CyvestiGO Graph API Documentation (draft)

NOTE:

- actual parameters and output from the APIs to be confirm.
- as graph generation / merging will be a long running process, after sending request, most of the time only a result `ID` will be returned for polling

Graph Generation with Playbook

```
# {base_url}/graph/generic

@post
def generic(event: list) -> http.ok:
    event = [
        {
            "node1ID": "1",
            "node2ID": "2",
            "linkLabel": "runs",
            "node1Label":"Processes",
            "node2Label":"Processes",
            "properties": [
                {..."node1"...},
                {..."node2"...},
                {..."link"...}
            ]
        }
    ]
    return http.ok
```

Graph Generation with query

```
# {base_url}/graph/query

@post
def query(query: dict) -> http.ok:
    query = {
        "coomputerName": "STEWART-DESKTOP",
        "fileName": "powershell.exe",
        "startTime": timeStamp(),
        "endTime": timeStamp()
    }
    return http.ok
```

Retrieving of Graph (Original)

```
# {base_url}/graph/original/{id}

@get
def original(graph_id: str) -> list:
    output = {
        "nodes": [...],
        "links": [...],
        "properties": [...]
    }

    # None == graph generation not complete
    return output or None
```

Retrieving of Graph (Original)

```
# {base_url}/graph/copy/{id}

@get
def copy(graph_id: str) -> list:
    output = {
        "nodes": [...],
        "links": [...],
        "properties": [...]
    }

    # None == graph generation not complete
    return output or None
```

Graph Merging

```
# {base_url}/graph/merge

@post
def merge(graph1_id: str, graph2_id: str) -> new_id:
    graph1 = retrieve_from_db(graph1_id)
    graph2 = retrieve_from_db(graph2_id)

    new_graph_id, new_graph = process_merge(graph1, graph2)
    return new_graph_id
```

Renaming of Items (Nodes or links)

```
# {base_url}/graph/{graph_id}/{item_id}/

@put
def rename(graph_id:str, item_id: str, new_name: str):
    graph1 = retrieve_from_db(graph_id)
    result: bool = rename_item(graph1, item_id, new_name)
    if result:
        return http.ok
    return http.bad_request
```

Deleting of Items (Nodes or links)

```
# {base_url}/graph/{graph_id}/{item_id}/

@delete
def delete(graph_id:str, item_id: str):
    graph1 = retrieve_from_db(graph_id)
    result: bool = delete_item(graph1, item_id)
    if result:
        return http.ok
    return http.bad_request
```

Tagging of Properties

```
# {base_url}/graph/{graph_id}/properties/

@put
def properties(graph_id:str, property_id: str, tag: str):
    graph1 = retrieve_from_db(graph_id)
    result: bool = update_property_tag(graph1, property_id, tag)
    if result:
        return http.ok
    return http.bad_request
```