protmapper Documentation

Release 0.0.28

John A. Bachman, Benjamin M. Gyori

Aug 29, 2023

Contents

1 Protmapper modules reference **1**
- 1.1 Protmapper API . . . 1
- 1.2 UniProt client . . . 5
- 1.3 PhosphoSite client . . . 11
- 1.4 Resource management . . . 13

2 REST API **15**
- 2.1 *map_to_human_ref* . . . 15
- 2.2 *map_sitelist_to_human_ref* . . . 16
- 2.3 Optional arguments . . . 16

Python Module Index **19**

Index **21**

CHAPTER 1

Protmapper modules reference

1.1 Protmapper API

exception protmapper.api.**InvalidSiteException**
Bases: Exception

class protmapper.api.**MappedSite**(*up_id*, *valid*, *orig_res*, *orig_pos*, *error_code=None*, *mapped_id=None*, *mapped_res=None*, *mapped_pos=None*, *description=None*, *gene_name=None*)
Bases: object

Represent details of a site that was mapped.

up_id
The UniProt ID of the protein whose site was mapped.

Type str

error_code
One of several strings indicating an error in retrieving the protein sequence, or None if there was no error. Error codes include 'NO_UNIPROT_ID' if the given gene name could not be converted into a Uniprot ID; 'UNIPROT_HTTP_NOT_FOUND' if the given Uniprot ID resulted in a 404 Not Found error from the Uniprot web service; or 'UNIPROT_HTTP_OTHER' if it was any other type of Uniprot web service error. Any other unexpected errors in getting the sequence are assigned the 'UNIPROT_OTHER' code. If the error code is not None, the *orig_res* and *orig_pos* fields will be set (based on the query arguments) but all other fields will be None.

Type str or None

valid
True if the original site was valid with respect to the given protein, False otherwise. Further, in case of an error (if error_code is not None), it is set to None.

Type bool

orig_res
The original amino acid residue that was mapped.

Type str

orig_pos
The original amino acid position that was mapped.

Type str

mapped_id
The Uniprot ID for the protein containing the mapped site. If *up_id* is the Uniprot ID for the human reference sequence, in most cases this will match; however, exceptions will occur if the site position refers to a site that is unique to a particular isoform.

Type str

mapped_res
The mapped amino acid residue.

Type str

mapped_pos
The mapped amino acid position.

Type str

description
A description of the mapping that was done, comes from a fixed set of codes of types of mapping that were performed.

Type str

gene_name
The standard (HGNC) gene name of the protein that was mapped.

Type str

has_mapping()
Return True if the original site was mapped successfully.

Returns True if a mapping was successfully obtained for the site, False otherwise.

Return type bool

not_invalid()
Return True if the original site is not known to be invalid.

Returns True if the original site is valid or if there is an error code, which implicitly means that the validity of the original site could not be established. False otherwise.

Return type bool

class protmapper.api.**ProtMapper**(*site_map=None*, *use_cache=False*, *cache_path=None*)
Bases: object

Use curated site information to standardize modification sites in stmts.

Parameters

- **site_map** (dict (as returned by *load_site_map()*)) – A dict mapping tuples of the form *(gene, orig_res, orig_pos)* to a tuple of the form *(correct_res, correct_pos, comment)*, where *gene* is the string name of the gene (canonicalized to HGNC); *orig_res* and *orig_pos* are the residue and position to be mapped; *correct_res* and *correct_pos* are the corrected residue and position, and *comment* is a string describing the reason for the mapping (species error, isoform error, wrong residue name, etc.).

- **use_cache** (*Optional[bool]*) – If True, the SITEMAPPER_CACHE_PATH from the config (or environment) is loaded and cached mappings are read and written to the given path. Otherwise, no cache is used. Default: False

Examples

Fixing site errors on both the modification state of an agent (MAP2K1) and the target of a Phosphorylation statement (MAPK1):

```
>>> map2k1_phos = Agent('MAP2K1', db_refs={'UP':'Q02750'}, mods=[
... ModCondition('phosphorylation', 'S', '217'),
... ModCondition('phosphorylation', 'S', '221')])
>>> mapk1 = Agent('MAPK1', db_refs={'UP':'P28482'})
>>> stmt = Phosphorylation(map2k1_phos, mapk1, 'T','183')
>>> (valid, mapped) = default_mapper.map_sites([stmt])
>>> valid
[]
>>> mapped  # doctest:+IGNORE_UNICODE
[
MappedStatement:
    original_stmt: Phosphorylation(MAP2K1(mods: (phosphorylation, S, 217),␣
↪(phosphorylation, S, 221)), MAPK1(), T, 183)
    mapped_mods: (('MAP2K1', 'S', '217'), ('S', '218', 'off by one'))
                 (('MAP2K1', 'S', '221'), ('S', '222', 'off by one'))
                 (('MAPK1', 'T', '183'), ('T', '185', 'off by two; mouse sequence
↪'))
    mapped_stmt: Phosphorylation(MAP2K1(mods: (phosphorylation, S, 218),␣
↪(phosphorylation, S, 222)), MAPK1(), T, 185)
]
>>> ms = mapped[0]
>>> ms.original_stmt
Phosphorylation(MAP2K1(mods: (phosphorylation, S, 217), (phosphorylation, S,␣
↪221)), MAPK1(), T, 183)
>>> ms.mapped_mods # doctest:+IGNORE_UNICODE
[(('MAP2K1', 'S', '217'), ('S', '218', 'off by one')), (('MAP2K1', 'S', '221'), (
↪'S', '222', 'off by one')), (('MAPK1', 'T', '183'), ('T', '185', 'off by two;␣
↪mouse sequence'))]
>>> ms.mapped_stmt
Phosphorylation(MAP2K1(mods: (phosphorylation, S, 218), (phosphorylation, S,␣
↪222)), MAPK1(), T, 185)
```

get_psp_mapping(*orig_id*, *query_id*, *gene_name*, *res*, *pos*, *query_pos*, *mapping_code*)

Wrapper around Phosphosite queries that performs peptide remapping.

The function is called with a uniprot ID, residue, and position combination that is used to query the phosphosite_client for a valid corresponding site on the human reference protein. The *mapping_code* is provided by the caller to indicate the type of mapping being attempted (e.g., human isoform, mouse, rat, methionine). If a valid mapping is obtained, this is the error code that is applied. If a valid mapping is obtained but it is for a human isoform, this indicates that the queried site exists only on a human isoform and not on the human reference protein, and the code *ISOFORM_SPECIFIC_SITE* is used. If the site returned by the phosphosite_client is at a position that does not match the Uniprot reference sequence (which can happen when the queried site and the PhosphositePlus protein sequences both exclude the initial methionine), the site is remapped to the Uniprot reference sequence using the peptide information for the site in PhosphositePlus. In these cases, the mapping code *REMAPPED_FROM_PSP_SEQUENCE* is used.

Parameters

- **orig_id** (*str*) – Original Uniprot ID of the protein to be mapped.
- **query_id** (*str*) – Uniprot ID of the protein being queried for sites. This may differ from *orig_id* if the orthologous mouse or rat protein is being checked for sites.
- **gene_name** (*str*) – Gene name of the protein.
- **res** (*str*) – Residue of the site to be mapped.
- **pos** (*str*) – Position of the site to be mapped.
- **query_pos** (*str*) – Position being queried for a mapping. This differs from *pos* when off-by-one (methionine) errors are being checked.
- **mapping_code** (*str*) – Mapping code to apply in case of a successful mapping, e.g. *INFERRED_ALTERNATIVE_ISOFORM*, *INFERRED_MOUSE_SITE*, etc.

Returns MappedSite object containing the mapping, or None indicating that no mapping was found.

Return type *MappedSite* or None

static map_peptide_to_human_ref(*prot_id*, *prot_ns*, *peptide*, *site_pos*)

Return a mapped site for a given peptide.

Parameters

- **prot_id** (*str*) – A Uniprot ID or HGNC gene symbol for the protein.
- **prot_ns** (*str*) – One of 'uniprot' or 'hgnc' indicating the type of ID given.
- **peptide** (*str*) – A string of amino acid symbols representing a peptide.
- **site_pos** (*int*) – A site position within the peptide. Note: site_pos is 1-indexed.

Returns The MappedSite object gives information on results of mapping the site. See *protmapper.api.MappedSite* documentation for details.

Return type *MappedSite*

map_sitelist_to_human_ref(*site_list*, ***kwargs*)

Return a list of mapped sites for a list of input sites.

Parameters **site_list** (*list of tuple*) – Each tuple in the list consists of the following entries: (prot_id, prot_ns, residue, position).

Returns A list of MappedSite objects, one corresponding to each site in the input list.

Return type list of *protmapper.api.MappedSite*

map_to_human_ref(*prot_id*, *prot_ns*, *residue*, *position*, *do_methionine_offset=True*, *do_orthology_mapping=True*, *do_isoform_mapping=True*)

Check an agent for invalid sites and look for mappings.

Look up each modification site on the agent in Uniprot and then the site map.

Parameters

- **prot_id** (*str*) – A Uniprot ID or HGNC gene symbol for the protein.
- **prot_ns** (*str*) – One of 'uniprot' or 'hgnc' indicating the type of ID given.
- **residue** (*str*) – Residue to map on the protein to check for validity and map.
- **position** (*str*) – Position of the residue to check for validity and map.

- **do_methionine_offset** (*boolean*) – Whether to check for off-by-one errors in site position (possibly) attributable to site numbering from mature proteins after cleavage of the initial methionine. If True, checks the reference sequence for a known modification at 1 site position greater than the given one; if there exists such a site, creates the mapping. Default is True.
- **do_orthology_mapping** (*boolean*) – Whether to check sequence positions for known modification sites in mouse or rat sequences (based on PhosphoSitePlus data). If a mouse/rat site is found that is linked to a site in the human reference sequence, a mapping is created. Default is True.
- **do_isoform_mapping** (*boolean*) – Whether to check sequence positions for known modifications in other human isoforms of the protein (based on PhosphoSitePlus data). If a site is found that is linked to a site in the human reference sequence, a mapping is created. Default is True.

Returns The MappedSite object gives information on results of mapping the site. See *protmapper.api.MappedSite* documentation for details.

Return type *MappedSite*

protmapper.api.**default_mapper = <protmapper.api.ProtMapper object>**
A default instance of *ProtMapper* that contains the site information found in resources/curated_site_map.csv'.

protmapper.api.**load_site_map**(*path*)
Load the modification site map from a file.

The site map file should be a comma-separated file with six columns:

```
UniprotId: Uniprot ID of protein
Gene: Gene name
OrigRes: Original (incorrect) residue
OrigPos: Original (incorrect) residue position
CorrectRes: The correct residue for the modification
CorrectPos: The correct residue position
Comment: Description of the reason for the error.
```

Parameters **path** (*string*) – Path to the tab-separated site map file.

Returns A dict mapping tuples of the form *(uniprot_id, orig_res, orig_pos)* to a tuple of the form *(correct_res, correct_pos, comment)*, where *uniprot_id* is the Uniprot ID of the protein; *orig_res* and *orig_pos* are the residue and position to be mapped; *correct_res* and *correct_pos* are the corrected residue and position, and *comment* is a string describing the reason for the mapping (species error, isoform error, wrong residue name, etc.).

Return type dict

1.2 UniProt client

protmapper.uniprot_client.**get_chains**(*protein_id*)
Return the list of cleaved chains for the given protein.

Parameters **protein_id** (*str*) – The UniProt ID of the protein whose cleaved chains are to be returned.

Returns A list of Feature named tuples representing each chain.

Return type list of Feature

protmapper.uniprot_client.**get_entrez_id**(*protein_id*)
Return the Entrez ID given a protein ID.

Parameters **protein_id** (*str*) – UniProt ID of the protein

Returns Entrez ID of the corresponding gene or None if not available.

Return type str or None

protmapper.uniprot_client.**get_family_members**(*family_name*, *human_only=True*)
Return the HGNC gene symbols which are the members of a given family.

Parameters

- **family_name** (*str*) – Family name to be queried.
- **human_only** (*bool*) – If True, only human proteins in the family will be returned. Default: True

Returns **gene_names** – The HGNC gene symbols corresponding to the given family.

Return type list

protmapper.uniprot_client.**get_feature_by_id**(*feature_id*)
Return a Feature based on its unique feature ID.

Parameters **feature_id** (*str*) – A Feature ID, of the form PRO_*.

Returns A Feature with the given ID.

Return type *Feature* or None

protmapper.uniprot_client.**get_feature_of**(*feature_id*)
Return the UniProt ID of the protein to which the given feature belongs.

Parameters **feature_id** (*str*) – A Feature ID, of the form PRO_*.

Returns A UniProt ID corresponding to the given feature, or None if not available (generally shouldn't happen, unless the feature ID is invalid).

Return type str or None

protmapper.uniprot_client.**get_features**(*protein_id*)
Return a list of features (chains, peptides) for a given protein.

Parameters **protein_id** (*str*) – The UniProt ID of the protein whose features are to be returned.

Returns A list of Feature named tuples representing each Feature.

Return type list of Feature

protmapper.uniprot_client.**get_function**(*protein_id*)
Return the function description of a given protein.

Parameters **protein_id** (*str*) – The UniProt ID of the protein.

Returns The function description of the protein.

Return type str

protmapper.uniprot_client.**get_gene_name**(*protein_id*, *web_fallback=True*)
Return the gene name or canonical protein name for the given UniProt ID.

If available, this function returns the primary gene name provided by UniProt. If not available, the primary protein name is returned.

Parameters

- **protein_id** (*str*) – UniProt ID to be mapped.
- **web_fallback** (*Optional[bool]*) – If True and the offline lookup fails, the UniProt web service is used to do the query.

Returns **gene_name** – The gene name corresponding to the given Uniprot ID.

Return type str

protmapper.uniprot_client.**get_gene_synonyms**(*protein_id: str*) → List[str]
Return a list of synonyms for the gene corresponding to a protein.

Note that synonyms here also include the official gene name as returned by get_gene_name.

Parameters **protein_id** – The UniProt ID of the protein to query

Returns The list of synonyms of the gene corresponding to the protein

protmapper.uniprot_client.**get_hgnc_id**(*protein_id*)
Return the HGNC ID given the protein id of a human protein.

Parameters **protein_id** (*str*) – UniProt ID of the human protein

Returns **hgnc_id** – HGNC ID of the human protein

Return type str

protmapper.uniprot_client.**get_id_from_entrez**(*entrez_id*)
Return the UniProt ID given the Entrez ID of a gene.

Parameters **entrez_id** (*str*) – Entrez ID of the gene

Returns UniProt ID of the corresponding protein or None if not available.

Return type str or None

protmapper.uniprot_client.**get_id_from_mgi**(*mgi_id*)
Return the UniProt ID given the MGI ID of a mouse protein.

Parameters **mgi_id** (*str*) – The MGI ID of the mouse protein.

Returns **up_id** – The UniProt ID of the mouse protein.

Return type str

protmapper.uniprot_client.**get_id_from_mgi_name**(*mgi_name: str*) → Optional[str]
Return the UniProt ID given the MGI name of a mouse protein.

Parameters **mgi_name** (*str*) – The MGI name of the mouse protein.

Returns **up_id** – The UniProt ID of the mouse protein.

Return type str

protmapper.uniprot_client.**get_id_from_mnemonic**(*uniprot_mnemonic*)
Return the UniProt ID for the given UniProt mnemonic.

Parameters **uniprot_mnemonic** (*str*) – UniProt mnemonic to be mapped.

Returns **uniprot_id** – The UniProt ID corresponding to the given Uniprot mnemonic.

Return type str

protmapper.uniprot_client.**get_id_from_rgd**(*rgd_id*)
Return the UniProt ID given the RGD ID of a rat protein.

Parameters **rgd_id** (*str*) – The RGD ID of the rat protein.

Returns **up_id** – The UniProt ID of the rat protein.

Return type str

protmapper.uniprot_client.**get_id_from_rgd_name**(*rgd_name: str*) → Optional[str]
Return the UniProt ID given the RGD name of a rat protein.

Parameters **rgd_name** (*str*) – The RGD name of the rat protein.

Returns **up_id** – The UniProt ID of the rat protein.

Return type str

protmapper.uniprot_client.**get_ids_from_refseq**(*refseq_id*, *reviewed_only=False*)
Return UniProt IDs from a RefSeq ID".

Parameters

- **refseq_id** (*str*) – The RefSeq ID of the protein to map.
- **reviewed_only** (*Optional[bool]*) – If True, only reviewed UniProt IDs are returned. Default: False

Returns A list of UniProt IDs corresponding to the RefSeq ID.

Return type list of str

protmapper.uniprot_client.**get_length**(*protein_id*)
Return the length (number of amino acids) of a protein.

Parameters **protein_id** (*str*) – UniProt ID of a protein.

Returns **length** – The length of the protein in amino acids.

Return type int

protmapper.uniprot_client.**get_mgi_id**(*protein_id*)
Return the MGI ID given the protein id of a mouse protein.

Parameters **protein_id** (*str*) – UniProt ID of the mouse protein

Returns **mgi_id** – MGI ID of the mouse protein

Return type str

protmapper.uniprot_client.**get_mnemonic**(*protein_id*, *web_fallback=False*)
Return the UniProt mnemonic for the given UniProt ID.

Parameters

- **protein_id** (*str*) – UniProt ID to be mapped.
- **web_fallback** (*Optional[bool]*) – If True and the offline lookup fails, the UniProt web service is used to do the query.

Returns **mnemonic** – The UniProt mnemonic corresponding to the given Uniprot ID.

Return type str

protmapper.uniprot_client.**get_modifications**(*protein_id: str*) → List[Tuple[str, int]]
Return a list of modifications for a protein.

Parameters **protein_id** – The UniProt ID of the protein to query

Returns The list of modifications of the protein, each represented as a tuple of residue description string and position string.

protmapper.uniprot_client.**get_mouse_id**(*human_protein_id*)
Return the mouse UniProt ID given a human UniProt ID.

Parameters **human_protein_id** (*str*) – The UniProt ID of a human protein.

Returns **mouse_protein_id** – The UniProt ID of a mouse protein orthologous to the given human protein.

Return type str

protmapper.uniprot_client.**get_organism_id**(*protein_id*)
Return the Taxonomy ID of the organism that a protein belongs to.

Parameters **protein_id** (*str*) – The UniProt ID of a protein.

Returns The Taxonomy ID of the organism the protein belongs to or None if not available.

Return type str or None

protmapper.uniprot_client.**get_primary_id**(*protein_id*)
Return a primary entry corresponding to the UniProt ID.

Parameters **protein_id** (*str*) – The UniProt ID to map to primary.

Returns **primary_id** – If the given ID is primary, it is returned as is. Otherwise the primary IDs are looked up. If there are multiple primary IDs then the first human one is returned. If there are no human primary IDs then the first primary found is returned.

Return type str

protmapper.uniprot_client.**get_protein_synonyms**(*protein_id*)
Return a list of synonyms for a protein.

Note that this function returns protein synonyms as provided by UniProt. The get_gene_synonym returns synonyms given for the gene corresponding to the protein, and get_synonyms returns both.

Parameters **protein_id** (*str*) – The UniProt ID of the protein to query

Returns **synonyms** – The list of synonyms of the protein

Return type list[str]

protmapper.uniprot_client.**get_rat_id**(*human_protein_id*)
Return the rat UniProt ID given a human UniProt ID.

Parameters **human_protein_id** (*str*) – The UniProt ID of a human protein.

Returns **rat_protein_id** – The UniProt ID of a rat protein orthologous to the given human protein

Return type str

protmapper.uniprot_client.**get_rgd_id**(*protein_id*)
Return the RGD ID given the protein id of a rat protein.

Parameters **protein_id** (*str*) – UniProt ID of the rat protein

Returns **rgd_id** – RGD ID of the rat protein

Return type str

protmapper.uniprot_client.**get_signal_peptide**(*protein_id*, *web_fallback=True*)
Return the position of a signal peptide for the given protein.

Parameters

- **protein_id** (*str*) – The UniProt ID of the protein whose signal peptide position is to be returned.
- **web_fallback** (*Optional[bool]*) – If True the UniProt web service is used to download information when the local resource file doesn't contain the right information.

Returns A Feature named tuple representing the signal peptide.

Return type *Feature*

protmapper.uniprot_client.**get_synonyms**(*protein_id*)
Return synonyms for a protein and its associated gene.

Parameters **protein_id** (*str*) – The UniProt ID of the protein to query

Returns **synonyms** – The list of synonyms of the protein and its associated gene.

Return type list[str]

protmapper.uniprot_client.**is_human**(*protein_id*)
Return True if the given protein id corresponds to a human protein.

Parameters **protein_id** (*str*) – UniProt ID of the protein

Returns

Return type True if the protein_id corresponds to a human protein, otherwise False.

protmapper.uniprot_client.**is_mouse**(*protein_id*)
Return True if the given protein id corresponds to a mouse protein.

Parameters **protein_id** (*str*) – UniProt ID of the protein

Returns

Return type True if the protein_id corresponds to a mouse protein, otherwise False.

protmapper.uniprot_client.**is_rat**(*protein_id*)
Return True if the given protein id corresponds to a rat protein.

Parameters **protein_id** (*str*) – UniProt ID of the protein

Returns

Return type True if the protein_id corresponds to a rat protein, otherwise False.

protmapper.uniprot_client.**is_reviewed**(*protein_id*)
Return True if the UniProt ID corresponds to a reviewed entry.

Parameters **protein_id** (*str*) – The UniProt ID to check.

Returns

Return type True if it is a reviewed entry, False otherwise.

protmapper.uniprot_client.**is_secondary**(*protein_id*)
Return True if the UniProt ID corresponds to a secondary accession.

Parameters **protein_id** (*str*) – The UniProt ID to check.

Returns

Return type True if it is a secondary accessing entry, False otherwise.

protmapper.uniprot_client.**query_protein**
Retrieve the XML entry for a given protein.

Parameters **protein_id** – The UniProt ID of the protein to look up.

Returns An ElementTree representation of the XML entry for the protein.

protmapper.uniprot_client.**verify_location**(*protein_id*, *residue*, *location*)
Return True if the residue is at the given location in the UP sequence.

Parameters

- **protein_id** (*str*) – UniProt ID of the protein whose sequence is used as reference.
- **residue** (*str*) – A single character amino acid symbol (Y, S, T, V, etc.)
- **location** (*str*) – The location on the protein sequence (starting at 1) at which the residue should be checked against the reference sequence.

Returns

- *True if the given residue is at the given position in the sequence*
- *corresponding to the given UniProt ID, otherwise False.*

protmapper.uniprot_client.**verify_modification**(*protein_id*, *residue*, *location=None*)

Return True if the residue at the given location has a known modifiation.

Parameters

- **protein_id** (*str*) – UniProt ID of the protein whose sequence is used as reference.
- **residue** (*str*) – A single character amino acid symbol (Y, S, T, V, etc.)
- **location** (*Optional[str]*) – The location on the protein sequence (starting at 1) at which the modification is checked.

Returns

- *True if the given residue is reported to be modified at the given position*
- *in the sequence corresponding to the given UniProt ID, otherwise False.*
- *If location is not given, we only check if there is any residue of the*
- *given type that is modified.*

1.3 PhosphoSite client

class protmapper.phosphosite_client.**PhosphoSite**(*GENE*, *PROTEIN*, *ACC_ID*, *HU_CHR_LOC*, *MOD_RSD*, *SITE_GRP_ID*, *ORGANISM*, *MW_kD*, *DOMAIN*, *SITE_7_AA*, *LT_LIT*, *MS_LIT*, *MS_CST*, *CST_CAT*)

Bases: tuple

ACC_ID
Alias for field number 2

CST_CAT
Alias for field number 13

DOMAIN
Alias for field number 8

GENE
Alias for field number 0

HU_CHR_LOC
Alias for field number 3

LT_LIT
Alias for field number 10

MOD_RSD
Alias for field number 4

MS_CST
Alias for field number 12

MS_LIT
Alias for field number 11

MW_kD
Alias for field number 7

ORGANISM
Alias for field number 6

PROTEIN
Alias for field number 1

SITE_7_AA
Alias for field number 9

SITE_GRP_ID
Alias for field number 5

class protmapper.phosphosite_client.**PspMapping**(*mapped_id*, *mapped_res*, *mapped_pos*, *motif*, *respos*)

Bases: tuple

mapped_id
Alias for field number 0

mapped_pos
Alias for field number 2

mapped_res
Alias for field number 1

motif
Alias for field number 3

respos
Alias for field number 4

protmapper.phosphosite_client.**has_data**()
Check if the PhosphoSite data is available and can be loaded.

Returns True if the data can be loaded, False otherwise.

Return type bool

protmapper.phosphosite_client.**map_to_human_site**(*up_id*, *mod_res*, *mod_pos*)
Find site on human ref seq corresponding to (possibly non-human) site.

Parameters

- **up_id** (*str*) – Uniprot ID of the modified protein (generally human, rat, or mouse).
- **mod_res** (*str*) – Modified amino acid residue.
- **mod_pos** (*str*) – Amino acid sequence position.

Returns Returns amino acid position on the human reference sequence corresponding to the site on the given protein.

Return type str

protmapper.phosphosite_client.**sites_only**(*exclude_isoforms=False*)
Return PhosphositePlus data as a flat list of proteins and sites.

Parameters **exclude_isoforms** (*bool*) – Whether to exclude sites for protein isoforms. Default is False (includes isoforms).

Returns Each tuple consists of (uniprot_id, residue, position).

Return type list of tuples

1.4 Resource management

class protmapper.resources.**Feature**(*type*, *begin*, *end*, *name*, *id*, *is_main*)
Bases: tuple

begin
Alias for field number 1

end
Alias for field number 2

id
Alias for field number 4

is_main
Alias for field number 5

name
Alias for field number 3

type
Alias for field number 0

class protmapper.resources.**ResourceManager**(*resource_map*)
Bases: object

Class to manage a set of resource files.

Parameters **resource_map** (*dict*) – A dict that maps resource file IDs to a tuple of resource file names and download functions.

download_resource_file(*resource_id*, *cached=True*)
Download the resource file corresponding to the given ID.

Parameters

- **resource_id** (*str*) – The ID of the resource.
- **cached** (*Optional[bool]*) – If True, the download is a pre-processed file from S3, otherwise the download is obtained and processed from the primary source. Default: True

get_create_resource_file(*resource_id*, *cached=True*)
Return the path to the resource file, download if it doesn't exist.

Parameters

- **resource_id** (*str*) – The ID of the resource.
- **cached** (*Optional[bool]*) – If True, the download is a pre-processed file from S3, otherwise the download is obtained and processed from the primary source. Default: True

Returns The path to the resource file.

Return type str

get_download_fun(*resource_id*)
Return the download function for the given resource.

Parameters **resource_id** (*str*) – The ID of the resource.

Returns The download function for the given resource.

Return type function

get_resource_file(*resource_id*)
Return the path to the resource file with the given ID.

Parameters **resource_id** (*str*) – The ID of the resource.

Returns The path to the resource file.

Return type str

get_resource_ids()
Return a list of all the resource IDs managed by this manager.

has_resource_file(*resource_id*)
Return True if the resource file exists for the given ID.

Parameters **resource_id** (*str*) – The ID of the resource.

Returns True if the resource file exists, false otherwise.

Return type bool

CHAPTER 2

REST API

The Protmapper REST API allows interacting with the Protmapper through HTTP requests. The REST API takes GET or POST request with a JSON payload.

The REST API exposes the following endpoints:

2.1 *map_to_human_ref*

This endpoint takes 4 arguments: *prot_id*, *prot_ns*, *residue*, and *position* and returns a JSON representation of a MappedSite object.

Example

Input:

```
{"prot_id": "MAPK1",
 "prot_ns": "hgnc",
 "residue": "T",
 "position": "183"}
```

Output:

```
{
 "description": "INFERRED_MOUSE_SITE",
 "error_code": null,
 "gene_name": "MAPK1",
 "mapped_id": "P28482",
 "mapped_pos": "185",
 "mapped_res": "T",
 "orig_pos": "183",
 "orig_res": "T",
 "up_id": "P28482",
```

(continues on next page)

(continued from previous page)


```
 "valid": false
}
```


2.2 *map_sitelist_to_human_ref*

This endpoint takes a single *site_list* argument which is a list of lists where each list consists of exactly 4 elements in the following order: *prot_id*, *prot_ns*, *residue*, and *position*. The response is a list of MappedSite object represented as JSON.

Example

Input:


```
{"site_list": [
    ["MAPK1","hgnc","T","185"],
    ["MAPK1", "hgnc", "T", "183"]
    ]
}
```


Output:


```
[
 {
  "description": "VALID",
  "error_code": null,
  "gene_name": "MAPK1",
  "mapped_id": null,
  "mapped_pos": null,
  "mapped_res": null,
  "orig_pos": "185",
  "orig_res": "T",
  "up_id": "P28482",
  "valid": true
 },
 {
  "description": "INFERRED_MOUSE_SITE",
  "error_code": null,
  "gene_name": "MAPK1",
  "mapped_id": "P28482",
  "mapped_pos": "185",
  "mapped_res": "T",
  "orig_pos": "183",
  "orig_res": "T",
  "up_id": "P28482",
  "valid": false
 }
]
```


2.3 Optional arguments

Both endpoints take the following optional boolean arguments which are *true* by default:

- *do_methionine_offset*
- *do_orthology_mapping*
- *do_isoform_mapping*
- genindex
- modindex
- search

Python Module Index

p

protmapper.api, 1
protmapper.phosphosite_client, 11
protmapper.resources, 13
protmapper.rest_api, 15
protmapper.uniprot_client, 5

Index

A

ACC_ID (*protmapper.phosphosite_client.PhosphoSite attribute*), 11

B

begin (*protmapper.resources.Feature attribute*), 13

C

CST_CAT (*protmapper.phosphosite_client.PhosphoSite attribute*), 11

D

default_mapper (*in module protmapper.api*), 5
description (*protmapper.api.MappedSite attribute*), 2
DOMAIN (*protmapper.phosphosite_client.PhosphoSite attribute*), 11
download_resource_file() (*protmapper.resources.ResourceManager method*), 13

E

end (*protmapper.resources.Feature attribute*), 13
error_code (*protmapper.api.MappedSite attribute*), 1

F

Feature (*class in protmapper.resources*), 13

G

GENE (*protmapper.phosphosite_client.PhosphoSite attribute*), 11
gene_name (*protmapper.api.MappedSite attribute*), 2
get_chains() (*in module protmapper.uniprot_client*), 5
get_create_resource_file() (*protmapper.resources.ResourceManager method*), 13
get_download_fun() (*protmapper.resources.ResourceManager method*), 14
get_entrez_id() (*in module protmapper.uniprot_client*), 5
get_family_members() (*in module protmapper.uniprot_client*), 6
get_feature_by_id() (*in module protmapper.uniprot_client*), 6
get_feature_of() (*in module protmapper.uniprot_client*), 6
get_features() (*in module protmapper.uniprot_client*), 6
get_function() (*in module protmapper.uniprot_client*), 6
get_gene_name() (*in module protmapper.uniprot_client*), 6
get_gene_synonyms() (*in module protmapper.uniprot_client*), 7
get_hgnc_id() (*in module protmapper.uniprot_client*), 7
get_id_from_entrez() (*in module protmapper.uniprot_client*), 7
get_id_from_mgi() (*in module protmapper.uniprot_client*), 7
get_id_from_mgi_name() (*in module protmapper.uniprot_client*), 7
get_id_from_mnemonic() (*in module protmapper.uniprot_client*), 7
get_id_from_rgd() (*in module protmapper.uniprot_client*), 7
get_id_from_rgd_name() (*in module protmapper.uniprot_client*), 8
get_ids_from_refseq() (*in module protmapper.uniprot_client*), 8
get_length() (*in module protmapper.uniprot_client*), 8
get_mgi_id() (*in module protmapper.uniprot_client*), 8
get_mnemonic() (*in module protmap-*

per.uniprot_client), 8
get_modifications() (in module protmapper.uniprot_client), 8
get_mouse_id() (in module protmapper.uniprot_client), 8
get_organism_id() (in module protmapper.uniprot_client), 9
get_primary_id() (in module protmapper.uniprot_client), 9
get_protein_synonyms() (in module protmapper.uniprot_client), 9
get_psp_mapping() (protmapper.api.ProtMapper method), 3
get_rat_id() (in module protmapper.uniprot_client), 9
get_resource_file() (protmapper.resources.ResourceManager method), 14
get_resource_ids() (protmapper.resources.ResourceManager method), 14
get_rgd_id() (in module protmapper.uniprot_client), 9
get_signal_peptide() (in module protmapper.uniprot_client), 9
get_synonyms() (in module protmapper.uniprot_client), 10

H

has_data() (in module protmapper.phosphosite_client), 12
has_mapping() (protmapper.api.MappedSite method), 2
has_resource_file() (protmapper.resources.ResourceManager method), 14
HU_CHR_LOC (protmapper.phosphosite_client.PhosphoSite attribute), 11

I

id (protmapper.resources.Feature attribute), 13
InvalidSiteException, 1
is_human() (in module protmapper.uniprot_client), 10
is_main (protmapper.resources.Feature attribute), 13
is_mouse() (in module protmapper.uniprot_client), 10
is_rat() (in module protmapper.uniprot_client), 10
is_reviewed() (in module protmapper.uniprot_client), 10
is_secondary() (in module protmapper.uniprot_client), 10

L

load_site_map() (in module protmapper.api), 5
LT_LIT (protmapper.phosphosite_client.PhosphoSite attribute), 11

M

map_peptide_to_human_ref() (protmapper.api.ProtMapper static method), 4
map_sitelist_to_human_ref() (protmapper.api.ProtMapper method), 4
map_to_human_ref() (protmapper.api.ProtMapper method), 4
map_to_human_site() (in module protmapper.phosphosite_client), 12
mapped_id (protmapper.api.MappedSite attribute), 2
mapped_id (protmapper.phosphosite_client.PspMapping attribute), 12
mapped_pos (protmapper.api.MappedSite attribute), 2
mapped_pos (protmapper.phosphosite_client.PspMapping attribute), 12
mapped_res (protmapper.api.MappedSite attribute), 2
mapped_res (protmapper.phosphosite_client.PspMapping attribute), 12
MappedSite (class in protmapper.api), 1
MOD_RSD (protmapper.phosphosite_client.PhosphoSite attribute), 11
motif (protmapper.phosphosite_client.PspMapping attribute), 12
MS_CST (protmapper.phosphosite_client.PhosphoSite attribute), 12
MS_LIT (protmapper.phosphosite_client.PhosphoSite attribute), 12
MW_kD (protmapper.phosphosite_client.PhosphoSite attribute), 12

N

name (protmapper.resources.Feature attribute), 13
not_invalid() (protmapper.api.MappedSite method), 2

O

ORGANISM (protmapper.phosphosite_client.PhosphoSite attribute), 12
orig_pos (protmapper.api.MappedSite attribute), 2
orig_res (protmapper.api.MappedSite attribute), 1

P

PhosphoSite (class in protmapper.phosphosite_client), 11
PROTEIN (protmapper.phosphosite_client.PhosphoSite attribute), 12
ProtMapper (class in protmapper.api), 2

protmapper.api (*module*), 1
protmapper.phosphosite_client (*module*), 11
protmapper.resources (*module*), 13
protmapper.rest_api (*module*), 15
protmapper.uniprot_client (*module*), 5
PspMapping (*class in protmapper.phosphosite_client*), 12

Q

query_protein (*in module protmapper.uniprot_client*), 10

R

ResourceManager (*class in protmapper.resources*), 13
respos (*protmapper.phosphosite_client.PspMapping attribute*), 12

S

SITE_7_AA (*protmapper.phosphosite_client.PhosphoSite attribute*), 12
SITE_GRP_ID (*protmapper.phosphosite_client.PhosphoSite attribute*), 12
sites_only() (*in module protmapper.phosphosite_client*), 12

T

type (*protmapper.resources.Feature attribute*), 13

U

up_id (*protmapper.api.MappedSite attribute*), 1

V

valid (*protmapper.api.MappedSite attribute*), 1
verify_location() (*in module protmapper.uniprot_client*), 10
verify_modification() (*in module protmapper.uniprot_client*), 11