restdb.io cheat sheet (v0.1)

> To make life easier for restdb.io developers, we have assembled a compact version of our docs for reference. Here you'll find reference to the Collection API, the Media Content API, the Meta data API, Mail API, Authentication API and the Codehook APIs.

REST API URLs are prefixed with the full database name.

```
https://{db-name}.restdb.io/rest/{collection}[.<format>]?q={}&h={}&...
```

For example, a query against the customer collection in the database *mydb-fex0*:

```
https://mydb-fex0.restdb.io/rest/customers?q={"name": "Jane"}
```

Collection API

Operation	Resource	Description
GET	`/rest/{collection}`	Get list of JSON documents from database collection.
GET	`/rest/{collection}/ID`	Get one document from a collection. ID must be a valid ObjectID.
GET	`/rest/{collection}/ID/{subcollection}`	Get list of documents from subcollection (subcollection is field name of type child).
GET	`/rest/{collection}/ID/{subcollection}/ID`	Get document from subcollection (subcollection is field name of type child) and ID is a valid ObjectID.
POST	`/rest/{collection}`	Create a new document in a collection. Request body is a valid JSON document.
POST	`/rest/{collection}/ID/{subcollection}`	Create a child document in a sub collection.
POST *array*	`/rest/{collection}`	Post array data. Request body is an array of JSON documents. Optional parameter `?validate=false`, for bulk inserts without validation. Will also work for child documents.
PUT	`/rest/{collection}/ID`	Update a document in a collection. Request body is a valid JSON document.

Operation	Resource	Description
PATCH	`/rest/{collection}/ID`	Update one or more properties on a document in a collection. Request body is a valid JSON object.
DELETE	`/rest/{collection}/ID`	Delete a document in a collection.
DELETE *array*	`/rest/{collection}/*`	Delete an array of documents in a collection. Request body must be an array of ID's.
DELETE *query*	`/rest/{collection}/*?q={...}`	Delete a list of documents in a collection. List is based on query in parameter `?q={...}`. Only allowed with a full access API-key or from a Codehook.

Parameters for GET operations

Parameter	Description
q	Database query as a valid JSON string: `/rest/people?q={"firstname":"Karen", "lastname":"Johnson", age: 39}`
h	Query hints to specify: fields, max, skip and orderby. Aggregation parameters can also be specified as hints, read more here : `/rest/people?q={}&h={"$fields": {"title":1},"$max":10,"$skip":5,"$orderby":{"body":1}}`
filter	Performs a text search and retrieves all matching documents: `/rest/people?q={}&filter=johnson`
sort	Specifies which field(s) should be used to sort the result. You can add multiple fields by simply adding another sort parameter. Default sort=_id: `/rest/people?q={}&sort=lastname`
dir	Sort direction. Allowed values are 1 (ascending) and -1 (descending). Used together with sort. Multiple dir parameters can be used in conjunction with sort: `/rest/people?q={}&sort=lastname&dir=-1`
skip	Where to start in the result set. Used for paging: `/rest/people?skip=100`
max	Maximum number of records retrieved: `/rest/people?max=20`

Parameter	Description
totals	Returns an object with both data and total count. Totals equals to max parameter or default 1000: Output from query `/rest/people?totals=true` returns the object `{data: [ … ], totals: { total: 100, count: 40, skip: 0, max: 1000}}`
totals and count	Returns an object with no data and just the total count: `/rest/people&totals=true&count=true`, output from query -> `{data: [], totals: { count: 42}}`
metafields	Displays internal restdb.io fields (_keywords, _tags, _version, _created, _createdby, _mock): `/rest/people?metafields=true`
groupby	Group output from query by a field: `/rest/people?groupby=gender`
aggregate	Perform aggregation function on data groups. Useful together with groupby parameter. Avaliable functions; SUM, AVG, MIN, MAX and COUNT Docs: `/rest/people?groupby=gender&aggregate=SUM:weight&aggregate=AVG:age`
format	Output format from request. .html returns a standard markup for all fields. .js (Javascript) returns a script that can be included in a web page for search engine indexing. .json standard JSON format: `/rest/people.html`
apikey	A valid apikey, full access key or a CORS enabled key: `/rest/people?&apikey=4321fd234f0876...`. Not recommended for production. Use header field `x-apikey` instead
idtolink	Inserts canonical URLs for image references and object references: `/rest/people?idtolink=true`
flatten	Used together with idtolink. Extract links as properties on root object: `/rest/people?flatten=true`
referencedby	Return all items that refers to a record. Requires a record _id in the query or path: `/rest/people/56011150e1321c7300000001?referencedby=true`
fetchmediadata	Replace media ID's with a full record from the media archive: `/rest/people?fetchmediadata=true`
fetchchildren	Insert records from parent-child relation on parent record: `/rest/company?fetchchildren=true`

Data aggregation API

Function	Format	Comment	Example
Min	MIN:field	Returns object	`h={"$aggregate":["MIN:score"]}`
Max	MAX:field	Returns object	`h={"$aggregate":["MAX:score"]}`
Avg	AVG:field	Returns value	`h={"$aggregate":["AVG:score"]}`
Sum	SUM:field	Returns value	`h={"$aggregate":["SUM:score"]}`
Count	COUNT:property	Returns value with chosen property name	`h={"$aggregate": ["COUNT:nplayers"]}`
Groupby	$groupby: ["field", ...]	Returns "groupkey":[array]	`h={"$groupby":["category"]}`
Groupby (dates)	$groupby: ["$YEAR:field", ...]	Predefined values for: $YEAR, $MONTH, $DAY, $HOUR, $SEC	`h={"$groupby": ["$YEAR:registered"]}`
Groupby (dates with formats)	$groupby: ["$DATE:format", ...]	Format strings for: ss, hh, mm, dd, MM, YY. All formats at momentjs.com	`h={"$groupby": ["$DATE.MMM:registered"]}`
Grand totals	$aggregate-grand-total: true	Recursive aggregation functions of groups	`h={"$groupby":["category"], "$aggregate": ["AVG:score"], "$aggregate-grand-total": true}`

Media Content API

Operation	Resource	Description
GET	`/media/ID`	Get binary data for media object with ID. ID is a valid ObjectID for an object in the media archive or an existing filename. Parameter s options; `?s=t` (thumbnail), `?s=w` (web) and `?s=o` (original size). No API-key required. Use parameter `?download=true` if you want the image or file to be downloaded instead of displayed.
GET	`/media/ID/meta`	Get JSON structure with media object metadata. Requires API-key.
POST	`/media`	Post file(s) using the multipart/formdata protocol, view example. Requires API-key.
PUT	`/media`	*Not yet implemented*
DELETE	`/media/ID`	Delete media content with ID. Requires API-key.

Meta Data API

Operation	Resource	Description
GET	`/rest/_meta`	Get meta data for the database as a JSON object.
GET	`/rest/{collection}/_meta`	Get meta data for the collection as a JSON object.

Mail API

Operation	Resource	Description
POST	`/mail`	Send email. Request body contains one document: `{"to": "...", "subject": "...", "html": "...", "company": "...", "sendername": "..."}` , or an array of documents `[{…},{…}]`

Authentication API

Operation	Resource	Description
POST	`/auth/jwt`	Generate a new JWT token. Body must contain a path to a secret and a payloan with JWT claims, e.g `{"secret": "path from global settings", "payload": {"email": "xxx@example.com"}}`
GET	`/auth/userinfo`	Get data about a user. Returns email, displayname and image.
POST	`/auth/logout`	Logout a user, invalidates the login token. This token can no longer be used for API access.

Codehook API

Database operation hooks

Codehook	Available parameters	Description
`beforeGET(req, res)`	req.query, req.hint, res.end([optional error])	Called before a GET operation
`beforePUT(req, res)`	req.body, res.end([optional error])	before a PUT
`afterPUT(req, res)`	req.body, req.result	after a PUT
`beforePOST(req, res)`	req.body, res.end([optional data, error])	before a POST
`afterPOST(req, res)`	req.body, req.result, res.end()	after a POST
`beforeDELETE(req, res)`	req.body, res.end([optional error])	before a DELETE
`afterDELETE(req, res)`	req.body	after a DELETE

Codehook	Available parameters	Description
`beforePATCH(req, res)`	req.body, res.end([optional error])	before a PATCH
`afterPATCH(req, res)`	req.body, req.result	after a PATCH
`runJob(req, res)`	res.end("Optional message to log")	triggers on a crontab expression

Codehook database API

Function	Parameters	Description
`db.get(path, query, hint, callback)`	path: REST url, query: query object, hint: hint object, callback: function(error, data)	get operation to the current database. E.g. `db.get('/rest/customer', {}, {}, function(error, data) {...}`
`db.put(path, data, callback)`	path: REST url, data: JSON, callback: function(error, data)	put operation to the current database
`db.patch(path, data, callback)`	path: REST url, data: JSON, callback: function(error, data)	patch operation to the current database
`db.post(path, data, callback)`	path: REST url, data: JSON, callback: function(error, data)	post operation to the current database
`db.delete(path, data, callback)`	path: REST url, data: JSON, callback: function(error, data)	delete operation to the current database

Codehook network API

Function	Parameters	Description
`request(options, callback)`	options: json, callback(error, response, body)	Network API
`slack(options, callback)`	options: message, slackhookurl, channel, callback: function(result)	Send a message to Slack
`sendMail(options, callback)`	options: to, subject, html, company, callback: function(result)	Send a html email

Codehook utility function API

Function	Parameters	Description
`log.debug(str, ...)`	Variable list of arguments log.info, log. debug, log. error, log.fatal	Writes output to Rest inspector

Function	Parameters	Description
`async.series(funcarray, callback)`	Array of functions	Call an array of functions in a sequence and callback when all are done. See examples.
`async.waterfall(funcarray, callback)`	Array of functions	First function calls next etc. in sequence passing parameters along and finally callback when all are done. See examples.
`async.apply(function, arguments)`	Apply one function with arguments	Creates a continuation function with some arguments already applied. Add to array and use with async.series. See examples.
`template(str, context)`	Run Handlebars on str with context json data	Handlebars templating function
`markdown(str)`	Run Markdown parser on str	Returns valid HTML
`auth.decode_jwt(token, callback)`	A valid JWT token	callback with (err, decodedjwt) as arguments
`auth.verify_jwt(token, secret, callback)`	A valid JWT token and your secret	callback with (err, decodedjwt) as arguments
`verifyHash(token, secret, 'RSA-SHA256', 'base64');`	token-string, secret-string, algorithm, encoding	Verify crypto tokens, e.g. Shopify webhook token