APIs Short User Guide (v.2)

Date: April 5th, 2020
Edited by: Teodoro Laino
Summary

1. INTRODUCTION
   1.1. Fair use policy

2. API calls
   2.1 Basic Information
      2.1.1 Creation and Recovery of an API key
      2.1.2 Common data structure of API calls
   2.2. First, create a project
      2.2.1. API response code
      2.2.2. Example with curl
   2.3. How to create a new prediction
      2.3.1. API response code
      2.3.2. API Response code (in case of errors)
      2.3.3. Example with curl
   2.4. Recover prediction attempts using the prediction ID
      2.4.1. API response code
      2.4.2. Example with curl
   2.5. List all stored projects
      2.5.1. API response code
      2.5.2. Example with curl
   2.6. List all attempts in a project
      2.6.1. API response code
      2.6.2. Example with curl
   2.7. Generate more outcomes
      2.7.1. API response code
      2.7.2. Example with curl
   2.8. Retrosynthesis: Automatic or Interactive?
   2.9. Automatic Retrosynthesis: task submission
      2.9.1. API response code
      2.9.2. Example with curl
   2.10. Automatic Retrosynthesis: task retrieval
      2.10.1. API response code
      2.10.2. Example with curl
2.11. Interactive Retrosynthesis: task submission
   2.11.1. API response code
   2.11.2. Example with curl
2.12. Interactive Retrosynthesis: expansion task
   2.12.1. API response code
   2.12.2. Example with curl
2.13. Interactive Retrosynthesis: retrieve expansions
   2.13.1. API response code
   2.13.2. Example with curl
2.14. Interactive Retrosynthesis: select an expansion
   2.14.1. API response code
   2.14.2. Example with curl
2.15. Retrosynthesis: retrieve queue status
   2.15.1. API response code
   2.15.2. Example with curl
2.16. Retrosynthesis: retrieve PDF
   2.16.1. API response code
   2.16.2. Example with curl
2.17. Paragraph to Actions
   2.17.1. API response code
   2.17.2. Example with curl

Changes to V1: added description of functionalities 2.7 to 2.14 (more outcomes, automatic retrosynthesis and interactive retrosynthesis)
1. INTRODUCTION

This is a short user guide to allow the usage of the IBM® RXN for Chemistry services in third party applications through APIs calls.

The entire system is based on an authentication mechanism via APIs keys. Moreover, any actions performed through APIs will be appearing as well in the user dashboard when accessing the service through the web-services.

Please, note that the access to the portal through APIs is done under best effort and subject to changes without prior notice. APIs are provided on an "as is" basis, without warranty of any kind.

1.1. Fair use policy

In order to guarantee a fair usage of the free service to anybody on Earth, the calls made via the API interface will be limited to max. 6 requests per minute with a minimum time of 2 seconds between two consecutive calls. Please, note that the system orchestrating the API calls ignores if a call is successfully executed or not. For example, let’s consider the following scenario:

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
</tr>
<tr>
<td>100ms</td>
<td>300ms</td>
<td>500ms</td>
<td>700ms</td>
<td>900ms</td>
<td>2200ms</td>
<td></td>
</tr>
</tbody>
</table>

The call (1) will be successfully executed. From call (2) to call (6) the system will return an error message (503 Service Unavailable) which means that the server is currently unable to handle the request due to a usage policy (too short time between consecutive calls). The implication is that this is a temporary condition which will be alleviated at the beginning of the next minute.

Call (7) is respecting the 2 seconds minimum time but is actually the 7th request done within the same minute and for this reason will return the same error (503). All the counters will be reset at the start of the subsequent clock minute.

If interested in high-throughput usage of these APIs please get in touch with the IBM® RXN for Chemistry team (https://rxn.res.ibm.com) for dedicated cloud instances or on premise installation.
2. API calls

2.1 Basic Information
In this section we will review basic information on how to generate an API key, the structure of the API calls and responses.

Here, we will document the essential API calls to run basic functionalities (reaction prediction/retrosynthesis/etc.). A more extensive list of API calls is available online: https://rxn.res.ibm.com/rxn/api/swagger-ui.html, with limited documentation.

2.1.1 Creation and Recovery of an API key
To use the IBM® RXN for Chemistry services you have first to generate an API key.

To generate the token, log-in into your personal account page of RXN and go on your profile page (top-right corner of the main dashboard). Below your personal information, you will find an API key box. Click on the “GENERATE” button and it will create your own token (starting with apk–…) to use in the subsequent API calls:

Unlike the authentication mechanism for accessing the services through a browser, this API key does not expire (there is no time session limit) and must not be shared with other people to preserve your own privacy.
2.1.2 Common data structure of API calls

All API calls need to have the base URL of the services: *BaseUrl: rxn.res.ibm.com*

Moreover, the header of all calls must contain:

```
Content-Type: application/json
Authorization: apk-...
```

The structure of the answer to the request is:

```
{
  Payload: {},
  "metadata": {
    "uiMessages": {
      "errors": [],
      "infos": [],
      "warnings": []
    }
  }
}
```

Payload contains the real answer to the service request and the metadata section contains errors, info or warnings (if any).
2.2. First, create a project

To run prediction, similarly to the web-interface, you have first to create a project folder where your prediction will be stored.

URL: BaseUrl+/rxn/api/api/v1/projects
Method: POST
Body
{
  "name": "Test API",
  "invitations": [{
    "email": "email@email.nop"
  }]
}

Mandatory fields:

- name
- invitations (can be empty, but must be present)
2.2.1. API response code

The API response code contains several information among which, the “id” is the most relevant to proceed with the prediction capabilities.
2.2.2. Example with curl

```
```

Output:

```
{"payload":{"id":"5c532f56d6cb760019ea34NV","createdOn":1548955478522,"createdBy":"ba7a9ee4-6de6-44bc-b0b0-6eb17dc63cNV","modifiedOn":1548955478522,"modifiedBy":"ba7a9ee4-6de6-44bc-b0b0-6eb17dc63cNV","name":"Test API","description":null,"attempts":[]},"visibility":null,"computedFields":null,"embed":null,"metadata":null,"metadata":{"uiMessages":{"errors":[],"infos":[]},"warnings":[]}}
```
2.3. How to create a new prediction

To create a new prediction is it necessary to know the project folder ID (see 2.2. First, create a project) where the result of the prediction will be stored. As seen in the previous section the ID is a parameter returned by the API response code.

In our example, for the project “Test API” we obtained a project ID of “id”:"5c2c95fed6cb7600019e6f17". Given a smile, representing a single or multiple number of input molecules (such as: C(O)1=C(O)C=C(C2=C(O)C=C([H])C=[H])C=C2(C([H])C=C([H])C=C2(C([H])C=C2=C1O.[H].[H].[H])), the API call for the prediction is constructed as:

```
URL: BaseUrl + /rxn/api/api/v1/predictions/pr?projectId=...
Method: POST
Body
{
    "reactants":"C(O)1=C(O)C=C(C2=C(O)C=C([H])C=[H])C=C2(C([H])C=C([H])C=C2(C([H])C=C2=C1O.[H].[H].[H])"
}
```

Mandatory fields:

- **projectId** = the project ID returned when the project folder is created
- **reactants** = the smile representing the set of starting molecules

2.3.1. API response code

As for the project ID, the response code of a prediction contains the prediction ID, which is needed to recover later the result of the prediction.

```
{"payload": {
    "id": "5c2cab97d6cb7600019e6f21",
    "createdOn": 1546431383554,
    "createdBy": "99f7928e-5992-484e-a0dc-9638a788ab47",
    "modifiedOn": 1546431383554,
    "modifiedBy": "99f7928e-5992-484e-a0dc-9638a788ab47",
    "request": {
        "reactants": "C(O)1=C(O)C=C(C2=C(O)C=C([H])C=[H])C=C2(C([H])C=C([H])C=C2(C([H])C=C2=C1O.[H].[H]. [H])"
    }
}
```
2.3.2. API Response code (in case of errors)

In case of malformed or erroneous SMILES, or in case of prediction errors, the API response will try to do its best to provide a feedback on why the prediction request failed.

An example of a failure:

```json
{
    "payload": null,
    "metadata": {
        "uiMessages": {
            "errors": [
                {
                    "code": null,
                    "message": "ExplicitValenceError, RDKit ERROR: [12:39:34] Explicit valence for atom # 0 H, 3, is greater than permitted - reactants: (H)(C)(C)C.C.CC, 400, RDKitError",
                    "type": "ERROR",
                    "fieldId": null,
                    "target": "TOAST"
                }
            ],
            "infos": [],
            "warnings": []
        }
    }
}
```
2.3.3. Example with curl

curl --data '{ "reactants" :

Output:

{"payload":{"id":"5c532fa7d6cb7600019ea345","createdOn":1548955559319,"createdBy":"ba7a9ee4-6de6-44bc-b0b0-6eb17dc63ca8","modifiedOn":1548955559319,"modifiedBy":"ba7a9ee4-6de6-44bc-b0b0-6eb17dc63ca8", "request":{"reactants":"C(O)1=C(O)C=C(C2=C(O)C=C(\[H\])C(\[H\])=C2C2=C(\[H\])C(\[H\])C=C2)C=C1O.\[H\].\[H\].\[H\], "reagents":[],"mol":""},"taskId":"7604bfc3-de1c-4569-b991-73efb1c7d738","mol":'', "type":"pr", "status":"NEW", "attempts":null, "computedFields":{},"embed":{},"metadata":{},"metadata":{"uiMessages":{"errors":[],"infos":[]}}}}
2.4. Recover prediction attempts using the prediction ID

Using the prediction ID returned in the API prediction response code, one can recover the attempt outcome generated in the corresponding prediction:

URL: BaseUrl + /rxn/api/api/v1/predictions/<PREDICTIONID>
Method: GET

Mandatory fields:

PREDICTIONID (es: 5c2cab97d6cb7600019e6f21)

2.4.1. API response code

La risposta a questa richiesta contiene oltre che lo smiles i dati riguardanti l’accuratezza della risposta AttentionWeights (SVG in Base64) e le immagini delle molecole (SVG in Base64)

```json
{
   "payload": {
      "id": "5c2cab97d6cb7600019e6f21",
      "createdOn": 1546431383554,
      "createdBy": "99f7928e-5992-484e-a0dc-9638a788ab47",
      "modifiedOn": 1546431395414,
      "modifiedBy": "system",
      "request": {
         "reactants": "C(O)1=C(O)C=C(C2=C(O)C=C([H])C([H])=C2C2=C([H])C=C([H])C=C2)C=C1O.[H].[H].[H].",
         "reagents": [],
         "mol": ""
      },
      "projectId": "5c2c95fed6cb7600019e6f17",
      "taskId": "e8fc2e28-a11d-429a-9821-3e409ef739a1",
      "mol": "",
      "type": "pr",
      "status": "SUCCESS",
      "attempts": [
         {
            "id": "5c2caba3d6cb7600019e6f22",
            "createdOn": 1546431395402,
```
2.4.2. Example with curl

curl --header "Content-Type: application/json" --header "Authorization: apk-have-here-your-own-key" -X GET
https://rxn.res.ibm.com/rxn/api/api/v1/predictions/5c532fa7d6cb7600019ea345

Output:

{"payload":{"id":"5c5aaf22d6cb7600019ea809","createdOn":1549446946367,"createdBy":"ba7a9ee4-6de6-44bc-b0b0-6ebe17b63ca8","modifiedOn":1549446950420,"modifiedBy":"system","request":{"reactants":"C(O)\text{\text{1}}=C(O)C=C2=C(\text{O})C=C([\text{H}])C([\text{H}])=C2C2C(\text{[H]}=C([\text{H}])C=C2C1O.[\text{H}].[\text{H}].[\text{H}],"reagents":null,"mol":null,"projectId":"5c532f56d6cb7600019ea342","taskId":"af0e8a5-c2a9-4819-9e41-ad7e40421090","mol":null,"type":"pr","status":"SUCCESS","attempts":null,"projectId":"5c532f56d6cb7600019ea342","createdOn":1549446950408,"createdBy":"system","modifiedOn":1549446950583,"modifiedBy":null,"requestId":null,"smiles":"Oc1cc(-c2c(O)cccccc2-c2ccccc2)c(O)c10.[\text{H}].[\text{H}].[\text{H}]>>Oc1cc(-c2c(O)cccccc2-c2ccccc2)c(O)c10","attentionWeights":null,"confidence":0.13361498713493347,"message":"ok","reactionImage":null,"predictionId":null,"feedbacks":null,"firstAttempt":true,"predictTime":null,"totalTime":null,"computedFields":null,"embed":null,"computedFields":null,"embed":null,"metadata":null,"errors":null,"infos":null,"warnings":null}}
2.5. List all stored projects

URL: BaseUrl +/rxn/api/v1/projects/?page=0&size=30
Method: GET

2.5.1. API response code

Returns the list of projects belonging to API key owner.

```json
{
    "payload": {
        "content": [{
            "id": "5c2c95fed6cb7600019e6f17",
            "createdOn": 1546425854026,
            "createdBy": "99f7928e-5992-484e-a0dc-9638a788ab47",
            "modifiedOn": 1546425854026,
            "modifiedBy": "99f7928e-5992-484e-a0dc-9638a788ab47",
            "name": "Test API",
            "description": "null",
            "attempts": [],
            "visibility": "null",
            "computedFields": {},
            "embed": {},
            "metadata": {}
        },
        {
            "id": "5c2c95fed6cb7600019e6f18",
            "createdOn": 1546425854026,
            "createdBy": "99f7928e-5992-484e-a0dc-9638a788ab47",
            "modifiedOn": 1546425854026,
            "modifiedBy": "99f7928e-5992-484e-a0dc-9638a788ab47",
            "name": "Test API 2",
            "description": "null",
            "attempts": [],
            "visibility": "null",
            "computedFields": {},
            "embed": {},
            "metadata": {}
        },
        ...
    ],
    "totalPages": 1,
    "last": true,
    "totalElements": 7,
    "numberOfElements": 7,
    "sort": "null",
    "first": true,
    "size": 20
}
```
The API response code contains several information among which, the “ids” are the most relevant to proceed with the prediction capabilities inside those project containers.

2.5.2. Example with curl

```

Output:

{"payload":{"content":[{"id":"5b838235a5b9db0001ee1b4b","createdOn":1535345205378,"createdBy":"ba7a9ee4-6de6-44bc-b0b0-6eb17dc63ca8","modifiedOn":1535345205378,"modifiedBy":"ba7a9ee4-6de6-44bc-b0b0-6eb17dc63ca8","name":"test","description":null,"attempts":[]},"visibility":null,"computedFields":{},"embedded":{},"metadata":{},"id":"5b841f72d633050001f190a2","createdOn":1535385458169,"createdBy":"ba7a9ee4-6de6-44bc-b0b0-6eb17dc63ca8","modifiedOn":1535385458169,"modifiedBy":"ba7a9ee4-6de6-44bc-b0b0-6eb17dc63ca8","name":"test on Monday","description":null,"attempts":[]},"visibility":null,"computedFields":{},"embedded":{},"metadata":{},"id":"5bc7344d78d6520001ed4aa1","createdOn":1539781709804,"createdBy":"ba7a9ee4-6de6-44bc-b0b0-6eb17dc63ca8","modifiedOn":1539781709804,"modifiedBy":"ba7a9ee4-6de6-44bc-b0b0-6eb17dc63ca8","name":"test 2","description":null,"attempts":[]},"visibility":null,"computedFields":{},"embedded":{},"metadata":{},"id":"5be4204478d6520001ed4a1","createdOn":1541677124178,"createdBy":"ba7a9ee4-6de6-44bc-b0b0-6eb17dc63ca8","modifiedOn":1541677124178,"modifiedBy":"ba7a9ee4-6de6-44bc-b0b0-6eb17dc63ca8","name":"press day","description":null,"attempts":[]},"visibility":null,"computedFields":{},"embedded":{},"metadata":{},"id":"5bed994a78d6520001ed6dfb","createdOn":1542297930532,"createdBy":"ba7a9ee4-6de6-44bc-b0b0-6eb17dc63ca8","modifiedOn":1542297930532,"modifiedBy":"ba7a9ee4-6de6-44bc-b0b0-6eb17dc63ca8","name":"webinar18","description":null,"attempts":[]},"visibility":null,"computedFields":{},"embedded":{},"metadata":{},"id":"5be24775d6520001ed4a1","createdOn":1542297930532,"createdBy":"ba7a9ee4-6de6-44bc-b0b0-6eb17dc63ca8","modifiedOn":1542297930532,"modifiedBy":"ba7a9ee4-6de6-44bc-b0b0-6eb17dc63ca8"}}
```
Which is equivalent to the view one can get using the browser interface:
2.6. List all attempts in a project

**URL:** BaseUrl
+/<rxn/api/api/v1/projects/<PROJECTID>/attempts?raw=\&page=0&size=8 &sort=createdOn,DESC
  **Method:** GET

Mandatory fields:

- **PROJECTID** (es: 5c2c95fed6cb7600019e6f17)

Optional fields:

- **raw** (must be always set equal to: {})
- **page** (ie. 0, 1, 2 )
- **size** (ie. 8 elements per page)
- **sort**: (createdOn, DESC (ASC|DESC ))

In case these fields (defining the paging of the results) are not defined, the entire list of attempts will be retrieved (it may take a long time!).

2.6.1. API response code

Returns the list of attempts belonging to the given project ID, per page and sorted (according requested criteria).

```json
{
  "payload": {
    "content": [
      {
        "id": "5b894010b72b87000121f186",
        "createdOn": 1535721488737,
        "createdBy": "system",
        "modifiedOn": 1535721488817,
        "modifiedBy": "system",
        "name": "testfb_20180831_13:18:08.737",
        "projectId": "5b759c5e300be10001fc9dc6",
      }
```
"smiles": "O.OCC1CC2(CCC3(CCC3)C2)CC1O>>O=CC1CC2(CCC3(CCC3)C2)CC1O",
"attentionWeights": "<?xml version='1.0' encoding='utf-8' standalone='no'?>
<svg version='1.1' baseProfile='full'
 xmlns='http://www.w3.org/2000/svg'
 xmlns:rdkit='http://www.rdkit.org/xml'
 stroke-linecap:butt;stroke-linejoin:miter;stroke-opacity:1
</svg>"
"confidence": 0.5353702306747437,
"message": "ok",
"reactionImage": "<?xml version='1.0' encoding='iso-8859-1'?>
<svg version='1.1' baseProfile='full'
 xmlns='http://www.w3.org/2000/svg'
 xmlns:rdkit='http://www.rdkit.org/xml'
 stroke-linecap:butt;stroke-linejoin:miter;stroke-opacity:1
</svg>"
"predictionId": "5b89400cb72b87000121f185",
"feedbacks": null,
"firstAttempt": true,
"predictTime": null,
"totalTime": null,
"computedFields": {},
"embed": {},
"metadata": {}
},
{
"id": "5b7708d4ec909700010e48a9",
"createdOn": 1534527700063,
"createdBy": "system",
"modifiedOn": 1534527700101,
"modifiedBy": "system",
"name": "testfb_20180817_17:41:40.062",
"projectId": "5b759c5e300be10001fc9dc6",
"smiles": "COC(=O)C1=CC2C=CC1CC(C=O)CC1>>COC(=O)C1=CC2C=CC1CC2",
"attentionWeights": "<?xml version='1.0' encoding='utf-8' standalone='no'?>
<svg version='1.1' baseProfile='full'
 xmlns='http://www.w3.org/2000/svg'
 xmlns:rdkit='http://www.rdkit.org/xml'
 stroke-linecap:butt;stroke-linejoin:miter;stroke-opacity:1
</svg>"
"confidence": 0.08226326107978821,
"message": "ok",
"reactionImage": "<?xml version='1.0' encoding='iso-8859-1'?>
<svg version='1.1' baseProfile='full'
 xmlns='http://www.w3.org/2000/svg'
 xmlns:rdkit='http://www.rdkit.org/xml'
 stroke-linecap:butt;stroke-linejoin:miter;stroke-opacity:1
</svg>"
"predictionId": "5b7708cdec909700010e48a8",
"feedbacks": null,
"firstAttempt": true,
"predictTime": null,
2.6.2. Example with curl

```
```

Output:

```
{"payload":{"content":{"id":"5c5aaf26d6cb7600019ea80a","createdOn":1549446950487,"createdBy":"system","modifiedOn":1549446950583,"modifiedBy":"system","name":"Test API_20190206_09:55:50.407","projectId":"5c532f56d6cb7600019ea342","smiles":"Oc1cc(-c2c(O)cccc2-c2cccc2)cc(O)c10.[H].[H].[H].>>Oc1cc(-c2c(O)cccc2-c2cccc2)cc(O)c10","attentionWeights":null,"confidence":0.13361498713493347,"message":"ok","reactionImage":null,"totalTime":null,"computedFields":{},"embed":{},"metadata":{}}}
```
Which is equivalent to the view you can have using the browser and navigating in the “API Test” project folder:

![Image of the web interface showing attempts and reaction collection for test reactions API_20190206_09:55:50.407 and API_20190131_17:26:05.398]
2.7. Generate more outcomes

URL: BaseUrl
+/<rxn/api/api/v1/predictions/prb?predictionId=<PREDICTIONID>&projectId=<PROJECTID>
Method: POST
Body
{
  "reactants":"C(O)1=C(O)C=C(C2=C(O)C=C([H])C([H])=C2C2=C([H])C=C([H])C =C2)C=C1[O].[H].[H].[H]
}

Mandatory fields:

- **PROJECTID** = the project ID returned when the project folder is created
- **reactants** = the smile representing the set of starting molecules
- **PREDICTIONID** = the prediction ID of the reaction for which one wishes to generate more outcomes (es: 5c2cab97d6cb7600019e6f21)

2.7.1. API response code

As for the prediction ID, the response code of a generate more outcome contains the prediction ID, which is needed to recover later the result of the prediction (see 2.4. Recover prediction attempts using the prediction ID).

```json
{"payload":
{
  "id":"5c2cab97d6cb7600019e6f21",
  "metadata":{},
  "embed":{},
  "computedFields":{},
  "createdOn":1584823120898,
  "createdBy":"ba7a9ee4-6de6-44bc-b0b06eb17dcsada8",
  "modifiedOn":1584813889819,
  "modifiedBy":"ba7a9ee4-6de6-44bc-b0b06eb17dcsada8",
  "request":{"reactants":"C1=CC=CC1[S+]|(C1=CC=CC1)C1=CC=CC=C1.O",
"reagents":[],"mol":""},
```
2.7.2. Example with curl

curl --header "Content-Type: application/json" --header "Authorization: apk-have-here-your-own-key" -X POST
'https://rxn.res.ibm.com/rxn/api/api/v1/predictions/prb?projectId=5c532fa7d6cb7600019ea345&predictionId=5e76565c48260b00010598d0'
--data '{ "reactants": "C(O)1=C(O)C=C(C2=C(O)C=C([H])C([H])=C2C2=C([H])C=C([H])C=C2)C=C1O.[H].[H].[H].mol":"\n" }

Output:

{"payload":{"id":"5e76565c48260b00010598d0","metadata":{},"embed":{},"computedFields":{}},"createdOn":1584813660898,"createdBy":"ba7a9ee4-6de6-44bc-b0b0-6eb17dc63ca8","modifiedOn":1584813889819,"modifiedBy":"ba7a9ee4-6de6-44bc-b0b0-6eb17dc63ca8","request":{"reactants":"C(O)1=C(O)C=C(C2=C(O)C=C([H])C([H])=C2C2=C([H])C=C([H])C=C2)C=C1O.[H].[H].[H].mol":"\n"},"projectId":"5c532fa7d6cb7600019ea345","taskId":"514f45e6-d5ef-4bd6-b025-0162e4f11e17","mol":"\n","type":"prb","status":"PREDICTION_READY","errorMessage":null,"attempts":null,"metadata":{"uiMessages":{"errors":[]},"infos":[]},"warnings":[]},"extendedPagination":{}}

Using the same prediction ID of the original reaction prediction you can visualize all the outcomes generated by this API call.
2.8. Retrosynthesis: Automatic or Interactive?

When should you choose an interactive VS an automatic?

The algorithm of the automatic retrosynthesis uses holistic (inspired by the Bayesian probability) to determine the optimal branch of the tree to expand in order to arrive in the shortest number of steps to commercially available materials. To optimize speed and performance, we never construct an entire tree of retrosynthetic options but focus exclusively on those disconnections that have the greatest chance to bring the design to completion in the smallest number of steps. We designed the retrosynthetic tool in this way to provide an interactive use and a better experience for users on the IBM RXN for Chemistry platform.

The holistic is of course perfectible and will undergo constant improvements in the future. For this reason, whenever your retrosynthetic task does not produce anything meaningful (i.e., the target is disconnected into itself), we recommend trying the interactive mode, in which you design the retrosynthetic tree choosing the steps from a list of options recommended by the AI algorithm.

The first more (automatic) is fully human-independent. But if you prefer to have control and be only guided among different possible disconnection then revert to the interactive mode, where the AI will only be your assistant in the retrosynthesis design. AI recommends and you choose!

The API call is identical. It is possible to switch between an automatic or interactive retrosynthesis by specifying accordingly the value of the API key: "isInteractive".

For automatic retrosynthesis, there is little else to do after the submission of the API call rather than retrieve the results (see 2.10. Automatic Retrosynthesis: task retrieval).

For interactive retrosynthesis, you need to construct the retrosynthetic tree using your domain expertise. This consists in calling, retrieving, choosing and setting the optimal sequences to complete your retrosynthetic design. Next sections will detail the operations needed to construct your interactive retrosynthesis if you decide to compile it with the use of APIs. Of course, we recommend for this task to use the online graphical interface at https://rxn.res.ibm.com.
2.9. Automatic Retrosynthesis: task submission

URL: BaseUrl +/rxn/api/api/v1/retrosynthesis/rs?projectId=<PROJECTID>

Method: POST
Body
{
  "isInteractive": false,  // this key decides if you want to run automatic or interactive
  "parameters": {
    "availability_pricing_threshold": 0,
    "available_smiles": "string" or null,
    "exclude_smiles": "string or null",
    "exclude_substructures": "string" or null,
    "exclude_target_molecule": true,
    "fap": 0,
    "max_steps": 0,
    "nbeams": 0,
    "pruning_steps": 0
  },
  "product": "SMILE"
}

Mandatory fields:

- **isInteractive** = false for automatic retrosynthesis, true for interactive retrosynthesis
- **availability_pricing_threshold** = Maximum price in USD per mg/ml of commercially available compounds that will be considered available precursors for the retrosynthesis
- **available_smiles** = Smiles of molecules available as precursors
- **exclude_smiles** = Smiles of molecules to exclude from set of precursors
- **exclude_substructures** = Smiles of substructures to exclude from precursors
- **exclude_target_molecule** = true/false to exclude product (when commercially available)
- **fap** = Forward Acceptance Probability. Every retrosynthetic step is evaluated with the forward prediction model. The step is retrained if the forward confidence is greater than FAP.
- **max_steps** = maximum number of retrosynthetic steps
- **nbeams** = maximum number of beams exploring the hypertree
- **pruning_steps** = number of interval steps to prune the explored hypertree
- **product** = SMILE of the molecule to retrosynthesise
Depending on the value of the parameters, the retrosynthesis can be a computationally intensive task.

### 2.9.1. API response code

Returns the list of information related to the submitted retrosynthesis. A WAITING status is also returned and indicates that the retrosynthesis is in the queue.

```
BaseResponse<RetrosynthesisDTO> ✓ {
  metadata
  payload
    RetrosynthesisDTO ✓ {
      computedFields
      createdAt
      createdOn
      embed
      errorMessage
      id
      isInteractive
      legend
      metaData
      metadata
      model
      modifiedBy
      modifiedOn
      molecules
      projectID
      sequences
      siblings
      status
      task
        RetrosynthesisResultTask ✓ {
        ...
      }
      wizardParameters
        RetrosynthesisResultWizardParameters ✓ {
        ...
      }
    }
  }
```

### 2.9.2. Example with curl

```
curl --header "Content-Type: application/json" --header "Authorization: apk-have-here-your-own-key" -X POST --data '{"isInteractive":false,"parameters":{
  "availability_pricing_threshold": 1,
  "availability_smiles": null,
  "exclude_smiles": null,
  "exclude_substructures": null,
  "exclude_target_molecule": true,
  "fap": 0.60,
  "max_steps": 3,
  "nbeams": 10,
  "pruning_steps": 2,
  "product": "C(Cl)1CCCCC1"}
}'}
https://rxn.res.ibm.com/rxn/api/api/v1/retrosynthesis/rs?projectId=5c532f56d6cb7600019ea342`
```
The evaluation of the retrosynthesis is an asynchronous process. As it may take few minutes, once it is launched, you can check with additional APIs (see 2.10. Automatic Retrosynthesis: task retrieval) the status of the retrosynthesis and eventually retrieve the results.
2.10. Automatic Retrosynthesis: task retrieval

URL: BaseUrl +/rxn/api/api/v1/retrosynthesis/\<RETROSYNTHESISID>
Method: GET

Mandatory fields:

- \texttt{RETROSYNTHESIS=} this is the ID returned by the submission API of the retrosynthesis (es: 5e777d0b48260b134105bf2b)
2.10.1. API response code

When the retrosynthetic task is completed (task.status = SUCCESS) the API response will contain all possible sequences (build in hierarchical tree format) as well as the list of all possible retrosynthetic single step for each molecule (siblings).
2.10.2. Example with curl

```
curl --header "Content-Type: application/json" --header "Authorization: apk-have-here-your-own-key" -X GET
'https://rxn.res.ibm.com/rxn/api/api/v1/retrosynthesis/5e777d0b48260b134105bf2b'
```

Output:

```
{payload...
"sequences":[{"id":"5e788ae548260b000105ecf4","metadata":{"hcomplexity":[0.9284365780213366,0.9332128269631699],"status":"DONE"} ....
"tree":["id":"5e788ae548260b000105ecf7","metadata":{},"computedFields":{},"createdOn":1584958181564,"createdBy":"system","modifiedOn":1584958181564,"modifiedById":"system","moleculeId":"5d0a7756e5d96c000118119db","retrosynthesizedId":"5e788abd48260b000105ecf2","sequenceId":"5e788ae548260b000105ecf4","projectId":"5e762b9648260b0001059324","smiles":"ClC1CCCCC1","confidence":0.999,"rclass":"Hydroxy to chloro","hasFeedback":false,"feedback":null,"children":[]},"computedFields":{},"createdOn":1584958181539,"createdById":"system","modifiedOn":1584958181539,"modifiedById":"system","moleculeId":"5b7b46f1ec909700010e824b","retrosynthesizedId":"5e788abd48260b000105ecf2","sequenceId":"5e788ae548260b000105ecf4","projectId":"5e762b9648260b0001059324","smiles":"ClP(Cl)(Cl)(Cl)Cl","confidence":0.0,"rclass":"Undefined","hasFeedback":false,"feedback":null,"children":[]},"computedFields":{},"createdOn":1584958181556,"createdById":"system","modifiedOn":1584958181556,"modifiedById":"system","moleculeId":"5b7b3cb0ec909700010e80aa","retrosynthesizedId":"5e788abd48260b000105ecf2","sequenceId":"5e788ae548260b000105ecf4","projectId":"5e762b9648260b0001059324","smiles":"OC1CCCC1","confidence":0.0,"rclass":"Undefined","hasFeedback":false,"feedback":null,"children":[]}
```

The amount of information retrieved by this API call is quite large. The different retrosynthetic sequences are organized in a tree. You can parse that information for each sequence to generate the corresponding retrosynthetic tree.
2.11. Interactive Retrosynthesis: task submission

URL: BaseUrl +/rxn/api/api/v1/retrosynthesis/rs?projectId=<PROJECTID>

Method: POST
Body
{
   "isInteractive": true, // this key decides If you want to run and automatic or interactive
   "parameters": {
      "availability_pricing_threshold": 0,
   },
   "product": "SMILE"
}

Mandatory fields:

- isInteractive= false for automatic retrosynthesis, true for interactive retrosynthesis
- availability_pricing_threshold= Maximum price in USD per mg/ml of commercially available compounds that will be considered available precursors for the retrosynthesis
- product= SMILE of the molecule to retrosynthesise
- PROJECTID (es: 5c2c95fed6cb7600019e6f17)

The first call is required only to initiate the retrosynthetic task. The expansion and selection of the corresponding tree is achieved with the use of the subsequent APIs.

2.11.1. API response code

Returns the list of information related to the submitted retrosynthesis. A WAITING status is also returned and indicates that the retrosynthesis is in the queue.
2.11.2. Example with curl

```
curl --header "Content-Type: application/json" --header "Authorization: apk-have-here-your-own-key" -X POST --data '{ "isInteractive": true, "parameters": { "availability_pricing_threshold": 1 }, "product": "C(Cl)1CCCCC1" }' https://rxn.res.ibm.com/rxn/api/api/v1/retrosynthesis/rs?projectId=5c532f56d6c87600019ea342'
```

Output:

```
{
  "payload":{"id":"5e899c5948260b000107d777","metadata":{},"embed":{},"computedFields":{},"createdOn":1586076761397,"createdBy":"ba7a9ee4-6de6-44bc-b0b0-6eb17dc63ca8","modifiedOn":1586076761397,"modifiedBy":"ba7a9ee4-6de6-44bc-b0b0-6eb17dc63ca8","projectId":"5e762b9648260b0001059324","status":"DONE","errorMessage":null,"task":null,"molecules":null,"sibling":null,"status":null,"Embed":null}
}
The information about the tree (highlighted in green) will have to be stored and transmitted when selecting and assigning a specific expansion (see 2.11. Interactive Retrosynthesis: task submission). In red is highlighted the RETROSYNTHESISID and in orange the NODEID. These information needs to be intercepted and store for subsequent calls.
The retrosynthesis appears also in the web application, under the user account/project:
2.12. Interactive Retrosynthesis: expansion task

URL: BaseUrl +/rxn/api/v1/retrosynthesis/<RETROSYNTHESISID>/expand-molecule/<NODEID>
Method: GET

Mandatory fields:

- NODEID=the target node to expand (see 2.11. Interactive Retrosynthesis: task submission)
- RETROSYNTHESISID= this is the ID returned by the submission API of the retrosynthesis (es: 5e777d0b48260b134105bf2b) (see 2.11. Interactive Retrosynthesis: task submission)

2.12.1. API response code

```json
{
    "status": "WAITING",
    "task": {
        "counters": {
            "backward_prediction": 0,
            "forward_prediction": 0
        },
        "expected_start": 0,
        "finished_time": 0,
        "metaData": {
            "additionalProp1": {},
            "additionalProp2": {},
            "additionalProp3": {}
        },
        "queue_position": 0,
        "queued_time": 0,
        "retrieval_time": 0,
        "retry": 0,
        "started_time": 0,
        "status": "NEW",
        "task_id": "string"
    }
}
```
2.12.2. Example with curl

```bash
curl --header "Content-Type: application/json" --header "Authorization: apk-have-here-your-own-key" -X GET 'https://rxn.res.ibm.com/rxn/api/api/v1/retrosynthesis/5e899c5948260b000107d777/expand-molecule/5e899c5948260b000107d779'
```

Output:

```json
{"payload":{"task":{"finished_time":0.0,"metaData":null,"started_time":0.0,"status":"WAITING","expected_start":1.586009903895212E9,"queue_position":0,"task_id":"5e899db72f5c865fb216a29c","retrieval_time":0.0,"retry":0,"counters":{"backward_prediction":0,"forward_prediction":0},"molecules":[],"siblings":null,"metaData":null,"status":"WAITING","legend":null},"metaData":null,"status":"WAITING","legend":null},"messages":{"errors":[],"infos":[],"warnings":[],"extendedPagination":null}}
```

This API calls the single-step retrosynthetic engine to provide a single step retrosynthesis (including not only reactants but also reagents) for the node target molecule. This API, as the automatic retrosynthesis is task based. In order to inspect the outcome, it will be necessary to call the relative API to query the status/result of the single-step retrosynthetic engine providing the corresponding task_id (in green).
2.13. Interactive Retrosynthesis: retrieve expansions

URL: BaseUrl +/api/v1/retrosynthesis/< RETROSYNTHESISID>/expand-molecule/result/< TASKID>
Method: GET

Mandatory fields:

- **TASKID**: the ID of the task returned by the task expansion API (see 2.12. Interactive Retrosynthesis: expansion task)
- **RETROSYNTHESISID**: this is the ID returned by the submission API of the retrosynthesis (ex: 5e777d0b48260b134105bf2b)

2.13.1. API response code

The API response is similar to the response of the automatic retrosynthesis.

```
{
    "molecules": [...],
    "sequences": [...],
    "siblings": [...],
    "task": {...}
}
```

2.13.2. Example with curl

```
curl --header "Content-Type: application/json" --header "Authorization: apk-have-here-your-own-key" -X GET
'https://rxn.res.ibm.com/rxn/api/api/v1/retrosynthesis/5e899c5948260b000107d777/expand-molecule/result/5e899db72f5c865fb216a29c'
```

Output:

```
{
    "payload":{"task":{"finished_time":1.586077112449E9,"metaData":{},"queued_time":null,"started_time":1.586077111895E9,"status":"DONE","expected_start":0.
```
In the next example, a molecule is hdrochlorinated: **C1CCCCC1.Cl**.

The molecule undergoes a chlorination reaction: **C1CCCCC1.ClP(Cl)(Cl)(Cl)Cl**.

Another molecule experiences a chlorination: **C1CCCCC1.O=S(Cl)Cl.c1ccncc1**.

A reaction involving an alkenel hydrochlorination: **C1=CCCCC1.Cl**.

A chlorination reaction: **C1CCCCC1.Cl**.

A molecule undergoes a chlorination: **C1CCCCC1.Cl**.

In the following example, a molecule is chlorinated: **C1CCCCC1.O=S(Cl)Cl.c1ccncc1**.

Another alkenel hydrochlorination reaction: **C1=CCCCC1.Cl**.

A molecule experiences a chlorination: **C1CCCCC1.ClP(Cl)(Cl)(Cl)Cl**.

Another molecule undergoes a chlorination: **C1CCCCC1.O=S(Cl)Cl.c1ccncc1**.

A molecule is chlorinated: **C1CCCCC1.Cl**.

A molecule undergoes a chlorination: **C1CCCCC1.O=S(Cl)Cl.c1ccncc1**.

Another molecule experiences a chlorination: **C1CCCCC1.Cl**.

The molecule undergoes a chlorination: **C1CCCCC1.O=S(Cl)Cl.c1ccncc1**.
In red are highlighted few possible reactions that can be assigned (see 2.14. Interactive Retrosynthesis: select an expansion) to the molecule to expand. This information (one among the highlighted ones) needs to be combined with the information provided by the interactive task submission (see 2.11. Interactive Retrosynthesis: task submission) as input for the selection/assignment of a specific expansion (see 2.14. Interactive Retrosynthesis: select an expansion).
2.14. Interactive Retrosynthesis: select an expansion

URL: BaseUrl +/api/v1/retrosynthesis/< RETROSYNTHESISID>/tree/<NODEID>
Method: PATCH

Mandatory fields:

- NODEID: NODEID= the target node to expand (see 2.11. Interactive Retrosynthesis: task submission)

- RETROSYNTHESIS= this is the ID returned by the submission API of the retrosynthesis (es: 5e777d0b48260b134105bf2b)

- { "children": [
    {
      "isExpandable": true,
      "moleculeId": "string",
      "smiles": "string"
    }, ...
] }

2.14.1. API response code

{
  "status": "NEW",
  "task": {
    "counters": { "backward_prediction": 0, "forward_prediction": 0 },
    "expected_start": 0, "finished_time": 0, "metaData": { "additionalProp1": {},
      "additionalProp2": {}, "additionalProp3": {} },
    "queue_position": 0, "queued_time": 0, "retrieval_time": 0, "retry": 0,
    "started_time": 0, "status": "NEW", "task_id": "string"
  }
}
2.14.2. Example with curl

curl --data-binary 
'{"id":"5e899c5948260b000107d777","metadata":{},"embed":{},"computedFields":{},"createdOn":1586076761413,"createdBy":"ba7a9ee4-6de6-44bc-b0b0-6eb17dc63ca8","modifiedOn":1586076761413,"modifiedBy":"ba7a9ee4-6de6-44bc-b0b0-6eb17dc63ca8","moleculeId":"5e78717a48260b000105ecb4","retrosynthesisId" :"5e899c5948260b000107d777","sequenceId":"5e899c5948260b000107d778","projectId":"5e762b9648260b0001059324","smiles":"C(Cl)1CCCCC1","hasFeedback":null,"feedback":null,"confidence":0.999,"confidenceTag":"H","label":null,"rclass":"Hydroxy to chloro","smiles":"ClP(Cl)(Cl)(Cl)Cl.OC1CCCCC1","children":null,"smiles":"ClP(Cl)(Cl)(Cl)Cl","metaData":{"borderColor":"#28a30d"},"moleculeId":"5b7b0d80ec9097000107d775","isExpandable":false},"smiles":"OC1CCCCC1","metaData":{"borderColor":"#28a30d"},"moleculeId":"5b7b3cb0ec909700010800a","isExpandable":false},"metaData":null,"isExpandable":true,"siblingId":null}' 
--header "Content-Type: application/json" 
--header " Authorization: apk-have-here-your-own-key" 
-X PATCH

'https://rxn.res.ibm.com/rxn/api/api/v1/retrosynthesis/5e899c5948260b000107d777/tree/5e899c5948260b000107d779'

Output:

```json
{
"payload":{"id":"5e899c5948260b000107d777","metadata":{},"embed":{},"computedFields":{},"createdOn":1586076761397,"createdBy":"ba7a9ee4-6de6-44bc-b0b0-6eb17dc63ca8","modifiedOn":1586076761397,"modifiedBy":"ba7a9ee4-6de6-44bc-b0b0-6eb17dc63ca8","projectId":null,"status":null,"errorObject":null,"molecules" :{"id":"5e78717a48260b000105ecb4","metadata":{},"embed":{},"computedFields":{},"createdOn":1584951674746,"createdBy":"ba7a9ee4-6de6-44bc-b0b0-6eb17dc63ca8","modifiedOn":1584951674746,"modifiedBy":"ba7a9ee4-6de6-44bc-b0b0-6eb17dc63ca8","sequences":null,"moleculeImage":null},"sequences":null,"moleculeImage":null}
```
The entire process can be iterated until the domain experts decide it to be complete. In this specific example none of the children is expandable. In the event, where one of the children leaves is instead expandable, it is sufficient to iterate the API calls from 2.12 to 2.14, until desired (here above the NODEID is highlighted in magenta).

The retrosynthesis is consequently updated in the web application, under the user account/project:
2.15. Retrosynthesis: retrieve queue status

URL: BaseUrl +/rxn/api/api/v1/retrosynthesis/queue-state
Method: GET

2.15.1. API response code

```
{
  "payload": {
    "itemsInQueue": 2,
    "willBeCompletedOn": null,
    "metadata": {
      "uiMessages": {
        "errors": [],
        "infos": [],
        "warnings": []
      },
      "extendedPagination": {}  
    }
  }
}
```

2.15.2. Example with curl

```
```

Output:

```
{"payload":{"itemsInQueue":1,"willBeCompletedOn":1585893597000},"metadata":{"uiMessages":{"errors":[]},"infos":[]},"warnings":[]},"extendedPagination":[]}
```
2.16. Retrosynthesis: retrieve PDF

URL: BaseUrl +/rxn/api/v1/retrosynthesis/<RETROSYNTHESISID>/sequences/<SEQUENCESID>/download-pdf
Method: GET

Mandatory fields:

- **SEQUENCESID**= this is the ID of the sequence returned by the retrosynthesis API for which one wants to generate the PDF (es.: 5e788ae548260b770105ecf4)
- **RETROSYNTHESISID**= this is the ID returned by the submission API of the retrosynthesis (es: 5e777d0b48260b134105bf2b)

2.16.1. API response code

The answer to the API call contains the PDF (Content-type: application/pdf) in bytes.

2.16.2. Example with curl

```
curl --header "Content-Type: application/json" --header "Authorization: apk-have-here-your-own-key" -X GET -X GET
'https://rxn.res.ibm.com/rxn/api/api/v1/retrosynthesis/5e777d0b48260b134105bf2b/sequences/5e788ae548260b770105ecf4/download-pdf'
```

Output:

```
%PDF-1.4
% ReportLab Generated PDF document http://www.reportlab.com
1 0 obj
<< /F1 2 0 R /F2 4 0 R /F3 6 0 R /F4 7 0 R >>
endobj
2 0 obj
```
The PDF contains a human readable representation of that specific sequence.
2.17. Paragraph to Actions

URL: BaseUrl +/rxn/api/api/v1/retrosynthesis/paragraph-actions
Method: POST

Mandatory fields:

- {"paragraph": "text describing a recipe"}

2.17.1. API response code

```
{
  "payload": {,
  "actionSequence": "",
  "metadata":{"uiMessages":{
  "errors":[],"infos":[],"warnings":[]}},"extendedPagination":{}
}
}
```

2.17.2. Example with curl

```
curl --header "Content-Type: application/json" --header "Authorization: apk-have-here-your-own-key" -X POST --data-binary '{"paragraph":"To a stirred solution of 7-(difluoromethylsulfonyl)-4-fluoro-indan-1-one (110 mg, 0.42 mmol) in methanol (4 mL) was added sodium borohydride (24 mg, 0.62 mmol). The reaction mixture was stirred at ambient temperature for 1 hour. Saturated aqueous NH4Cl solution was added dropwise. The mixture was extracted with EtOAc. The combined organic layers were washed with water and brine, dried and concentrated in vacuo to give 7-(difluoromethylsulfonyl)-4-fluoro-indan-1-ol (100 mg, 90%), which was used in the next step without further purification. LCMS ESI (+) m/z 267 (M+H); ESI (\u2212) m/z 311 (M\u2212H+46)."'}
'https://rxn.res.ibm.com/rxn/api/api/v1/retrosynthesis/paragraph-actions'
```
Makesolution with 7-(difluoromethylsulfonyl)-4-fluoro-indan-1-one (110 mg, 0.42 mmol) and methanol (4 mL); add SLN; add sodium borohydride (24 mg, 0.62 mmol). Stir for 1 hour at ambient temperature. Add saturated aqueous NH₄Cl solution dropwise. Extract with EtOAc. Collect layer organic; wash with water; wash with brine; dry with material unknown; concentrate. Yield 7-(difluoromethylsulfonyl)-4-fluoro-indan-1-ol (100 mg, 90%). Noaction.