ProtoPRED API Documentation

The ProtoPRED prediction platform (https://protopred.protoqsar.com/) will be accessible through an API.

Base URL: https://protopred.protoqsar.com/API/v2/

Note: The API does not support GET requests. All input must be included in the POST request.

Authentication

A preliminary user has been created with the following credentials. However, additional users for ONTOX partners could be defined and added to our database to ensure proper access.

- `"account_token": "1JX3LP"`
- `"account_secret_key": "A8X9641JM"`
- `"account_user": "OOntox"`

Request parameters

Parameter	Type	Required	Description
module	string	Yes	The module name. Only "ProtoPHYSCHEM" and "ProtoADME" are currently supported
models_list	string	Yes	Comma-separated list in the format model_property:model_name
input_type	string	Yes	Either SMILES_TEXT or SMILES_FILE
input_data	string / xlsx / json	Yes	This parameter accepts a SMILES string, an excel, or a JSON, depending on the input type specified
output_type	string	No	Defaults to JSON. Use "XLSX" for excel output

Available modules and models

Models are grouped by property and can be combined in a single request:

e.g.: `"models_list":"model_phys:water_solubility, model_tox:eye_irritation"`

Note:

- Both "model_type" and "model_name" are case-insensitive.
- Spaces will be ignored; for example, `"MODEL_Phys : water_solubility"` is also considered valid.

The following modules and models are available:

Module	Model property	Model name	Full name
ProtoPHYSCHEM	model_phys	melting_point	7.2 Melting point
		boiling_point	7.3 Boiling point
		vapour_pressure	7.5 Vapour pressure
		water_solubility	7.7 Water solubility
		log_kow	7.8 Partition coefficient (log Kow/log P)
		log_d	7.8 Partition coefficient (log D)
		surface_tension	Surface tension
ProtoADME	model_abs	bioavailability20	Bioavailability 20%
		bioavailability30	Bioavailability 30%
		caco-2_permeability	Caco-2 permeability
		p-gp_inhibitor	P-glycoprotein inhibitor
		p-gp_substrate	P-glycoprotein substrate
		skin_permeability	Skin permeability
		human_intestinal_absorption	Human intestinal absorption
	model_met	CYP450_1A2_inhibitor	CYP450 1A2 inhibitor
		CYP450_1A2_substrate	CYP450 1A2 substrate
		CYP450_2C19_inhibitor	CYP450 2C19 inhibitor
		CYP450_2C19_substrate	CYP450 2C19 substrate
		CYP450_2C9_inhibitor	CYP450 2C9 inhibitor
		CYP450_2D6_inhibitor	CYP450 2D6 inhibitor
		CYP450_2D6_substrate	CYP450 2D6 substrate
		CYP450_3A4_inhibitor	CYP450 3A4 inhibitor
		CYP450_3A4_substrate	CYP450 3A4 substrate
	model_dist	blood-brain_barrier	Blood-brain barrier penetration
		plasma-protein_binding	Plasma protein binding
		volume_of_distribution	Volume of distribution
	model_exc	half-life	Half-life
		human_liver_microsomal	Human liver microsomal stability
		OATP1B1	OATP1B1 inhibitor
		OATP1B3	OATP1B3 inhibitor
		BSEP	BSEP inhibitor

Input options

The ProtoPRED API supports multiple ways to submit input data for predictions. You can provide either a single molecule or a batch of molecules using text, files (xlsx/json), or embedded JSON.

Note: All the examples in this document are in python and the script containing them will be provided.

1. Single SMILES as text

- The parameter "input_type" must be set to "SMILES_TEXT".
- The parameter "input_data" must be defined as a string containing a molecule in SMILES codification (e.g., "CCCCC").
- Send the request as a standard form POST.

Example:

```
import requests, json

query = {
        "account_token": "1JX3LP",
        "account_secret_key": "A8X9641JM",
        "account_user": "OOntox",
        "module": "ProtoPHYSCHEM",
        "input_type": "SMILES_TEXT",
        "input_data": "CCCCC",
        "models_list": "model_phys:water_solubility"
}
response = requests.post("https://protopred.protoqsar.com/API/v2/",
data=query)

with open(r"C:\path\to\output.json", "w") as f:
    json.dump(response.json(), f, indent=4, ensure_ascii=False)

print(response.json())
```

2. File upload (EXCEL or JSON)

- The parameter "input_type" must be set to "SMILES_FILE".
- The input file would need to be defined inside the "input_data" parameter, using the request extra parameter named "files".
- Accepted formats:
  - Excel (.xlsx) with at least one column named "SMILES" containing sanitized smiles.
  - JSON (.json) structured as a dictionary, where each entry represents one molecule identified by an ID, and includes at least one field named "SMILES" containing sanitized smiles.
- The metadata fields or columns allowed in the file are: "SMILES", "CAS", "Chemical name", "EC number" and "Structural formula".

JSON file example:

```
{
	"ID_1": {
		"SMILES": "C1=CC(=O)C=CC1=O"
	},
	"ID_2": {
		"SMILES": "CCCCC",
		"CAS": "10-66-0",
		"Chemical name": "Pentane",
		"EC number": "203-692-4",
		"Structural formula": "C5H12"
	}

}
```

```
file_path = r"C:\path\to\JSON_input.json"
```

Excel file example:

	A	B	C	D	E
1	**CAS**	**EC number**	**Structural formula**	**Chemical name**	**SMILES**
2	94-52-0	202-341-2	C7H5N3O2	6-nitro-1H-benzimidazole	O=[N+]([O-])c1ccc2nc[nH]c2c1
3	10-66-0	203-692-4	C5H12	Pentane	CCCCC
4	19052-63-2	203-405-2	C6H4O2	cyclohexa-2,5-diene-1,4-dione	C1=CC(=O)C=CC1=O

```
file_path = r"C:\path\to\EXCEL.xlsx"
```

Script example:

```
import requests, json


query = {
    "account_token": "1JX3LP",
    "account_secret_key": "A8X9641JM",
    "account_user": "OOntox",
    "module": "ProtoPHYSCHEM",
    "input_type": "SMILES_FILE",
    "models_list": "model_phys:water_solubility"
}
files = {"input_data": open(file_path, "rb")}

response = requests.post("https://protopred.protoqsar.com/API/v2/",
data=query, files=files)

with open(r"C:\path\to\output.json", "w") as f:
    json.dump(response.json(), f, indent=4, ensure_ascii=False)

print(response.json())
```

3. Embedded JSON in the request body

This method allows to send the data directly in the request body, rather than as a file upload.

- The parameter "input_type" must be set to "SMILES_FILE".
- The dictionary of molecules must be defined as "input_data".

Example:

```
import requests, json

query = {
    "account_token": "1JX3LP",
    "account_secret_key": "A8X9641JM",
    "account_user": "OOntox",
    "module": "ProtoPHYSCHEM",
    "input_type": "SMILES_FILE",
    "models_list": "model_phys:water_solubility",
    "input_data":
            {
                "ID_1": {
                    "SMILES": "C1=CC(=O)C=CC1=O"
                },
                "ID_2": {
                    "SMILES": "CCCCC",
                    "CAS": "10-66-0",
                    "Chemical name": "Pentane",
                    "EC number": "203-692-4",
                    "Structural formula": "C5H12"
                }
            }
}
```

- The query may be JSON-encoded using json.dumps() and passed through the data parameter.

```
response = requests.post("https://protopred.protoqsar.com/API/v2/",
data= json.dumps(query))
```

- Alternatively, the query can be passed directly using the "json" parameter.

```
response = requests.post("https://protopred.protoqsar.com/API/v2/",
json=query)
```

- Once the request is completed, the response can be saved locally and printed for review.

```
with open(r"C:\path\to\output.json", "w") as f:
    json.dump(response.json(), f, indent=4, ensure_ascii=False)

print(response.json())
```

4. JSON file as complete request

This method allows sending the full request content by loading it from an external .json file.

- The file must include all required fields, such as "module", "input_type", "models_list", and "input_data".
- The file is sent as the body of the request, using binary mode (rb).

JSON file example:

```
{
   "account_token": "1JX3LP",
    "account_secret_key": "A8X9641JM",
    "account_user": "OOntox",
    "module": "ProtoPHYSCHEM",
    "input_type": "SMILES_FILE",
    "models_list": "model_phys: water_solubility",
     "input_data": {
             "ID_1": {
                 "SMILES": "C1=CC(=O)C=CC1=O"
             },
             "ID_2": {
                 "SMILES": "CCCCC",
                 "CAS": "10-66-0",
                 "Chemical name": "Pentane",
                 "EC number": "203-692-4",
                 "Structural formula": "C5H12"
             },
             "ID_3": {
                 "SMILES": "O=[N+]([O-])c1ccc2nc[nH]c2c1",
                 "CAS": "94-52-0",
                 "Chemical name": "6-nitro-1H-benzimidazole",
                 "EC number": "202-341-2",
                 "Structural formula": "C7H5N3O2"
             }
         }
}
```

Script example:

```
import requests, json

query = open(r"C:\path\to\request_body_API.json", "rb")

response = requests.post("https://protopred.protoqsar.com/API/v2/",
data=query)

with open(r"output_option5.json", "w") as f:
    json.dump(response.json(), f, indent=4, ensure_ascii=False)

print(response.json())
```

Output options

Format	Description
JSON (default)	Returns a structured dictionary per model.
XLSX	Use "output_type": "XLSX" to get an excel file.

JSON file example:


```
{
    "Water solubility": [
        {
            "ID": "ID_1",
            "Chemical name": "-",
            "EC number": "-",
            "Structural formula": "-",
            "CAS": "-",
            "SMILES": "C1=CC(=O)C=CC1=O",
            "Experimental value*": "11.1 g/L",
            "Predicted value": "18.3 g/L",
            "Experimental value (model units)*": "-0.99 log mol/L",
            "Predicted value (model units)": "-0.77 log mol/L",
            "Probability": "NaN",
            "Experimental numerical": 11.1,
            "Predicted numerical": 18.3,
            "Experimental numerical (model units)": -0.9885,
            "Predicted numerical (model units)": -0.7717,
            "Applicability domain**": "Inside (T/L/E/R)"
        },
        {
            "ID": "ID_2",
            "Chemical name": "Pentane",
            "EC number": "203-692-4",
            "Structural formula": "C5H12",
            "CAS": "10-66-0",
            "SMILES": "CCCCC",
            "Experimental value*": "0.038 g/L",
            "Predicted value": "0.066 g/L",
            "Experimental value (model units)*": "-3.28 log mol/L",
            "Predicted value (model units)": "-3.04 log mol/L",
            "Probability": "NaN",
            "Experimental numerical": 0.038,
            "Predicted numerical": 0.066,
            "Experimental numerical (model units)": -3.2785,
            "Predicted numerical (model units)": -3.0369,
            "Applicability domain**": "Inside (T/L/E/R)"
        }
    ],
    "Melting point": [
        {
            "ID": "ID_1",
            "Chemical name": "-",
```

```
            "EC number": "-",
            "Structural formula": "-",
            "CAS": "-",
            "SMILES": "C1=CC(=O)C=CC1=O",
            "Experimental value*": "115.9 ºC",
            "Predicted value": "112.9 ºC",
            "Experimental value (model units)*": "2.59 log K",
            "Predicted value (model units)": "2.59 log K",
            "Probability": "NaN",
            "Experimental numerical": 115.9,
            "Predicted numerical": 112.9,
            "Experimental numerical (model units)": 2.589782103291143,
            "Predicted numerical (model units)": 2.58663,
            "Applicability domain**": "Inside (T/L/E/R)"
        },
        {
            "ID": "ID_2",
            "Chemical name": "Pentane",
            "EC number": "203-692-4",
            "Structural formula": "C5H12",
            "CAS": "10-66-0",
            "SMILES": "CCCCC",
            "Experimental value*": "-128.6 ºC",
            "Predicted value": "-128.6 ºC",
            "Experimental value (model units)*": "2.16 log K",
            "Predicted value (model units)": "2.16 log K",
            "Probability": "NaN",
            "Experimental numerical": -128.6,
            "Predicted numerical": -128.6,
            "Experimental numerical (model units)": 2.156700552582017,
            "Predicted numerical (model units)": 2.16014,
            "Applicability domain**": "Inside (T/L/E/R)"
        }

    ]
}
```


Excel file structure:

When the "output_type" parameter is set to "XLSX" (case-insensitive), the API returns a structured excel file (.xlsx) with different sheets:

- The first sheet ("General information") provides general information about the models and the input type, and provides a complete information to understand the results.
- The second sheet ("Summary") provides summarized results.
- The subsequent sheets are named after each model using sentence case (e.g. Water solubility), and contain detailed prediction information.