x-bte extension doc

Oct 07, 2021

Contents

CHAPTER 1

Introduction

The goal of this extension is to facilitate the automatic retrieval of single-hop knowledge graph data in the format of subject-predicate-object (e.g. ChemicalSubstance – treats – Disease) from APIs by intelligent agents, such as BioThings Explorer. This is achieved through documenting single-hop knowledge graph retrieval operations that an individual OpenAPI operation can perform. The knowledge graph retrieval operation should be defined using the BioLink Data Model, e.g. each input/output node should be categorized using Biolink classes and ID prefixes, edges should be labeled using valid Biolink relationship types.

CHAPTER 2

Topics

2.1 x-bte-kgs-operations Object

Describe list of single-hop knowledge graph retrieval operations that a single OpenAPI operation can perform.

Properties		
Property name	Type	Description
x-bte-kgs-operations	[x-bte-kgs-operation Object | Reference Object]	A list of single-hop knowledge graph retrieval operations that an OpenAPI operation can perform. The list can use the Reference Object to link to x-bte-kgs operation defined in components

2.1.1 x-bte-kgs-operations example

The following example defines two x-bte-kgs-operations (ChemicalSubstance – physically_interacts_with – Gene && Gene – physically_interacts_with – ChemicalSubstance) associated with the GET operation of the /interactions endpoint.

```
{
    "interactions.json": {
        "get": {
            "parameters": [
                {
                    "in": "query",
                    "name": "drugs"
                },
                {
                    "in": "query",
                    "name": "genes
                }
```

(continues on next page)

(continued from previous page)

```
            ],
            "x-bte-kgs-operations": [
                {
                    "inputs": [
                        {
                            "id": "biolink:CHEMBL.COMPOUND",
                            "semantic": "biolink:ChemicalSubstance"
                        }
                    ],
                    "outputs": [
                        {
                            "id": "biolink:NCBIGene",
                            "semantic": "biolink:Gene"
                        }
                    ],
                    "parameters": {
                        "drugs": "{inputs[0]}"
                    },
                    "predicate": "biolink:physically_interacts_with",
                    "supportBatch": False,
                    "responseMapping": {
                        "NCBIGene": "matchedTerms.interactions.geneEntrezId",
                        "publication": "matchedTerms.interactions.pmids"
                    }
                },
                {
                    "inputs": [
                        {
                            "id": "biolink:NCBIGene",
                            "semantic": "biolink:Gene"
                        }
                    ],
                    "outputs": [
                        {
                            "id": "biolink:CHEMBL.COMPOUND",
                            "semantic": "biolink:ChemicalSubstance"
                        }
                    ],
                    "parameters": {
                        "genes": "{inputs[0]}"
                    },
                    "predicate": "biolink:physically_interacts_with",
                    "supportBatch": False,
                    "responseMapping": {
                        "CHEMBL.COMPOUND": "matchedTerms.interactions.drugChemblId",
                        "publication": "matchedTerms.interactions.pmids"
                    }
                }
            ]
        }
    }
}
```

2.2 x-bte-kgs-operation Object

Describe a single-hop knowledge graph retrieval operation.

The x-bte-kgs-operation object contains 3 parts:

- Single-hop knowledge graph association

  Metadata information describing the knowledge retrieval operation, including the input, output, predicate and source. One kgs-operation may have more than one inputs or outputs, but it should have exactly one predicate to capture the relationship between the input(s) and output(s).

- API Operation

  Describe how to structure the API call in order to retrieve the knowledge, including request body and parameters. Other relevant information to perform API query, e.g. server URL, path, HTTP method can be inferred from the server object and path object.

- Response Mapping

  Map individual fields in the API response to their corresponding concepts in the Biolink model.

2.2.1 Properties

Properties		
Property name	Type	Description
inputs	[x-bte-kgs-node Object]	Specifies the list of inputs for the single-hop knowledge graph retrieval operation, including the inputsemantic type and input identifier type.
outputs	[x-bte-kgs-node Object]	Specifies the list of inputs for the single-hop knowledge graph retrieval operation, including the inputsemantic type and input identifier type.
predicate	String	Specifies the predicate for the kgs operation, in other words, the relationship between the inputs and outputs.
source	String	Specifies the source database which provides the association.
parameters	x-bte-parameter	An object to hold parameter names and their corresponding values. If the parameter corresponds to one of the inputs, should use the following notation $inputs[index]. For example, $inputs[0] means this parameter correspond to the first element of the inputs.
requestBody	x-bte-requestBody	**An object representing the** request body. If a parameter corresponds to one of the inputs, should use the following notation $inputs[index]. For example, $inputs[0] means this parameter correspond to the first element of the inputs.
supportBatch	Boolean	Indicate whether the operation support batch query.
inputSeparators	String	**Describe the operator used** to separate inputs in a batch query. Only need to specify when supportBatch is True. Default value is ",".
responseMapping	x-bte-response -mapping Objet	Provide one-to-one map between individual field in the API response and the corresponding concept in the biolink model.
2.2. x-bte-kgs-operation Object		**7**
useTemplating	Boolean	Indicate whether to use nunjucks templating.
templateImputs	Object	An object in which to delcare any

2.2.2 x-bte-kgs-operations example

The following example defines one x-bte-kgs-operation (ChemicalSubstance – physically_interacts_with – Gene).

```
   {
       "x-bte-kgs-operations": [
           {
               "inputs": [
                   {
                       "id": "biolink:CHEMBL.COMPOUND",
                       "semantic": "biolink:ChemicalSubstance"
                   }
               ],
               "outputs": [
                   {
                       "id": "biolink:NCBIGene",
                       "semantic": "biolink:Gene"
                   }
               ],
               "parameters": {
                   "drugs": "{inputs[0]}"
               },
               "predicate": "biolink:physically_interacts_with",
               "supportBatch": False,
               "responseMapping": {
                   "NCBIGene": "matchedTerms.interactions.geneEntrezId",
                   "publication": "matchedTerms.interactions.pmids"
               }
           }
       ]
   }

Templated x-bte operations query
********************************
```

To use templated queries, first enable query templating with the property `useTemplating: true`. `queryInputs` takes the place of `{inputs[0]}` to reference input IDs, while other variables, delcared in the annotation under `templateInputs`, may be referenced.

Any part of `parameters` or `requestBody.body` will be rendered through Nunjucks, meaning that any Nunjucks recognized templating will be applied. Templates are rendered per-property of `parameters` and `requestBody.body`, unless `requestBodyType: object` is set, in which case the entirety of `body` is expected to be a string and will be parsed as JSON into an object after being rendered. This, in concert with `header: application/json` allows JSON to be send as the body of a POST request.

A number of custom filter functions have been defined, as listed below:

- `substr(begin, end)`: slice a string
- `addPrefix(prefix, delim)`: add a prefix, with `delim` between prefix and string defaulting to `:`
- `rmPrefix(delim)`: remove a prefix by splitting by delimiter and removing first string, with delimiter defaulting to `:`. If no prefix is found, the string is returned.
- `replPrefix(prefix, delim)` replace a prefix by using `rmPrefix` and `addPrefix` in order, using same delimiter.
- `wrap(start, end)`: wrap the input string between `start` and `end`, or `start` and `start` if end is not provided.

- `joinSafe(delim)`: Join the entries of an array by `delim`, or `,` if none is provided. If a string is provided instead of an array, the string is simply returned.

2.2.3 Templated Example

The following example defines one x-bte-kgs-operation in yaml format.

```yaml
disease-gene-templated:
  - useTemplating: true ## flag to say templating is being used below
    inputs:
      - id: UMLS
        semantic: Disease
    templateInputs:
      desiredField: disgenet.genes_related_to_disease
    requestBodyType: object
    requestBody:
      body:
        requestBody:
          body: >-
            {
              "q": [
                {% for input in queryInputs %}
                  ["{{input}}", "Definitive"]{% if loop.revindex0 %},{% endif %}
                {% endfor %}
              ],
              "scopes": ["entrezgene", "clingen.clinical_validity.classification"]
            }
      header: application/json
    parameters:
      fields: "{{ desiredField }}"
    outputs:
      - id: NCBIGene
        semantic: Gene
    predicate: related_to
    source: "infores:disgenet"
    response_mapping:
      "$ref": "#/components/x-bte-response-mapping/disease-gene"
```

`useTemplating` Enables templating. `templateInputs` allows us to define static variables to use in our templates. `requestBodyType` states that the request body will be parsed as JSON, while the header allows the request to be sent as such. `parameters.fields` makes use of our static veriable: `fields` will evaluate to the value of `desiredField`.

Our template generates a Biothings-compatible batch query in JSON format. if `queryInputs` were an array such as `['aaa', 'bbb']`, the request body would render as such:

```
{
    "q": [
        ["aaa", "Definitive"],
        ["bbb", "Definitve"]
    ],
    "scopes": ["entrezgene", "clingen.clinical_validity.classification"]
}
```

We make use of a for loop to dynamically create each `[input, Definitive]` array, and an if statement checking how many iterations until the final (0-indexed) in order to avoid inserting a comma at the end of the array of arrays.

2.3 x-bte-kgs-node Object

Describe a node in a meta knowledge graph. Used to describe the inputs and outputs of a single-hop knowledge graph retrieval operation.

2.3.1 Properties

Properties		
Property name	Type	Description
id	String	The identifier used to represent the node, e.g. NCBIGene. The value should be prefixed with "biolink:".
semantic	String	The semantic type used to represent the node, e.g. Gene. The value should be prefixed with "biolink:".

2.3.2 x-bte-kgs-node example

The following example represents a x-bte-kgs-node object with identifier as "NCBIGene" from the biolink model and semantic type as "Gene" from the biolink model.

```
{
    "id": "biolink:NCBIGene",
    "semantic": "biolink:Gene"
}
```

2.4 x-bte-parameter Object

An object to hold parameter names and their corresponding values. If the value of the parameter is constant for the single-hop knowledge graph operation, use the const value in the object. If the value of the parameter is not constant and correspond to one of the inputs of the knowledge graph operation, use the notation $inputs[index], where the index refers to the index of the input in the inputs list. For example, {inputs[0]} represents the first element of the inputs list.

2.4.1 Properties

Properties		
Property name	Type	Description
parameterName	String	The value of the parameter.

2.4.2 x-bte-parameter example

The following example represents a x-bte-kgs-parameter, where the interaction_type parameter takes a constant value "gene2chemical", whereas the value parameter corresponds to the first inputs.

```
{
    "interaction_type": "gene2chemical",
    "value": "{inputs[0]}"
}
```

2.5 x-bte-requestBody Object

An object representing the request body. If the value of the requestBody parameter is constant for the single-hop knowledge graph operation, use the const value in the object. If the value of the parameter is not constant and correspond to one of the inputs of the knowledge graph operation, use the notation $inputs[index], where the index refers to the index of the input in the inputs list. For example, {inputs[0]} represents the first element of the inputs list.

2.5.1 Properties

Properties		
Property name	Type	Description
parameterName	String	The value of the parameter.

2.5.2 x-bte-requestBody example

The following example represents a x-bte-requestBody object, where the scopes parameter takes a constant value "entrezgene", whereas the q parameter corresponds to the first inputs. .. code-block:: json

{ "q": "{inputs[0]}",

"value": "entrezgene"

}

2.6 x-bte-response-mapping Object

Provide one-to-one map between individual field in the API response and the corresponding concept in the Biolink model.

2.6.1 Properties

Properties		
Property name	Type	Description
biolinkConceptName	String	Map between individual fiefield in API response and corresponding concept name in Biolink model. Nested fields should be represented using the dot notation.

2.6.2 x-bte-response-mapping example

The following example represents a x-bte-response-mapping object, where the nested field “go.CC.id” correspond to the Biolink concept GO, and the “go.CC.pubmed” correspond to the Biolink concept publication. .. code-block:: json

{ “GO”: “go.CC.id”,

“publication”: “go.CC.pubmed”

}

CHAPTER 3

Indices and tables

- genindex
- modindex
- search