TeamDesk REST API

Design Goals

For a long time TeamDesk has a SOAP API. Over the years we've found advantages of SOAP protocols as well as its shortcomings. Some of them are:

- SOAP is built on top of XML and HTTP and adds own semantics in addition, or superseding the one existing in HTTP – in order to use SOAP, you should understand not only SOAP related stuff, but have advanced knowledge of XML and HTTP protocols.
- SOAP makes heavy use of XML namespaces which proved to be a recurring problem for many users.
- While SOAP is intended to be a self-describable protocol (via Web Services Description Language - WSDL), there are SOAP clients that misinterpret this information – PHP for example, and generate incorrect calls to a web service resulting hours of debugging the code that seems to be written right.

Web landscape has changed since we released SOAP API for TeamDesk back in 2006. Now, many web sites make heavy use of REST and JSON. Dynamically typed languages rule the web and JSON, being simple to parse and produce, is an ideal data protocol for them. Moreover, strongly typed languages such as C++ also have libraries to deal with the data in JSON format.

So, while designing next version of the API our goals were:

- Make JSON first-class output format.
- Build an API based solely on HTTP protocol semantics.
- Pass credentials together with request to avoid separate login calls.
- Allow cross-origin resource sharing (CORS).
- Make calls require only couple lines of code.
- Address certain design shortcomings of SOAP API

Have we reached these goals? We hope so. Want to query the Default View of a Test table in a Test API application (21995) in HTML format? Open your browser and type:

```
https://www.teamdesk.net/secure/api/v2/21995/Test/Default%20View/select.html
```

You'll be prompted for login (type **test@test.com**) and password (**pwd**) and then there is your data.

Using jQuery? Easy! Create authorization token in application's setup section, set *authtoken* variable and let jQuery handle the rest.

```
var authtoken = "0123456789ABCDEF0123456789ABCDEF";
$.getJSON(
	"https://www.teamdesk.net/secure/api/v2/21995/" + authtoken +
	"/Test/Default%20View/select.json",
	function(data) {
		/* here we have data */
	}
);
```

In PHP? Still easy!

```
$authtoken = "0123456789ABCDEF0123456789ABCDEF";
$data = json_decode(
	file_get_contents(
	"https://www.teamdesk.net/secure/api/v2/21995/" . $authtoken .
	"/Test/Default%20View/select.json"
));
```

In C#?

```
string authtoken = "0123456789ABCDEF0123456789ABCDEF";
object data = new JavaScriptSerializer().DeserializeObject(
	new WebClient().DownloadString(
	"https://www.teamdesk.net/secure/api/v2/21995/" + authtoken +
	"/Test/Default%20View/select.json"
));
```

Please note that simplicity does not mean efficiency. Samples above are dead simple but in order to make API operations more effective you may want to enable compression and caching that will require little bit more complicated web client setup.

Note for SOAP API users

REST API is a radical departure from SOAP API in terms of both data formats and operation logic. Please read this documentation carefully.

REST API Base URL

When you check your app's address you'll likely see in your browser's address bar a sort of

```
https://www.teamdesk.net/secure/db/21995/...
```

The number that follows the **/db/** is your *application id*. Changing **/db/** to **/api/v2/** and removing the remainder of the URL after the number will serve as a base URL for all API calls. Directly under the URL resides "playground" page to test API calls.

```
https://www.teamdesk.net/secure/api/v2/21995/
```

TeamDesk Enterprise and dbFLEX users will need to adjust **www.teamdesk.net** to their appropriate domain, but URL pattern is the same across all products – that's

```
https://{YOUR_DOMAIN}/secure/api/v2/{APPID}/
```

Authorization

All API methods require authorization. There are several ways to authorize the call.

API Token

API token allows you to bind the user of the application to the unique identifier to perform API calls without exposing user's credentials. Moreover, token is application specific – that's it gives an access only to the application it is defined for.

Existing tokens are listed and new ones can be created under *Setup > Database > Integration API > REST API Authorization Tokens*. You can create multiple tokens for one user to use in different contexts, so that token removal will disable data access in from one content but not the others. For example, you can issue several tokens for third-party developers to let them make API calls under single user account. Revoking the token will disable access for one developer but not the others.

In order to authorize API call you can send the token via Authorization HTTP Header

```
GET https://www.teamdesk.net/secure/api/v2/21995/user.json
Authorization: Bearer 0123456789ABCDEF0123456789ABCDEF
```

Or embed token into URL after application ID such as

```
https://www.teamdesk.net/secure/api/v2/21995/0123456789ABCDEF0123456789ABCDEF/user.json
```

though, while embedding looks simple, please keep in mind that request URL may leave traces in upstream proxies and Internet providers' logs; sending token via header is a bit more secure.

Basic Access Authentication

In addition to token authorization API supports HTTP basic access authentication scheme. This scheme is a well-supported by every HTTP client. Moreover, many interactive clients, such as browser or Microsoft Excel, will prompt for username and password via dialog box if credentials are not provided.

Re-use authorization cookie

This method is only suitable when building calls from HTML snippets embedded into TeamDesk pages. When minus sign (-) is added after application id we'll try to re-use existing TeamDesk authorization cookie to authenticate the user. Please note that this method **won't work in Call URL actions**.

```
https://www.teamdesk.net/secure/api/v2/21995/-/user.json
```

Output format

You should specify method's output format by appending appropriate extension to a method name: .json for JSON output, .xml for XML output. Select and Retrieve methods support two additional formats: HTML (.html) and CSV (.csv). The only exclusion to this rule is an Attachment method as it returns the content of the file in its native format.

JSON and XML outputs closely map to each other wherever possible. Arrays in JSON are indicated by square brackets while objects are indicated with curly braces and object properties and their values are in **"key": value** format. In XML arrays and object are both tags. Arrays tags wrap all their items and usually named in plural, object tags wrap their individual properties and named in singular. Object properties usually have simple content. To illustrate both formats let's compare them side by side:

```
// JSON
// application properties
{
  "id": "21995",
  "name": "API Test",
  // an array...
  "tables": [
    // ...of table descriptors
    {
      "id": 115442,
      "recordName": "Test"
    },
    {
      "id": 115443,
      "recordName": "Test2"
    }
    //, { ... }
  ]
}
```

```
<!-- XML -->
<!-- top level is always Response -->
<Response>
  <id>21995</id>
  <name>API Test</name>
  <!-- an array... -->
  <tables>
    <!-- ...of table descriptors -->
    <table>
      <id>115442</id>
      <recordName>Test</recordName>
    </table>
    <table>
      <id>115443</id>
      <recordName>Test2</recordName>
    </table>
    <!-- ... -->
  </tables>
</Response>
```

Output Compression

API methods are able to compress output traffic via *gzip* or *deflate* compression. Given that both JSON and XML are text based formats with highly repetitive content, compression can effectively save 60% to 80% of response size. Many HTTP clients use simple flags to enable compression (if not enabled by default) and make decompression process completely transparent.

For example .NET's HttpClientHandler and HttpWebRequest, Java's HttpClientBuilder, JavaScript's XMLHttpRequest and one of the most widely spread HTTP client libraries, libcurl supports such an option. For other HTTP clients, please consult your client's documentation.

Output Caching

All responses from data retrieval methods are cacheable by default – you can improve the performance even more by performing conditional HTTP requests to check whether the data was modified and skip loading the content if your cached copy is still fresh. As with compression, many HTTP clients support it out of stock and make it completely transparent for you – JavaScript's XMLHttpRequest, NET's HttpWebRequest, Java's CachingHttpClientBuilder. If caching is not supported it is fairly easy to write thin wrapper to implement this logic for the API given that we are using its small subset.

a. Only GET requests are cacheable.
b. Responses with Cache-Control header containing "no-cache" are not cacheable.

First time send the request unconditionally:

```
GET https://www.teamdesk.net/secure/api/v2/21995/user.json
```

You'll get *ETag* header and the content.

```
HTTP/1.1 200 OK
Cache-Control: private, must-revalidate, max-age=0
ETag: "0123456789"

{ id: 1, name: "John", ... }
```

Use URL including query parameters to store the value of the *ETag* header and content of the response in a file system or database.

Next time find stored *ETag* from the URL and query parameters and issue the request with *If-None-Match* header set to *ETag* value.

```
GET https://www.teamdesk.net/secure/api/v2/21995/user.json
If-None-Match: "01234.56789"
```

You'll get back either

```
HTTP/1.1 304 Not Modified
```

when it is OK to use cached copy or new *ETag* and content to update the cache with

```
HTTP/1.1 200 OK
Cache-Control: private, must-revalidate, max-age=0
```

```
ETag: "43210.98765"

{ id: 1, name: "Jane", ... }
```

Errors

Error condition is indicated response with one of the 4xx HTTP status code and the content containing short error descriptor in either JSON or XML format unless otherwise noted. If the method is requested to return XML format, error's output format will be XML, for all the other formats error output format is JSON:

```
// JSON
{
    "error":  403,                          // the copy of HTTP status
    "message": "View does not exist",       // the message text
    "code":    3200,                        // optional code for the message, if standard
    "source":  "Some View"                  // optional location hint, e.g. parameter name
}


<!-- XML -->
<Error>
    <error>403</status>                     <!-- HTTP status -->
    <message>View does not exist</message>  <!-- the message text -->
    <code>3200</code>                       <!-- optional code -->
    <source>Some View</source>              <!-- optional hint -->
</Error>
```

We use limited subset of 4xx HTTP status codes.

Code	Status	Description
400	Bad request	One of the request parameters is missing or malformed. Check "source" property of the error descriptor for the hint.
401	Unauthorized	No authorization is provided. Interactive clients can bring up login/password prompt. No error descriptor is returned.
403	Forbidden	Invalid authentication token or user credentials; Access to the table, column or view is denied; Record not found or access forbidden Not enough privileges to perform record creation or update.
405	Invalid Method	Invalid method name
409	Conflict	Unable to perform record creation or update due to application constraints.
413	Request Entity Too Large	Client attempts to send more than 20Mb of data at once. No error descriptor is returned.
414	Request URI Too Long	Client attempts to send more than 16K of data in a query string. No error descriptor is returned.
415	Invalid Media Type	Client sends the data in a format API does not understand.
500	Internal Server Error	Internal server error; Error from a data access backend – these ones will be eventually re-qualified to error codes above.

REST API Methods

User method

Use this method to retrieve about the user. We do not recommend treating the response as a fixed structure as we reserve the right to **extend** the output with more information at any time.

Request

```
GET https://www.teamdesk.net/secure/api/v2/{appid}/user.{json|xml}
```

```
GET https://www.teamdesk.net/secure/api/v2/21995/user.json
```

Response

```
{
  "id":        12345,
  "email":     "test@test.com",
  "firstName": "Test",
  "lastName":  "User",
  "role":      "Default Role",
  "culture":   "en-US",
  // as defined in IANA TimeZone Database
  "timezone":  "America/Chicago",
  "admin":     "CustomizeApplication, ManageUsers, ManageData"
}
```

Describe (Application) Method

Use this method to retrieve application's description and the list of tables. We do not recommend treating the response as a fixed structure as we reserve the right to **extend** the output with more information at any time.

Request

```
GET https://www.teamdesk.net/secure/api/v2/{appid}/describe.{json|xml}
```

Describe Test API application

```
GET https://www.teamdesk.net/secure/api/v2/21995/describe.json
```

Response

```
{
  "id": "21995",
  "name": "API Test",
  "description": "This is a Test API application",
  "culture": "en-US",
  // as defined in IANA TimeZone Database
  "timeZone": "America/Chicago",
  // an array or table descriptors
  "tables": [
    {
      "id": 115442,
      "recordName": "Test",  // you can address the table by this name
      "recordsName": "Tests",
      "alias": "t_115442",   // or by alias
      "showTab": true,
      "color": "#0061B0"
    }
    // more tables...
    //, { ... }
  ]
}
```

Describe (Table) Method

Use this method to retrieve table's structure. We do not recommend treating the response as a fixed structure as we reserve the right to **extend** the output with more information at any time.

Parameters

- *table*: the singular name of the table or its *alias* as returned by the Describe (Application) method. Please note that this parameter is embedded into the URL before the method name.

Request

```
GET https://www.teamdesk.net/secure/api/v2/{appid}/{table}/describe.{json|xml}
```

Describe Test table in Test API application

```
GET https://www.teamdesk.net/secure/api/v2/21995/Test/describe.json
```

Response

```
{
  "id": 115442,
  "recordName": "Test",
  "recordsName": "Tests",
  "alias": "t_115442",
  "showTab": true,
  "color": "#0061B0"
  "allowAdd": true,
  "key": "Id",
  "columns": [ // an array of column descriptors
    {
      "id": 2445930,
      "name": "Text",
      "alias": "f_2445930",
      "type": "Text",
      "dataOptions": "AllowAddSimilar, AllowFind",
      "displayOptions": "ShowInViews",
      "width": 40
    },
    // ...
  ],
  "views": [// an array of view descriptors
    {
      "id": 730247,
      "type": "Table",
      "name": "Default View",
      "alias": "v_730247",
      "showInMenu": true,
      "actions": "Add, Edit, View, Delete"
    },
    // ...
  ],
}
```

Select (Table) Method

This method allows you to construct a query to obtain records from the table.

Parameters

- *table*: the singular name of the table or its *alias* as returned by the Describe (Application) method. Please note that this parameter is embedded into the URL before the method name.
- *column*: the name or an alias of the column to query can appear multiple times. If omitted or star (*) is specified, API returns the data for all updateable columns. You can combine star with other column names.
- *filter*: optional, allows you to specify filtering criteria in syntax described in Formula Language Reference.
- *sort*: the name or an alias of the column to sort by can appear multiple times. Sort order can be specified by appending //ASC or //DESC to a column name, ascending order is the default
- *top*: optional, a number of records to return in a range 1...500. Default is 500.
- *skip*: number of records to skip before returning the result. This parameter can be used to organize paginated output, for example: skip=0&top=200 (page 1), skip=200&top=200 (page 2), etc.

Request

```
GET https://www.teamdesk.net/secure/api/v2/{appid}/{table}/select.{json|xml}?parameters
```

Query updatable columns from first 500 records in a Test table

```
GET https://www.teamdesk.net/secure/api/v2/21995/Test/select.json
```

Query Text, Date columns from records 5-10 in a Test table, sort by Date descending

```
GET https://www.teamdesk.net/secure/api/v2/21995/Test/select.json?column=Text&column=Date
&sort=Date//DESC&skip=5&top=5
```

Query all updateable columns plus Date Modified column

```
GET https://www.teamdesk.net/secure/api/v2/21995/Test/select.json?column=*&column=Date%20
Modified
```

Response (JSON)

The data is returned as the array of JavaScript objects. Each object's field corresponds to a column queried.

Null values reported as **null.**

Checkboxes reported as **true** or **false**.

Numeric columns are JavaScript numbers

Durations are reported as a number of seconds.

Text columns are strings.

Since JavaScript has no literal form for dates we are returning dates, times and timestamps as a string in YYYY-MM-DDTHH:MM:SS[+/-]ZZ:ZZ format. The pattern is easily recognizable and can be then converted to appropriate date and time supporting object. Date columns are always reported as a midnight values with zero offset, e.g. **yyyy-mm-dd**T00:00:00+00:00, times are reported as 1/1/0001 dates and zero offset, e.g. 0001-01-01T**hh:mm:ss**+00:00. Timestamps are reported in user's time zone with offset to UTC to allow further recalculations.

Users are reported as strings in a form of Name <email>. The name can be then used to render the data while email uniquely identifies the user.

Object fields starting with @row are row properties – internal ID, actions allowed to perform and the color string if row colorization formula is provided.

```
[
 {
   "@row.id": 12,                          // internal row ID
   "@row.allow": "Edit, Delete",           // comma separated allowed actions
   "Id": "60",                             // autonumber is string
   "Text": "Text",                         // text is string
   "Multiline": "Multi\r\nText",           // line separators are encoded appropriately
   "Checkbox": true,                       // checkbox is either true or false
   "Date": "2014-11-18T00:00:00+00:00",    // date is UTC midnight
   "Time": "0001-01-01T17:26:00+00:00",    // time is in 1/1/0001 UTC
   "Number": 1234567,                      // number is ...well, number
   "Email": "kir@skyeytech.com",
   "Phone": "+12345678",
   "URL": "http://www.teamdesk.net",
   "User": "test user <test@test.com>",    // name <email>
   "Duration": 86400,                      // number of seconds
   "Timestamp": "2014-11-18T17:26:00-06:00", // in a local user's timezone
 }
 // , {...}
]
```

Response (XML)

Top level element is named <Response> and contains a set of <row> elements.
Row properties are reported as attributes of <row> element. Subtags of <row> elements are column values. Null values reported as elements with no content and i:nil="true" attribute. Checkboxes Date, times, durations and null values are reported in a format appropriate for XML.

Any XML name character that does not conform to the XML 1.0 spec (fourth edition) recommendation is escaped as _xHHHH_. The HHHH string stands for the four-digit hexadecimal UCS-2 code for the character in most significant bit first order. For example, the name Order Details is encoded as Order_x0020_Details. The underscore character does not need to be escaped unless it is followed by a character sequence that together with the underscore can be misinterpreted as an escape sequence when decoding the name. For example, Order_Details is not encoded, but Order_x0020_ is encoded as Order_x005f_x0020_. No short forms are allowed.

```
<!-- URL to the row schema is reported for conforming XML readers but not required -->
<Response>
  <row id="12" allow="Edit Delete">
    <Id>60</Id>
    <Text>Text</Text>
    <Multiline>Multi
Text</Multiline>
    <Checkbox>true</Checkbox>
    <!-- yyyy-mm-dd -->
    <Date>2014-11-18</Date>
    <!-- hh:mm:ss -->
    <Time>17:26:00</Time>
    <Number>1234567.000000</Number>
    <Email>kir@skyeytech.com</Email>
    <Phone>+12345678</Phone>
    <URL>http://www.teamdesk.net</URL>
    <User>test user &lt;test@test.com &gt;</User>
    <!-- PTsecondsS -->
    <Duration>PT86400S</Duration>
    <!-- yyyy-mm-ddThh:mm:ss+/-ZZ:ZZ -->
    <Timestamp>2014-11-18T17:26:00+02:00</Timestamp>
    <!-- Null value -->
    <Test_Null i:nil="true"/>
  </row>
  <!-- more rows -->
</Response>
```

Also, XML output can be used to create dynamic read-only link between the data in TeamDesk and Excel. Try Excel's *Data* tab, *Get External Data* ribbon, *From Other Sources* dropdown, *From XML Data Import* menu and paste URL as the File Name.

Select (Table) Method with aggregation

Select (Table) method is also capable to calculate aggregate function over the data group or complete data set.

To perform calculation, add double slash and a suffix that indicates a function to calculate to a column name, such as Number//MAX. Functions available are:

Function	Types	Description
COUNT	All types	Calculates count of records in a group or complete set.
SUM	Numeric, Duration	Calculates total of numeric/duration column over a group or complete set.
AVG	Numeric, Duration	Calculates average of numeric/duration column over a group or complete set.
MIN	All types	Calculates minimum value in a column over a group or complete set.
MAX	All types	Calculates maximum value in a column over a group or complete set.
STDEV	Numeric, Duration	Calculates statistical deviation of all values in numeric/duration column over a group or complete set.
STDEVP	Numeric, Duration	Calculates statistical deviation for the population of all values in numeric/duration column over a group or complete set.
VAR	Numeric, Duration	Calculates statistical variance of all values in numeric/duration column over a group or complete set.
VARP	Numeric, Duration	Calculates statistical variance for the population of all values in numeric/duration column over a group or complete set.

To set column(s) to group by add double slash and a grouping suffix to a column name(s):

Group by	Types	Description
EQ	All types	Group by equal value
FW	Text	Group by first word
FL	Text	Group by first letter
SS	Duration, Time, Timestamp	Group by second
MI	Duration, Time, Timestamp	Group by minute
HH	Duration, Time, Timestamp	Group by hour
DD	Date, Duration, Timestamp	Group by day
MM	Date, Timestamp	Group by month
QQ	Date, Timestamp	Group by quarter
YY	Date, Timestamp	Group by year
.001	Numeric	Group by one thousandth
.01	Numeric	Group by one hundredth
.1	Numeric	Group by one tenth
1	Numeric	Group by integer value
10	Numeric	Group by ten
100	Numeric	Group by one hundred

1K	Numeric	Group by one thousand
10K	Numeric	Group by ten thousands
100K	Numeric	Group by hundred thousands
1M	Numeric	Group by one million

If function to calculate is specified all the other columns must contain group type suffix.

By default, group values are sorted in ascending order. To reverse direction, specify *column//DESC* via sort parameter.

```
GET
https://www.teamdesk.net/secure/api/v2/21995/Test/select.json?column=Date//MM&column=Number//SUM&sort=Date//DESC
```

When group columns are present, first row of the output, indicated by "@row.type": "Grand" (or type="Grand" attribute of the row element in XML) contains value calculated over all groups.

Select (View) Method

This method allows you to construct a query to obtain records from the table via defined view.

Parameters

- *table*: the singular name of the table or its *alias* as returned by the Describe (Application) method. Please note that this parameter is embedded into the URL before the method name.
- *view*: the name or an alias of the view to query. Please note that this parameter is embedded into the URL after the table name/alias.
- *filter*: optional, allows you to specify filtering criteria that will be applied in addition to a view filter. The syntax is described in Formula Language Reference.
- *top*: optional, a number of records to return in a range 1...500. Default is 500.
- *skip*: number of records to skip before returning the result. This parameter can be used to organize paginated output, for example: skip=0&top=200 (page 1), skip=200&top=200 (page 2), etc.

Request

```
GET https://www.teamdesk.net/secure/api/v2/{appid}/{table}/{view}/select.{json|xml}?params
```

Retrieve the data from List All views in a Test table.

```
GET https://www.teamdesk.net/secure/api/v2/21995/Test/List%20All/select.json
```

Response

Response matches the format returned by Select (Table) method. If the view calculates aggregated values the output may contain additional rollup rows identified by "@row.type" attribute in JSON output or "type" attribute in XML output – Grand for grand total, and RowTotal and ColumnTotal to indicate corresponding total values in crosstab views.

Retrieve method

This method allows you to retrieve records from a table given either their keys or ids. The order the records are returned may not match the order ids/keys are passed in.

Parameters

- *table*: the singular name of the table or its *alias* as returned by the Describe (Application) method. Please note that this parameter is embedded into the URL before the method name.
- *column*: the name or an alias of the column to query can appear multiple times. If omitted or star (*) is specified, API returns the data for all updateable columns. You can combine star with other column names.
- *key* or *id*: the value of key column or internal record id to retrieve. Can appear multiple times. You cannot mix keys with ids. You cannot retrieve more than 500 records at once.

Request

```
GET https://www.teamdesk.net/secure/api/v2/{appid}/{table}/retrieve.{json|xml}?params
```

Retrieve Id and Text column from records with keys 56 and 57

```
GET https://www.teamdesk.net/secure/api/v2/21995/Test/retrieve.json?column=Id&column=Text
&key=56&key=57
```

Retrieve Id and Text column from records with ids 56 and 57

```
GET https://www.teamdesk.net/secure/api/v2/21995/Test/retrieve.json?column=Id&column=Text
&id=56&id=57
```

Response

Response matches the format returned by Select (Table) method.

Document Method

This method allows you to render document for selected records from a table given either their ids. The order the records matches the order ids are passed in.

Parameters

- *table*: the singular name of the table or its *alias* as returned by the Describe (Application) method. Please note that this parameter is embedded into the URL before the method name.
- *document*: the name of the document or its *alias* as returned by the Describe (Table). Please note that this parameter is embedded into the URL before the method name.
- *id*: the value of internal record id. Can appear multiple times. You cannot render the document for more than 500 records at once.

Request

```
GET https://www.teamdesk.net/secure/api/v2/{appid}/{table}/{document}/document?params
```

Render document for Id and Text column from records with id 156 and 157

```
GET https://www.teamdesk.net/secure/api/v2/21995/Test/SampleDoc/document?id=156&id=157
```

Response

Generated document file. Output format (DOCX or PDF) matches document settings defined in the database.

Create/Update/Upsert methods

These three methods are similar by the way they perform, format of the data passed in and returned back.

Create method always tries to create a record and will fail if value of the key column is passed in and the record with such key already exists. Column with no value supplied will be assigned with default value.

Update method, in contrast, requires either internal record's id or a key column to be present and the record to exist in the database. If both are supplied, record's id takes precedence.

Upsert combines the functionality of former two, updating the record if it exists or creating new one.

Please note that **workflow rules are enabled** by default and **each record is processed individually.** For each record passed in methods return short status description to indicate whether processing was successful or failed.

All three methods return an array of short status descriptors, one for each row. Descriptor consist of status code (200 – updated, 201 – created, 304 – not modified or 4xx code in case of error), row's id and key and an array of standard error descriptors if any.

Parameters

- *table*: the singular name of the table or its *alias* as returned by the Describe (Application) method. Please note that this parameter is embedded into the URL before the method name.
- *workflow:* pass 0 to suppress running workflow rules. The user should have *ManageData* administrative permissions otherwise method call will fail.
- *match:* applicable to Update and Upsert methods only and allows to override matching logic. If omitted, methods try to find update the record using key column. If present, it's value should be the name or alias of unique column. Methods then will try to find the record based on the value of unique column. This might be useful for integration with external systems that use different record keying scheme, by e-mail, for example.

Input

All three methods are invoked via POST request. While method extension defines output format, *Content-Type* HTTP header is required and defines the type of the content you are passing in – *application/json* for JSON payload or *text/xml* for XML payload, making it possible to send XML and receive JSON or vice versa.

JSON and XLM data formats match those returned by data retrieval methods but requirements to the format somewhat relaxed. JSON input should be an *array of objects*. Each object consists of key-value pairs; key name should match column name or its alias. XML document should consist of a root element with any name; second level elements denote rows and also can have any names; third level elements denote columns within a row and should be named after column names or their aliases. Since rows processed individually, they do not have to have uniform structure – you can pass one set of data for one row and another set of data for another row.

Format of the column value in both JSON and XML depends on a column type as described in a following table (assuming column name is "c").

Type	JSON	XML
Null	"c": null, "c": ""	<c/> <c></c>
Text, Phone, URL, Email	"c": "text"	<c>text</c>
Multiline	"c": "first line**\n**second line" "c": "first line**\r\n**second line"	<c>first line second line</c>
Checkbox	"c": true "c": false	<c>true</c> <c>false </c>
Numeric	"c": 4.5 "c": -13.2	<c>4.5</c> <c>-13.2</c>
Date	"c": "2004-12-18" "c": "2004-12-18T12:33:42" "c": "2004-12-18T12:33:42-06:00" *(only date part is taken)*	<c>2014-12-18</c> <c>2014-12-1818T12:33:42</c> <c>2004-12-18T12:33:42-06:00</c> *(only date part is taken)*
Time	"c": "12:33:42" "c": "2004-12-18T12:33:42" "c": "2004-12-18T12:33:42-06:00" *(only time part is taken)*	<c>12:33:42</c> <c>2014-12-1818T12:33:42</c> <c>2004-12-18T12:33:42-06:00</c> *(only time part is taken)*
Timestamp	"c": "2004-12-18T12:33:42" "c": "2004-12-18T12:33:42-06:00" *(time zone, if included, is used to coerce time to user's local time)*	<c>2014-12-1818T12:33:42</c> <c>2004-12-18T12:33:42-06:00</c> *(time zone, if included, is used to coerce time to user's local time)*
Duration	"c": 86400 *(seconds)* "c": "PT86400S" *(as in XML format for duration)*	<c>86400</c> *(seconds)* <c>PT86400S</c> *(as in XML format for duration)*
User	"c": "test user <test@test.com>" "c": "test@test.com"	<c>test user <test@test.com></c> <c>test@test.com</c>

Request

```
POST https://www.teamdesk.net/secure/api/v2/{appid}/{table}/upsert.{json|xml}
Content-Type: {application/json|text/xml}
```

Pass batch of 3 records for table Test to Upsert method in JSON format

```
POST https://www.teamdesk.net/secure/api/v2/21995/Test/upsert.json
Content-Type: application/json

[
    {  // will update by key
       "Id": "68",
       "Text": "Update #1",
       "Date": "2014-12-18",
       "Multiline": null
    },
    {  // will update by id
       "@row.id": 12,
       "Id": "69",
       "Text": "Update #2",
    },
    {  // will create
       "Date": "2014-12-18"
       "Time": "12:34:50"
    },
]
```

The method will respond with something like:

```
[
  {
    "status": 200, // updated
    "id": 20,
    "key": "68"
  },
  {
    "status": 400, // error
    "id": 12,
    "key": "69",
    "errors": [
      {
        "error": 409,
        "message": "Cannot write duplicate value \"69\" into column \"Id\""
      }
    ]
  },
  {
    "status": 201, // created
    "id": 48,
    "key": "91"
  }
]
```

Let's do the same in XML format

```
POST https://www.teamdesk.net/secure/api/v2/21995/Test/upsert.xml
Content-Type: text/xml

<Request>
    <row>
       <Id>68</Id>
       <Text>Update #1</Text>
       <Date>2014-12-18</Date>
       <Multiline/>
    </row>
    <row id="12">
       <Id>69</Id>
       <Text>Update #2</Text>
```

```
    </row>
</Request>
```

The response will be a sort of

```
<Response>
  <row>
    <id>20</id>
    <key>68</key>
    <status>304</status>
  </row>
  <row>
    <id>12</id>
    <key>69</key>
    <status>400</status>
    <errors>
      <Error>
        <error>409</error>
        <message>Cannot write duplicate value "69" into column "Id"</message>
      </Error>
    </errors>
    <id>12</id>
    <key>69</key>
    <status>400</status>
  </row>
</Response>
```

Sending attachment data

In SOAP API we had separate SetAttachment method but now workflow triggers are run by default and setting the data first and then modifying attachment may result in running modification triggers multiple times. In order to avoid that, modification methods support sending files side by side with records data.

In this case the data format resembles plain text e-mail messages with attachments – the body of the request should be formatted according to multipart/related rules, for example:

```
POST https://www.teamdesk.net/secure/api/v2/21995/Test/upsert.json
Content-Type: multipart/related; boundary=example-1

--example-1
Content-ID: <file-1>
Content-Type: text/plain;
Content-Disposition: attachment; filename="sample.txt"

This is sample text file
--example-1
Content-Type: application/json

[{ "Id": "69", "File": "cid:file-1" }]
--example-1--
```

Request's *Content-Type* header should be set to *multipart/related* and its *boundary* parameter should result to some unique string. Two minus signs followed by boundary string indicates beginning of the part. The request should end with two minus signs followed by boundary string and two more minus signs. Part's headers are separated from the data by an empty line.

The part considered a file if *Content-Disposition* header is present and contains *filename* parameter. *Content-ID* header is used to uniquely identify file's data. The part without *filename* parameter

assumed to contain record's data. Now to refer to a file simply set the value of attachment column to **cid:***file-content-id*.

The order of the parts plays no role. Data part may precede, follow or placed in between file parts.

Delete Method

This method allows you to delete records with matching keys or ids.

Parameters

- *table*: the singular name of the table or its *alias* as returned by the Describe (Application) method. Please note that this parameter is embedded into the URL before the method name.
- *key* or *id*: the value of key column or internal record id to retrieve. Can appear multiple times. You cannot mix keys with ids. You cannot retrieve more than 500 records at once.
- *workflow:* pass 0 to suppress running workflow rules. The user should have *ManageData* administrative permissions otherwise method call will fail.
- *purge:* pass 1 to bypass recycle bin and delete record immediately.

Each record is deleted individually. If parameters passed validation successfully, HTTP status 200 is returned even in case there was a failure to delete row(s). For each key/id passed in the method returns short status descriptor. Status descriptor consist of status code (200 indicates successful deletion), a copy of an id or a key, and, in case of error, standard error descriptor containing explanatory message.

Request

```
GET https://www.teamdesk.net/secure/api/v2/{appid}/{table}/delete.{json|xml}?parameters
```

Deletes records with keys 56 and 57

```
GET https://www.teamdesk.net/secure/api/v2/21995/Test/delete.json?key=56&key=57
```

Deletes records with ids 56 and 57, attempts to suppress workflow rules

```
GET https://www.teamdesk.net/secure/api/v2/21995/Test/delete.xml?id=56&id=57&workflow=0
```

Response (JSON)

```
[ // An array of status descriptors
  {
    "status": 200, // Deleted successfully
    "id": 56,
  },
  {
    "status": 403,    // In case of error non-200 status is returned
    "id": 57,
    "error": {        // And the standard error descriptor
      "error": 403,
      "code": 4000,
      "message": "Record is not found or not accessible"
    }
  }
]
```

Response (XML)

```
<Response>
  <row>
    <id>56</id>
    <status>200</status>
  </row>
  <row>
    <id>57</id>
    <status>403</status>
    <error>
      <error>403</error>
      <code>4000</code>
      <message>Record is not found or not accessible</message>
    </error>
  </row>
</Response>
```

Updated Method

This method allows you to retrieve the list of records that where updated in a range of dates. Recently modified records are returned on top. There is no limit to the number of records returned.

Parameters

- *table*: the singular name of the table or its *alias* as returned by the Describe (Application) method. Please note that this parameter is embedded into the URL before the method name.
- *from*: optional, the timestamp indicating the start of the range. Time zone, if included is used to convert the supplied time to user's local time.
- *to*: optional, the timestamp indicating the end of the range. Time zone is handled the same way as in from parameter.

Request

```
GET https://www.teamdesk.net/secure/api/v2/{appid}/{table}/updated.{json|xml}?params
```

Retrieve records from Test table updated in 2014

```
GET https://www.teamdesk.net/secure/api/v2/21995/Test/updated.json?from=2014-01-
01Z&to=2014-12-31T23:59:59Z
```

Response

The response is identical to Select (Table) method when queried for key, date created and date modified columns aliased as "key", "created" and "modified" for uniformity.

```
[
  {
    "@row.id": 12,
    "key": "60",
    "created": "2014-12-03T18:02:31.79+02:00",
    "modified": "2014-12-03T18:05:22.46+02:00"
  },
  {
    "@row.id": 13,
    "key": "61",
    "created": "2014-12-04T18:30:24.39+02:00",
    "modified": "2014-12-04T18:30:24.39+02:00"
  }
  // , { ... } more rows
]
```

Deleted Method

This method allows you to retrieve the list of records in the recycle bin that where deleted in a range of dates. Recently deleted records are returned on top. There is no limit to the number of records returned. Administrators with *ManageData* permission will receive back the list of records deleted by any user. Other users will receive back only records they have deleted themselves.

Parameters

- *table*: the singular name of the table or its *alias* as returned by the Describe (Application) method. Please note that this parameter is embedded into the URL before the method name.
- *from*: optional, the timestamp indicating the start of the range. Time zone, if included is used to convert the supplied time to user's local time.
- *to*: optional, the timestamp indicating the end of the range. Time zone is handled the same way as in from parameter.

Request

```
GET https://www.teamdesk.net/secure/api/v2/{appid}/{table}/deleted.{json|xml}?params
```

Retrieve records from Test table updated in 2014

```
GET https://www.teamdesk.net/secure/api/v2/21995/Test/deleted.json?from=2014-01-
01Z&to=2014-12-31T23:59:59Z
```

Response

The response is identical to Select (View) method when queried for Record Picker view plus two additional columns named *deletedby* and *deleted*.

```
[
  {
    "@row.id": 14,
    "Text": "AAA",
    "deletedby": "test user <test@test.com>",
    "deleted": "2014-12-16T17:44:36.523+02:00"
  },
  // , { ... } more rows
]
```

Attachment Method

Parameters

- *table*: the singular name of the table or its *alias* as returned by the Describe (Application) method. Please note that this parameter is embedded into the URL before the method name.
- *column*: the name or an alias of the column to query. Please note that this parameter is embedded into the URL after the table name/alias.
- *key* or *id*: the value of key column or internal record id to retrieve. You should supply either key or id but not both, unless you are requesting data from public attachment column. In this case, these parameters are optional.

And either:

- *revision:* the revision to retrieve the file for. The column should have Allow See Revision History property checked to enable to access the information about previous revisions.

Or

- *guid*: globally unique identifier of the file.

Please note that output format should not be specified for this method as it returns the content of the file. Also, you can use HEAD HTTP method to retrieve file metadata for examination without retrieving its content.

Request

```
GET|HEAD
https://www.teamdesk.net/secure/api/v2/{appid}/{table}/{column}/attachment?parameters
```

Or

```
GET|HEAD https://www.teamdesk.net/secure/api/v2/{appid}/{table}/{column}/attachment/{guid}
```

Retrieves the file metadata identified by GUID from the column File of in a table Test via HEAD.

```
HEAD https://www.teamdesk.net/secure/api/v2/21995/Test/File/attachment/5c98ad47-c6ca-437d-
8c2f-cb2f929c54b8
```

Response

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Content-Disposition: attachment; filename=DSC02688.JPG; filename*=utf-8''DSC02688.JPG
X-Revision: 2
Content-Type: image/jpeg
Content-Length: 326133
Last-Modified: Wed, 03 Dec 2014 16:02:31 GMT
X-Author: test user <test@test.com>
```

Retrieves latest file revision from the column File of the record with id 56 in a table Test.

```
GET https://www.teamdesk.net/secure/api/v2/21995/Test/File/attachment?id=56
```

Response

```
HTTP/1.1 200 OK
Cache-Control: private, must-revalidate, max-age=0
ETag: "1234567890.1234567890"
Content-Disposition: attachment; filename=DSC02688.JPG; filename*=utf-8''DSC02688.JPG
X-Revision: 2
Content-Type: image/jpeg
Content-Length: 326133
Last-Modified: Wed, 03 Dec 2014 16:02:31 GMT
X-Author: test user <test@test.com>

... binary data ...
```

Attachments Method

This method allows you to retrieve the information about the files stored in attachment column.

Parameters

- *table*: the singular name of the table or its *alias* as returned by the Describe (Application) method. Please note that this parameter is embedded into the URL before the method name.
- *column*: the name or an alias of the column to query. Please note that this parameter is embedded into the URL after the table name/alias.
- *key* or *id*: the value of key column or internal record id to retrieve. You should supply either key or id but not both.
- *revisions:* optional, default to 1, the number of revisions to report the information for. The column should have Allow See Revision History property checked to enable to access the information about previous revisions.

Request

```
GET https://www.teamdesk.net/secure/api/v2/{appid}/{table}/{column}/attachments.{json|xml}
?parameters
```

Retrieve list of attachments from File column in a Test table from a record with id 56

```
GET https://www.teamdesk.net/secure/api/v2/21995/Test/File/attachments.json?id=56
```

Response (JSON)

```
[
  {
    "name": "DSC02688.JPG",
    "revision": 2,
    "type": "image/jpeg",
    "size": 326133,
    "guid": "5c98ad47-c6ca-437d-8c2f-cb2f929c54b8",
    "author": "test user <test@test.com>",
    "created": "2014-12-03T18:02:31.787+02:00"
  }
  //, { ... }
]
```

Response (XML)

```
<Response>
  <attachment>
    <name>DSC02688.JPG</name>
    <revision>2</revision>
    <type>image/jpeg</type>
    <size>326133</size>
    <guid>5c98ad47-c6ca-437d-8c2f-cb2f929c54b8</guid>
    <author>Kirill Bondar &lt;kir@skyeytech.com&gt;</author>
    <created>2014-12-03T18:02:31.787+02:00</created>
  </attachment>
  <!-- more revisions -->
</Response>
```

Setup | User method

Allows adding or modifying the user in your database. Authorized user must have *ManageUsers* privilege to call this method.

Parameters:

- *email:* user's email. Can be either email address as is or take the form of "Name" <email>. If user account does not exist it is created. In this case, account's first and last names properties will be pre-filled from name part, if present.
- *role:* The role to assign. Pass empty string (e.g. *role=*) to disable user.
- *defaultSet:* 1 to include user in default user set. 0 otherwise.
- *external*: 1 if the user is external user.
- *invite*: 1 to send default invitation letter, 0 to skip sending. Default is 0.

Request

```
GET https://www.teamdesk.net/secure/api/v2/{appid}/setup/user.{json|xml}?parameters
```

Response (JSON)

```
{
  // 201 – created, 200 – updated
  "status": 201,
  // authorization ticket for invitation email
  "ticket": "5c98ad47-c6ca-437d-8c2f-cb2f929c54b8"
}
```

Response (XML)

```
<Response>
  <status>201</status>
  <ticket>5c98ad47-c6ca-437d-8c2f-cb2f929c54b8</ticket>
</Response>
```