

CODE

```
var Airtable = require('airtable');
var base = new Airtable({apiKey:
'YOUR_SECRET_API_TOKEN'}).base('appPVwvxpSYRf5d0K');

base('Users').update([
  {
    "id": "recLkv8HyT1Avv8z9",
    "fields": {
      "Name": "Alice Mugenzi",
      "User ID": 101,
      "Email": "alice.mugenzi@email2201.com",
      "Role": "Admin",
      "Status": "Active",
      "Last Login": "2025-08-17",
      "Profile": [
        "recOornkxOinHnWQG"
      ],
      "Customer Profile": [
        "recaF22BK3FJUAi6s"
      ],
      "User Summary (AI)": {
        "state": "generated",
        "value": "Alice Mugenzi is an active admin user who last
logged in 1 day ago. She manages system settings and oversees user
roles.",
        "isStale": true
      }
    }
  },
  {
    "id": "recqAIeDLQE7sp7fS",
    "fields": {
      "Name": "Jean Uwimana",
      "User ID": 102,
      "Email": "jean.uwimana@email42f4.com",
      "Role": "Agent",
      "Status": "Active",
      "Last Login": "2025-08-18",
      "Profile": [
        "recX8ujF2lUzDBNXh"
      ],
      "Customer Profile": [
        "recZ28mz9NEqopI0E"
      ],
      "User Summary (AI)": {
        "state": "generated",
        "value": "Jean Uwimana is an active sales agent in the North
region. He logged in today and handles customer orders.",
        "isStale": true
      }
    }
  },
  {
    "id": "recm9iPNk1c9gfReL",
    "fields": {
      "Name": "Claudine Mukamana",
      "User ID": 103,
      "Email": "claudine.mukamana@email4c41.com",
      "Role": "Customer",
      "Status": "Active",
      "Last Login": "2025-08-15",
      "Profile": [
        "recOzIlfCcSHaUwcs"
      ],
      "Customer Profile": [
        "rec1hkIEnKuHyGdtC"
      ],
      "User Summary (AI)": {
        "state": "generated",
        "value": "Claudine Mukamana is an active customer from
Kayonza district. She last logged in 3 days ago to check her recent
orders.",
        "isStale": true
      }
    }
  }
], function(err, records) {
```

```
  if (err) {
    console.error(err);
    return;
  }
  records.forEach(function(record) {
    console.log(record.get('Name'));
  });
});
```

OUTPUT

```
"Alice Mugenzi"
"Jean Uwimana"
"Claudine Mukamana"
```

Delete Users records

To delete Users records, use the `destroy` method. Note that table names and table ids can be used interchangeably. Using table ids means table name changes do not require modifications to your API request.

The first parameter to `destroy` is an array of up to 10 record IDs to delete.

You can also set the first parameter to a record ID to delete a single record. Click here to show an example.

CODE

```
var Airtable = require('airtable');
var base = new Airtable({apiKey:
'YOUR_SECRET_API_TOKEN'}).base('appPVwvxpSYRf5d0K');

base('Users').destroy(['recLkv8HyT1Avv8z9', 'recqAIeDLQE7sp7fS'],
function(err, deletedRecords) {
  if (err) {
    console.error(err);
    return;
  }
  console.log('Deleted', deletedRecords.length, 'records');
});
```

OUTPUT

```
Deleted 2 records
```

AGENT PROFILES TABLE

The id for Agent Profiles is `tblb29mUicMjTIiER`. Table ids and table names can be used interchangeably in API requests. Using table ids means table name changes do not require modifications to your API request.

Fields

Each record in the Agent Profiles table contains the following fields:

Field names and field ids can be used interchangeably. Using field ids means field name changes do not require modifications to your API request. We recommend using field ids over field names where possible, to reduce modifications to your API request if the user changes the field name later.

FIELD NAME	FIELD ID	TYPE	DESCRIPTION

Agent ID	`fld78w8tunOmq9WVc`	Text	`string`
			A single line of text.
			
			EXAMPLE VALUES
			`"AGT-001"`
			`"AGT-002"`
			`"AGT-003"`
			`"AGT-004"`
			`"AGT-005"`

User	`fldlCsfx9seFsQ0qR`	Link to another record	`array of record IDs (strings)`
			Array of linked records IDs from the Users table.
			
			EXAMPLE VALUE
			`["rec8116cdd76088af", "rec245db9343f55e8", "rec4f3bade67ff565"]`

Team	`fldWAITQLidAlSbky`	Single select	`string`
			Selected option name.
			
			When creating or updating records, if the choice string does not exactly match an existing option, the request will fail with an `INVALID_MULTIPLE_CHOICE_OPTIONS` error unless the `typecast` parameter is enabled. If `typecast` is enabled, a new choice will be created if one does not exactly match.
			
			POSSIBLE VALUES
			`[ "Team A", "Team B", "Team C", "Team D" ]`

Region	`fldTVCM1VngVIwK1F`	Single select	`string`
			Selected option name.
			
			When creating or updating records, if the choice string does not exactly match an existing option, the request will fail with an `INVALID_MULTIPLE_CHOICE_OPTIONS` error unless the `typecast` parameter is enabled. If `typecast` is enabled, a new choice will be created if one does not exactly match.
			
			POSSIBLE VALUES
			`[ "North", "South", "East", "West" ]`

Sales	`fldv7EdvQ9ppt5jzF`	Link to another record	`array of record IDs (strings)`
			Array of linked records IDs from the Orders table.
			
			EXAMPLE VALUE
			`["rec8116cdd76088af", "rec245db9343f55e8", "rec4f3bade67ff565"]`

Commission Rate	`fldRKb4kg9GhM5vSx`	Number	`number`
			An integer (whole number, e.g. 1, 32, 99). This field allows negative and positive numbers.
			
			EXAMPLE VALUES
			`0`
			`0`
			`0`
			`0`
			`0`

Profile	`fld7rfZvB`	Attach	`array of attachment objects` EXAMPLE VALUES

Photo xm4Xi3GH ment

Each attachment object may contain the following properties. To see which fields are required or optional, please consult the relevant section: retrieve, create, update, or delete.

Because this field is configured to show attachments in reversed order, the order of attachments in the app will be reversed compared to what you see here.

`id` string

unique attachment id

`url` string

url, e.g. "https://v5.airtableusercontent.com/foo".

Note: URLs returned will expire 2 hours after being returned from our API. If you want to persist the attachments, we recommend downloading them instead of saving the URL. See our support article for more information.

`filename` string

filename, e.g. "foo.jpg"

`size` number

file size, in bytes

`type` string

content type, e.g. "image/jpeg"

`width` number
`height` number

width/height, in pixels (these may be available if the attachment is an image)

`thumbnails.small.url` string
`thumbnails.large.url` string

url of small/large thumbnails (these may be available if the attachment is an image or document). See notes under `url` about the lifetime of these URLs.

`thumbnails.small.width` number
`thumbnails.small.height` number
`thumbnails.large.width` number
`thumbnails.large.height` number

width/height of small/large thumbnails, in pixels (these will be available if the corresponding thumbnail url is available)


```
[
    {
        "id": "att82tcWWDJAuVOQq",
        "width": 600,
        "height": 401,
        "url":
"https://v5.airtableusercontent.com/v3/u/44/44/1755612000000/zr7IQZ1V
NfTA68feTIUuqQ/pEPnbcvE-
WK3Uk55fDHnDbwEm4tcLKViwbpi8rWlrCbiYCtbpydNM8qsovCgtmWCdXGsf22yRBN3Gh
g7UkVKx7WcjFyXllK6cgq_k_IvYDR-
5qvrQ5vrAEeT5qlHLDLhrys_1HO98NUIAyuQUsh-
GC5G_BeHliB28tjFKqVQ3f8/UFoR7pDtWt-TMm2gZwwKZhOPGtVtnAWfOTXR5EQV4ec",
        "filename": "abstract_19.png",
        "size": 118031,
        "type": "image/png",
        "thumbnails": {
            "small": {
                "url":
"https://v5.airtableusercontent.com/v3/u/44/44/1755612000000/kmsT4vAq
zTbGdIwN74Ij_A/N61zzGfxaJJdDTnhOwd0-QvP8Db-
jQIYsuBrm0SPNlACYqSUpMeSTPZD867Vh_onXHFp5MpJbkqDSZsBlA3xsoK2a3cSbgH9w
EcRkce35XOLC_lauNRbs42O6Cdpi4tenvRwgBM4buckztDN4uKnGg/bzEwCstNVXAMZ5i
fJItDgyXjWJlcGhgBWA2df-1Ut4o",
                "width": 54,
                "height": 36
            },
            "large": {
                "url":
"https://v5.airtableusercontent.com/v3/u/44/44/1755612000000/DAyCq_iH
6iglEzyqxuWHrw/v2NOOZCakskcx06XYX-bRNnAi9p2PWcAeS-
jO2464chdjNnfsQcaBi_0JBUVnNXbV2VMnN0c66ETRFIo9Qn8GT8dktFr5GLtqS8H-
0OPN1SpqNs4jkBUlQOtAEo3Zi0jSBujBrZhtt_lSmP7VobGfQ/F-
i5JyWVmuTXwkJre5C5mI2Hki49WeoWYvsH-poCuG4",
                "width": 600,
                "height": 401
            },
            "full": {
                "url":
"https://v5.airtableusercontent.com/v3/u/44/44/1755612000000/VE3aYoJY
tugtV6UG0ogx6Q/tLpIMGuFlC57Z9u6FM0RBWzNabGUBmrZeG9Uuz6_fU2Erv6FLLRZwO
ZRTN1GBFWpeDyMgScFem4GX5gQFqlKo06bD2bx__0ct-V8_7Aki8I50GgIr3f-
D8f8AIQfpBFJYDzydAJmEQHe2XvvvImNMQ/kHr9MAjRTcNZSMSP1IN8EjsFFnQdmTApUe
06B0yF9ac",
                "width": 600,
                "height": 401
            }
        }
    }
]
[
    {
        "id": "attkKZUwHXfQU2aG9",
        "width": 600,
        "height": 401,
        "url":
"https://v5.airtableusercontent.com/v3/u/44/44/1755612000000/YD4XLAYZ
KmCHMDgFpn3aOQ/OaMG_I8vG3stH0l_1uJT_wLuR0RwgaVYhcCk_rpl6ymNwOe-
WRtcOBC8aq2SCiNEMRquj1uYi44AWTbJrSaEh4gb8JuttTWbawY7xfVOvHhapSFx5m_QA
4QqGGLjhywtPRzdtS9c4ZAxmGg_rYhy2uC5IPaZMtvVBd3003mdbQg/kFImV_DEaYSswS
YceFE7cUxqC81OhHzzQHkAgRSScqY",
        "filename": "abstract_36.png",
        "size": 250091,
        "type": "image/png",
        "thumbnails": {
            "small": {
                "url":
"https://v5.airtableusercontent.com/v3/u/44/44/1755612000000/NI4pqVOa
N4akHPKtVoIb1g/qlppCybQ07qlPux-
BxjsGV6_SemoA746fGyaPlasIwJmdYCTJF1PqJ3Xk2Tm-vYSPp9PdglGcxy-
4oen0MIgBmbURcmCisA5EK-
RO5ToeWXKlWhJJku4OxEVWm7l6Mjl2Uftj4tn7bEDUNF2c5PYuA/lEQ2X5HmLB7O5lRVb
vmWuIJuPJHOW2uJ0ksiukhlfJ0",
                "width": 54,
                "height": 36
            },
            "large": {
                "url":
"https://v5.airtableusercontent.com/v3/u/44/44/1755612000000/VCS_6H-
XJ-YmIhOaeIpPaQ/xjcNHGQ7TEe95dF-
Z_DIJvYtsGWDA9LsQGc3VBpflvZaY9NA_bUyLpwlJOFh5rR-
1ylgzaGveDETuSpz4tgQie2__ADTkbfkYM9EqnE6_-5a8ak7grVvjo6LaDz1sxOOVmGup
H4RJB0tOBsD-3pEkQ/ylMTGhZWlRrmmH6adqENt62Id4KridSZ4zBbWW0rRB8",
                "width": 600,
                "height": 401
```

```
            },
            "full": {
                "url":
"https://v5.airtableusercontent.com/v3/u/44/44/1755612000000/D2D2524U
MXpJTPeDo1fRYw/kaNseswKAq76vRwo-eomqs4RA8hmZdBweIn1p84iSXkmBBBfLLP-
2NBvSgYWcNaMjKL0cDyu2bHV0GkdbwXlxcubhmKR7LFYVd9kvbpK40NGoILJjxRToAl0N
uo5D7SB9kVLGie9GZv2uVUSYMKfgw/yLhZuTpUKe0aVED0vBchklh6_cOSzY1ixSafBn0
1xx0",
                "width": 600,
                "height": 401
            }
        }
    }
]
```


Total Sales Amount	fldLYYR9pztZofabh	Rollup	number or string Computed value: SUM(values) for Total Amount in Orders.	EXAMPLE VALUES 166500 89500 157000 67500 210000
Number of Sales	fldhonp54R3D33fHU	Count	number Number of linked Orders records.	EXAMPLE VALUES 2 1 1 1 1
Average Sale Value	fldCJd23V1KI3lnRG	Formula	number, string, array of numbers or strings Computed value: IF(Number of Sales , Total Sales Amount / Number of Sales , BLANK()).	EXAMPLE VALUES 83250 89500 157000 67500 210000
Estimated Commission Earned	fldGWUPExC4EMyluv	Formula	number, string, array of numbers or strings Computed value: IF(Total Sales Amount , Total Sales Amount * Commission Rate , BLANK()).	EXAMPLE VALUES 13320 6265.000000000001 14130 4050 15750
User Email	fld1kVM7ivgcDBjPn	Lookup	array of numbers, strings, booleans, or objects Array of Email fields in linked Users records.	EXAMPLE VALUES ["alice.mugenzi@email2201.com"] ["jean.uwimana@email42f4.com"] ["claudine.mukamana@email4c41.com"] ["eric.nshimiyimana@emaile11d.com"] ["marie.uwase@email330a.com"]
Last Login	fldEgKIQiepBL0w3B	Lookup	array of numbers, strings, booleans, or objects Array of Last Login fields in linked Users records.	EXAMPLE VALUES ["2025-08-17"] ["2025-08-18"] ["2025-08-15"] ["2025-07-30"] ["2025-08-16"]
Agent Performance Summa	fld4lAXH3WVUMk9EQ			

Summary (AI)

Agent Improvement Suggestions (AI) `fldlS0oQ2` `FMmE7jqs`

List Agent Profiles records

To list records in Agent Profiles, use the `select` method.

`select` returns a query object. To fetch the records matching that query, use the `eachPage` or `firstPage` method of the query object.

Returned records do not include any fields with "empty" values, e.g. `""`, `[]`, or `false`.

Note: Airtable's API only accepts request with a URL shorter than 16,000 characters. Encoded formulas may cause your requests to exceed this limit. To fix this issue you can instead make a POST request to `/v0/{baseId}/{tableIdOrName}/listRecords` while passing the parameters within the body of the request instead of the query parameters. See our support article on this for more information.

You can use the following parameters to filter, sort, and format the results:

fields
`array of strings`
optional

Only data for fields whose names are in this list will be included in the result. If you don't need every field, you can use this parameter to reduce the amount of data transferred.

For example, to only return data from Agent ID and User, pass in:

```
fields: ["Agent ID", "User"]
```

You can also perform the same action with field ids (they can be found in the fields section):

```
fields: ["fld78w8tunOmq9WVc", "fldlCsfx9seFsQ
```

filterByFormula
`string`
optional

A formula used to filter records. The formula will be evaluated for each record, and if the result is not `0`, `false`, `""`, `NaN`, `[]`, or `#Error!` the record will be included in the response. We recommend testing your formula in the Formula field UI before using it in your API request.

If combined with the `view` parameter, only records in that view which satisfy the formula will be returned.

The formula must be encoded first before passing it as a value. You can use this tool to not only encode the formula but also create the entire url you need. For example, to only include records where Agent ID isn't empty, pass in `NOT({Agent ID} = '')` as a parameter like this:

```
filterByFormula: "NOT({Agent ID} = '')"
```

Note: Airtable's API only accepts request with a URL shorter than 16,000 characters. Encoded formulas may cause your requests to exceed this limit. To fix this issue you can instead make a POST request to `/v0/{baseId}/{tableIdOrName}/listRecords` while passing the parameters within the body of the request instead of the query parameters. See our support article on this for more information.

maxRecords
`number`
optional

The maximum total number of records that will be returned in your requests. If this value is larger than `pageSize` (which is 100 by default), you may have to load multiple pages to reach this total. See the

CODE

```
var Airtable = require('airtable');
var base = new Airtable({apiKey:
'YOUR_SECRET_API_TOKEN'}).base('appPVwvxpSYRf5d0K');

base('Agent Profiles').select({
    // Selecting the first 3 records in Grid view:
    maxRecords: 3,
    view: "Grid view"
}).eachPage(function page(records, fetchNextPage) {
    // This function (`page`) will get called for each page of
records.

    records.forEach(function(record) {
        console.log('Retrieved', record.get('Agent ID'));
    });

    // To fetch the next page of records, call `fetchNextPage`.
    // If there are more records, `page` will get called again.
    // If there are no more records, `done` will get called.
    fetchNextPage();

}, function done(err) {
    if (err) { console.error(err); return; }
});
```

OUTPUT

```
Retrieved AGT-001
Retrieved AGT-002
Retrieved AGT-003
```

FETCH FIRST PAGE

```
// If you only want the first page of records, you can
// use `firstPage` instead of `eachPage`.
base('Agent Profiles').select({
    view: 'Grid view'
}).firstPage(function(err, records) {
    if (err) { console.error(err); return; }
    records.forEach(function(record) {
        console.log('Retrieved', record.get('Agent ID'));
    });
});
```

FETCH ADDITIONAL RECORD METADATA

```
// If you want to fetch the number of comments for each record,
// include the `recordMetadata` param.
base('Agent Profiles').select({
    recordMetadata: ['commentCount']
}).firstPage(function(err, records) {
    if (err) { console.error(err); return; }
    records.forEach(function(record) {
        console.log('Retrieved a record with', record.commentCount,
'comments');
    });
});
```

Pagination section below for more.

pageSize
`number`
optional

The number of records returned in each request. Must be less than or equal to 100. Default is 100. See the Pagination section below for more.

sort
`array of objects`
optional

A list of sort objects that specifies how the records will be ordered. Each sort object must have a `field` key specifying the name of the field to sort on, and an optional `direction` key that is either `"asc"` or `"desc"`. The default direction is `"asc"`.

The `sort` parameter overrides the sorting of the view specified in the `view` parameter. If neither the `sort` nor the `view` parameter is included, the order of records is arbitrary.

For example, to sort records by Agent ID in descending order, send these two query parameters:

```
sort%5B0%5D%5Bfield%5D=Agent%20ID
sort%5B0%5D%5Bdirection%5D=desc
```

For example, to sort records by Agent ID in descending order, pass in:

```
[{field: "Agent ID", direction: "desc"}]
```

view
`string`
optional

The name or ID of a view in the Agent Profiles table. If set, only the records in that view will be returned. The records will be sorted according to the order of the view unless the `sort` parameter is included, which overrides that order. Fields hidden in this view will be returned in the results. To only return a subset of fields, use the `fields` parameter.

cellFormat
`string`
optional

The format that should be used for cell values. Supported values are:

`json`: cells will be formatted as JSON, depending on the field type.

`string`: cells will be formatted as user-facing strings, regardless of the field type. The `timeZone` and `userLocale` parameters are required when using `string` as the `cellFormat`.

Note: You should not rely on the format of these strings, as it is subject to change.

The default is `json`.

timeZone
`string`
optional

The time zone that should be used to format dates when using `string` as the `cellFormat`. This parameter is required when using `string` as the `cellFormat`.

userLocale
`string`
optional

The user locale that should be used to format dates when using `string` as the `cellFormat`. This parameter is required when using `string` as the `cellFormat`.

returnFieldsByFieldId
`boolean`
optional

An optional boolean value that lets you return field objects where the key is the field id.

This defaults to `false`, which returns field objects where the key is the field name.

recordMetadata
`array of strings`
optional

An optional field that, if includes `commentCount`, adds a `commentCount` read only property on each record returned.

Pagination

The server returns one page of records at a time. Each page will contain `pageSize` records, which is 100 by default.

To fetch the next page of records, call `fetchNextPage`.

Pagination will stop when you've reached the end of your table. If the `maxRecords` parameter is passed, pagination will stop once you've reached this maximum.

Iteration may timeout due to client inactivity or server restarts. In that case, the client will receive a 422 response with error message `LIST_RECORDS_ITERATOR_NOT_AVAILABLE`. It may then restart iteration from the beginning.

Retrieve a Agent Profiles record

To retrieve an existing record in Agent Profiles table, use the `find` method.

Any "empty" fields (e.g. `""`, `[]`, or `false`) in the record will not be returned.

In attachment objects included in the retrieved record (Profile Photo), only `id`, `url`, and `filename` are always returned. Other attachment properties may not be included. Note: Attachment URLs returned will expire 2 hours after being returned from our API. If you want to persist the attachments, we recommend downloading them instead of saving the URL. See our support article for more information.

CODE

```
var Airtable = require('airtable');
var base = new Airtable({apiKey:
'YOUR_SECRET_API_TOKEN'}).base('appPVwvxpSYRf5d0K');

base('Agent Profiles').find('recOornkxOinHnWQG', function(err,
record) {
    if (err) { console.error(err); return; }
    console.log('Retrieved', record.id);
});
```

OUTPUT

```
Retrieved recOornkxOinHnWQG
```

Create Agent Profiles records

To create new records, use the `create` method. Note that table names and table ids can be used interchangeably. Using table ids means table name changes do not require modifications to your API request.

The first argument should be an array of up to 10 record objects. Each of these objects should have one key whose value is an inner object containing your record's cell values, keyed by either field name or field id.

Returns an array of record objects created if the call succeeded, including record IDs which will uniquely identify the records within AgriSales CRM .

To create new attachments in Profile Photo , set the field value to an array of attachment objects. When creating an attachment, `url` is required, and `filename` is optional. Airtable will download the file at the given `url` and keep its own copy of it. All other attachment object properties will be generated server-side soon afterward.

Note that in most cases the API does not currently return an error code for failed attachment object creation given attachment uploading happens in an asynchronous manner, such cases will manifest with the attachment object either being cleared from the cell or persisted with generated URLs that return error responses when queried. If the same attachment URL fails to upload multiple times in a short time interval then * the API may return an ATTACHMENTS_FAILED_UPLOADING error code in the details field of the response and the attachment object will * be cleared from the cell synchronously.

We also require URLs used to upload have the https:// or http:// protocol (Note: http:// support will be removed in the near future), have a limit of 3 max redirects, and a file size limit of 1GB. In addition, URLs must be publicly accessible, in cases where cookie authentication or logging in to access the file is required, the login page HTML will be downloaded instead of the file.

Attachment URLs returned will expire 2 hours after being returned from our API. If you want to persist the attachments, we recommend downloading them instead of saving the URL.

If too many attachments are uploaded within a short period of time, the server may return a partial failure on record creation with an "Attachment Upload Rate Too High" error. See our support article for more information.

CODE

```
var Airtable = require('airtable');
var base = new Airtable({apiKey:
'YOUR_SECRET_API_TOKEN'}).base('appPVwvxpSYRf5d0K');

base('Agent Profiles').create([
  {
    "fields": {
      "Agent ID": "AGT-001",
      "User": [
        "recLkv8HyT1Avv8z9"
      ],
      "Team": "Team A",
      "Region": "North",
      "Sales": [
        "reciKJhi7RNXV0p1X",
        "recqalJAapDxavwOz"
      ],
      "Commission Rate": 0.08,
      "Profile Photo": [
        {
          "url":
"https://v5.airtableusercontent.com/v3/u/44/44/1755612000000/zr7IQZ1V
NfTA68feTIUuqQ/pEPnbcvE-
WK3Uk55fDHnDbwEm4tcLKViwbpi8rWlrCbiYCtbpydNM8qsovCgtmWCdXGsf22yRBN3Gh
g7UkVKx7WcjFyXllK6cgq_k_IvYDR-
5qvrQ5vrAEeT5qlHLDLhrys_1HO98NUIAyuQUsh-
GC5G_BeHliB28tjFKqVQ3f8/UFoR7pDtWt-TMm2gZwwKZhOPGtVtnAWfOTXR5EQV4ec"
        }
      ],
      "Agent Performance Summary (AI)": {
        "state": "generated",
        "value": "Jane consistently exceeds targets in the North
region, demonstrating strong relationship-building skills with
customers and a high average sale value.",
        "isStale": true
      },
      "Agent Improvement Suggestions (AI)": {
        "state": "generated",
        "value": "Focus on digital sales tools and expand outreach to
underperforming districts for even higher results.",
        "isStale": true
```

Values for Total Sales Amount, Number of Sales, Average Sale Value, Estimated Commission Earned, User Email and Last Login are automatically computed by Airtable and cannot be directly created.

The Airtable API will perform best-effort automatic data conversion from string values if the `typecast` parameter is passed in (click to show example). Automatic conversion is disabled by default to ensure data integrity, but it may be helpful for integrating with 3rd party data sources.

You can also include a single record object at the top level. Click here to show an example.

```
      }
    }
  },
  {
    "fields": {
      "Agent ID": "AGT-002",
      "User": [
        "recqAIeDLQE7sp7fS"
      ],
      "Team": "Team B",
      "Region": "South",
      "Sales": [
        "recfH73uq2xdb1k0H"
      ],
      "Commission Rate": 0.07,
      "Profile Photo": [
        {
          "url":
"https://v5.airtableusercontent.com/v3/u/44/44/1755612000000/YD4XLAYZ
KmCHMDgFpn3aOQ/OaMG_I8vG3stH01_1uJT_wLuR0RwgaVYhcCk_rp16ymNwOe-
WRtcOBC8aq2SCiNEMRquj1uYi44AWTbJrSaEh4gb8JuttTWbawY7xfVOvHhapSFx5m_QA
4QqGGLjhywtPRzdt89c4ZAxmGg_rYhy2uC5IPaZMtvVBd3003mdbQg/kFImV_DEaYSswS
YceFE7cUxgC81OhHzzQHkAgRSScqY"
        }
      ],
      "Agent Performance Summary (AI)": {
        "state": "generated",
        "value": "Paul maintains solid sales with a high average per
transaction, showing reliability and strong follow-up.",
        "isStale": true
      },
      "Agent Improvement Suggestions (AI)": {
        "state": "generated",
        "value": "Increase cross-selling of new products and leverage
customer testimonials to boost conversions.",
        "isStale": true
      }
    }
  }
], function(err, records) {
  if (err) {
    console.error(err);
    return;
  }
  records.forEach(function (record) {
    console.log(record.getId());
  });
});
```

OUTPUT

```
recOornkxOinHnWQG
recX8ujF2lUzDBNXh
```

Update Agent Profiles records

To update Agent Profiles records, use the `update` or `replace` method. An `update` will only update the fields you include. Fields not included will be unchanged. A `replace` will perform a destructive update and clear all unincluded cell values. A `replace` call on Agent Profiles records will always fail. The example at the right uses the non-destructive `update` method. Click here to show a destructive `replace` call.

The first argument should be an array of up to 10 record objects. Each of these objects should have an `id` property representing the record ID and a `fields` property which contains all of your record's cell values by field name or field id for all of your record's cell values by field name.

To add attachments to Profile Photo, add new attachment objects to the existing array. Be sure to include all existing attachment objects that you wish to retain, to keep preexisting attachments providing `id` is required (which can be retrieved using the retrieve endpoint), other fields are ignored. For the new attachments being added, `url` is required, and `filename` is optional. To remove attachments, include the existing array of attachment objects, excluding any that you wish to remove.

Note that in most cases the API does not currently return an error code for failed attachment object creation given attachment uploading happens in an asynchronous manner, such cases will manifest with the attachment object either being cleared from the cell or persisted with generated URLs that return error responses when queried. If the same attachment URL fails to upload multiple

CODE

```
var Airtable = require('airtable');
var base = new Airtable({apiKey:
'YOUR_SECRET_API_TOKEN'}).base('appPVwvxpSYRf5d0K');

base('Agent Profiles').update([
  {
    "id": "recOornkxOinHnWQG",
    "fields": {
      "Agent ID": "AGT-001",
      "User": [
        "recLkv8HyT1Avv8z9"
      ],
      "Team": "Team A",
      "Region": "North",
      "Sales": [
        "reciKJhi7RNXV0p1X",
        "recqa1JAapDxavwOz"
      ],
      "Commission Rate": 0.08,
      "Profile Photo": [
        {
          "id": "att82tcWWDJAuVOQq"
        }
      ],
      "Agent Performance Summary (AI)": {
        "state": "generated",
```

responses when queried. If the same attachment URL fails to upload multiple times in a short time interval then * the API may return an ATTACHMENTS_FAILED_UPLOADING error code in the details field of the response and the attachment object will * be cleared from the cell synchronously.

We also require URLs used to upload have the https:// or http:// protocol (Note: http:// support will be removed in the near future), have a limit of 3 max redirects, and a file size limit of 1GB. In addition, URLs must be publicly accessible, in cases where cookie authentication or logging in to access the file is required, the login page HTML will be downloaded instead of the file.

If too many attachments are uploaded within a short period of time, the server may return a partial failure on record creation with an "Attachment Upload Rate Too High" error.

To link to new records in `User` and `Sales`, add new linked record IDs to the existing array. Be sure to include all existing linked record IDs that you wish to retain. To unlink records, include the existing array of record IDs, excluding any that you wish to unlink.

Values for `Total Sales Amount`, `Number of Sales`, `Average Sale Value`, `Estimated Commission Earned`, `User Email` and `Last Login` are automatically computed by Airtable and cannot be directly updated. You cannot clear these, even with a `replace` call.

Automatic data conversion for update actions can be enabled via `typecast` parameter. See create record for details.

You can also include a single record object at the top level. Click here to show an example.


```
        "value": "Jane consistently exceeds targets in the North
region, demonstrating strong relationship-building skills with
customers and a high average sale value.",
        "isStale": true
      },
      "Agent Improvement Suggestions (AI)": {
        "state": "generated",
        "value": "Focus on digital sales tools and expand outreach to
underperforming districts for even higher results.",
        "isStale": true
      }
    }
  },
  {
    "id": "recX8ujF2lUzDBNXh",
    "fields": {
      "Agent ID": "AGT-002",
      "User": [
        "recqAIeDLQE7sp7fS"
      ],
      "Team": "Team B",
      "Region": "South",
      "Sales": [
        "recfH73uq2xdb1k0H"
      ],
      "Commission Rate": 0.07,
      "Profile Photo": [
        {
          "id": "attkK2UwHXfQU2aG9"
        }
      ],
      "Agent Performance Summary (AI)": {
        "state": "generated",
        "value": "Paul maintains solid sales with a high average per
transaction, showing reliability and strong follow-up.",
        "isStale": true
      },
      "Agent Improvement Suggestions (AI)": {
        "state": "generated",
        "value": "Increase cross-selling of new products and leverage
customer testimonials to boost conversions.",
        "isStale": true
      }
    }
  },
  {
    "id": "recOzIlfCcSHaUwcs",
    "fields": {
      "Agent ID": "AGT-003",
      "User": [
        "recm9iPNk1c9gfReL"
      ],
      "Team": "Team C",
      "Region": "East",
      "Sales": [
        "recMkt174sEFSB3nY"
      ],
      "Commission Rate": 0.09,
      "Profile Photo": [
        {
          "id": "attVo9dkU3orFqz4I"
        }
      ],
      "Agent Performance Summary (AI)": {
        "state": "generated",
        "value": "Alice leads in both total sales and number of
deals, with excellent customer retention rates.",
        "isStale": true
      },
      "Agent Improvement Suggestions (AI)": {
        "state": "generated",
        "value": "Consider mentoring junior agents and exploring
automation to handle more clients efficiently.",
        "isStale": true
      }
    }
  }
], function(err, records) {
  if (err) {
    console.error(err);
    return;
  }
  records.forEach(function(record) {
    console.log(record.get('Agent ID'));
```

```
  });
});
```

OUTPUT

```
"AGT-001"
"AGT-002"
"AGT-003"
```

Delete Agent Profiles records

To delete Agent Profiles records, use the `destroy` method. Note that table names and table ids can be used interchangeably. Using table ids means table name changes do not require modifications to your API request.

The first parameter to `destroy` is an array of up to 10 record IDs to delete.

You can also set the first parameter to a record ID to delete a single record. Click here to show an example.

CODE

```
var Airtable = require('airtable');
var base = new Airtable({apiKey:
'YOUR_SECRET_API_TOKEN'}).base('appPVwvxpSYRf5d0K');

base('Agent Profiles').destroy(['recOornkxOinHnWQG',
'recX8ujF2lUzDBNXh'], function(err, deletedRecords) {
  if (err) {
    console.error(err);
    return;
  }
  console.log('Deleted', deletedRecords.length, 'records');
});
```

OUTPUT

```
Deleted 2 records
```

CUSTOMER PROFILES TABLE

The id for Customer Profiles is `tblt3X7CaKohHSBR9`. Table ids and table names can be used interchangeably in API requests. Using table ids means table

name changes do not require modifications to your API request.

Fields

Each record in the Customer Profiles table contains the following fields:

Field names and field ids can be used interchangeably. Using field ids means field name changes do not require modifications to your API request. We recommend using field ids over field names where possible, to reduce modifications to your API request if the user changes the field name later.

FIELD NAME	FIELD ID	TYPE	DESCRIPTION	EXAMPLE VALUES
Customer ID	fldvZf4sqRkytoSJN	Text	string A single line of text.	"CUST-001" "CUST-002" "CUST-003" "CUST-004" "CUST-005"
User	fldLVcq3G4fxRe1ST	Link to another record	array of record IDs (strings) Array of linked records IDs from the Users table.	["rec8116cdd76088af", "rec245db9343f55e8", "rec4f3bade67ff565"]
First Name	fldiAdOhyGKyydVut	Text	string A single line of text.	"Jean" "Aline" "Eric" "Claudine" "Samuel"
Last Name	fldEMPp2LEg1F9Tee	Text	string A single line of text.	"Uwimana" "Mukamana" "Nshimiyimana" "Iradukunda" "Habimana"
Telephone	fldf9lWfyBc3GyhNf	Text	string A single line of text.	"+250788123456" "+250788654321" "+250789112233" "+250782334455" "+250788998877"
District	fldhcRo9UAlovY2CY	Text	string A single line of text.	"Kigali" "Southern" "Eastern" "Northern" "Western"
Sector	fldS5hDHa46f4ZqpB	Text	string A single line of text.	"Gasabo" "Huye" "Rwamagana" "Musanze" "Rubavu"
Cell	flde3gpYVjvxj7SXd	Text	string A single line of text.	"Remera" "Ngoma" "Karenge" "Muhoza" "Gisenyi"
Village	fldxxXjWHvqXXtK6U	Text	string A single line of text.	"Nyabisindu" "Kigarama" "Bweramana" "Gashaki" "Mbugangari"
Order	fldChMBSj	Link to	array of record IDs	EXAMPLE VALUE

s	x9x2omnM	another record	(strings) Array of linked records IDs from the Orders table.	["rec8116cdd76088af", "rec245db9343f55e8", "rec4f3bade67ff565"]
Full Name	fldSuKGtbwtoQ8606	Formula	number, string, array of numbers or strings Computed value: First Name & " " & Last Name.	EXAMPLE VALUES "\"Jean Uwimana\"" "\"Aline Mukamana\"" "\"Eric Nshimiyimana\"" "\"Claudine Iradukunda\"" "\"Samuel Habimana\""
Order Count	flduukkGwg0JThD4P	Count	number Number of linked Orders records.	EXAMPLE VALUES 2 1 1 1 1
Total Orders Value	fldvS80A4rLogM2uI	Rollup	number or string Computed value: SUM(values) for Total Amount in Orders.	EXAMPLE VALUES 166500 89500 157000 67500 210000
Last Order Date	fldclgjtG2JXdrd2I	Rollup	number or string Computed value: MAX(values) for Order Date in Orders.	EXAMPLE VALUES "2025-08-13" "2025-08-15" "2025-08-10" "2025-07-30" "2025-08-17"
Order Status Summary	fldNJSNfXTx38SuuB	Rollup	number or string Computed value: ARRAYJOIN(values, ", ") for Status in Orders.	EXAMPLE VALUES "\"Delivered, Cancelled\"" "\"Pending\"" "\"Shipped\"" "\"Delivered\"" "\"Pending\""
Location Summary	fldegNvjHblBQtInP	Formula	number, string, array of numbers or strings Computed value: District & ", " & Sector & ", " & Cell & ", " & Village.	EXAMPLE VALUES "\"Kigali, Gasabo, Remera, Nyabisindu\"" "\"Southern, Huye, Ngoma, Kigarama\"" "\"Eastern, Rwamagana, Karenge, Bweramana\"" "\"Northern, Musanze, Muhoza, Gashaki\"" "\"Western, Rubavu, Gisenyi, Mbugangari\""
Customer Insights (AI)	fldHJ0UKojZ3UyRHA			
Customer Type (AI)	fldF01nVIgo0RGavl			
Deadline	flde4Sw3GxblGQBOi	Date	string (ISO 8601 formatted date) UTC date, e.g. "2014-09-05".	EXAMPLE VALUE "2014-09-05"

List Customer Profiles records

To list records in Customer Profiles, use the `select` method.

`select` returns a query object. To fetch the records matching that query, use the `eachPage` or `firstPage` method of the query object.

Returned records do not include any fields with "empty" values, e.g. `""`, `[]`, or `false`.

Note: Airtable's API only accepts request with a URL shorter than 16,000 characters. Encoded formulas may cause your requests to exceed this limit. To fix this issue you can instead make a POST request to `/v0/{baseId}/{tableIdOrName}/listRecords` while passing the parameters within the body of the request instead of the query parameters. See our support article on this for more information.

CODE

```
var Airtable = require('airtable');
var base = new Airtable({apiKey:
'YOUR_SECRET_API_TOKEN'}).base('appPVwvxpSYRf5d0K');

base('Customer Profiles').select({
    // Selecting the first 3 records in Grid view:
    maxRecords: 3,
    view: "Grid view"
}).eachPage(function page(records, fetchNextPage) {
    // This function (`page`) will get called for each page of
records.

    records.forEach(function(record) {
        console.log('Retrieved', record.get('Customer ID'));
    });
```

You can use the following parameters to filter, sort, and format the results:

fields
`array of strings`
optional

Only data for fields whose names are in this list will be included in the result. If you don't need every field, you can use this parameter to reduce the amount of data transferred.

For example, to only return data from `Customer ID` and `User`, pass in:

```
fields: ["Customer ID", "User"]
```

You can also perform the same action with field ids (they can be found in the fields section):

```
fields: ["fldvZf4sqRkytoSJN", "fldLVcq3G4fxRe1
```

filterByFormula
`string`
optional

A formula used to filter records. The formula will be evaluated for each record, and if the result is not `0`, `false`, `""`, `NaN`, `[]`, or `#Error!` the record will be included in the response. We recommend testing your formula in the Formula field UI before using it in your API request.

If combined with the `view` parameter, only records in that view which satisfy the formula will be returned.

The formula must be encoded first before passing it as a value. You can use this tool to not only encode the formula but also create the entire url you need. For example, to only include records where `Customer ID` isn't empty, pass in `NOT({Customer ID} = '')` as a parameter like this:

```
filterByFormula: "NOT({Customer ID} = '')"
```

Note: Airtable's API only accepts request with a URL shorter than 16,000 characters. Encoded formulas may cause your requests to exceed this limit. To fix this issue you can instead make a POST request to `/v0/{baseId}/{tableIdOrName}/listRecords` while passing the parameters within the body of the request instead of the query parameters. See our support article on this for more information.

maxRecords
`number`
optional

The maximum total number of records that will be returned in your requests. If this value is larger than `pageSize` (which is 100 by default), you may have to load multiple pages to reach this total. See the Pagination section below for more.

pageSize
`number`
optional

The number of records returned in each request. Must be less than or equal to 100. Default is 100. See the Pagination section below for more.

sort
`array of objects`
optional

A list of sort objects that specifies how the records will be ordered. Each sort object must have a `field` key specifying the name of the field to sort on, and an optional `direction` key that is either `"asc"` or `"desc"`. The default direction is `"asc"`.

The `sort` parameter overrides the sorting of the view specified in the `view` parameter. If neither the `sort` nor the `view` parameter is included, the order of records is arbitrary.

For example, to sort records by `Customer ID` in descending order, send these two query parameters:

```
sort%5B0%5D%5Bfield%5D=Customer%20ID
sort%5B0%5D%5Bdirection%5D=desc
```

For example, to sort records by `Customer ID` in descending order, pass in:

```
[{field: "Customer ID", direction: "desc"}]
```

```
    // To fetch the next page of records, call `fetchNextPage`.
    // If there are more records, `page` will get called again.
    // If there are no more records, `done` will get called.
    fetchNextPage();

}, function done(err) {
    if (err) { console.error(err); return; }
});
```

OUTPUT

```
Retrieved CUST-001
Retrieved CUST-002
Retrieved CUST-003
```

FETCH FIRST PAGE

```
// If you only want the first page of records, you can
// use `firstPage` instead of `eachPage`.
base('Customer Profiles').select({
    view: 'Grid view'
}).firstPage(function(err, records) {
    if (err) { console.error(err); return; }
    records.forEach(function(record) {
        console.log('Retrieved', record.get('Customer ID'));
    });
});
```

FETCH ADDITIONAL RECORD METADATA

```
// If you want to fetch the number of comments for each record,
// include the `recordMetadata` param.
base('Customer Profiles').select({
    recordMetadata: ['commentCount']
}).firstPage(function(err, records) {
    if (err) { console.error(err); return; }
    records.forEach(function(record) {
        console.log('Retrieved a record with', record.commentCount,
'comments');
    });
});
```

view
`string`
optional

The name or ID of a view in the Customer Profiles table. If set, only the records in that view will be returned. The records will be sorted according to the order of the view unless the `sort` parameter is included, which overrides that order. Fields hidden in this view will be returned in the results. To only return a subset of fields, use the `fields` parameter.

cellFormat
`string`
optional

The format that should be used for cell values. Supported values are:

`json`: cells will be formatted as JSON, depending on the field type.

`string`: cells will be formatted as user-facing strings, regardless of the field type. The `timeZone` and `userLocale` parameters are required when using `string` as the `cellFormat`.

Note: You should not rely on the format of these strings, as it is subject to change.

The default is `json`.

timeZone
`string`
optional

The time zone that should be used to format dates when using `string` as the `cellFormat`. This parameter is required when using `string` as the `cellFormat`.

userLocale
`string`
optional

The user locale that should be used to format dates when using `string` as the `cellFormat`. This parameter is required when using `string` as the `cellFormat`.

returnFieldsByFieldId
`boolean`
optional

An optional boolean value that lets you return field objects where the key is the field id.

This defaults to `false`, which returns field objects where the key is the field name.

recordMetadata
`array of strings`
optional

An optional field that, if includes `commentCount`, adds a `commentCount` read only property on each record returned.

Pagination

The server returns one page of records at a time. Each page will contain `pageSize` records, which is 100 by default.

To fetch the next page of records, call `fetchNextPage`.

Pagination will stop when you've reached the end of your table. If the `maxRecords` parameter is passed, pagination will stop once you've reached this maximum.

Iteration may timeout due to client inactivity or server restarts. In that case, the client will receive a 422 response with error message `LIST_RECORDS_ITERATOR_NOT_AVAILABLE`. It may then restart iteration from the beginning.

Retrieve a Customer Profiles record

To retrieve an existing record in Customer Profiles table, use the `find` method.

Any "empty" fields (e.g. `""`, `[]`, or `false`) in the record will not be returned.

CODE

```
var Airtable = require('airtable');
var base = new Airtable({apiKey:
'YOUR_SECRET_API_TOKEN'}).base('appPVwvxpSYRf5d0K');

base('Customer Profiles').find('recaF22BK3FJUAi6s', function(err,
```

```
record) {
    if (err) { console.error(err); return; }
    console.log('Retrieved', record.id);
});
```

OUTPUT

```
Retrieved recaF22BK3FJUAi6s
```

Create Customer Profiles records

To create new records, use the `create` method. Note that table names and table ids can be used interchangeably. Using table ids means table name changes do not require modifications to your API request.

The first argument should be an array of up to 10 record objects. Each of these objects should have one key whose value is an inner object containing your record's cell values, keyed by either field name or field id.

Returns an array of record objects created if the call succeeded, including record IDs which will uniquely identify the records within AgriSales CRM.

Values for Full Name, Order Count, Total Orders Value, Last Order Date, Order Status Summary and Location Summary are automatically computed by Airtable and cannot be directly created.

The Airtable API will perform best-effort automatic data conversion from string values if the `typecast` parameter is passed in (click to show example). Automatic conversion is disabled by default to ensure data integrity, but it may be helpful for integrating with 3rd party data sources.

You can also include a single record object at the top level. Click here to show an example.

CODE

```
var Airtable = require('airtable');
var base = new Airtable({apiKey:
'YOUR_SECRET_API_TOKEN'}).base('appPVwvxpSYRf5d0K');

base('Customer Profiles').create([
  {
    "fields": {
      "Customer ID": "CUST-001",
      "User": [
        "recLkv8HyT1Avv8z9"
      ],
      "First Name": "Jean",
      "Last Name": "Uwimana",
      "Telephone": "+250788123456",
      "District": "Kigali",
      "Sector": "Gasabo",
      "Cell": "Remera",
      "Village": "Nyabisindu",
      "Orders": [
        "reciKJhi7RNXV0p1X",
        "recqa1JAapDxavwOz"
      ],
      "Customer Insights (AI)": {
        "state": "generated",
        "value": "Customer is highly engaged, places regular orders,
and prefers early season purchases.",
        "isStale": true
      },
      "Customer Type (AI)": {
        "state": "generated",
        "value": "Commercial Farmer",
        "isStale": true
      }
    }
  },
  {
    "fields": {
      "Customer ID": "CUST-002",
      "User": [
        "recqAIeDLQE7sp7fS"
      ],
      "First Name": "Aline",
      "Last Name": "Mukamana",
      "Telephone": "+250788654321",
      "District": "Southern",
      "Sector": "Huye",
      "Cell": "Ngoma",
      "Village": "Kigarama",
      "Orders": [
        "recfH73uq2xdb1k0H"
      ],
      "Customer Insights (AI)": {
        "state": "generated",
        "value": "Occasional buyer, tends to order during harvest
season.",
        "isStale": true
      },
      "Customer Type (AI)": {
        "state": "generated",
        "value": "Smallholder",
        "isStale": true
      }
    }
  }
], function(err, records) {
  if (err) {
    console.error(err);
    return;
  }
  records.forEach(function (record) {
    console.log(record.getId());
```

```
  });
});
```

OUTPUT

```
recaF22BK3FJUAi6s
recZ28mz9NEqopI0E
```

Update Customer Profiles records

To update Customer Profiles records, use the `update` or `replace` method. An `update` will only update the fields you include. Fields not included will be unchanged. A `replace` will perform a destructive update and clear all unincluded cell values. A `replace` call on Customer Profiles records will always fail. The example at the right uses the non-destructive `update` method. Click here to show a destructive `replace` call.

The first argument should be an array of up to 10 record objects. Each of these objects should have an `id` property representing the record ID and a `fields` property which contains all of your record's cell values by field name or field id for all of your record's cell values by field name.

To link to new records in User and Orders, add new linked record IDs to the existing array. Be sure to include all existing linked record IDs that you wish to retain. To unlink records, include the existing array of record IDs, excluding any that you wish to unlink.

Values for Full Name, Order Count, Total Orders Value, Last Order Date, Order Status Summary and Location Summary are automatically computed by Airtable and cannot be directly updated. You cannot clear these, even with a `replace` call.

Automatic data conversion for update actions can be enabled via `typecast` parameter. See create record for details.

You can also include a single record object at the top level. Click here to show an example.

CODE

```
var Airtable = require('airtable');
var base = new Airtable({apiKey:
'YOUR_SECRET_API_TOKEN'}).base('appPVwvxpSYRf5d0K');

base('Customer Profiles').update([
  {
    "id": "recaF22BK3FJUAi6s",
    "fields": {
      "Customer ID": "CUST-001",
      "User": [
        "recLkv8HyT1Avv8z9"
      ],
      "First Name": "Jean",
      "Last Name": "Uwimana",
      "Telephone": "+250788123456",
      "District": "Kigali",
      "Sector": "Gasabo",
      "Cell": "Remera",
      "Village": "Nyabisindu",
      "Orders": [
        "reciKJhi7RNXV0p1X",
        "recqa1JAapDxavwOz"
      ],
      "Customer Insights (AI)": {
        "state": "generated",
        "value": "Customer is highly engaged, places regular orders,
and prefers early season purchases.",
        "isStale": true
      },
      "Customer Type (AI)": {
        "state": "generated",
        "value": "Commercial Farmer",
        "isStale": true
      }
    }
  },
  {
    "id": "recZ28mz9NEqopI0E",
    "fields": {
      "Customer ID": "CUST-002",
      "User": [
        "recqAIeDLQE7sp7fS"
      ],
      "First Name": "Aline",
      "Last Name": "Mukamana",
      "Telephone": "+250788654321",
      "District": "Southern",
      "Sector": "Huye",
      "Cell": "Ngoma",
      "Village": "Kigarama",
      "Orders": [
        "recfH73uq2xdb1k0H"
      ],
      "Customer Insights (AI)": {
        "state": "generated",
        "value": "Occasional buyer, tends to order during harvest
season.",
        "isStale": true
      },
      "Customer Type (AI)": {
        "state": "generated",
        "value": "Smallholder",
        "isStale": true
      }
    }
  },
  {
    "id": "rec1hkIEnKuHyGdtC",
    "fields": {
      "Customer ID": "CUST-003",
      "User": [
        "recm9iPNk1c9afReL"
```

```
      ],
      "First Name": "Eric",
      "Last Name": "Nshimiyimana",
      "Telephone": "+250789112233",
      "District": "Eastern",
      "Sector": "Rwamagana",
      "Cell": "Karenge",
      "Village": "Bweramana",
      "Orders": [
        "recMkt174sEFSB3nY"
      ],
      "Customer Insights (AI)": {
        "state": "generated",
        "value": "Frequent customer, high order value, potential for
upselling.",
        "isStale": true
      },
      "Customer Type (AI)": {
        "state": "generated",
        "value": "Commercial Farmer",
        "isStale": true
      }
    }
  }
], function(err, records) {
  if (err) {
    console.error(err);
    return;
  }
  records.forEach(function(record) {
    console.log(record.get('Customer ID'));
  });
});
```

OUTPUT

```
"CUST-001"
"CUST-002"
"CUST-003"
```

Delete Customer Profiles records

To delete Customer Profiles records, use the `destroy` method. Note that table names and table ids can be used interchangeably. Using table ids means table name changes do not require modifications to your API request.

The first parameter to `destroy` is an array of up to 10 record IDs to delete.

You can also set the first parameter to a record ID to delete a single record. Click here to show an example.

CODE

```
var Airtable = require('airtable');
var base = new Airtable({apiKey:
'YOUR_SECRET_API_TOKEN'}).base('appPVwvxpSYRf5d0K');

base('Customer Profiles').destroy(['recaF22BK3FJUAi6s',
'rec228mz9NEqopI0E'], function(err, deletedRecords) {
  if (err) {
    console.error(err);
    return;
  }
  console.log('Deleted', deletedRecords.length, 'records');
});
```

OUTPUT

```
Deleted 2 records
```

ORDERS TABLE

The id for Orders is `tblJTbpjFpkQEpIw5`. Table ids and table names can be used interchangeably in API requests. Using table ids means table name changes do not require modifications to your API request.

Fields

Each record in the Orders table contains the following fields:

Field names and field ids can be used interchangeably. Using field ids means field name changes do not require modifications to your API request. We recommend using field ids over field names where possible, to reduce modifications to your API request if the user changes the field name later.

FIELD NAME	FIELD ID	TYPE	DESCRIPTION
Name	`fldJnVxYxlJ01vm3N`	Text	`string` A single line of text.
Order ID	`fldAUpIUIiEfrpvYv`	Number	`number` An integer (whole number, e.g. 1, 32, 99). This field allows negative and positive numbers.
Customer	`fld89WoG2d253Osyu`	Link to another record	`array of record IDs (strings)` Array of linked records IDs from the Customer Profiles table.
Agent	`fldBoXJOltlubBJlp`	Link to another record	`array of record IDs (strings)` Array of linked records IDs from the Agent Profiles table.
Order Date	`fld9Kbi5fgrVAlORa`	Date	`string (ISO 8601 formatted date)` UTC date, e.g. "2014-09-05".
Status	`fldOKd4Sh`	Single select	`string` Selected option name

EXAMPLE VALUES

```
"Order #101"
"Order #102"
"Order #103"
"Order #104"
"Order #105"
```

EXAMPLE VALUES

```
101
102
103
104
105
```

EXAMPLE VALUE

```
["rec8116cdd76088af", "rec245db9343f55e8", "rec4f3bade67ff565"]
```

EXAMPLE VALUE

```
["rec8116cdd76088af", "rec245db9343f55e8", "rec4f3bade67ff565"]
```

EXAMPLE VALUES

```
"2025-08-12"
"2025-08-15"
"2025-08-10"
"2025-07-30"
"2025-08-17"
```

POSSIBLE VALUES

	f7BCsBM4		Selected option name. When creating or updating records, if the choice string does not exactly match an existing option, the request will fail with an `INVALID_MULTIPLE_CHOICE_OPTIONS` error unless the `typecast` parameter is enabled. If `typecast` is enabled, a new choice will be created if one does not exactly match.	["Pending", "Shipped", "Delivered", "Cancelled"]
Order Items	fldcVDMsE9gJ2rTwJ	Link to another record	`array of record IDs (strings)` Array of linked records IDs from the Order Line Items table.	EXAMPLE VALUE ["rec8116cdd76088af", "rec245db9343f55e8", "rec4f3bade67ff565"]
Total Amount	fldEvaj3NXioQVDB3	Currency	`number` Currency value. This field allows negative and positive numbers.	EXAMPLE VALUES 124500.00 89500.00 157000.00 67500.00 210000.00
Days to Delivery	fldosW4LiQY9G6Dns	Formula	`number, string, array of numbers or strings` Computed value: IF(Status = 'Delivered', DATETIME_DIFF(TODAY(), Order Date, 'days'), BLANK()).	EXAMPLE VALUES 7 20
Order Month	fldx9FWZIPN07XGMT	Formula	`number, string, array of numbers or strings` Computed value: DATETIME_FORMAT(Order Date, 'YYYY-MM').	EXAMPLE VALUES "\"2025-08\"" "\"2025-08\"" "\"2025-08\"" "\"2025-07\"" "\"2025-08\""
Customer District	fldsGDhH1xMCOQHRb	Lookup	`array of numbers, strings, booleans, or objects` Array of District fields in linked Customer Profiles records.	EXAMPLE VALUES ["Kigali"] ["Southern"] ["Eastern"] ["Northern"] ["Western"]
Agent Region	fldsjIll5M58QvAid	Lookup	`array of numbers, strings, booleans, or objects` Array of Region fields in linked Agent Profiles records.	EXAMPLE VALUES ["North"] ["South"] ["East"] ["West"] ["North"]
Order Item Count	fldMa7judvpLI3W71	Count	`number` Number of linked Order Line Items records.	EXAMPLE VALUES 1 1 1 1 1

Order Summary	fld9fOkUk uvzCqnrh
Order Risk Assessment	fldro4Ist nEjQ1dld

List Orders records

To list records in Orders, use the `select` method.

`select` returns a query object. To fetch the records matching that query, use the `eachPage` or `firstPage` method of the query object.

Returned records do not include any fields with "empty" values, e.g. `""`, `[]`, or `false`.

Note: Airtable's API only accepts request with a URL shorter than 16,000 characters. Encoded formulas may cause your requests to exceed this limit. To fix this issue you can instead make a POST request to `/v0/{baseId}/{tableIdOrName}/listRecords` while passing the parameters within the body of the request instead of the query parameters. See our support article on this for more information.

You can use the following parameters to filter, sort, and format the results:

fields
`array of strings`
optional

Only data for fields whose names are in this list will be included in the result. If you don't need every field, you can use this parameter to reduce the amount of data transferred.

For example, to only return data from Name and Order ID, pass in:

```
fields: ["Name", "Order ID"]
```

You can also perform the same action with field ids (they can be found in the fields section):

```
fields: ["fldJnVxYxlJ01vm3N", "fldAUpIUIiEfrpv
```

filterByFormula
`string`
optional

A formula used to filter records. The formula will be evaluated for each record, and if the result is not `0`, `false`, `""`, `NaN`, `[]`, or `#Error!` the record will be included in the response. We recommend testing your formula in the Formula field UI before using it in your API request.

If combined with the `view` parameter, only records in that view which satisfy the formula will be returned.

The formula must be encoded first before passing it as a value. You can use this tool to not only encode the formula but also create the entire url you need. For example, to only include records where Name isn't empty, pass in `NOT({Name} = '')` as a parameter like this:

```
filterByFormula: "NOT({Name} = '')"
```

Note: Airtable's API only accepts request with a URL shorter than 16,000 characters. Encoded formulas may cause your requests to exceed this limit. To fix this issue you can instead make a POST request to `/v0/{baseId}/{tableIdOrName}/listRecords` while passing the parameters within the body of the request instead of the query parameters. See our support article on this for more information.

maxRecords
`number`
optional

The maximum total number of records that will be returned in your requests. If this value is larger than `pageSize` (which is 100 by default), you may have to load multiple pages to reach this total. See the

CODE

```
var Airtable = require('airtable');
var base = new Airtable({apiKey:
'YOUR_SECRET_API_TOKEN'}).base('appPVwvxpSYRf5d0K');

base('Orders').select({
    // Selecting the first 3 records in Grid view:
    maxRecords: 3,
    view: "Grid view"
}).eachPage(function page(records, fetchNextPage) {
    // This function (`page`) will get called for each page of
records.

    records.forEach(function(record) {
        console.log('Retrieved', record.get('Name'));
    });

    // To fetch the next page of records, call `fetchNextPage`.
    // If there are more records, `page` will get called again.
    // If there are no more records, `done` will get called.
    fetchNextPage();

}, function done(err) {
    if (err) { console.error(err); return; }
});
```

OUTPUT

```
Retrieved Order #101
Retrieved Order #102
Retrieved Order #103
```

FETCH FIRST PAGE

```
// If you only want the first page of records, you can
// use `firstPage` instead of `eachPage`.
base('Orders').select({
    view: 'Grid view'
}).firstPage(function(err, records) {
    if (err) { console.error(err); return; }
    records.forEach(function(record) {
        console.log('Retrieved', record.get('Name'));
    });
});
```

FETCH ADDITIONAL RECORD METADATA

```
// If you want to fetch the number of comments for each record,
// include the `recordMetadata` param.
base('Orders').select({
    recordMetadata: ['commentCount']
}).firstPage(function(err, records) {
    if (err) { console.error(err); return; }
    records.forEach(function(record) {
        console.log('Retrieved a record with', record.commentCount,
'comments');
    });
});
```

Pagination section below for more.

pageSize
`number`
optional

The number of records returned in each request. Must be less than or equal to 100. Default is 100. See the Pagination section below for more.

sort
`array of objects`
optional

A list of sort objects that specifies how the records will be ordered. Each sort object must have a `field` key specifying the name of the field to sort on, and an optional `direction` key that is either `"asc"` or `"desc"`. The default direction is `"asc"`.

The `sort` parameter overrides the sorting of the view specified in the `view` parameter. If neither the `sort` nor the `view` parameter is included, the order of records is arbitrary.

For example, to sort records by Name in descending order, send these two query parameters:

```
sort%5B0%5D%5Bfield%5D=Name
sort%5B0%5D%5Bdirection%5D=desc
```

For example, to sort records by Name in descending order, pass in:

```
[{field: "Name", direction: "desc"}]
```

view
`string`
optional

The name or ID of a view in the Orders table. If set, only the records in that view will be returned. The records will be sorted according to the order of the view unless the `sort` parameter is included, which overrides that order. Fields hidden in this view will be returned in the results. To only return a subset of fields, use the `fields` parameter.

cellFormat
`string`
optional

The format that should be used for cell values. Supported values are:

`json`: cells will be formatted as JSON, depending on the field type.

`string`: cells will be formatted as user-facing strings, regardless of the field type. The `timeZone` and `userLocale` parameters are required when using `string` as the `cellFormat`.

Note: You should not rely on the format of these strings, as it is subject to change.

The default is `json`.

timeZone
`string`
optional

The time zone that should be used to format dates when using `string` as the `cellFormat`. This parameter is required when using `string` as the `cellFormat`.

userLocale
`string`
optional

The user locale that should be used to format dates when using `string` as the `cellFormat`. This parameter is required when using `string` as the `cellFormat`.

returnFieldsByFieldId
`boolean`
optional

An optional boolean value that lets you return field objects where the key is the field id.

This defaults to `false`, which returns field objects where the key is the field name.

recordMetadata
`array of strings`
optional

An optional field that, if includes `commentCount`, adds a `commentCount` read only property on each record returned.

Pagination

The server returns one page of records at a time. Each page will contain `pageSize` records, which is 100 by default.

To fetch the next page of records, call `fetchNextPage`.

Pagination will stop when you've reached the end of your table. If the `maxRecords` parameter is passed, pagination will stop once you've reached this maximum.

Iteration may timeout due to client inactivity or server restarts. In that case, the client will receive a 422 response with error message `LIST_RECORDS_ITERATOR_NOT_AVAILABLE`. It may then restart iteration from the beginning.

Retrieve a Orders record

To retrieve an existing record in Orders table, use the `find` method.

Any "empty" fields (e.g. `""`, `[]`, or `false`) in the record will not be returned.

CODE

```
var Airtable = require('airtable');
var base = new Airtable({apiKey:
'YOUR_SECRET_API_TOKEN'}).base('appPVwvxpSYRf5d0K');

base('Orders').find('reciKJhi7RNXV0p1X', function(err, record) {
    if (err) { console.error(err); return; }
    console.log('Retrieved', record.id);
});
```

OUTPUT

```
Retrieved reciKJhi7RNXV0p1X
```

Create Orders records

To create new records, use the `create` method. Note that table names and table ids can be used interchangeably. Using table ids means table name changes do not require modifications to your API request.

The first argument should be an array of up to 10 record objects. Each of these objects should have one key whose value is an inner object containing your record's cell values, keyed by either field name or field id.

Returns an array of record objects created if the call succeeded, including record IDs which will uniquely identify the records within AgriSales CRM.

Values for Days to Delivery, Order Month, Customer District, Agent Region and Order Item Count are automatically computed by Airtable and cannot be directly created.

The Airtable API will perform best-effort automatic data conversion from string values if the `typecast` parameter is passed in (click to show example). Automatic conversion is disabled by default to ensure data integrity, but it may be helpful for integrating with 3rd party data sources.

You can also include a single record object at the top level. Click here to show an example.

CODE

```
var Airtable = require('airtable');
var base = new Airtable({apiKey:
'YOUR_SECRET_API_TOKEN'}).base('appPVwvxpSYRf5d0K');

base('Orders').create([
  {
    "fields": {
      "Name": "Order #101",
      "Order ID": 101,
      "Customer": [
        "recaF22BK3FJUAi6s"
      ],
      "Agent": [
        "recOornkxOinHnWQG"
      ],
      "Order Date": "2025-08-12",
      "Status": "Delivered",
      "Order Items": [
        "rec8rb9IkSKKM29Wo"
      ],
      "Total Amount": 124500,
      "Order Summary": {
        "state": "generated",
        "value": "4 items: Maize Seed, Fertilizer, Pesticide,
Irrigation Kit",
        "isStale": true
      },
      "Order Risk Assessment": {
        "state": "generated",
        "value": "Low risk: Prompt delivery, full payment received",
        "isStale": true
      }
    }
  },
  {
    "fields": {
      "Name": "Order #102",
      "Order ID": 102,
      "Customer": [
        "recZ28mz9NEqopI0E"
      ],
      "Agent": [
        "recX8ujF2lUzDBNXh"
```

```
      ],
      "Order Date": "2025-08-15",
      "Status": "Pending",
      "Order Items": [
        "reckkCvI3AT4zRXi6"
      ],
      "Total Amount": 89500,
      "Order Summary": {
        "state": "generated",
        "value": "2 items: Bean Seed, Fertilizer",
        "isStale": true
      },
      "Order Risk Assessment": {
        "state": "generated",
        "value": "Medium risk: Awaiting payment confirmation",
        "isStale": true
      }
    }
  }
], function(err, records) {
  if (err) {
    console.error(err);
    return;
  }
  records.forEach(function (record) {
    console.log(record.getId());
  });
});
```

OUTPUT

```
reciKJhi7RNXV0p1X
recfH73uq2xdb1k0H
```

Update Orders records

To update Orders records, use the `update` or `replace` method. An `update` will only update the fields you include. Fields not included will be unchanged. A `replace` will perform a destructive update and clear all uninclueded cell values. A `replace` call on Orders records will always fail. The example at the right uses the non-destructive `update` method. Click here to show a destructive `replace` call.

The first argument should be an array of up to 10 record objects. Each of these objects should have an `id` property representing the record ID and a `fields` property which contains all of your record's cell values by field name or field id for all of your record's cell values by field name.

To link to new records in Customer, Agent and Order Items, add new linked record IDs to the existing array. Be sure to include all existing linked record IDs that you wish to retain. To unlink records, include the existing array of record IDs, excluding any that you wish to unlink.

Values for Days to Delivery, Order Month, Customer District, Agent Region and Order Item Count are automatically computed by Airtable and cannot be directly updated. You cannot clear these, even with a `replace` call.

Automatic data conversion for update actions can be enabled via `typecast` parameter. See create record for details.

You can also include a single record object at the top level. Click here to show an example.

CODE

```
var Airtable = require('airtable');
var base = new Airtable({apiKey:
'YOUR_SECRET_API_TOKEN'}).base('appPVwvxpSYRf5d0K');

base('Orders').update([
  {
    "id": "reciKJhi7RNXV0p1X",
    "fields": {
      "Name": "Order #101",
      "Order ID": 101,
      "Customer": [
        "recaF22BK3FJUAi6s"
      ],
      "Agent": [
        "recOornkxOinHnWQG"
      ],
      "Order Date": "2025-08-12",
      "Status": "Delivered",
      "Order Items": [
        "rec8rb9IkSKKM29Wo"
      ],
      "Total Amount": 124500,
      "Order Summary": {
        "state": "generated",
        "value": "4 items: Maize Seed, Fertilizer, Pesticide,
Irrigation Kit",
        "isStale": true
      },
      "Order Risk Assessment": {
        "state": "generated",
        "value": "Low risk: Prompt delivery, full payment received",
        "isStale": true
      }
    }
  },
  {
    "id": "recfH73uq2xdb1k0H",
    "fields": {
      "Name": "Order #102",
      "Order ID": 102,
      "Customer": [
        "recZ28mz9NEqopI0E"
      ],
      "Agent": [
        "recX8ujF2lUzDBNXh"
```

```
      ],
      "Order Date": "2025-08-15",
      "Status": "Pending",
      "Order Items": [
        "reckkCvI3AT4zRXi6"
      ],
      "Total Amount": 89500,
      "Order Summary": {
        "state": "generated",
        "value": "2 items: Bean Seed, Fertilizer",
        "isStale": true
      },
      "Order Risk Assessment": {
        "state": "generated",
        "value": "Medium risk: Awaiting payment confirmation",
        "isStale": true
      }
    }
  },
  {
    "id": "recMkt174sEFSB3nY",
    "fields": {
      "Name": "Order #103",
      "Order ID": 103,
      "Customer": [
        "rec1hkIEnKuHyGdtC"
      ],
      "Agent": [
        "recOzIlfCcSHaUwcs"
      ],
      "Order Date": "2025-08-10",
      "Status": "Shipped",
      "Order Items": [
        "rec3hK4SLVVcJYKTf"
      ],
      "Total Amount": 157000,
      "Order Summary": {
        "state": "generated",
        "value": "5 items: Maize Seed, Fertilizer, Herbicide,
Sprayer, Gloves",
        "isStale": true
      },
      "Order Risk Assessment": {
        "state": "generated",
        "value": "Low risk: Reliable customer, on-time shipment",
        "isStale": true
      }
    }
  }
], function(err, records) {
  if (err) {
    console.error(err);
    return;
  }
  records.forEach(function(record) {
    console.log(record.get('Name'));
  });
});
```

OUTPUT

```
"Order #101"
"Order #102"
"Order #103"
```

Delete Orders records

To delete Orders records, use the `destroy` method. Note that table names and table ids can be used interchangeably. Using table ids means table name changes do not require modifications to your API request.

The first parameter to `destroy` is an array of up to 10 record IDs to delete.

You can also set the first parameter to a record ID to delete a single record. Click here to show an example.

CODE

```
var Airtable = require('airtable');
var base = new Airtable({apiKey:
'YOUR_SECRET_API_TOKEN'}).base('appPVwvxpSYRf5d0K');

base('Orders').destroy(['reciKJhi7RNXV0p1X', 'recfH73uq2xdb1k0H'],
function(err, deletedRecords) {
  if (err) {
    console.error(err);
    return;
  }
  console.log('Deleted', deletedRecords.length, 'records');
});
```

OUTPUT

```
Deleted 2 records
```

```
Deleted 2 records
```

PRODUCTS TABLE

The id for Products is `tblhFrENfLsrg7TCi`. Table ids and table names can be used interchangeably in API requests. Using table ids means table name changes do not require modifications to your API request.

Fields

Each record in the Products table contains the following fields:

Field names and field ids can be used interchangeably. Using field ids means field name changes do not require modifications to your API request. We recommend using field ids over field names where possible, to reduce modifications to your API request if the user changes the field name later.

FIELD NAME	FIELD ID	TYPE	DESCRIPTION
Product Name	`fldFUGsjHD2Ta330y`	Text	`string` A single line of text.

EXAMPLE VALUES

```
"Maize Hybrid Seed"
"NPK 17-17-17 Fertilizer"
"Handheld Sprayer 16L"
"Tomato Seeds (Roma VF)"
"Organic Neem Pesticide"
```

SKU	`fldiMdlWu7H4u9MIJ`	Text	`string` A single line of text.

EXAMPLE VALUES

```
"SEED-001"
"FERT-002"
```

```
"EQUIP-003"
"SEED-004"
"PEST-005"
```

Description | `fldyXp8NNLiHI5cUa` | Long text | `string`

Multiple lines of text, which may contain "mention tokens", e.g. `<airtable:mention id="menE1i9oBaGX3DseR">@Alex</airtable:mention>`

EXAMPLE VALUES

```
"High-yield maize hybrid seed suitable for all regions. Disease resistant and fast germinating."
"Balanced NPK fertilizer for improved crop growth. Suitable for vegetables and cereals."
"Durable 16-liter manual sprayer for pesticides and fertilizers. Adjustable nozzle included."
"Roma VF tomato seeds with high disease resistance and excellent fruit quality. Early maturing."
"Natural neem-based pesticide for organic farming. Controls a wide range of pests safely."
```

Image | `fldSnILJXaxJljpUX` | Attachment | `array of attachment objects`

Each attachment object may contain the following properties. To see which fields are required or optional, please consult the relevant section: retrieve, create, update, or delete.

Because this field is configured to show attachments in reversed order, the order of attachments in the app will be reversed compared to what you see here.

`id` — `string`
unique attachment id

`url` — `string`
url, e.g. "https://v5.airtableusercontent.com/foo".

Note: URLs returned will expire 2 hours after being returned from our API. If you want to persist the attachments, we recommend downloading them instead of saving the URL. See our support article for more information.

`filename` — `string`
filename, e.g. "foo.jpg"

`size` — `number`
file size, in bytes

`type` — `string`
content type, e.g. "image/jpeg"

`width` — `number`
`height` — `number`
width/height, in pixels (these may be available if the attachment is an image)

`thumbnails.small.url` — `string`
`thumbnails.large.url` — `string`
url of small/large thumbnails (these may be available if the attachment is an image or document). See notes under `url` about the lifetime of these URLs.

`thumbnails.small.width` — `number`
`thumbnails.small.height` — `number`
`thumbnails.large.width` — `number`
`thumbnails.large.height` — `number`
width/height of small/large thumbnails, in pixels (these will be available if the corresponding thumbnail url is available)

EXAMPLE VALUES


```
[
    {
        "id": "att5F62EnM0hVF1SS",
        "width": 600,
        "height": 401,
        "url":
"https://v5.airtableusercontent.com/v3/u/44/44/1755612000000/UE7co6RvrNhwReeIQ8ozpQ/-3wVmYTN6s-hlACbftO_a9L-U2jLLEZdzVUS4MQPsyTpZNNfFH-vPP0sreKANzI5P3FPx-eEgruvgmejBwpcdoggoyEzJsfIFELH6FimVWjomSYhwoxbDLlRnc49WbET4FI_cFe7Ux9JAcHJLo0vDcNm2vP9P2nq21oF6izDNAc/7XdhGlzdjmgTgrS-TLNpI3o197bggqfk5wMmasKYNmQ",
        "filename": "abstract_20.png",
        "size": 194316,
        "type": "image/png",
        "thumbnails": {
            "small": {
                "url":
"https://v5.airtableusercontent.com/v3/u/44/44/1755612000000/vil8J7pabSTI8p7nr5PBIQ/ayOLXZ5IYNGAwenv-9bCKhI_GtPz_r6fKfeXE6O9ntgKl0jQdTn0wXBrpqLlIq5XqPXes3CO6bVJaIyM1JvliuASvEeqB44ItCnz0xsagwT4v25BvrkPjjecZEvgVsraNWi6oshOLRYWYH2b95T77w/_zTadjcyu0seiRqjy7CdBTNqh0aPgaJwVbG0ROD3J5k",
                "width": 54,
                "height": 36
            },
            "large": {
                "url":
"https://v5.airtableusercontent.com/v3/u/44/44/1755612000000/6eeRclZaRWATVCaJlFGilw/gUiAc13i9AP4xpNrZ1TegDvqleXxsPAaovJpJAsnNSYiGetWmiiG-LQJSgYG2FJl0H0uIUe5ZifVCypXdYOC_dveONHb8PjBGiX-w-6vl5vkvRs2tihNR3uJBzzsQQTQww7A54ttIncYOt-mHtEihg/Wf2i_TeSJ5jAHdSdYfdN9wYshk8pKZeJOheEBpFDDn8",
                "width": 600,
                "height": 401
            },
            "full": {
                "url":
"https://v5.airtableusercontent.com/v3/u/44/44/1755612000000/yMpIHZ8RCtqeiJ4wkWSKrQ/oaoluG7mN3MdGHYyzFT2xOd6e5jaOZDsbnyu1xQCbQc4fCjWcxla2hXqfI6JHVX2c9tIaI01BsnHha51Erv64U16nLYsWli55C_PRTqciV8E349TpVpdNbn6XWT30qZUslC58cyHS3aVQeG-hEY2OQ/B9z4xkRX_UfFfQX035o1lJlAB3DvRwW3ujl-1Ai22Mc",
                "width": 600,
                "height": 401
            }
        }
    }
]
[
    {
        "id": "att03ZnOwYNHvkIeP",
        "width": 600,
        "height": 401,
        "url":
"https://v5.airtableusercontent.com/v3/u/44/44/1755612000000/KTiKdSdppt8JXXtqOroqVQ/N0OuDmb-PSogEkYa-xlHJpN_98uQkhdCKOB65srH2be84uvnNmOoWTg4AMmt8mwDmg9nihg7nSslr2-DqFEgBi6IKAfgOx695oPc4b0wZxB8J6DYvL5g9xfrH80-AAjmXb-IRGPo8STMM_-IMVTy43ulgVfsq1uVEpC7ArYhuKs/_yCUPapiZd1VP9x1G_9P2xz1lReWSk0Z8ROZtOrUgKo",
        "filename": "abstract_2.png",
        "size": 354054,
        "type": "image/png",
        "thumbnails": {
            "small": {
```

```
                "url":
"https://v5.airtableusercontent.com/v3/u/44/44/1755612000000/Gh-
TvnHAdWE6CxD5rehD3Q/siDK6ivgOwzSJqm31roNrh9e1ye4Kj457FhLO4FxBFrND7KOg
2Y5WouoXSF9q1fO_PHehODEDfMsk9j83H9ScEbMsfuy7OXbRMaRUvnIosSBdfnsP25FBH
5M51xEtzpEPPOelwJb7gXypgulgO0aUA/ajG5rRh5Xrt0poT4GpiWPgFuVDTvgfXTe8hz
sfqeHPI",
                "width": 54,
                "height": 36
            },
            "large": {
                "url":
"https://v5.airtableusercontent.com/v3/u/44/44/1755612000000/ma7NlgkV
zr375zDD1UAc1w/mO1-
BDEf4YCK08X5tRo1JRsKmXQ4YVoM0SG1fK7NS11yIqOMSl21FXzV2vmGF53Y-
UECXmB3snWuhah0yH88Oiu3lbxg8cgP5U7NiofIboK3SoyQji9hwrb9aHsaBbbL-
515FkCmaor-OgKwWffneg/1ieoMEge4ffhz7_u_Ms4AOOBZOiDs2hZxayok4X9wp0",
                "width": 600,
                "height": 401
            },
            "full": {
                "url":
"https://v5.airtableusercontent.com/v3/u/44/44/1755612000000/B1wkbunz
QcsDLGEXpJtIEw/n9irwW5d62V3FGck6bWIQXeT_R9mlI1Kp3UyZncLyNYcJUwlrE06kc
ntyvYuSxuHLv0Ci6wrmk9-
Pap3oeAJQsylKJi7CuxZQUGJyMicLMMKvYLHBehXlM_d49E1XXy0L4PvyJKFK1e8B5BPR
FtPkA/1S4gsqBECZLSAhuzFiVNr4rLTAeNi0rx860YHIbvXVk",
                "width": 600,
                "height": 401
            }
        }
    }
]
```


Unit Price	fld16kHrjSOShXlSt	Currency	number Currency value. This field allows negative and positive numbers.	EXAMPLE VALUES 3500.00 2500.00 18000.00 1200.00 4200.00
Stock	fld05Qlw8c07mb4Hv	Number	number An integer (whole number, e.g. 1, 32, 99). This field allows negative and positive numbers.	EXAMPLE VALUES 120 60 25 200 40
Total Units Sold	fldjtnM6SFvHO9poU	Rollup	number or string Computed value: SUM(values) for Quantity in Order Line Items.	EXAMPLE VALUES 10 5 2 8 20
Order Line Items	fldjiCupbRo8XAHeg	Link to another record	array of record IDs (strings) Array of linked records IDs from the Order Line Items table.	EXAMPLE VALUE ["rec8116cdd76088af", "rec245db9343f55e8", "rec4f3bade67ff565"]
Total Revenue	fldU4aZrop0LZ1GUz	Rollup	number or string Computed value: SUM(values) for Total Price in Order Line Items.	EXAMPLE VALUES 35000 12500 36000 9600 84000
Low Stock Warning	fldS9kTae5dVjCGwz	Formula	number, string, array of numbers or strings Computed value: IF(Stock < 10, 'Low Stock', 'OK').	EXAMPLE VALUES "\"OK\"" "\"OK\"" "\"OK\"" "\"OK\"" "\"OK\""
Product Category	fldKVMFeu6MSzqMWB	Single select	string Selected option name. When creating or updating records, if the choice string does not exactly match an existing option, the request will fail with an	POSSIBLE VALUES ["Fertilizer", "Seed", "Pesticide", "Equipment", "Other"]

`INVALID_MULTIPLE_CHOICE_OPTIONS` error unless the `typecast` parameter is enabled. If `typecast` is enabled, a new choice will be created if one does not exactly match.

Product Summary (AI)	`fldne6NWte673zmxT`
Product Tag Suggestions (AI)	`fldXV3Wn399xmTIFd`

List Products records

To list records in Products, use the `select` method.

`select` returns a query object. To fetch the records matching that query, use the `eachPage` or `firstPage` method of the query object.

Returned records do not include any fields with "empty" values, e.g. `""`, `[]`, or `false`.

Note: Airtable's API only accepts request with a URL shorter than 16,000 characters. Encoded formulas may cause your requests to exceed this limit. To fix this issue you can instead make a POST request to `/v0/{baseId}/{tableIdOrName}/listRecords` while passing the parameters within the body of the request instead of the query parameters. See our support article on this for more information.

You can use the following parameters to filter, sort, and format the results:

fields
`array of strings`
optional

Only data for fields whose names are in this list will be included in the result. If you don't need every field, you can use this parameter to reduce the amount of data transferred.

For example, to only return data from Product Name and SKU, pass in:

```
fields: ["Product Name", "SKU"]
```

You can also perform the same action with field ids (they can be found in the fields section):

```
fields: ["fldFUGsjHD2Ta330y", "fldiMd1Wu7H4u9M
```

filterByFormula
`string`
optional

A formula used to filter records. The formula will be evaluated for each record, and if the result is not `0`, `false`, `""`, `NaN`, `[]`, or `#Error!` the record will be included in the response. We recommend testing your formula in the Formula field UI before using it in your API request.

If combined with the `view` parameter, only records in that view which satisfy the formula will be returned.

The formula must be encoded first before passing it as a value. You can use this tool to not only encode the formula but also create the entire url you need. For example, to only include records where Product Name isn't empty, pass in `NOT({Product Name} = '')` as a parameter like this:

```
filterByFormula: "NOT({Product Name} = '')"
```

Note: Airtable's API only accepts request with a URL shorter than 16,000 characters. Encoded formulas may cause your requests to exceed this limit. To fix this issue you can instead make a POST request to

CODE

```
var Airtable = require('airtable');
var base = new Airtable({apiKey:
'YOUR_SECRET_API_TOKEN'}).base('appPVwvxpSYRf5d0K');

base('Products').select({
    // Selecting the first 3 records in Grid view:
    maxRecords: 3,
    view: "Grid view"
}).eachPage(function page(records, fetchNextPage) {
    // This function (`page`) will get called for each page of
records.

    records.forEach(function(record) {
        console.log('Retrieved', record.get('Product Name'));
    });

    // To fetch the next page of records, call `fetchNextPage`.
    // If there are more records, `page` will get called again.
    // If there are no more records, `done` will get called.
    fetchNextPage();

}, function done(err) {
    if (err) { console.error(err); return; }
});
```

OUTPUT

```
Retrieved Maize Hybrid Seed
Retrieved NPK 17-17-17 Fertilizer
Retrieved Handheld Sprayer 16L
```

FETCH FIRST PAGE

```
// If you only want the first page of records, you can
// use `firstPage` instead of `eachPage`.
base('Products').select({
    view: 'Grid view'
}).firstPage(function(err, records) {
    if (err) { console.error(err); return; }
    records.forEach(function(record) {
        console.log('Retrieved', record.get('Product Name'));
    });
});
```

FETCH ADDITIONAL RECORD METADATA

```
// If you want to fetch the number of comments for each record,
// include the `recordMetadata` param.
base('Products').select({
    recordMetadata: ['commentCount']
}).firstPage(function(err, records) {
    if (err) { console.error(err); return; }
    records.forEach(function(record) {
        console.log('Retrieved a record with', record.commentCount,
'comments');
    });
});
```

`/v0/{baseId}/{tableIdOrName}/listRecords` while passing the parameters within the body of the request instead of the query parameters. See our support article on this for more information.

maxRecords
`number`
optional

The maximum total number of records that will be returned in your requests. If this value is larger than `pageSize` (which is 100 by default), you may have to load multiple pages to reach this total. See the Pagination section below for more.

pageSize
`number`
optional

The number of records returned in each request. Must be less than or equal to 100. Default is 100. See the Pagination section below for more.

sort
`array of objects`
optional

A list of sort objects that specifies how the records will be ordered. Each sort object must have a `field` key specifying the name of the field to sort on, and an optional `direction` key that is either `"asc"` or `"desc"`. The default direction is `"asc"`.

The `sort` parameter overrides the sorting of the view specified in the `view` parameter. If neither the `sort` nor the `view` parameter is included, the order of records is arbitrary.

For example, to sort records by Product Name in descending order, send these two query parameters:

```
sort%5B0%5D%5Bfield%5D=Product%20Name
sort%5B0%5D%5Bdirection%5D=desc
```

For example, to sort records by Product Name in descending order, pass in:

```
[{field: "Product Name", direction: "desc"}]
```

view
`string`
optional

The name or ID of a view in the Products table. If set, only the records in that view will be returned. The records will be sorted according to the order of the view unless the `sort` parameter is included, which overrides that order. Fields hidden in this view will be returned in the results. To only return a subset of fields, use the `fields` parameter.

cellFormat
`string`
optional

The format that should be used for cell values. Supported values are:

`json`: cells will be formatted as JSON, depending on the field type.

`string`: cells will be formatted as user-facing strings, regardless of the field type. The `timeZone` and `userLocale` parameters are required when using `string` as the `cellFormat`.

Note: You should not rely on the format of these strings, as it is subject to change.

The default is `json`.

timeZone
`string`
optional

The time zone that should be used to format dates when using `string` as the `cellFormat`. This parameter is required when using `string` as the `cellFormat`.

userLocale
`string`
optional

The user locale that should be used to format dates when using `string` as the `cellFormat`. This parameter is required when using `string` as the `cellFormat`.

returnFieldsByFieldId
`boolean`
optional

An optional boolean value that lets you return field objects where the key is the field id.

This defaults to `false`, which returns field objects

This defaults to `false`, which returns field objects where the key is the field name.

recordMetadata
`array of strings`
optional

An optional field that, if includes `commentCount`, adds a `commentCount` read only property on each record returned.

Pagination

The server returns one page of records at a time. Each page will contain `pageSize` records, which is 100 by default.

To fetch the next page of records, call `fetchNextPage`.

Pagination will stop when you've reached the end of your table. If the `maxRecords` parameter is passed, pagination will stop once you've reached this maximum.

Iteration may timeout due to client inactivity or server restarts. In that case, the client will receive a 422 response with error message `LIST_RECORDS_ITERATOR_NOT_AVAILABLE`. It may then restart iteration from the beginning.

Retrieve a Products record

To retrieve an existing record in Products table, use the `find` method.

Any "empty" fields (e.g. `""`, `[]`, or `false`) in the record will not be returned.

In attachment objects included in the retrieved record (Image), only `id`, `url`, and `filename` are always returned. Other attachment properties may not be included. Note: Attachment URLs returned will expire 2 hours after being returned from our API. If you want to persist the attachments, we recommend downloading them instead of saving the URL. See our support article for more information.

CODE

```
var Airtable = require('airtable');
var base = new Airtable({apiKey:
'YOUR_SECRET_API_TOKEN'}).base('appPVwvxpSYRf5d0K');

base('Products').find('recCeXjucglJneIEU', function(err, record) {
    if (err) { console.error(err); return; }
    console.log('Retrieved', record.id);
});
```

OUTPUT

```
Retrieved recCeXjucglJneIEU
```

Create Products records

To create new records, use the `create` method. Note that table names and table ids can be used interchangeably. Using table ids means table name changes do not require modifications to your API request.

The first argument should be an array of up to 10 record objects. Each of these objects should have one key whose value is an inner object containing your record's cell values, keyed by either field name or field id.

Returns an array of record objects created if the call succeeded, including record IDs which will uniquely identify the records within AgriSales CRM .

To create new attachments in Image , set the field value to an array of attachment objects. When creating an attachment, `url` is required, and `filename` is optional. Airtable will download the file at the given `url` and keep its own copy of it. All other attachment object properties will be generated server-side soon afterward.

Note that in most cases the API does not currently return an error code for failed attachment object creation given attachment uploading happens in an asynchronous manner, such cases will manifest with the attachment object either being cleared from the cell or persisted with generated URLs that return error responses when queried. If the same attachment URL fails to upload multiple times in a short time interval then * the API may return an ATTACHMENTS_FAILED_UPLOADING error code in the details field of the response and the attachment object will * be cleared from the cell synchronously.

We also require URLs used to upload have the https:// or http:// protocol (Note: http:// support will be removed in the near future), have a limit of 3 max redirects, and a file size limit of 1GB. In addition, URLs must be publicly accessible, in cases where cookie authentication or logging in to access the file is

CODE

```
var Airtable = require('airtable');
var base = new Airtable({apiKey:
'YOUR_SECRET_API_TOKEN'}).base('appPVwvxpSYRf5d0K');

base('Products').create([
  {
    "fields": {
      "Product Name": "Maize Hybrid Seed",
      "SKU": "SEED-001",
      "Description": "High-yield maize hybrid seed suitable for all
regions. Disease resistant and fast germinating.",
      "Image": [
        {
          "url":
"https://v5.airtableusercontent.com/v3/u/44/44/1755612000000/UE7co6Rv
rNhwReeIQ8ozpQ/-3wVmYTN6s-hlACbftO_a9L-U2jLLEZdzVUS4MQPsyTpZNNfFH-
vPP0sreKANzI5P3FPx-
eEgruvgmejBwpcdoggoyEzJsfIFELH6FimVWjomSYhwoxbDLlRnc49WbET4FI_cFe7Ux9
JAcHJLo0vDcNm2vP9P2nq21oF6izDNAc/7XdhGlzdjmgTgrS-
TLNpI3o197bggqfk5wMmasKYNmQ"
        }
      ],
      "Unit Price": 3500,
      "Stock": 120,
      "Order Line Items": [
        "rec8rb9IkSKKM29Wo"
      ],
      "Product Category": "Seed",
      "Product Summary (AI)": {
        "state": "generated",
        "value": "Ideal for farmers seeking robust maize harvests.
Disease resistance and fast germination make it a top choice.",
```

required, the login page HTML will be downloaded instead of the file.

Attachment URLs returned will expire 2 hours after being returned from our API. If you want to persist the attachments, we recommend downloading them instead of saving the URL.

If too many attachments are uploaded within a short period of time, the server may return a partial failure on record creation with an "Attachment Upload Rate Too High" error. See our support article for more information.

Values for `Total Units Sold`, `Total Revenue` and `Low Stock Warning` are automatically computed by Airtable and cannot be directly created.

The Airtable API will perform best-effort automatic data conversion from string values if the `typecast` parameter is passed in (click to show example). Automatic conversion is disabled by default to ensure data integrity, but it may be helpful for integrating with 3rd party data sources.

You can also include a single record object at the top level. Click here to show an example.

```
        "isStale": true
      },
      "Product Tag Suggestions (AI)": {
        "state": "generated",
        "value": "maize, hybrid, seed, high-yield, agriculture",
        "isStale": true
      }
    }
  },
  {
    "fields": {
      "Product Name": "NPK 17-17-17 Fertilizer",
      "SKU": "FERT-002",
      "Description": "Balanced NPK fertilizer for improved crop
growth. Suitable for vegetables and cereals.",
      "Image": [
        {
          "url":
"https://v5.airtableusercontent.com/v3/u/44/44/1755612000000/KTiKdSdp
pt8JXXtqOroqVQ/N0OuDmb-PSogEkYa-
xlHJpN_98uQkhdCKOB65srH2be84uvnNmOoWTg4AMmt8mwDmg9nihg7nSslr2-
DqFEgBi6IKAfgOx695oPc4b0wZxB8J6DYwL5g9xfrH80-AAjmXb-IRGPo8STMM_-
IMVTy43ulgVfsq1uVEpC7ArYhuKs/_yCUPapiZd1VP9x1G_9P2xz1lReWSk0Z8ROZtOrU
gKo"
        }
      ],
      "Unit Price": 2500,
      "Stock": 60,
      "Order Line Items": [
        "reckkCvI3AT4zRXi6"
      ],
      "Product Category": "Fertilizer",
      "Product Summary (AI)": {
        "state": "generated",
        "value": "Boosts plant growth and yield with balanced
nutrients. Especially effective for vegetables and cereals.",
        "isStale": true
      },
      "Product Tag Suggestions (AI)": {
        "state": "generated",
        "value": "fertilizer, NPK, balanced, crop growth,
vegetables",
        "isStale": true
      }
    }
  }
], function(err, records) {
  if (err) {
    console.error(err);
    return;
  }
  records.forEach(function (record) {
    console.log(record.getId());
  });
});
```

OUTPUT

```
recCeXjucglJneIEU
recyYAWHWM64HCHK2
```

Update Products records

To update `Products` records, use the `update` or `replace` method. An `update` will only update the fields you include. Fields not included will be unchanged. A `replace` will perform a destructive update and clear all unincluded cell values. A `replace` call on `Products` records will always fail. The example at the right uses the non-destructive `update` method. Click here to show a destructive `replace` call.

The first argument should be an array of up to 10 record objects. Each of these objects should have an `id` property representing the record ID and a `fields` property which contains all of your record's cell values by field name or field id for all of your record's cell values by field name.

To add attachments to `Image`, add new attachment objects to the existing array. Be sure to include all existing attachment objects that you wish to retain, to keep preexisting attachments providing `id` is required (which can be retrieved using the retrieve endpoint), other fields are ignored. For the new attachments being added, `url` is required, and `filename` is optional. To remove attachments, include the existing array of attachment objects, excluding any that you wish to

CODE

```
var Airtable = require('airtable');
var base = new Airtable({apiKey:
'YOUR_SECRET_API_TOKEN'}).base('appPVwvxpSYRf5d0K');

base('Products').update([
  {
    "id": "recCeXjucglJneIEU",
    "fields": {
      "Product Name": "Maize Hybrid Seed",
      "SKU": "SEED-001",
      "Description": "High-yield maize hybrid seed suitable for all
regions. Disease resistant and fast germinating.",
      "Image": [
        {
          "id": "att5F62EnM0hVF1SS"
        }
      ],
      "Unit Price": 3500,
      "Stock": 120
```

remove.

Note that in most cases the API does not currently return an error code for failed attachment object creation given attachment uploading happens in an asynchronous manner, such cases will manifest with the attachment object either being cleared from the cell or persisted with generated URLs that return error responses when queried. If the same attachment URL fails to upload multiple times in a short time interval then * the API may return an ATTACHMENTS_FAILED_UPLOADING error code in the details field of the response and the attachment object will * be cleared from the cell synchronously.

We also require URLs used to upload have the https:// or http:// protocol (Note: http:// support will be removed in the near future), have a limit of 3 max redirects, and a file size limit of 1GB. In addition, URLs must be publicly accessible, in cases where cookie authentication or logging in to access the file is required, the login page HTML will be downloaded instead of the file.

If too many attachments are uploaded within a short period of time, the server may return a partial failure on record creation with an "Attachment Upload Rate Too High" error.

To link to new records in `Order Line Items`, add new linked record IDs to the existing array. Be sure to include all existing linked record IDs that you wish to retain. To unlink records, include the existing array of record IDs, excluding any that you wish to unlink.

`Description` may contain "mention tokens". A *mention token* corresponds to a "@mention" in Airtable's user interface; here in the API it will look like `<airtable:mention id="menE1i9oBaGX3DseR">@Alex</airtable:mention>`. Mention tokens cannot be created via this API and should be left intact (or wholly removed) when updating long text fields.

Values for `Total Units Sold`, `Total Revenue` and `Low Stock Warning` are automatically computed by Airtable and cannot be directly updated. You cannot clear these, even with a `replace` call.

Automatic data conversion for update actions can be enabled via `typecast` parameter. See create record for details.

You can also include a single record object at the top level. Click here to show an example.


```
      "Order Line Items": [
        "rec8rb9IkSKKM29Wo"
      ],
      "Product Category": "Seed",
      "Product Summary (AI)": {
        "state": "generated",
        "value": "Ideal for farmers seeking robust maize harvests.
Disease resistance and fast germination make it a top choice.",
        "isStale": true
      },
      "Product Tag Suggestions (AI)": {
        "state": "generated",
        "value": "maize, hybrid, seed, high-yield, agriculture",
        "isStale": true
      }
    }
  },
  {
    "id": "recyYAWHWM64HCHK2",
    "fields": {
      "Product Name": "NPK 17-17-17 Fertilizer",
      "SKU": "FERT-002",
      "Description": "Balanced NPK fertilizer for improved crop
growth. Suitable for vegetables and cereals.",
      "Image": [
        {
          "id": "att03ZnOwYNHvkIeP"
        }
      ],
      "Unit Price": 2500,
      "Stock": 60,
      "Order Line Items": [
        "reckkCvI3AT4zRXi6"
      ],
      "Product Category": "Fertilizer",
      "Product Summary (AI)": {
        "state": "generated",
        "value": "Boosts plant growth and yield with balanced
nutrients. Especially effective for vegetables and cereals.",
        "isStale": true
      },
      "Product Tag Suggestions (AI)": {
        "state": "generated",
        "value": "fertilizer, NPK, balanced, crop growth,
vegetables",
        "isStale": true
      }
    }
  },
  {
    "id": "recoUf2X7aSLG4iXz",
    "fields": {
      "Product Name": "Handheld Sprayer 16L",
      "SKU": "EQUIP-003",
      "Description": "Durable 16-liter manual sprayer for pesticides
and fertilizers. Adjustable nozzle included.",
      "Image": [
        {
          "id": "att6vCXZx2Zz03onU"
        }
      ],
      "Unit Price": 18000,
      "Stock": 25,
      "Order Line Items": [
        "rec3hK4SLVVcJYKTf"
      ],
      "Product Category": "Equipment",
      "Product Summary (AI)": {
        "state": "generated",
        "value": "Essential tool for even application of
agrochemicals. Large capacity for efficient fieldwork.",
        "isStale": true
      },
      "Product Tag Suggestions (AI)": {
        "state": "generated",
        "value": "sprayer, equipment, pesticide, fertilizer, manual",
        "isStale": true
      }
    }
  }
], function(err, records) {
  if (err) {
    console.error(err);
    return;
```

```
  }
  records.forEach(function(record) {
    console.log(record.get('Product Name'));
  });
});
```

OUTPUT

```
"Maize Hybrid Seed"
"NPK 17-17-17 Fertilizer"
"Handheld Sprayer 16L"
```

Delete Products records

To delete `Products` records, use the `destroy` method. Note that table names and table ids can be used interchangeably. Using table ids means table name changes do not require modifications to your API request.

The first parameter to `destroy` is an array of up to 10 record IDs to delete.

You can also set the first parameter to a record ID to delete a single record. Click here to show an example.

CODE

```
var Airtable = require('airtable');
var base = new Airtable({apiKey:
'YOUR_SECRET_API_TOKEN'}).base('appPVwvxpSYRf5d0K');

base('Products').destroy(['recCeXjucglJneIEU', 'recyYAWHWM64HCHK2'],
function(err, deletedRecords) {
  if (err) {
    console.error(err);
    return;
  }
  console.log('Deleted', deletedRecords.length, 'records');
});
```

OUTPUT

```
Deleted 2 records
```

ORDER LINE ITEMS TABLE

The id for Order Line Items is `tblsLfAioYzMuyEIb`. Table ids and table names can be used interchangeably in API requests. Using table ids means table name changes do not require modifications to your API request.

Fields

Each record in the Order Line Items table contains the following fields:

Field names and field ids can be used interchangeably. Using field ids means field name changes do not require modifications to your API request. We recommend using field ids over field names where possible, to reduce modifications to your API request if the user changes the field name later.

FIELD NAME	FIELD ID	TYPE	DESCRIPTION	EXAMPLE VALUES

Name	`fld9CV784ELrr64Xs`	Text	`string`
			A single line of text.
				`"Maize Hybrid Seed"`
				`"Organic Fertilizer"`
				`"Drip Irrigation Kit"`
				`"Pesticide - Neem Oil"`
				`"Hybrid Tomato Seedlings"`

Order	`fldgpeDz9sRBgE5sP`	Link to another record	`array of record IDs (strings)` Array of linked records IDs from the Orders table.	`["rec8116cdd76088af", "rec245db9343f55e8", "rec4f3bade67ff565"]`
Product	`fld3ygouM3Z8YmowD`	Link to another record	`array of record IDs (strings)` Array of linked records IDs from the Products table.	`["rec8116cdd76088af", "rec245db9343f55e8", "rec4f3bade67ff565"]`

Quantity	`fldZMIhumQMRE4rB5`	Number	`number`
			An integer (whole number, e.g. 1, 32, 99). This field allows negative and positive numbers.
				`10`
				`5`
				`2`
				`8`
				`20`

Unit Price (from Product)	`fldrb1bqicIEfFfK5`	Lookup	`array of numbers, strings, booleans, or objects`
			Array of Unit Price fields in linked Products records.
				`[ 3500 ]`
				`[ 2500 ]`
				`[ 18000 ]`
				`[ 1200 ]`
				`[ 4200 ]`

Total Price	`fld9bjfl3ySezW1hX`	Formula	`number, string, array of numbers or strings`
			Computed value: Quantity * Unit Price (from Product).
				`35000`
				`12500`
				`36000`
				`9600`
				`84000`

Order Date	`fld1PBHtWiQ9WrQwE`	Lookup	`array of numbers, strings, booleans, or objects`
			Array of Order Date fields in linked Orders records.
				`[ "2025-08-12" ]`
				`[ "2025-08-15" ]`
				`[ "2025-08-10" ]`
				`[ "2025-07-30"`

Product Description	fldv13TVhvXdS7FbG	Lookup	array of numbers, strings, booleans, or objects Array of Description fields in linked Products records.	EXAMPLE VALUES ["High-yield maize hybrid seed suitable for all regions. Disease resistant and fast germinating."] ["Balanced NPK fertilizer for improved crop growth. Suitable for vegetables and cereals."] ["Durable 16-liter manual sprayer for pesticides and fertilizers. Adjustable nozzle included."] ["Roma VF tomato seeds with high disease resistance and excellent fruit quality. Early maturing."] ["Natural neem-based pesticide for organic farming. Controls a wide range of pests safely."]
Line Item Notes	fldfE9TAsllU0R91Y	Long text	string Multiple lines of text, which may contain "mention tokens", e.g. <airtable:mention id="menE1i9oBaGX3DseR">@Alex</airtable:mention>	EXAMPLE VALUES "Urgent delivery requested for upcoming planting season." "Customer prefers eco-friendly packaging." "Installation support required as per customer note." "Customer requests MSDS documentation." "Order for staggered delivery: split into two shipments."
Line Item Summary (AI)	fldiEkXX1c4Kpx2rw			
Potential Issue/Flag (AI)	fldHuK5k4QJaXiQhD			

List Order Line Items records

To list records in Order Line Items, use the `select` method.

`select` returns a query object. To fetch the records matching that query, use the `eachPage` or `firstPage` method of the query object.

Returned records do not include any fields with "empty" values, e.g. `""`, `[]`, or `false`.

Note: Airtable's API only accepts request with a URL shorter than 16,000 characters. Encoded formulas may cause your requests to exceed this limit. To fix this issue you can instead make a POST request to `/v0/{baseId}/{tableIdOrName}/listRecords` while passing the parameters within the body of the request instead of the query parameters. See our support article on this for more information.

You can use the following parameters to filter, sort, and format the results:

fields
`array of strings`
optional

Only data for fields whose names are in this list will be included in the result. If you don't need every field, you can use this parameter to reduce the amount of data transferred.

For example, to only return data from Name and Order, pass in:

```
fields: ["Name", "Order"]
```

You can also perform the same action with field ids (they can be found in the fields section):

```
fields: ["fld9CV784ELrr64Xs", "fldgpeDz9sRBgE5
```

CODE

```
var Airtable = require('airtable');
var base = new Airtable({apiKey:
'YOUR_SECRET_API_TOKEN'}).base('appPVwvxpSYRf5d0K');

base('Order Line Items').select({
    // Selecting the first 3 records in Grid view:
    maxRecords: 3,
    view: "Grid view"
}).eachPage(function page(records, fetchNextPage) {
    // This function (`page`) will get called for each page of
records.

    records.forEach(function(record) {
        console.log('Retrieved', record.get('Name'));
    });

    // To fetch the next page of records, call `fetchNextPage`.
    // If there are more records, `page` will get called again.
    // If there are no more records, `done` will get called.
    fetchNextPage();

}, function done(err) {
    if (err) { console.error(err); return; }
});
```

OUTPUT

```
Retrieved Maize Hybrid Seed
Retrieved Organic Fertilizer
Retrieved Drip Irrigation Kit
```

FETCH FIRST PAGE

```
// If you only want the first page of records, you can
// use `firstPage` instead of `eachPage`.
```

filterByFormula
`string`
optional

A formula used to filter records. The formula will be evaluated for each record, and if the result is not `0`, `false`, `""`, `NaN`, `[]`, or `#Error!` the record will be included in the response. We recommend testing your formula in the Formula field UI before using it in your API request.

If combined with the `view` parameter, only records in that view which satisfy the formula will be returned.

The formula must be encoded first before passing it as a value. You can use this tool to not only encode the formula but also create the entire url you need. For example, to only include records where Name isn't empty, pass in `NOT({Name} = '')` as a parameter like this:

```
filterByFormula: "NOT({Name} = '')"
```

Note: Airtable's API only accepts request with a URL shorter than 16,000 characters. Encoded formulas may cause your requests to exceed this limit. To fix this issue you can instead make a POST request to `/v0/{baseId}/{tableIdOrName}/listRecords` while passing the parameters within the body of the request instead of the query parameters. See our support article on this for more information.

maxRecords
`number`
optional

The maximum total number of records that will be returned in your requests. If this value is larger than `pageSize` (which is 100 by default), you may have to load multiple pages to reach this total. See the Pagination section below for more.

pageSize
`number`
optional

The number of records returned in each request. Must be less than or equal to 100. Default is 100. See the Pagination section below for more.

sort
`array of objects`
optional

A list of sort objects that specifies how the records will be ordered. Each sort object must have a `field` key specifying the name of the field to sort on, and an optional `direction` key that is either `"asc"` or `"desc"`. The default direction is `"asc"`.

The `sort` parameter overrides the sorting of the view specified in the `view` parameter. If neither the `sort` nor the `view` parameter is included, the order of records is arbitrary.

For example, to sort records by Name in descending order, send these two query parameters:

```
sort%5B0%5D%5Bfield%5D=Name
sort%5B0%5D%5Bdirection%5D=desc
```

For example, to sort records by Name in descending order, pass in:

```
[{field: "Name", direction: "desc"}]
```

view
`string`
optional

The name or ID of a view in the Order Line Items table. If set, only the records in that view will be returned. The records will be sorted according to the order of the view unless the `sort` parameter is included, which overrides that order. Fields hidden in this view will be returned in the results. To only return a subset of fields, use the `fields` parameter.

cellFormat
`string`
optional

The format that should be used for cell values. Supported values are:

`json`: cells will be formatted as JSON, depending on the field type.

`string`: cells will be formatted as user-facing strings, regardless of the field type. The `timeZone` and `userLocale` parameters are required when using `string` as the `cellFormat`.

```
base('Order Line Items').select({
    view: 'Grid view'
}).firstPage(function(err, records) {
    if (err) { console.error(err); return; }
    records.forEach(function(record) {
        console.log('Retrieved', record.get('Name'));
    });
});
```

FETCH ADDITIONAL RECORD METADATA

```
// If you want to fetch the number of comments for each record,
// include the `recordMetadata` param.
base('Order Line Items').select({
    recordMetadata: ['commentCount']
}).firstPage(function(err, records) {
    if (err) { console.error(err); return; }
    records.forEach(function(record) {
        console.log('Retrieved a record with', record.commentCount,
'comments');
    });
});
```

`string` as the `cellFormat`.

Note: You should not rely on the format of these strings, as it is subject to change.

The default is `json`.

timeZone
`string`
optional

The time zone that should be used to format dates when using `string` as the `cellFormat`. This parameter is required when using `string` as the `cellFormat`.

userLocale
`string`
optional

The user locale that should be used to format dates when using `string` as the `cellFormat`. This parameter is required when using `string` as the `cellFormat`.

returnFieldsByFieldId
`boolean`
optional

An optional boolean value that lets you return field objects where the key is the field id.

This defaults to `false`, which returns field objects where the key is the field name.

recordMetadata
`array of strings`
optional

An optional field that, if includes `commentCount`, adds a `commentCount` read only property on each record returned.

Pagination

The server returns one page of records at a time. Each page will contain `pageSize` records, which is 100 by default.

To fetch the next page of records, call `fetchNextPage`.

Pagination will stop when you've reached the end of your table. If the `maxRecords` parameter is passed, pagination will stop once you've reached this maximum.

Iteration may timeout due to client inactivity or server restarts. In that case, the client will receive a 422 response with error message `LIST_RECORDS_ITERATOR_NOT_AVAILABLE`. It may then restart iteration from the beginning.

Retrieve a Order Line Items record

To retrieve an existing record in Order Line Items table, use the `find` method.

Any "empty" fields (e.g. `""`, `[]`, or `false`) in the record will not be returned.

CODE

```
var Airtable = require('airtable');
var base = new Airtable({apiKey:
'YOUR_SECRET_API_TOKEN'}).base('appPVwvxpSYRf5d0K');

base('Order Line Items').find('rec8rb9IkSKKM29Wo', function(err,
record) {
    if (err) { console.error(err); return; }
    console.log('Retrieved', record.id);
});
```

OUTPUT

```
Retrieved rec8rb9IkSKKM29Wo
```

Create Order Line Items records

To create new records, use the `create` method. Note that table names and table ids can be used interchangeably. Using table ids means table name changes do not require modifications to your API request.

The first argument should be an array of up to 10 record objects. Each of these objects should have one key whose value is an inner object containing your

CODE

```
var Airtable = require('airtable');
var base = new Airtable({apiKey:
'YOUR_SECRET_API_TOKEN'}).base('appPVwvxpSYRf5d0K');

base('Order Line Items').create([
```

record's cell values, keyed by either field name or field id.

Returns an array of record objects created if the call succeeded, including record IDs which will uniquely identify the records within AgriSales CRM.

Values for Unit Price (from Product), Total Price, Order Date and Product Description are automatically computed by Airtable and cannot be directly created.

The Airtable API will perform best-effort automatic data conversion from string values if the `typecast` parameter is passed in (click to show example). Automatic conversion is disabled by default to ensure data integrity, but it may be helpful for integrating with 3rd party data sources.

You can also include a single record object at the top level. Click here to show an example.

```
  {
    "fields": {
      "Name": "Maize Hybrid Seed",
      "Order": [
        "reciKJhi7RNXV0p1X"
      ],
      "Product": [
        "recCeXjucglJneIEU"
      ],
      "Quantity": 10,
      "Line Item Notes": "Urgent delivery requested for upcoming
planting season.",
      "Line Item Summary (AI)": {
        "state": "generated",
        "value": "10 units of Maize Hybrid Seed ordered at
2,500/unit. Total: 25,000.",
        "isStale": true
      },
      "Potential Issue/Flag (AI)": {
        "state": "generated",
        "value": "No issues detected.",
        "isStale": true
      }
    }
  },
  {
    "fields": {
      "Name": "Organic Fertilizer",
      "Order": [
        "recfH73uq2xdb1k0H"
      ],
      "Product": [
        "recyYAWHWM64HCHK2"
      ],
      "Quantity": 5,
      "Line Item Notes": "Customer prefers eco-friendly packaging.",
      "Line Item Summary (AI)": {
        "state": "generated",
        "value": "5 bags of Organic Fertilizer at 1,800 each. Total:
9,000.",
        "isStale": true
      },
      "Potential Issue/Flag (AI)": {
        "state": "generated",
        "value": "Check stock levels: low inventory flagged.",
        "isStale": true
      }
    }
  }
], function(err, records) {
  if (err) {
    console.error(err);
    return;
  }
  records.forEach(function (record) {
    console.log(record.getId());
  });
});
```

OUTPUT

```
rec8rb9IkSKKM29Wo
reckkCvI3AT4zRXi6
```

Update Order Line Items records

To update Order Line Items records, use the `update` or `replace` method. An `update` will only update the fields you include. Fields not included will be unchanged. A `replace` will perform a destructive update and clear all unincluded cell values. A `replace` call on Order Line Items records will always fail. The example at the right uses the non-destructive `update` method. Click here to show a destructive `replace` call.

The first argument should be an array of up to 10 record objects. Each of these objects should have an `id` property representing the record ID and a `fields` property which contains all of your record's cell values by field name or field id for all of your record's cell values by field name.

To link to new records in Order and Product, add new linked record IDs to the existing array. Be sure to include all existing linked record IDs that you wish to retain. To unlink records, include the existing array of record IDs, excluding any that you wish to unlink.

CODE

```
var Airtable = require('airtable');
var base = new Airtable({apiKey:
'YOUR_SECRET_API_TOKEN'}).base('appPVwvxpSYRf5d0K');

base('Order Line Items').update([
  {
    "id": "rec8rb9IkSKKM29Wo",
    "fields": {
      "Name": "Maize Hybrid Seed",
      "Order": [
        "reciKJhi7RNXV0p1X"
      ],
      "Product": [
        "recCeXjucglJneIEU"
      ],
      "Quantity": 10,
```

that you wish to unlink.

Line Item Notes may contain "mention tokens". A *mention token* corresponds to a "@mention" in Airtable's user interface; here in the API it will look like `<airtable:mention id="menE1i9oBaGX3DseR">@Alex</airtable:mention>`. Mention tokens cannot be created via this API and should be left intact (or wholly removed) when updating long text fields.

Values for Unit Price (from Product), Total Price, Order Date and Product Description are automatically computed by Airtable and cannot be directly updated. You cannot clear these, even with a `replace` call.

Automatic data conversion for update actions can be enabled via `typecast` parameter. See create record for details.

You can also include a single record object at the top level. Click here to show an example.

```
      "Line Item Notes": "Urgent delivery requested for upcoming
planting season.",
      "Line Item Summary (AI)": {
        "state": "generated",
        "value": "10 units of Maize Hybrid Seed ordered at
2,500/unit. Total: 25,000.",
        "isStale": true
      },
      "Potential Issue/Flag (AI)": {
        "state": "generated",
        "value": "No issues detected.",
        "isStale": true
      }
    }
  },
  {
    "id": "reckkCvI3AT4zRXi6",
    "fields": {
      "Name": "Organic Fertilizer",
      "Order": [
        "recfH73uq2xdb1k0H"
      ],
      "Product": [
        "recyYAWHWM64HCHK2"
      ],
      "Quantity": 5,
      "Line Item Notes": "Customer prefers eco-friendly packaging.",
      "Line Item Summary (AI)": {
        "state": "generated",
        "value": "5 bags of Organic Fertilizer at 1,800 each. Total:
9,000.",
        "isStale": true
      },
      "Potential Issue/Flag (AI)": {
        "state": "generated",
        "value": "Check stock levels: low inventory flagged.",
        "isStale": true
      }
    }
  },
  {
    "id": "rec3hK4SLVVcJYKTf",
    "fields": {
      "Name": "Drip Irrigation Kit",
      "Order": [
        "recMkt174sEFSB3nY"
      ],
      "Product": [
        "recoUf2X7aSLG4iXz"
      ],
      "Quantity": 2,
      "Line Item Notes": "Installation support required as per
customer note.",
      "Line Item Summary (AI)": {
        "state": "generated",
        "value": "2 Drip Irrigation Kits at 12,000 each. Total:
24,000.",
        "isStale": true
      },
      "Potential Issue/Flag (AI)": {
        "state": "generated",
        "value": "Potential delay: installation support may affect
delivery timeline.",
        "isStale": true
      }
    }
  }
], function(err, records) {
  if (err) {
    console.error(err);
    return;
  }
  records.forEach(function(record) {
    console.log(record.get('Name'));
  });
});
```

OUTPUT

```
"Maize Hybrid Seed"
"Organic Fertilizer"
"Drip Irrigation Kit"
```

Delete Order Line Items records

To delete Order Line Items records, use the `destroy` method. Note that table names and table ids can be used interchangeably. Using table ids means table name changes do not require modifications to your API request.

The first parameter to `destroy` is an array of up to 10 record IDs to delete.

You can also set the first parameter to a record ID to delete a single record. Click here to show an example.

CODE

```
var Airtable = require('airtable');
var base = new Airtable({apiKey:
'YOUR_SECRET_API_TOKEN'}).base('appPVwvxpSYRf5d0K');

base('Order Line Items').destroy(['rec8rb9IkSKKM29Wo',
'reckkCvI3AT4zRXi6'], function(err, deletedRecords) {
  if (err) {
    console.error(err);
    return;
  }
  console.log('Deleted', deletedRecords.length, 'records');
});
```

OUTPUT

```
Deleted 2 records
```

ERRORS

The AgriSales CRM API follows HTTP status code semantics. 2xx codes signify success, 4xx mostly represent user error, 5xx generally correspond to a server error. The error messages will return a JSON-encoded body that contains `error` and `message` fields. Those will provide specific error condition and human-readable message to identify what caused the error.

Success code

`200`	OK	Request completed successfully.

User error codes

These errors generally indicate a problem on the client side. If you are getting one of these, check your code and the request details.

`400`	Bad Request	The request encoding is invalid; the request can't be parsed as a valid JSON.
`401`	Unauthorized	Accessing a protected resource without authorization or with invalid credentials.
`402`	Payment Required	The account associated with the API key making requests hits a quota that can be increased by upgrading the Airtable account plan.
`403`	Forbidden	Accessing a protected resource with API credentials that don't have access to that resource.
`404`	Not Found	Route or resource is not found. This error is returned when the request hits an undefined route, or if the resource doesn't exist (e.g. has been deleted).
`413`	Request Entity Too Large	The request exceeded the maximum allowed payload size. You shouldn't encounter this under normal use.
`422`	Invalid Request	The request data is invalid. This includes most of the base-specific validations. You will receive a detailed error message and code pointing to the exact issue.
`429`	Too Many Requests	The API is limited to 5 requests per second per base. You will receive a 429 status code and a message "Rate limit exceeded. Please try again later" and will need to wait 30 seconds before subsequent requests will succeed. To learn more about rate limits, please visit our Rate Limits guide.

Server error codes

These errors generally represent an error on our side. In the event of a 5xx error code, detailed information about the error will be automatically recorded, and Airtable's developers will be notified.

`500`	Internal Server Error	The server encountered an unexpected condition.
`502`	Bad Gateway	Airtable's servers are restarting or an unexpected outage is in progress. You should generally not receive this error, and requests are safe to retry.
`503`	Service Unavailable	The server could not process your request in time. The server could be temporarily unavailable, or it could have timed out processing your request. You should retry the request with backoffs.