Anki-Connect

#anki #anki-connect #extension #gpl license #japanese #python #yomichan

Anki-Connect enables external applications such as Yomichan to communicate with Anki over a simple HTTP API. Its capabilities include executing queries against the user's card deck, automatically creating new cards, and more. Anki-Connect is compatible with the latest stable (2.1.x) releases of Anki; older versions (2.0.x and below) are no longer supported.

Installation

The installation process is similar to other Anki plugins and can be accomplished in three steps:

1. Open the `Install Add-on` dialog by selecting `Tools | Add-ons | Get Add-ons...` in Anki.
2. Input 2055492159 into the text box labeled `Code` and press the `OK` button to proceed.
3. Restart Anki when prompted to do so in order to complete the installation of Anki-Connect.

Anki must be kept running in the background in order for other applications to be able to use Anki-Connect. You can verify that Anki-Connect is running at any time by accessing `localhost:8765` in your browser. If the server is running, you will see the message `Anki-Connect` displayed in your browser window.

Notes for Windows Users

Windows users may see a firewall nag dialog box appear on Anki startup. This occurs because Anki-Connect runs a local HTTP server in order to enable other applications to connect to it. The host application, Anki, must be unblocked for this plugin to function correctly.

Notes for MacOS Users

Starting with Mac OS X Mavericks, a feature named *App Nap* has been introduced to the operating system. This feature causes certain applications which are open (but not visible) to be placed in a suspended state. As this behavior causes Anki-Connect to stop working while you have another window in the foreground, App Nap should be disabled for Anki:

1. Start the Terminal application.
2. Execute the following commands in the terminal window:

```
defaults write net.ankiweb.dtop NSAppSleepDisabled -bool true
```

Application Interface for Developers

Anki-Connect exposes internal Anki features to external applications via an easy to use API. After being installed, this plugin will start an HTTP server on port 8765 whenever Anki is launched. Other applications (including browser extensions) can then communicate with it via HTTP requests.

By default, Anki-Connect will only bind the HTTP server to the `127.0.0.1` IP address, so that you will only be able to access it from the same host on which it is running. If you need to access it over a network, you can change the binding address in the configuration. Go to Tools->Add-ons->AnkiConnect->Config and change the "webBindAddress" value. For example, you can set it to `0.0.0.0` in order to bind it to all network interfaces on your host. This also requires a restart for Anki.

Sample Invocation

Every request consists of a JSON-encoded object containing an `action`, a `version`, contextual `params`, and a `key` value used for authentication (which is optional and can be omitted by default). Anki-Connect will respond with an object containing two fields: `result` and `error`. The `result` field contains the return value of the executed API, and the `error` field is a description of any exception thrown during API execution (the value `null` is used if execution completed successfully).

Sample successful response:

```
{"result": ["Default", "Filtered Deck 1"], "error": null}
```

Samples of failed responses:

```
{"result": null, "error": "unsupported action"}
```

```
{"result": null, "error": "guiBrowse() got an unexpected keyword argument
```

For compatibility with clients designed to work with older versions of Anki-Connect, failing to provide a `version` field in the request will make the version default to 4. Furthermore, when the provided version is level 4 or below, the API response will only contain the value of the `result`; no `error` field is available for error handling.

You can use whatever language or tool you like to issue request to Anki-Connect, but a couple of simple examples are included below as reference.

Powershell

```
(Invoke-RestMethod -Uri http://localhost:8765 -Method Post -Body '{"actic
```

Python

```
import json
import urllib.request

def request(action, **params):
    return {'action': action, 'params': params, 'version': 6}

def invoke(action, **params):
    requestJson = json.dumps(request(action, **params)).encode('utf-8')
    response = json.load(urllib.request.urlopen(urllib.request.Request('h
    if len(response) != 2:
        raise Exception('response has an unexpected number of fields')
    if 'error' not in response:
        raise Exception('response is missing required error field')
    if 'result' not in response:
        raise Exception('response is missing required result field')
    if response['error'] is not None:
        raise Exception(response['error'])
    return response['result']

invoke('createDeck', deck='test1')
result = invoke('deckNames')
print('got list of decks: {}'.format(result))
```

JavaScript

```
function invoke(action, version, params={}) {
    return new Promise((resolve, reject) => {
        const xhr = new XMLHttpRequest();
        xhr.addEventListener('error', () => reject('failed to issue reque
        xhr.addEventListener('load', () => {
            try {
                const response = JSON.parse(xhr.responseText);
                if (Object.getOwnPropertyNames(response).length != 2) {
                    throw 'response has an unexpected number of fields';
                }
                if (!response.hasOwnProperty('error')) {
                    throw 'response is missing required error field';
                }
```

```
                    throw response.error;
                }
                resolve(response.result);
            } catch (e) {
                reject(e);
            }
        });

        xhr.open('POST', 'http://127.0.0.1:8765');
        xhr.send(JSON.stringify({action, version, params}));
    });
}

await invoke('createDeck', 6, {deck: 'test1'});
const result = await invoke('deckNames', 6);
console.log(`got list of decks: ${result}`);
```

Authentication

Anki-Connect supports requiring authentication in order to make API requests. This support is *disabled* by default, but can be enabled by setting the `apiKey` field of Anki-Config's settings (Tools->Add-ons->AnkiConnect->Config) to a desired string. If you have done so, you should see the `requestPermission` API request return `true` for `requireApiKey`. You then must include an additional parameter called `key` in any further API request bodies, whose value must match the configured API key.

Hey, could you add a new action to support $FEATURE?

The primary goal for Anki-Connect was to support real-time flash card creation from the Yomichan browser extension. The current API provides all the required actions to make this happen. I recognise that the role of Anki-Connect has evolved from this original vision, and I am happy to review new feature requests.

With that said, *this project operates on a self-serve model*. If you would like a new feature, create a PR. I'll review it and if it looks good, it will be merged in. *Requests to add new features without accompanying pull requests will not be serviced*. Make sure that your pull request meets the following criteria:

- Attempt to match style of the surrounding code.
- Have accompanying documentation with examples.
- Have accompanying tests that verify operation.
- Implement features useful in other applications.

Supported Actions

available for use. Search parameters are passed to Anki, check the docs for more information: https://docs.ankiweb.net/searching.html

- Card Actions
- Deck Actions
- Graphical Actions
- Media Actions
- Miscellaneous Actions
- Model Actions
- Note Actions
- Statistic Actions

Card Actions

`getEaseFactors`

- Returns an array with the ease factor for each of the given cards (in the same order).

  ▶ *Sample request:*
  ▶ *Sample result:*

`setEaseFactors`

- Sets ease factor of cards by card ID; returns `true` if successful (all cards existed) or `false` otherwise.

  ▶ *Sample request:*
  ▶ *Sample result:*

`setSpecificValueOfCard`

- Sets specific value of a single card. Given the risk of wreaking havor in the database when changing some of the values of a card, some of the keys require the argument "warning_check" set to True. This can be used to set a card's flag, change it's ease factor, change the review order in a filtered deck and change the column "data" (not currently used by anki apparantly), and many other values. A list of values and explanation of their respective utility can be found at AnkiDroid's wiki.

  ▶ *Sample request:*
  ▶ *Sample result:*

`suspend`

- ▶ *Sample result:*

`unsuspend`

- Unsuspend cards by card ID; returns `true` if successful (at least one card was previously suspended) or `false` otherwise.

  - ▶ *Sample request:*
  - ▶ *Sample result:*

`suspended`

- Check if card is suspended by its ID. Returns `true` if suspended, `false` otherwise.

  - ▶ *Sample request:*
  - ▶ *Sample result:*

`areSuspended`

- Returns an array indicating whether each of the given cards is suspended (in the same order). If card doesn't exist returns `null`.

  - ▶ *Sample request:*
  - ▶ *Sample result:*

`areDue`

- Returns an array indicating whether each of the given cards is due (in the same order). *Note*: cards in the learning queue with a large interval (over 20 minutes) are treated as not due until the time of their interval has passed, to match the way Anki treats them when reviewing.

  - ▶ *Sample request:*
  - ▶ *Sample result:*

`getIntervals`

- Returns an array of the most recent intervals for each given card ID, or a 2-dimensional array of all the intervals for each given card ID when `complete` is `true`. Negative intervals are in seconds and positive intervals in days.

  - ▶ *Sample request 1:*
  - ▶ *Sample result 1:*
  - ▶ *Sample request 2:*

- Returns an array of card IDs for a given query. Functionally identical to `guiBrowse` but doesn't use the GUI for better performance.

  ▶ *Sample request:*
  ▶ *Sample result:*

`cardsToNotes`

- Returns an unordered array of note IDs for the given card IDs. For cards with the same note, the ID is only given once in the array.

  ▶ *Sample request:*
  ▶ *Sample result:*

`cardsModTime`

- Returns a list of objects containings for each card ID the modification time. This function is about 15 times faster than executing `cardsInfo`.

  ▶ *Sample request:*
  ▶ *Sample result:*

`cardsInfo`

- Returns a list of objects containing for each card ID the card fields, front and back sides including CSS, note type, the note that the card belongs to, and deck name, last modification timestamp as well as ease and interval.

  ▶ *Sample request:*
  ▶ *Sample result:*

`forgetCards`

- Forget cards, making the cards new again.

  ▶ *Sample request:*
  ▶ *Sample result:*

`relearnCards`

- Make cards be "relearning".

  ▶ *Sample request:*
  ▶ *Sample result:*

- ▶ *Sample request:*
- ▶ *Sample result:*

Deck Actions

`deckNames`

- Gets the complete list of deck names for the current user.

  - ▶ *Sample request:*
  - ▶ *Sample result:*

`deckNamesAndIds`

- Gets the complete list of deck names and their respective IDs for the current user.

  - ▶ *Sample request:*
  - ▶ *Sample result:*

`getDecks`

- Accepts an array of card IDs and returns an object with each deck name as a key, and its value an array of the given cards which belong to it.

  - ▶ *Sample request:*
  - ▶ *Sample result:*

`createDeck`

- Create a new empty deck. Will not overwrite a deck that exists with the same name.

  - ▶ *Sample request:*
  - ▶ *Sample result:*

`changeDeck`

- Moves cards with the given IDs to a different deck, creating the deck if it doesn't exist yet.

  - ▶ *Sample request:*
  - ▶ *Sample result:*

`deleteDecks`

▶ *Sample result:*

`getDeckConfig`

- Gets the configuration group object for the given deck.

▶ *Sample request:*
▶ *Sample result:*

`saveDeckConfig`

- Saves the given configuration group, returning `true` on success or `false` if the ID of the configuration group is invalid (such as when it does not exist).

▶ *Sample request:*
▶ *Sample result:*

`setDeckConfigId`

- Changes the configuration group for the given decks to the one with the given ID. Returns `true` on success or `false` if the given configuration group or any of the given decks do not exist.

▶ *Sample request:*
▶ *Sample result:*

`cloneDeckConfigId`

- Creates a new configuration group with the given name, cloning from the group with the given ID, or from the default group if this is unspecified. Returns the ID of the new configuration group, or `false` if the specified group to clone from does not exist.

▶ *Sample request:*
▶ *Sample result:*

`removeDeckConfigId`

- Removes the configuration group with the given ID, returning `true` if successful, or `false` if attempting to remove either the default configuration group (ID = 1) or a configuration group that does not exist.

▶ *Sample request:*
▶ *Sample result:*

`getDeckStats`

Graphical Actions

`guiBrowse`

- Invokes the *Card Browser* dialog and searches for a given query. Returns an array of identifiers of the cards that were found. Query syntax is documented here.

  Optionally, the `reorderCards` property can be provided to reorder the cards shown in the *Card Browser*. This is an array including the `order` and `columnId` objects. `order` can be either `ascending` or `descending` while `columnId` can be one of several column identifiers (as documented in the Anki source code). The specified column needs to be visible in the *Card Browser*.

  ▶ *Sample request:*
  ▶ *Sample result:*

`guiSelectNote`

- Finds the open instance of the *Card Browser* dialog and selects a note given a note identifier. Returns `true` if the *Card Browser* is open, `false` otherwise.

  ▶ *Sample request:*
  ▶ *Sample result:*

`guiSelectedNotes`

- Finds the open instance of the *Card Browser* dialog and returns an array of identifiers of the notes that are selected. Returns an empty list if the browser is not open.

  ▶ *Sample request:*
  ▶ *Sample result:*

`guiAddCards`

- Invokes the *Add Cards* dialog, presets the note using the given deck and model, with the provided field values and tags. Invoking it multiple times closes the old window and *reopen the window* with the new provided values.

  Audio, video, and picture files can be embedded into the fields via the `audio`, `video`, and `picture` keys, respectively. Refer to the documentation of `addNote` and `storeMediaFile` for an explanation of these fields.

▶ *Sample result:*

`guiEditNote`

- Opens the *Edit* dialog with a note corresponding to given note ID. The dialog is similar to the *Edit Current* dialog, but:
  - has a Preview button to preview the cards for the note
  - has a Browse button to open the browser with these cards
  - has Previous/Back buttons to navigate the history of the dialog
  - has no bar with the Close button

▶ *Sample request:*
▶ *Sample result:*

`guiCurrentCard`

- Returns information about the current card or `null` if not in review mode.

▶ *Sample request:*
▶ *Sample result:*

`guiStartCardTimer`

- Starts or resets the `timerStarted` value for the current card. This is useful for deferring the start time to when it is displayed via the API, allowing the recorded time taken to answer the card to be more accurate when calling `guiAnswerCard`.

▶ *Sample request:*
▶ *Sample result:*

`guiShowQuestion`

- Shows question text for the current card; returns `true` if in review mode or `false` otherwise.

▶ *Sample request:*
▶ *Sample result:*

`guiShowAnswer`

- Shows answer text for the current card; returns `true` if in review mode or `false` otherwise.

▶ *Sample request:*

- Answers the current card; returns `true` if succeeded or `false` otherwise. Note that the answer for the current card must be displayed before before any answer can be accepted by Anki.

  ▶ *Sample request:*
  ▶ *Sample result:*

`guiUndo`

- Undo the last action / card; returns `true` if succeeded or `false` otherwise.

  ▶ *Sample request:*
  ▶ *Sample result:*

`guiDeckOverview`

- Opens the *Deck Overview* dialog for the deck with the given name; returns `true` if succeeded or `false` otherwise.

  ▶ *Sample request:*
  ▶ *Sample result:*

`guiDeckBrowser`

- Opens the *Deck Browser* dialog.

  ▶ *Sample request:*
  ▶ *Sample result:*

`guiDeckReview`

- Starts review for the deck with the given name; returns `true` if succeeded or `false` otherwise.

  ▶ *Sample request:*
  ▶ *Sample result:*

`guiImportFile`

- Invokes the *Import... (Ctrl+Shift+I)* dialog with an optional file path. Brings up the dialog for user to review the import. Supports all file types that Anki supports. Brings open file dialog if no path is provided. Forward slashes must be used in the path on Windows. Only supported for Anki 2.1.52+.

  ▶ *Sample request:*

- Schedules a request to gracefully close Anki. This operation is asynchronous, so it will return immediately and won't wait until the Anki process actually terminates.

  ▶ *Sample request:*
  ▶ *Sample result:*

`guiCheckDatabase`

- Requests a database check, but returns immediately without waiting for the check to complete. Therefore, the action will always return `true` even if errors are detected during the database check.

  ▶ *Sample request:*
  ▶ *Sample result:*

Media Actions

`storeMediaFile`

- Stores a file with the specified base64-encoded contents inside the media folder. Alternatively you can specify a absolute file path, or a url from where the file shell be downloaded. If more than one of `data`, `path` and `url` are provided, the `data` field will be used first, then `path`, and finally `url`. To prevent Anki from removing files not used by any cards (e.g. for configuration files), prefix the filename with an underscore. These files are still synchronized to AnkiWeb. Any existing file with the same name is deleted by default. Set `deleteExisting` to false to prevent that by letting Anki give the new file a non-conflicting name.

  ▶ *Sample request (relative path):*
  ▶ *Sample result (relative path):*
  ▶ *Sample request (absolute path):*
  ▶ *Sample result (absolute path):*
  ▶ *Sample request (url):*
  ▶ *Sample result (url):*

`retrieveMediaFile`

- Retrieves the base64-encoded contents of the specified file, returning `false` if the file does not exist.

  ▶ *Sample request:*
  ▶ *Sample result:*

- ▶ *Sample request:*
- ▶ *Sample result:*

`getMediaDirPath`

- Gets the full path to the `collection.media` folder of the currently opened profile.

  - ▶ *Sample request:*
  - ▶ *Sample result:*

`deleteMediaFile`

- Deletes the specified file inside the media folder.

  - ▶ *Sample request:*
  - ▶ *Sample result:*

Miscellaneous Actions

`requestPermission`

- Requests permission to use the API exposed by this plugin. This method does not require the API key, and is the only one that accepts requests from any origin; the other methods only accept requests from trusted origins, which are listed under `webCorsOriginList` in the add-on config. `localhost` is trusted by default.

  Calling this method from an untrusted origin will display a popup in Anki asking the user whether they want to allow your origin to use the API; calls from trusted origins will return the result without displaying the popup. When denying permission, the user may also choose to ignore further permission requests from that origin. These origins end up in the `ignoreOriginList`, editable via the add-on config.

  The result always contains the `permission` field, which in turn contains either the string `granted` or `denied`, corresponding to whether your origin is trusted. If your origin is trusted, the fields `requireApiKey` (`true` if required) and `version` will also be returned.

  This should be the first call you make to make sure that your application and Anki-Connect are able to communicate properly with each other. New versions of Anki-Connect are backwards compatible; as long as you are using actions which are available in the reported Anki-Connect version or earlier, everything should work fine.

  - ▶ *Sample request:*

- Gets the version of the API exposed by this plugin. Currently versions `1` through `6` are defined.

  ▶ *Sample request:*
  ▶ *Sample result:*

`apiReflect`

- Gets information about the AnkiConnect APIs available. The request supports the following params:

  - `scopes` - An array of scopes to get reflection information about. The only currently supported value is `"actions"`.
  - `actions` - Either `null` or an array of API method names to check for. If the value is `null`, the result will list all of the available API actions. If the value is an array of strings, the result will only contain actions which were in this array.

  The result will contain a list of which scopes were used and a value for each scope. For example, the `"actions"` scope will contain a `"actions"` property which contains a list of supported action names.

  ▶ *Sample request:*
  ▶ *Sample result:*

`sync`

- Synchronizes the local Anki collections with AnkiWeb.

  ▶ *Sample request:*
  ▶ *Sample result:*

`getProfiles`

- Retrieve the list of profiles.

  ▶ *Sample request:*
  ▶ *Sample result:*

`loadProfile`

- Selects the profile specified in request.

  ▶ *Sample request:*
  ▶ *Sample result:*

- ▶ *Sample request:*
- ▶ *Sample result:*

`exportPackage`

- Exports a given deck in `.apkg` format. Returns `true` if successful or `false` otherwise. The optional property `includeSched` (default is `false`) can be specified to include the cards' scheduling data.

- ▶ *Sample request:*
- ▶ *Sample result:*

`importPackage`

- Imports a file in `.apkg` format into the collection. Returns `true` if successful or `false` otherwise. Note that the file path is relative to Anki's collection.media folder, not to the client.

- ▶ *Sample request:*
- ▶ *Sample result:*

`reloadCollection`

- Tells anki to reload all data from the database.

- ▶ *Sample request:*
- ▶ *Sample result:*

Model Actions

`modelNames`

- Gets the complete list of model names for the current user.

- ▶ *Sample request:*
- ▶ *Sample result:*

`modelNamesAndIds`

- Gets the complete list of model names and their corresponding IDs for the current user.

- ▶ *Sample request:*
- ▶ *Sample result:*

- ▶ *Sample request:*
- ▶ *Sample result:*

`findModelsByName`

- Gets a list of models for the provided model names from the current user.

  - ▶ *Sample request:*
  - ▶ *Sample result:*

`modelFieldNames`

- Gets the complete list of field names for the provided model name.

  - ▶ *Sample request:*
  - ▶ *Sample result:*

`modelFieldDescriptions`

- Gets the complete list of field descriptions (the text seen in the gui editor when a field is empty) for the provided model name.

  - ▶ *Sample request:*
  - ▶ *Sample result:*

`modelFieldFonts`

- Gets the complete list of fonts along with their font sizes.

  - ▶ *Sample request:*
  - ▶ *Sample result:*

`modelFieldsOnTemplates`

- Returns an object indicating the fields on the question and answer side of each card template for the given model name. The question side is given first in each array.

  - ▶ *Sample request:*
  - ▶ *Sample result:*

`createModel`

- Creates a new model to be used in Anki. User must provide the `modelName`, `inOrderFields` and `cardTemplates` to be used in the model. There are optional fields

the card names will be `Card 1`, `Card 2`, and so on.

▶ *Sample request:*
▶ *Sample result:*

`modelTemplates`

- Returns an object indicating the template content for each card connected to the provided model by name.

▶ *Sample request:*
▶ *Sample result:*

`modelStyling`

- Gets the CSS styling for the provided model by name.

▶ *Sample request:*
▶ *Sample result:*

`updateModelTemplates`

- Modify the templates of an existing model by name. Only specifies cards and specified sides will be modified. If an existing card or side is not included in the request, it will be left unchanged.

▶ *Sample request:*
▶ *Sample result:*

`updateModelStyling`

- Modify the CSS styling of an existing model by name.

▶ *Sample request:*
▶ *Sample result:*

`findAndReplaceInModels`

- Find and replace string in existing model by model name. Customise to replace in front, back or css by setting to true/false.

▶ *Sample request:*
▶ *Sample result:*

`modelTemplateRename`

`modelTemplateReposition`

- Repositions a template in an existing model.

  The value of `index` starts at 0. For example, an index of `0` puts the template in the first position, and an index of `2` puts the template in the third position.

  ▶ *Sample request:*
  ▶ *Sample result:*

`modelTemplateAdd`

- Adds a template to an existing model by name. If you want to update an existing template, use `updateModelTemplates`.

  ▶ *Sample request:*
  ▶ *Sample result:*

`modelTemplateRemove`

- Removes a template from an existing model.

  ▶ *Sample request:*
  ▶ *Sample result:*

`modelFieldRename`

- Rename the field name of a given model.

  ▶ *Sample request:*
  ▶ *Sample result:*

`modelFieldReposition`

- Reposition the field within the field list of a given model.

  The value of `index` starts at 0. For example, an index of `0` puts the field in the first position, and an index of `2` puts the field in the third position.

  ▶ *Sample request:*
  ▶ *Sample result:*

`modelFieldAdd`

- Creates a new field within a given model.

▶ *Sample result:*

`modelFieldRemove`

- Deletes a field within a given model.

▶ *Sample request:*
▶ *Sample result:*

`modelFieldSetFont`

- Sets the font for a field within a given model.

▶ *Sample request:*
▶ *Sample result:*

`modelFieldSetFontSize`

- Sets the font size for a field within a given model.

▶ *Sample request:*
▶ *Sample result:*

`modelFieldSetDescription`

- Sets the description (the text seen in the gui editor when a field is empty) for a field within a given model.

  Older versions of Anki (2.1.49 and below) do not have field descriptions. In that case, this will return with `false`.

▶ *Sample request:*
▶ *Sample result:*

Note Actions

`addNote`

- Creates a note using the given deck and model, with the provided field values and tags. Returns the identifier of the created note created on success, and `null` on failure.

  Anki-Connect can download audio, video, and picture files and embed them in newly created notes. The corresponding `audio`, `video`, and `picture` note members are optional and can be omitted. If you choose to include any of them, they should contain a

with an MD5 hash that matches the provided value. This is useful for avoiding the saving of error pages and stub files. The `fields` member is a list of fields that should play audio or video, or show a picture when the card is displayed in Anki. The `allowDuplicate` member inside `options` group can be set to true to enable adding duplicate cards. Normally duplicate cards can not be added and trigger exception.

The `duplicateScope` member inside `options` can be used to specify the scope for which duplicates are checked. A value of `"deck"` will only check for duplicates in the target deck; any other value will check the entire collection.

The `duplicateScopeOptions` object can be used to specify some additional settings:

- `duplicateScopeOptions.deckName` will specify which deck to use for checking duplicates in. If undefined or `null`, the target deck will be used.
- `duplicateScopeOptions.checkChildren` will change whether or not duplicate cards are checked in child decks. The default value is `false`.
- `duplicateScopeOptions.checkAllModels` specifies whether duplicate checks are performed across all note types. The default value is `false`.

▶ *Sample request:*
▶ *Sample result:*

`addNotes`

- Creates multiple notes using the given deck and model, with the provided field values and tags. Returns an array of identifiers of the created notes (notes that could not be created will have a `null` identifier). Please see the documentation for `addNote` for an explanation of objects in the `notes` array.

▶ *Sample request:*
▶ *Sample result:*

`canAddNotes`

- Accepts an array of objects which define parameters for candidate notes (see `addNote`) and returns an array of booleans indicating whether or not the parameters at the corresponding index could be used to create a new note.

▶ *Sample request:*
▶ *Sample result:*

`canAddNotesWithErrorDetail`

be used to create a new note.
    - `error` contains an explanation of why a note cannot be added.
  - ▶ *Sample request:*
  - ▶ *Sample result:*

`updateNoteFields`

- Modify the fields of an existing note. You can also include audio, video, or picture files which will be added to the note with an optional `audio`, `video`, or `picture` property. Please see the documentation for `addNote` for an explanation of objects in the `audio`, `video`, or `picture` array.

  > ***Warning**: You must not be viewing the note that you are updating on your Anki browser, otherwise the fields will not update. See this issue for further details.*

  - ▶ *Sample request:*
  - ▶ *Sample result:*

`updateNote`

- Modify the fields and/or tags of an existing note. In other words, combines `updateNoteFields` and `updateNoteTags`. Please see their documentation for an explanation of all properties.

  Either `fields` or `tags` property can be omitted without affecting the other. Thus valid requests to `updateNoteFields` also work with `updateNote`. The note must have the `fields` property in order to update the optional audio, video, or picture objects.

  If neither `fields` nor `tags` are provided, the method will fail. Fields are updated first and are not rolled back if updating tags fails. Tags are not updated if updating fields fails.

  > ***Warning** You must not be viewing the note that you are updating on your Anki browser, otherwise the fields will not update. See this issue for further details.*

  - ▶ *Sample request:*
  - ▶ *Sample result:*

`updateNoteTags`

- Set a note's tags by note ID. Old tags will be removed.

  - ▶ *Sample request:*
  - ▶ *Sample result:*

- ▶ *Sample request:*
- ▶ *Sample result:*

`addTags`

- Adds tags to notes by note ID.

  - ▶ *Sample request:*
  - ▶ *Sample result:*

`removeTags`

- Remove tags from notes by note ID.

  - ▶ *Sample request:*
  - ▶ *Sample result:*

`getTags`

- Gets the complete list of tags for the current user.

  - ▶ *Sample request:*
  - ▶ *Sample result:*

`clearUnusedTags`

- Clears all the unused tags in the notes for the current user.

  - ▶ *Sample request:*
  - ▶ *Sample result:*

`replaceTags`

- Replace tags in notes by note ID.

  - ▶ *Sample request:*
  - ▶ *Sample result:*

`replaceTagsInAllNotes`

- Replace tags in all the notes for the current user.

  - ▶ *Sample request:*
  - ▶ *Sample result:*

`findNotes`

`notesInfo`

- Returns a list of objects containing for each note ID the note fields, tags, note type and the cards belonging to the note.

  ▶ *Sample request:*
  ▶ *Sample result:*

`deleteNotes`

- Deletes notes with the given ids. If a note has several cards associated with it, all associated cards will be deleted.

  ▶ *Sample request:*
  ▶ *Sample result:*

`removeEmptyNotes`

- Removes all the empty notes for the current user.

  ▶ *Sample request:*
  ▶ *Sample result:*

Statistic Actions

`getNumCardsReviewedToday`

- Gets the count of cards that have been reviewed in the current day (with day start time as configured by user in anki)

  ▶ *Sample request:*
  ▶ *Sample result:*

`getNumCardsReviewedByDay`

- Gets the number of cards reviewed as a list of pairs of `(dateString, number)`

  ▶ *Sample request:*
  ▶ *Sample result:*

`getCollectionStatsHTML`

- Gets the collection statistics report

- Requests all card reviews for a specified deck after a certain time. `startID` is the latest unix time not included in the result. Returns a list of 9-tuples `(reviewTime, cardID, usn, buttonPressed, newInterval, previousInterval, newFactor, reviewDuration, reviewType)`

  ▶ *Sample request:*
  ▶ *Sample result:*

`getReviewsOfCards`

- Requests all card reviews for each card ID. Returns a dictionary mapping each card ID to a list of dictionaries of the format:

  ```
  {
      "id": reviewTime,
      "usn": usn,
      "ease": buttonPressed,
      "ivl": newInterval,
      "lastIvl": previousInterval,
      "factor": newFactor,
      "time": reviewDuration,
      "type": reviewType,
  }
  ```

  The reason why these key values are used instead of the more descriptive counterparts is because these are the exact key values used in Anki's database.

  ▶ *Sample request:*
  ▶ *Sample result:*

`getLatestReviewID`

- Returns the unix time of the latest review for the given deck. 0 if no review has ever been made for the deck.

  ▶ *Sample request:*
  ▶ *Sample result:*

`insertReviews`

- Inserts the given reviews into the database. Required format: list of 9-tuples `(reviewTime, cardID, usn, buttonPressed, newInterval,`