graphspace-manual Documentation

Release 1.0

Aditya Bharadwaj

May 20, 2021

Contents

Table of Contents

CHAPTER 1

Introduction

Networks have become ubiquitous in systems biology. Visualization is a crucial component in their analysis. However, collaborations within research teams in network biology are hampered by software systems that are either specific to a computational algorithm, create visualizations that are not biologically meaningful, or have limited features for sharing networks and visualizations.

We present GraphSpace, a web-based platform that fosters team science by allowing collaborating research groups to easily store, interact with, lay out, and share networks. Users can upload richly-annotated networks, irrespective of the algorithms or software used to generate them. Users can create private groups, invite other users to join groups, and share networks with group members. A user may search for networks for specific attributes and for the presence of a specific node or nodes. A powerful layout editor allows users to efficiently modify node positions, edit node and edge styles, save new layouts, and share them with other users. Users may make networks public and provide a persistent URL in a publication, enabling others to explore these networks. While users can view and interact with GraphSpace networks anonymously, several features become available after a user creates an account and logs in. We describe the main features of GraphSpace in this documentation.

GraphSpace is being developed by T. M. Murali ‘s research group in the Department of Computer Science at Virginia Tech.

CHAPTER 2

Quick Tour of GraphSpace

2.1 Welcome Screen

The welcome page greets a user when the user visits GraphSpace. The Welcome Screen is designed to access commonly used features of GraphSpace like:

- *Create Account*
- *Log In*
- *Upload a graph*
- *List of uploaded graphs*

2.1.1 Create Account

The user can create an account on GraphSpace by following the given steps:

1. Click on the `Create Account` button on the top navigation bar. This will trigger a create account window/pop-up to be displayed.
2. Enter the your email address.
3. Enter a password for your account.
4. Verify your password by entering the same password again.
5. Click on `Submit` button to create the account with the given email address and password.

Create Account

2.1.2 Log In

The user can log-in to GraphSpace by following the given steps:

1. Click on the `Log In` button on the top navigation bar. This will trigger a log-in window/pop-up to be displayed.

2. Enter your email address.
3. Enter you password.
4. Click on `Submit` button to log in with the given email address.
5. If you have forgotten you password, click on the `Forgot Password` link.

Log In

For the rest of the tutorial, we assume that the user has already created an account. We also assume that the user belongs to a group with shared networks or that the user uploaded networks via the upload page or the RESTful API. The tutorial specifically steps through how `user1@example.com` interacts with GraphSpace after logging in.

2.1.3 Upload a graph

The user can upload a graph on GraphSpace by following the given steps:

1. Go to the Upload Graph Page by clicking on the `Upload Graph` button on Home Page.
2. Enter a unique name for the new graph.
3. Select a CYJS file which contains the graph information.
4. Select a JSON file which contains the style information for the graph. (Optional Step)
5. Click on `Submit` button to upload the graph using the selected files.
6. Once the graph has been uploaded, GraphSpace will provide a unique URL through which the user may interact with the graph represented by the uploaded files.

You can use the following sample files to learn how to upload graphs to GraphSpace:

- Sample network file
- Sample style file

Upload a graph

2.1.4 List of uploaded graphs

The user can go to a page that lists the graphs accessible by the user on GraphSpace by following the given steps:

- Click on the button titled `Graphs` on the top navigation bar. OR
- Click on the `My Graphs` button on the Home Page.

In this example, the user owns 33 graphs, can access 64 public graphs and 33 graphs are shared with this user. List of uploaded graphs

2.2 Searching within Multiple Graphs

The search interface in GraphSpace allows a user to search for networks that have a specific attribute or tag and contain one or more nodes using simple syntax on Graphs Page by following the given steps:

1. Enter the name of the graph/node or specific network attribute mapping you are searching for in the search bar.
2. Press `Enter` key or click on the `Search` button.

In this example, the user searches for the list for graphs that contain the protein (node) CTNNB1 (the symbol for β-catenin, a transcriptional regulator in the Wnt signaling pathway). The reduced list of graphs are the graphs where protein (node) name, label or aliases contain CTNNB1 as a substring. The match is case-insenstive. In the following example, There are six graphs owned by the user and thirty-two public graphs that contain this protein. Each link in the Graph Name column will take the user to a specific graph with the search term highlighted. In this example, the user clicks on the graph with the name KEGG-Wnt-signaling-pathway and reaches the graph for the Wnt pathway with the searched node highlighted.

Searching within Multiple Graphs

2.3 Searching within a Single Graph

The user can search for node or edges within a given graph on GraphSpace by following the given steps:

1. Enter the name of the node or an edge you are searching for in the search bar.
2. The nodes or edges are highlighted automatically as you type in the name of the node or edge in the search bar.

In the following example, the user searches for the graph for two proteins (nodes) CTNNB1 and WNT using the query ctnnb1, wnt. This search query highlights the proteins where protein (node) name, label or aliases contains CTNNB1 or WNT as a substring (case-insensitive). In the following example, the graph contains four nodes which match the given query.

Searching nodes within a Single Graph

In the following example, the user searches the graph for edges from Wnt to Fzd using the query Wnt::Fzd. GraphSpace highlights any edge whose tail node matches wnt and whose head node matches fzd. In the following example, the graph contains three edges which match the given query.

Searching edges within a Single Graph

2.4 Interacting with a Graph

In this section, we examine different ways to interact with an individual network on its page. The information that appears in following examples must be included in the JSON files that are uploaded by the network owner, as described in the Network Model section. An individual network page is designed to access features like:

- *Graph Information*
- *Edge and Node Information*
- *Export Graph*
- *Change Layout*
- *Share Layout*

2.4.1 Graph Information

As its name suggests, the Graph Information panel displays information about the entire graph, e.g., a legend of node and edge shapes and colors. The description attribute in the JSON for the network specifies this content. The user can go to Graph Information panel by clicking on the Graph Information link above the graph.

Graph Details

Please refer to this section for more information.

2.4.2 Edge and Node Information

Clicking on a node or edge pops up a panel with information on that node or edge. The `popup` attribute for the node in the network JSON specifies this content.

Edge and Node Information

Please refer to this section for more information.

2.4.3 Export Graph

GraphSpace allows users to export a graph as an image file or a JSON file. GraphSpace does not support any other export formats since it relies on Cytoscape.js for this functionality, which implements only export to PNG, JPG and JSON format.

In the following example, the user is exporting the graph as an image in PNG format.

Export Graph

Please refer to this section for more information.

2.4.4 Change Layout

Layouts provide a powerful means to organize nodes within a network. The following figures illustrate how a user may access and view automatically generated and previously saved layouts.

GraphSpace allows users to change layout using the following steps:

1. Click on the `Change Layout` button to view available layout options.
2. The `Change Layout` panel provides two alternatives: - **Select Layout Algorithm** - List of layout algorithms supported by GraphSpace through its use of Cytoscape.js. - **Select Saved Layout** - List of layout saved by the user using GraphSpace. The user has created them in earlier sessions by manually modifying the positions of nodes and edges created by some automatic layout algorithm and saving the layout.
3. Click on a layout option to change the current layout.

In the following example, the user selects to view the layout titled "manual-top-to-bottom". To create this layout, the user manually moved nodes so that extracellular ligands and receptors (triangles) appear on the top while transcription factors (rectangles) appear at the bottom. The user can click on the `Back` button to return to the main menu.

Change Layout

Please refer to this section for more information.

2.4.5 Share Layout

GraphSpace allows users to share a layout using the following steps:

1. Click on the `Layouts` link above the graph.
2. The layouts panel shows two types of layouts:
   - **Private Layouts** - The user has created them in earlier sessions by manually modifying the positions of nodes and edges. But the user has not shared them with any other user.
   - **Shared Layouts** - These layouts were created by the user who has access to this graph and shared the layout with other users who have access to this graph.

3. Click on the `Share` link next to the layout name of a private layout you want to share with other users who have access to this graph.

Note: The icons next to each layout name allow the user to (i) change its name, (ii) share it with other users who have access to this graph, (iii) delete this layout.

Share Layout

Please refer to this section for more information.

CHAPTER 3

Terminology

This section describes GraphSpace concepts and terms that are used throughout the documentation.

3.1 Anonymous vs. Registered user

An **anonymous** user is anyone that is using GraphSpace without being logged in. An anonymous user does not have access to the REST API and may not be a member of a group. Such a user may upload graphs anonymously via a web interface. We will delete graphs that are uploaded after 30 days.

A **registered** user is anyone that is logged into GraphSpace. By logging into GraphSpace, a user has full access to the REST API and is allowed to be a member of groups. Graphs uploaded by a registered user will remain in that user's account unless explicitly deleted by the user.

3.2 Graph visibility: Private, Shared, or Public

The visibility of a graph uploaded by a registered user can have one of three states: private, shared, or public. The visibility of a graph controls who can view the graph and interact with it upon visiting the URL associated with that graph.

3.2.1 Private

When a registered user uploads a graph to GraphSpace either through the REST API or through the web interface, its visibility is **private**, i.e., only the graph owner (the user who uploaded it) can view it.

3.2.2 Shared

A graph owner may share a graph with one or more groups. At this point, the graph's visibility is **shared**, i.e., all members of these groups may view the graph.

3.2.3 Public

An owner may make a specific network or all networks associated with a tag public, e.g., as accompanying information for a publication. All public networks and tags can be accessed using a URL to GraphSpace. For example, several automated reconstructions of human signaling pathways (Ritz et al., 2016) appear at http://graphspace.org/graphs?tags=2015-npj-sysbio-appl-pathlinker.

When an anonymous user uploads a graph, it is by default a public graph, i.e., it is accessible to any user who visits GraphSpace and knows the URL of the graph.

3.3 Groups

A group is a collection of GraphSpace users. For example, if there are multiple researchers who are collaborating a project, a group may be created containing all of them. Another use case for a group is for all students registered in a network biology course.

Groups lie at the core of GraphSpace's functionality. Any GraphSpace user may create a group and add other GraphSpace users as members of the group using their email address. Group owners can also invite other users by sharing a signup link for the group. Groups are private by design. A member of a group may share any network he or she owns with the group. Only members of the group can view the shared network. Groups provide a level of privacy that is intermediate between the default (a network is visible only to its owner) and public networks (a network is visible to anyone).

A **group owner** is the creator of the group. Any GraphSpace user can create a group by visiting the Groups page and clicking the "Create group" button. The group owner may

- Invite any GraphSpace user that has an account to be a member of their group.
- Remove any member from the group.
- Unshare any graph that has already been shared by the members of the group

A **group member** is a user who is a part of a group. (A group owner is trivially a member of the group.) A group member may

- Share a graph owned by him or her with a group.
- Unshare a previously shared graph.
- Share a layout for a previously shared graph.
- Unshare a previously shared layout.

The Groups page provides access to all the groups owned by the user (by clicking the link "Owner of") as well as the groups of which the user is a member (via the link "Member of"). The user will have the option to delete a group (if the user wowns it) or to unfollow a group (if the user is a member).

Clicking on the name of a group will open a new page that lists all the graphs in that group. This page will also contain a panels to search for nodes and edges in graphs belonging to that group and a panel to search for graphs in that group that match specific tags. If the user owns the group, there will be an interface to add or delete members. If the user is a member of group, the user can only see who are the other members of the group. Notes:

- Users must have GraphSpace accounts and be logged in in order to access group functions.
- By design, groups are private, i.e., a GraphSpace user does not have access to the names of groups of which the user is not a member.
- GraphSpace does not have a mechanism wherein a user may request to become a member of a group. All memberships must be managed by private communication.

CHAPTER 4

GraphSpace Network Model

GraphSpace networks follow a Cytoscape.js supported JSON format that separates the *specification of the network structure* (nodes and edges) from the *description of the visual styles of the nodes and edges* (e.g., colors, widths, shapes, labels).

1. *CYJS Format* - Format is defined by Cytoscape.js for storing network structure and data information.
2. *Stylesheet JSON format* - Format is defined by Cytoscape.js for storing the visual styles of the nodes and edges (e.g., colors, widths, shapes, labels) in CSS-based JSON format.

Graph information in these formats can be exported from or imported to Cytoscape (version 3.1.1 or greater).

4.1 CYJS Format

GraphSpace only supports one of the Cytoscape.js supported JSON formats for storing network structure and data information:

```
{
    "elements":{  // Elements JSON
        "nodes":[ // List of Node Objects
            {
                "data": { // Node Data Attributes
                    "id": ...
                },
                "position": { // Node Position Attributes
                    "x": ...
                    "y": ...
                }
            },
            .
            .
        ],
        "edges":[ // List of Edge Objects
            {
                "data": { // Edge Data Attributes
                    "source": ...,
                    "target": ...
                }
            },
            .
            .
        ]
    },
    "data": {   // Graph Data Attributes
        "title": ...,
        "tags": [..],
        "description": ...
    }
}
```

A Cytoscape (v3.1 or later) user can easily export their graph in the above mentioned JSON format. We call the format as `CYJS format` because the extension of the exportable JSON file from Cytoscape App is `.cyjs`.

CYJS format structure consists of two core parts:

1. *Elements JSON* - An object specifying the list of *nodes* and *edges* in the graph.
2. *Graph Data Attributes* - An object specifying name-value pairs describing the graph.

Note: Any deviation from this format may result in GraphSpace rejecting the graph or problems in rendering the graph.

4.1.1 Elements JSON

The elements JSON object contains two types of lists:

1. *Nodes*: An array of *node objects* describing the nodes in the graph.
2. *Edges*: An array of *edge objects* describing the edges in the graph.

4.1.2 Graph Data Attributes

Cytoscape.js supports a network-level `data` section in the JSON file specifying name-value pairs describing the graph. In this section, GraphSpace gives users the freedom to include any attributes such as `name`, `title`, `description`, and `tags` that best characterize the network. GraphSpace displays the `name` attribute to identify networks in the list that a user can access and in the list that match search results. When the user accesses a specific network, GraphSpace displays the `title` above the layout of the graph and the content of the `description` attribute in the tab called Graph Information. The Graph Information tab for an individual network displays all its attributes and their values.

Graph also allows the users to search for networks with specific attribute values as described here. Cytoscape supports `Graph Data Attributes` for both import and export. The `Graph Data Attributes` are mapped to the Cytoscape network table for a network on import.

Please refer to the *list of graph data attributes treated specially by GraphSpace* to make the best use of GraphSpace's features.

4.1.3 Node Object

```
{
    "data": {     // Node Data Attributes
        "id": ... // unique identifier for the node
    },
    "position": { // Position Attributes
        "x": ...
        "y": ...
    }
}
```

The `Node Object` describes a node in the graph. A `Node Object` typically contains two types of attributes:

1. **Node Data Attributes**:

   Node Data Attributes specify name-value pairs describing the node. Cytoscape requires that each Node Object should have an `id` or `name` data attribute which can uniquely identify the element in the graph. The users can define more data-attributes to describe the node. But, if the attributes are not specially treated by GraphSpace, they will be treated as "opaque". This means that GraphSpace will store or transmit the data without any processing. Please refer to *list of node data attributes treated specially by GraphSpace* to make the best use of GraphSpace's features.

   Note: GraphSpace uses `id` and `name` attributes interchangeably. If both attributes are specified, we overwrite the `name` attribute with the value provided in `id` attribute.

2. **Position Attributes**:

   Position Attributes specify the model position of the node in the graph layout. For example, `{ x:  100, y: 100 }` specifies a point 100 pixels to the right and 100 pixels down from the top-left corner of the viewport at zoom 1 and pan (0,0).

4.1.4 Edge Object

```
{
    "data": {     // Node Data Attributes
        "source": ..., // identifier for the source node
        "target": ...  // identifier for the target node
    }
}
```

The `Edge Object` describes an edge in the graph. An `Edge Object` typically contains a data object which is defined as following:

- **Edge Data Attributes**:

  Edge Data Attributes specify name-value pairs describing the edge. Cytoscape requires that each Edge Object should have `source` and `target` data attributes which can respectively identify the source and target nodes for the edge in the graph. The users can define more data-attributes to describe the edge. But, if the attributes are not specially treated by GraphSpace, they will be treated as "opaque". This means that GraphSpace will store or transmit the data attributes without any processing. Please refer to *list of edge data attributes treated specially by GraphSpace* to make the best use of GraphSpace's features.

4.1.5 Graph Data Attributes Attributes Treated Specially by GraphSpace

GraphSpace gives users the freedom to include any attributes that best characterize the network. If the attributes are not specially treated by GraphSpace, they will be treated as "opaque". This means that GraphSpace will store or transmit the data without any processing.

- required:
  - name – text – Name of the graph. Refer to query semantics section to find how name attribute is used for searching graphs. The maximum allowed length of the name is 256 characters.
  - tags – list of strings – Used to categorize graphs. See the section on organizing graphs using tags for more information.
  - description – text – May be HTML formatted string. May be link to image hosted elsewhere. May simly be a string which contains information such as a legend or significance of the graph. This information is displayed in the tab called Graph Informtaion.
- optional:
  - title – text – Name that is displayed above the layout of the graph.

4.1.6 Node Data Attributes Attributes Treated Specially by GraphSpace

GraphSpace gives users the freedom to include any attributes that best characterize the node. If the attributes are not specially treated by GraphSpace, they will be treated as "opaque". This means that GraphSpace will store or transmit the data without any processing.

- required:
  - id or name – text – A unique id representing the node. If both attributes are specified, we overwrite the name attribute with the value provided in id attribute. GraphSpace uses it to search for nodes with a matching name.
- optional:
  - label – text – The text that is displayed inside of the node unless it is overidden by the content style-attribute in the *stylesheet JSON*. GraphSpace uses it to search for nodes with a matching name.
  - aliases - list of strings - A list of alternative identifiers for the node. GraphSpace uses it to search for nodes with a matching name.
  - popup - text - A string that will be displayed in a popup window when the user clicks the node. This string can be HTML-formatted information, e.g., Gene Ontology annotations and database links for a protein; or types, mechanism, and database sources for an interaction.
  - k - integer - An integer-valued attribute for this node, which denotes a rank. Through this attribute, GraphSpace allows the user to filter nodes and edges in a network visualization.

4.1.7 Edge Data Attributes Attributes Treated Specially by GraphSpace

GraphSpace gives users the freedom to include any attributes that best characterize the edge. If the attributes are not specially treated by GraphSpace, they will be treated as "opaque". This means that GraphSpace will store or transmit the data without any processing.

- required:
  - source – text – The id of the node where the edge is coming from.
  - target – text – The id of the node where edge is going to.
- optional:
  - popup - text - A string that will be displayed in a popup window when the user clicks the edge. This string can be HTML-formatted information, e.g., Gene Ontology annotations and database links for a protein; or types, mechanism, and database sources for an interaction.

  - k - integer - An integer-valued attribute for this edge, which denotes a rank. Through this attribute, GraphSpace allows the user to filter nodes and edges in a network visualization.

4.1.8 Sample JSON

The following example is a CYJS formatted JSON accepted by GraphSpace.

```
{
    "elements": {
        "nodes": [
            {
                "data": {
                    "id":"P4314611",
                    "label": "DCC",
                    "k": 0
                }
            },
            {
                "data": {
                    "id":"P0810711",
                    "label": "This is an example\n of how to use new lines for the␣
↪content of\n a node.",
                    "k": 0
                }
            }
        ],
        "edges": [
            {
                "data": {
                    "source": "P4314611",
                    "target": "P0810711",
                    "k": 0
                }
            }
        ]

    },
    "data": {
        "title": "Graph Name",
        "description": "Description of graph.. can also point to an image hosted␣
↪elsewhere",
        "tags": [
            "tutorial"
        ]
    }
}
```

4.2 Stylesheet JSON Format

Cytoscape and Cytoscape.js are sharing a concept called Style. This is a collection of mappings from data point to network property. Cytoscape can export its Styles into CSS-based Cytoscape.js JSON.

Style in Cytoscape.js follows CSS conventions as closely as possible. In most cases, a property has the same name and behaviour as its corresponding CSS namesake. However, the properties in CSS are not sufficient to specify the style of some parts of the graph. In that case, additional properties are introduced that are unique to Cytoscape.js. For simplicity and ease of use, specificity rules are completely ignored in stylesheets by Cytoscape.js. For a given style property for a given element, the last matching selector wins.

A Cytoscape (v3.1 or later) user can export all Styles into one JSON file from **File | Export | Style** and select Cytoscape.js JSON as its format.

Note: Cytoscape.js does not support all of Cytoscape Network Properties. Those properties will be ignored or simplified when you export to JSON Style file.

4.2.1 Style Properties

Please refer to CytoscapeJS documentation for list of style properties supported by Cytoscape.js. We thank them for the excellent documentation of their framework.

4.2.2 Sample JSON

```
{
    "style": [
        {
            "selector": "node[name='P4314611']",
            "style": {
                "border-color": "#888",
                "text-halign": "center",
                "text-valign": "center",
                "border-width": "2px",
                "height": "50px",
                "width": "50px",
                "shape": "ellipse",
                "background-blacken": "0.1",
                "background-color": "yellow"
            }
        },
        {
            "selector": "node[name='P0810711']",
            "style": {
                "text-halign": "center",
                "text-valign": "center",
                "text-outline-color": "#888",
                "text-outline-width": "2px",
                "border-color": "black",
                "border-width": "5px",
                "height": "150px",
                "shape": "ellipse",
                "color": "black",
                "border-style": "double",
                "text-wrap": "wrap",
                "background-blacken": "0",
                "width": "150px",
                "background-color": "red"
            }
        },
        {
            "selector": "edge[name='P4314611-P0810711']",
            "style": {
                "curve-style": "bezier",
                "line-style": "dotted",
                "width": "12px",
                "line-color": "blue",
                "source-arrow-color": "yellow",
                "target-arrow-shape": "triangle"
            }
        }
    ]
}
```

CHAPTER 5

Uploading Graphs

GraphSpace networks follow the JavaScript Object Notation (JSON) format supported by Cytoscape.js. GraphSpace allows a user to upload a network in three different ways:

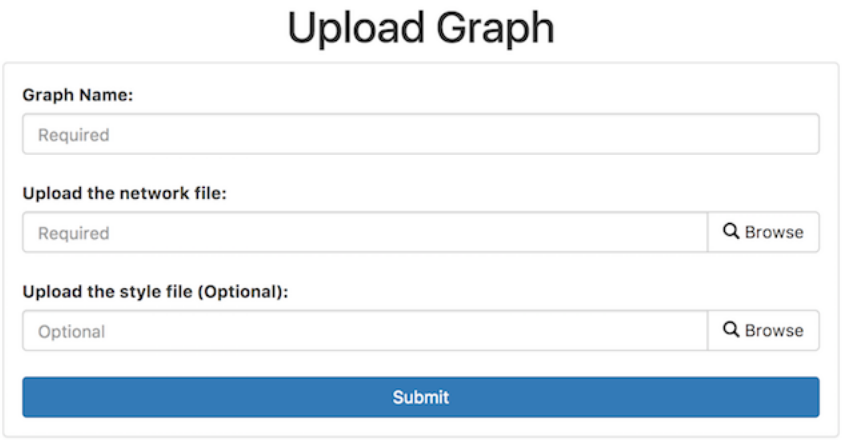
1. Users may create their own JSON files and upload networks one-by one via the *web interface* at http://graphspace.org/upload.
2. Cytoscape users may export their visually-coded networks from Cytoscape and import them directly into GraphSpace. This functionality works as follows. Since v3.1, Cytoscape has supported export of the structure of networks into JSON files and the visual elements of networks in JSON-based style files. GraphSpace can import both these types of files at http://graphspace.org/upload. Moreover, users can import a Cytoscape style file when they are editing the layout of a network in GraphSpace. In the future, we intend to develop a Cytoscape app to make the integration between the two systems seamless.
3. A comprehensive RESTful API and a Python module called "graphspace_python" facilitate bulk uploads of networks. Both the API and the module are easy to integrate into software pipelines. Please refer to the Programmer's Guide for more information.

5.1 Upload Page

GraphSpace provides a *simple web interface* for uploading individual graphs. Once the graph has been uploaded, GraphSpace will provide a unique URL through which the user may interact with the graph represented by the uploaded graph files.

If a user has an account and is logged in, *this interface* will upload the graph directly into the user's account, much like using the REST API. If a user does not have an account or is not logged in, this upload functionality will provide a unique URL through which the user may interact with the graph represented by the uploaded file. **Note: After 30 days, we will delete all graphs that are uploaded for unregistered users of GraphSpace.**

5.1.1 Upload Graph Form



Upload Graph

Graph Name:

Required

Upload the network file:

Required

Upload the style file (Optional):

Optional

Upload Graph

The upload graph form has three input fields:

Graph Name

The name of the graph. GraphSpace allows users to search graphs by their name. It is a required field. The maximum allowed length of the name is 256 characters.

Network File

The network file containing the graphs structure and data information in CYJS Format. It is a required field.

Style File

The network file containing the graphs style information in Stylesheet JSON Format. It is a optional field. If left empty, GraphSpace will use *default style* for displaying the graphs.

Default Graph Style

GraphSpace uses following default style for all graphs. The default style values are used when:

- The user does not upload a *style file* for the graph. OR
- The user uploads the style file which doesn't overrides the default style values.

```
[
    {
        'selector': 'edge',
        'style': {
            'curve-style': 'bezier',
            'line-style': 'solid',
            'line-color': 'black'
        }
    },
    {
        'selector': 'node',
        'style': {
            'content': 'data(label)',
            'shape': 'ellipse',
            'background-color': 'yellow',
            'border-color': '#888',
            'text-halign': 'center',
            'text-valign': 'center'
        }
    }
]
```

CHAPTER 6

Viewing Graphs

When viewing a graph, you will notice several tabs above the graph. Clicking on a tab link will reveal the tab. The Graph Page has following 5 tabs:

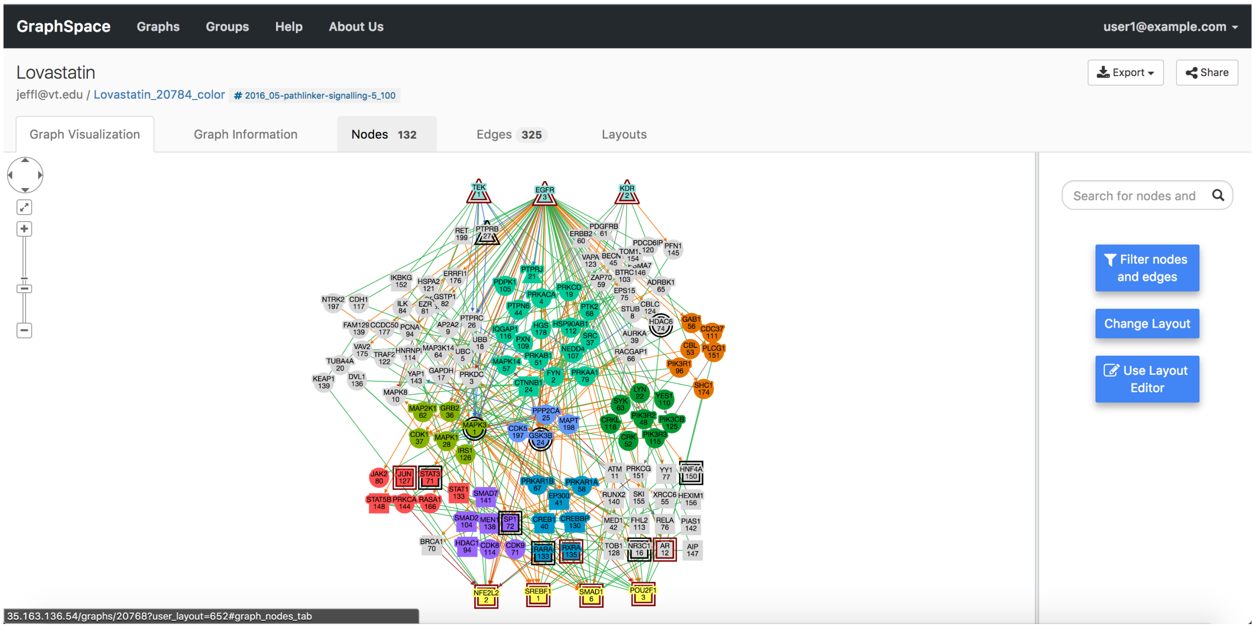
- *Graph Visualization Tab*
- *Graph Information Tab*
- *Edges Tab*
- *Nodes Tab*
- *Layouts Tab*

6.1 Graph Visualization Tab

The Graph Visualization Tab has two sections:

- Left section shows the actual graph.
- Right section shows multiple interaction options.

In this example, the graph has 325 edges and 132 nodes.

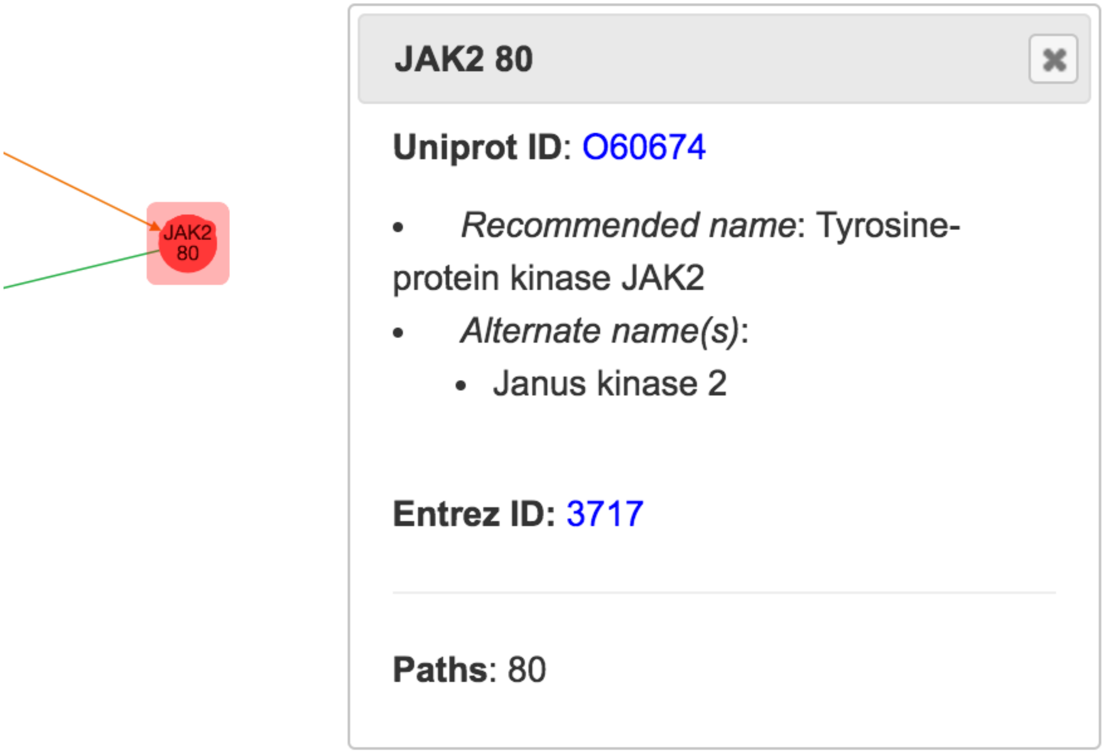


.Graph Visualization Tab

6.1.1 Node and Edge Popups

Each node or edge in a graph may have information embedded in it via the popup attribute. Clicking on a node or edge will *highlight* them. If the clicked node/edge contains popup attribute, popup attribute's value will appear in a pop-up box; otherwise, no popup will be shown. If the popup attribute's value is formatted in HTML, GraphSpace will interpret it appropriately else it will use it as free-text information.

The image below shows an example of the popup shown for a node JAK2.



JAK2 80

Uniprot ID: O60674

- *Recommended name*: Tyrosine-protein kinase JAK2
- *Alternate name(s)*:
  - Janus kinase 2

Entrez ID: 3717

Paths: 80

Graph Visualization Tab

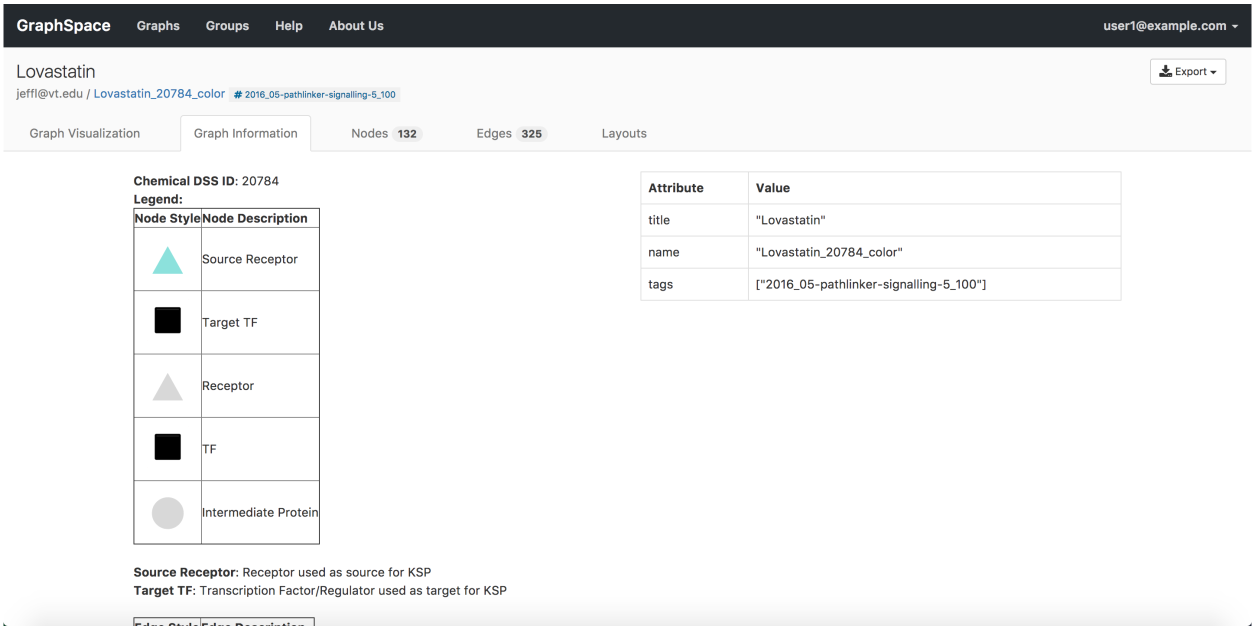
6.1.2 Highlighted Graph Elements

GraphSpace highlights nodes/edges if they are selected. If selected, GraphSpace highlights the elements by adding an overlay around the elements in red color.

6.2 Graph Information Tab

As its name suggests, the `Graph Information` tab displays information about the entire graph, e.g., a legend of node and edge shapes and colors. The `description` attribute in the JSON for the network specifies this content. The `Graph Information` tab for an individual network also displays all its attributes and their values. The user can go to `Graph Information` tab by clicking on the `Graph Information` tab link above the graph.

The image below shows an example of Graph Information Tab when user clicks on the `Graph Information` tab link:



Lovastatin

jeffl@vt.edu / Lovastatin_20784_color # 2016_05-pathlinker-signalling-5_100

Chemical DSS ID: 20784

Legend:

Node Style	Node Description
▲	Source Receptor
■	Target TF
▲	Receptor
■	TF
●	Intermediate Protein

Source Receptor: Receptor used as source for KSP

Target TF: Transcription Factor/Regulator used as target for KSP

Attribute	Value
title	"Lovastatin"
name	"Lovastatin_20784_color"
tags	["2016_05-pathlinker-signalling-5_100"]

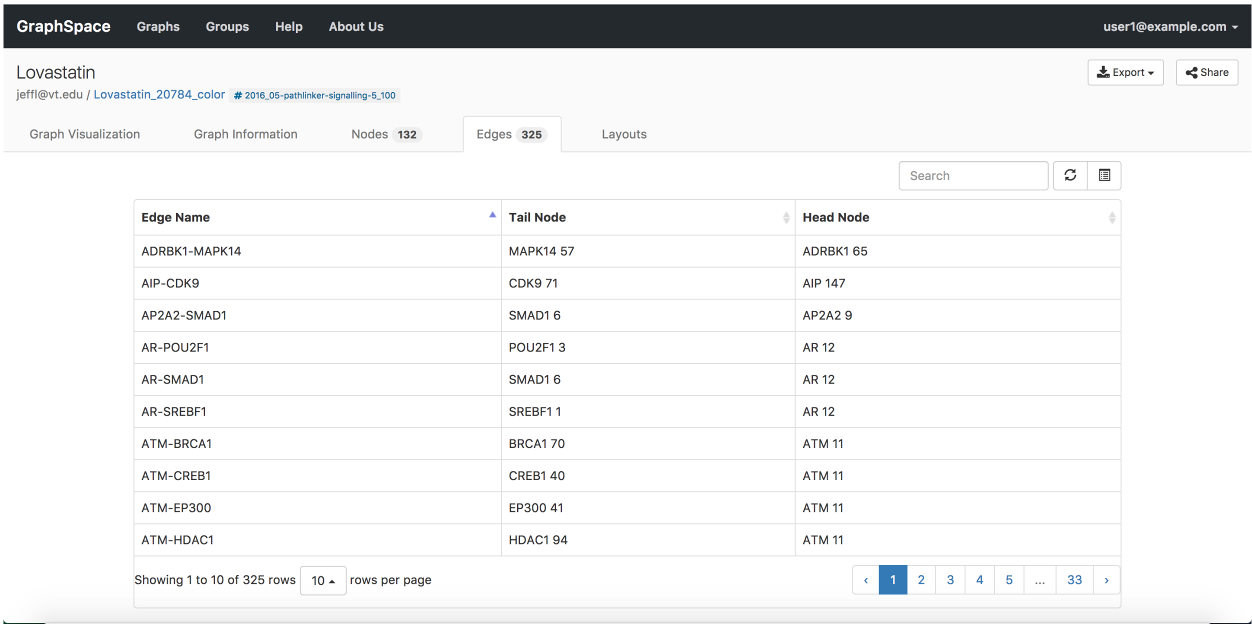
Graph Information Tab Image

6.3 Edges Tab

As its name suggests, the Edges tab displays information about the edges in a table format. The table contains following columns:

1. **Edge Name** - Name of the edge.
2. **Tail Node** - Label of the tail node.
3. **Head Node** - Label of the head node.

The image below shows an example of Edges Tab when a user clicks on the Edges tab link:



Lovastatin

jeffl@vt.edu / Lovastatin_20784_color # 2016_05-pathlinker-signalling-5_100

Edge Name	Tail Node	Head Node
ADRBK1-MAPK14	MAPK14 57	ADRBK1 65
AIP-CDK9	CDK9 71	AIP 147
AP2A2-SMAD1	SMAD1 6	AP2A2 9
AR-POU2F1	POU2F1 3	AR 12
AR-SMAD1	SMAD1 6	AR 12
AR-SREBF1	SREBF1 1	AR 12
ATM-BRCA1	BRCA1 70	ATM 11
ATM-CREB1	CREB1 40	ATM 11
ATM-EP300	EP300 41	ATM 11
ATM-HDAC1	HDAC1 94	ATM 11

Showing 1 to 10 of 325 rows 10 rows per page

‹ 1 2 3 4 5 ... 33 ›

Graph

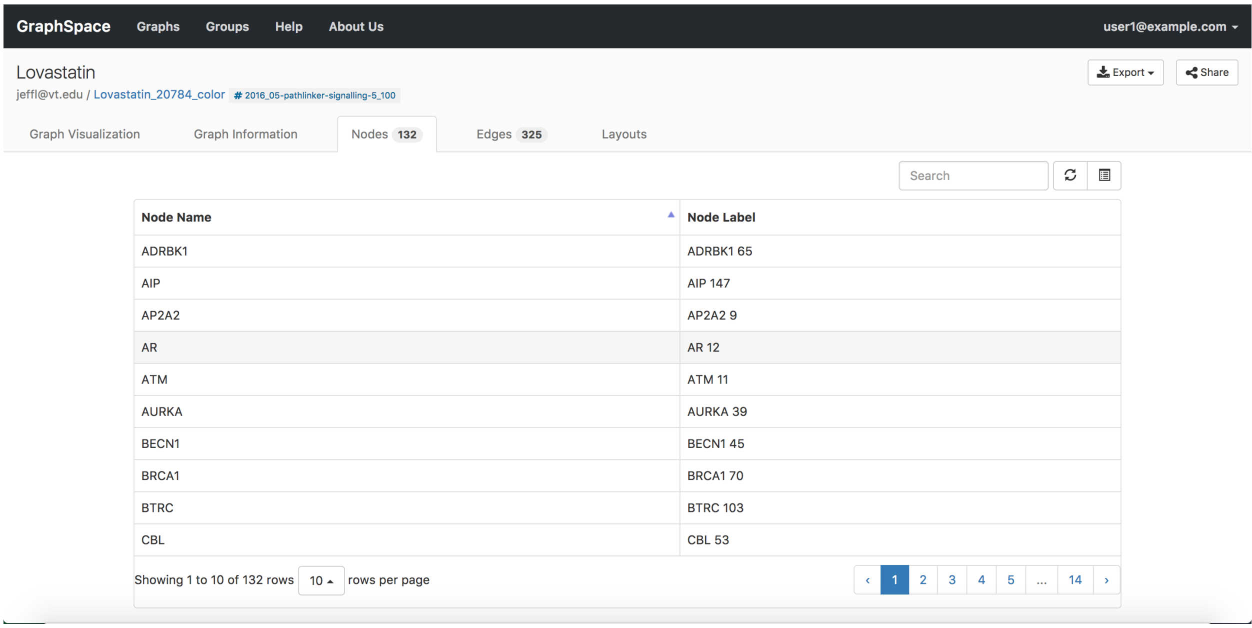
Edges Tab Image

6.4 Nodes Tab

As its name suggests, the `Nodes` tab displays information about the nodes in a table format. The table contains following columns:

1. **Node Name** - Name of the node.
2. **Node Label** - Label for the node.

The image below shows an example of Nodes Tab when user clicks on the `Nodes` tab link:



Node Name	Node Label
ADRBK1	ADRBK1 65
AIP	AIP 147
AP2A2	AP2A2 9
AR	AR 12
ATM	ATM 11
AURKA	AURKA 39
BECN1	BECN1 45
BRCA1	BRCA1 70
BTRC	BTRC 103
CBL	CBL 53

Showing 1 to 10 of 132 rows 10 rows per page

Graph Nodes Tab Image

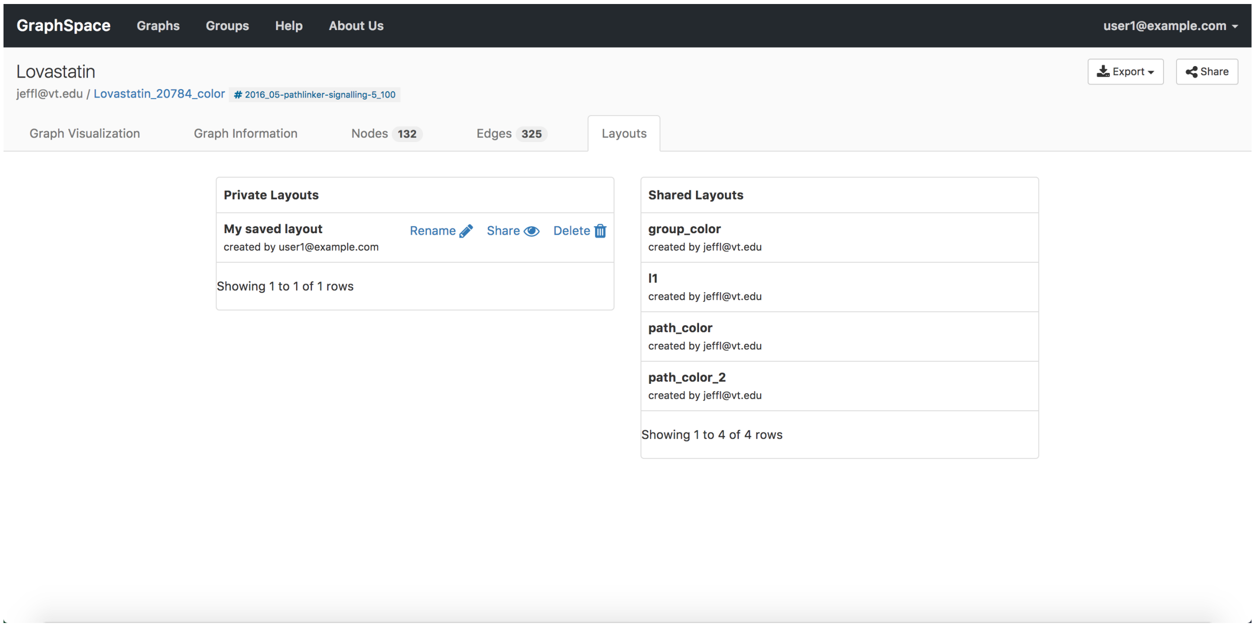
6.5 Layouts Tab

As its name suggests, the `Layouts` tab displays information about the layouts in a table format. The tab contains two types of layouts:

- **Private Layouts** - The user has created them in earlier sessions by manually modifying the positions of nodes and edges. But the user has not shared them with any other user.
- **Shared Layouts** - These layouts were created by the user who has access to this graph and shared the layout with other users who have access to this graph.

The icons next to each layout name allow the user to (i) change its name, (ii) share/unshare it with other users who have access to this graph, (iii) delete this layout.

The image below shows an example of Layouts Tab when user clicks on the `Layouts` tab link. In this example, the user has created 1 private layout and has access to four shared layouts.



Lovastatin

jeffl@vt.edu / Lovastatin_20784_color # 2016_05-pathlinker-signalling-5_100

Private Layouts

My saved layout
created by user1@example.com

Showing 1 to 1 of 1 rows

Shared Layouts

group_color
created by jeffl@vt.edu

l1
created by jeffl@vt.edu

path_color
created by jeffl@vt.edu

path_color_2
created by jeffl@vt.edu

Showing 1 to 4 of 4 rows

.Graph Layouts Tab Image

CHAPTER 7

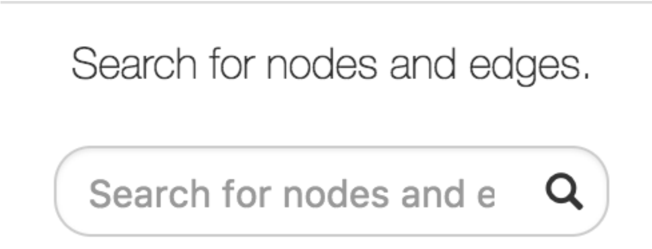
Interacting with Graphs

In this section, we examine different ways to interact with an individual network on its page. The information that appears in following examples must be included in the JSON files that are uploaded by the network owner, as described in the Network Model section. An individual network page is designed to access features like:

- *Searching within a graph*
- *Exporting a graph*
- *Changing the graph layout*
- *Setting default layout*
- *Filtering nodes and edges*

7.1 Search

This feature provides a search bar that allows the user to find nodes/edges that match the search terms. See Search Query Semantics section for more information on query semantics. The matching nodes or edges are highlighted automatically as you type in the query in the search bar.



Search for nodes and edges.

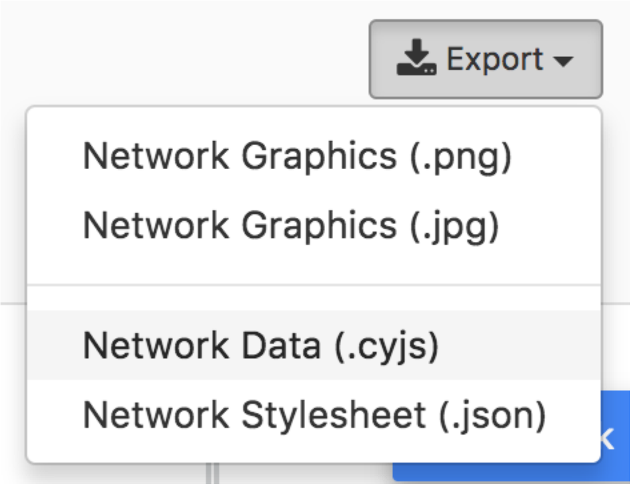
Search Bar

7.2 Export

GraphSpace allows users to export a graph in the following formats:

1. **Network Graphics (.png)** - Export the current graph view as a PNG image.
2. **Network Graphics (.jpg)** - Export the current graph view as a JPEG image.
3. **Network Data (.png)** - Export the graph in CYJS format.
4. **Network Graphics (.png)** - Export the graph stylesheet in Stylesheet JSON format.

GraphSpace does not support any other export formats since it relies on Cytoscape.js for this functionality, which implements only export to PNG, JPG and JSON format.



Search Bar

7.3 Change Layout

Layouts provide a powerful means to organize nodes within a network. GraphSpace allows users to change layout using the following steps:

1. Click on the `Change Layout` button to view available layout options.
2. The `Change Layout` panel provides two alternatives: - **Select Layout Algorithm** - This section lists all the automatic network layout algorithms supported by GraphSpace through its use of Cytoscape.js. - **Select Saved Layout** - This section lists layouts saved by the user. The user has created them in earlier sessions by using the layout editor or manually modifying the positions of nodes and edges created by some automatic layout algorithm and saving the layout.
3. Click on a layout option to change the current layout.

In the following example, the user selects to view the layout titled `manual-top-to-bottom`. To create this layout, the user manually moved nodes so that extracellular ligands and receptors (triangles) appear on the top while transcription factors (rectangles) appear at the bottom. The user can click on the `Back` button to return to the main menu.

Change Layout

7.4 Set Default Layout

Default layout is the layout which is used by default, whenever a user visits the page for a graph. A layout can be set as a default layout for a graph only if it is shared with other users who have access to the graph. Default layout for a graph can only be set by the owner of the graph.

If a layout is shared with other users who have access to the graph, user can click on the `Set as Default Layout` button to set the layout as the default layout for the graph.

If a layout is set as default layout, user can click on the `Remove as Default Layout` button to unset the layout as the default layout for the graph.

7.5 Filter nodes and edges

Graph algorithms may output networks where nodes and edges can be ranked, e.g., by path index or by weight/score. GraphSpace allows each node and edge to have an integer-valued data-attribute called `k` that specifies the rank of the node or the edge. For any network that contains this attribute (and only for such networks), GraphSpace displays the "Filter nodes and edges" panel with a `Current rank` slider. Changing the value in the `Current rank` slider hides all nodes and edges whose k values are less than or equal to the value in the slider. The possible values in this slider range from 1 to the maximum value of `k` in the graph. This interface element allows the user to unveil the network gradually in real time and gain intuition about how the network expands or contracts as this threshold changes.

The following example shows a user stepping through a graph using this slider.

Filter nodes and edges

CHAPTER 8

Editing Layouts

GraphSpace includes a powerful and intuitive layout editor that allows the user to quickly select nodes with specific properties (e.g., color and shape), to organize the selected nodes in defined shapes (e.g., a horizontal line or a filled circle), and to adjust the spacing between them. In addition, the user can manually move nodes into any configuration. The layout editor also allows users to manually change the visual properties of nodes and edges. Finally, the user can save the layout, make this layout appear by default, and share it with other viewers of the network. Only users that have access (i.e., owned or shared via a group) can manipulate a network layout.

The user can activate this functionality by clicking the `Use Layout Editor` button on the page for an individual network.

8.1 Start Tour

The `Start Tour` button walks the user through all the features provided in the tool pallette. The Exit layout editor button gives a user the option to save the current layout and/or go back to the original view of the current graph. In the layout editor, the user may wield the tool pallette on the right hand side to quickly re-arrange the structure of the graph.

8.2 Edit selected nodes

A GraphSpace user can change visual properties of a node by following the given steps:

1. Select one or more nodes either by *color* or *shape* or using the gestures supported by Cytoscape.js.
2. Click on the `Edit selected nodes` button to launch the *node editor*.
3. Change the visual properties of the selected nodes.
4. Click on `OK` if you want to save the changes else click on the `Cancel` button to discard the changes.

Once the user enters the layout editor, they can select nodes using the gestures supported by Cytoscape.js (e.g., clicking to select one node, using the `Shift` button on the keyboard and the mouse to select multiple nodes) or the tool palette for selecting nodes in the layout editor. In this example, the user selects the node called `CTNNB1` and opens the node editor by clicking on the `Edit selected nodes` button. The user may select more than one node. At this point, the user can apply these changes by clicking on the `OK` button or discard them by clicking on the `Cancel` button.

editing nodes

8.2.1 Node Editor

GraphSpace that allows a user to select one or more nodes based on *color* and/or *shape*, and to then change their visual properties using an easy-to-use interface. The current interface allows user to change the following properties:

1. **Color**: The background color of the node.
2. **Shape**: The shape of the node. GraphSpace currently supports following shapes - rectangle, roundrectangle, ellipse, triangle, pentagon, hexagon, heptagon, octagon, star, diamond, vee, or rhomboid.
3. **Width**: The width of the node in pixels.
4. **Height**: The height of the node in pixels.
5. **Label**: The label inside the node.. In addition to specifying the value of a property outright, the developer may also use a mapper to dynamically specify the property value. For example, data(k) would map a property to the value in an element's `k` data-attribute. This feature can used for visualizing node annotations in the layouts.

The user can apply the changes by clicking on the `OK` button or discard them by clicking on the `Cancel` button in the node editor.

The following image shows the visual properties of a node before and after the editing using the node editor. In this example, user has made the following changes: (i) background color from green to pink (ii) shape from rectangle to ellipse (iii) width and height from 45px to 100px (iv) label from 'CTNNB1' to 'Updated CTNNB1'.

When a user launches the node editor, it displays the values of style properties of the selected node if only one node is selected. Whereas if more than one node is selected, the node editor will display the values of style properties as blank. The following image shows an example of node editor when i) only one node is selected ii) more than one node is selected.

8.3 Edit selected edges

A GraphSpace user can change visual properties of an edge by following the given steps:

1. Select one or more edges either by using the gestures supported by Cytoscape.js.
2. Click on the `Edit selected edges` button to launch the *edge editor*.
3. Change the visual properties of the selected edges.
4. Click on `OK` if you want to save the changes else click on `Cancel` button to cancel the changes.

Once the user enters the layout editor, they can select edges using the gestures supported by Cytoscape.js (e.g., clicking to select one edge, using the `Shift` button on the keyboard and the mouse to select multiple nodes). In this example, user selects the edge between `LRP6` and `DAB2` by clicking on it and opens the edge editor by clicking on `Edit selected edges` button.. The user may select more than one edge.

editing edges

8.3.1 Edge Editor

GraphSpace that allows a user to select one or more edges based on *color* and/or *shape*, and to then change their visual/style properties using an easy-to-use interface. The current interface allows user to change the following properties:

1. **Line Color**: The colour of the edge.
2. **Line Style**: The style of the edge. GraphSpace currently supports following line styles - solid, dotted, or dashed.
3. **Width**: The width of an edge in pixels.
4. **Source Arrow Shape**: The shape of the edges source arrow
5. **Target Arrow Shape**: The shape of the edges target arrow. GraphSpace currently suuports following arrow styles - tee, triangle, triangle-tee, triangle-backcurve, square, circle, diamond, or none

The user can apply the changes by clicking on the `OK` button or discard them by clicking on the `Cancel` button in the edge editor.

The following image shows the visual properties of an edge before and after the editing using the edge editor. In this example, user has made the following changes: (i) line color from grey to red (ii) line style from solid to dashed (iii) width from 3px to 10px.

When a user launches the edge editor, it displays the current style values of the selected edge if only one edge is selected. Whereas if more than one edge is selected, the node editor will display the values of style properties as blank.

8.4 Show node labels

Unchecking the `Show node labels` checkbox takes a user of GraphSpace to a simplified view of the graph that hides node names.

toggle node labels

8.5 Select by shape

The `Select by shape` section allows the user to choose nodes in the graph based on nodes `shape` style property. For example, if a user wishes to select all nodes that are rectangles or ellipses, he/she will select both "Ellipse" and "Rectangle" in the tool pallete.

select nodes by shape

8.6 Select by color

The `Select by color` section allows the user to choose nodes in the graph based on nodes `background-color` style property. For example, if a user wishes to select all nodes that are yellow or green, he/she will select both yellow and green.

select nodes by color

8.7 Unselect All Nodes

A user may click on the `Unselect All Nodes` button to reset all selections.

8.8 Undo and Redo

The Undo and Redo buttons allow the user to undo or repeat all actions, including selection, editing and the arrange functions described below.

undo redo changes

8.9 Arrange nodes

The `Arrange nodes` section allows a user to arrange all selected nodes into regions of different shapes. The following animation shows the arrangements of selected nodes that users may construct by clicking different buttons in this section.

CHAPTER 9

Searching Graphs

The search interface in GraphSpace allows a user to search for networks that have a specific attribute or tag and contain one or more nodes using *simple syntax*. When the user visits a matching network, GraphSpace highlights the nodes that match the search term. An identical search interface on the page for an individual network enables the user to refine the query. This interface also enables the user to search for specific edges within the network. GraphSpace allows the user to record the URL to the network with the search terms, for sharing or for further study.

9.1 Query Semantics

GraphSpace supports three types of search terms:

1. a single string, e.g., `wnt`. In this case, GraphSpace will return a network (a) if its `name` attribute contains the query as a substring or (b) if any node in the network has a `name`, `label`, or `aliases` attribute that contains the query as a substring.
2. two strings separated by a colon, e.g., `name:wnt`. Here, GraphSpace will return a network if the `data` section of its JSON representation contains an attribute called `name` whose value contains `wnt` as a substring.
3. two strings separated by two colons, e.g., `wnt::fzd`: This type of search term is only available when a user is searching a specific network. GraphSpace will highlight every edge that connects a node that matches `wnt` to a node that matches `fzd`. This search ignores the direction of the edge.

All searches are case-insensitive. A user may specify more than one search term. When the user searches all networks, GraphSpace returns only those networks that match all the search terms. When the user is visualizing an individual network and searching within it, GraphSpace highlights nodes and edges that match any search term

9.2 Searching within Multiple Graphs

The user can search for search for networks that have a specific attribute or tag and contain one or more nodes using simple syntax on Graphs Page by following the given steps:

1. Enter the name of the graph/node or specific network attribute mapping you are searching for in the search bar.

2. Press `Enter` key or click on the `Search` button.

In this example, the user searches for the list for graphs that contain the protein (node) `CTNNB1` (the symbol for β-catenin, a transcriptional regulator in the Wnt signaling pathway). The reduced list of graphs are the graphs where protein (node) `name`, `label` or `aliases` contain `CTNNB1` as a substring. The match is case-insenstive. In the following example, There are six graphs owned by the user and thirty-two public graphs that contain this protein. Each link in the `Graph Name` column will take the user to a specific graph with the search term highlighted. In this example, the user clicks on the graph with the name `KEGG-Wnt-signaling-pathway` and reaches the graph for the Wnt pathway with the searched node highlighted.

Searching within Multiple Graphs

9.3 Searching within a Single Graph

The user can search for node or edges within a given graph on GraphSpace by following the given steps:

1. Enter the name of the node or an edge you are searching for in the search bar.
2. The nodes or edges are highlighted automatically as you type in the name of the node or edge in the search bar.

In the following example, the user searches for the graph for two proteins (nodes) `CTNNB1` and `WNT` using the query `ctnnb1, wnt`. This search query highlights the proteins where protein (node) `name`, `label` or `aliases` contains `CTNNB1` or `WNT` as a substring (case-insensitive). In the following example, the graph contains four nodes which match the given query.

Searching nodes within a Single Graph

In the following example, the user searches the graph for edges from `Wnt` to `Fzd` using the query `Wnt::Fzd`. GraphSpace highlights any edge whose tail node matches `wnt` and whose head node matches `fzd`. In the following example, the graph contains three edges which match the given query.

Searching edges within a Single Graph

CHAPTER 10

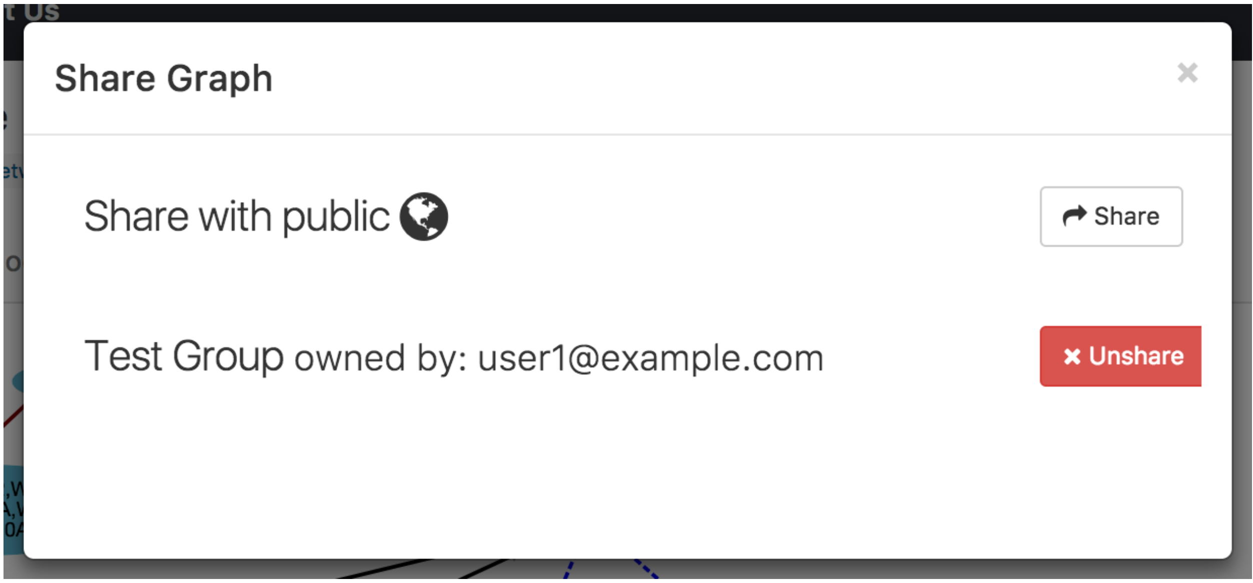
Sharing Graphs

A user may share a graph and its layouts with groups to which the user belongs.

10.1 Sharing graphs with group(s)

There are two ways to share a specific graph owned by the user: the REST API and the `Share` Button. A user may share a graph with multiple groups. In order to share a graph, the user must own it. In addition, the user must be a member or the owner of the group he/she wants to share the graph with. Sharing a graph allows all the members in the group to access the graph. Un-sharing a graph from a graph means that no one else in the group will be able to access that graph anymore.

This image shows the sharing panel displayed when a user clicks on the `Share` button on the Graph Page. In the following example, the graph is only shared with a group name 'Test Group' created by user1@example.com. The user can click on the Share button next to `Share with public` text to share the graph with everyone.



Share Graph

Share with public

Test Group owned by: user1@example.com

Share layout panel

10.2 Sharing layouts with group(s)

Similar to a graph, a layout may also be shared with a group. In order for a layout to be shared with a group, the graph must already be shared with the group. Currently, the only way to share a layout is through the Layouts tab. Initially, only the creator of a saved layout is allowed to access it. Note that a layout may be created by a user who is not the owner of the graph, as long as the layout creator can access the graph (because the owner has shared it). Sharing a layout changes its access as follows:

1. Public graphs

   The layout is also publicly available, i.e., to all users of GraphSpace.

2. Shared graphs

   The layout is accessible to every user who is a member of a group with which the graph is shared, as long as the layout creater is also a member of that group.

GraphSpace](http://www.graphspace.org) allows users to share a layout using the following steps:

1. Click on the `Layouts` link above the graph.
2. The layouts panel shows two types of layouts:
   - **Private Layouts** - The user has created them in earlier sessions by manually modifying the positions of nodes and edges. But the user has not shared them with any other user.
   - **Shared Layouts** - These layouts were created by the user who has access to this graph and shared the layout with other users who have access to this graph.
3. Click on the `Share` link next to the layout name of a private layout you want to share with other users who have access to this graph.

Note: The icons next to each layout name allow the user to (i) change its name, (ii) share it with other users who have access to this graph, (iii) delete this layout.

Share Layout

CHAPTER 11

Organizing Graphs Using Tags

GraphSpace uses tags as a mechanism for grouping a set of graphs, e.g., all the graphs that a user may desire to make public upon the publication of a paper. A graph may have any number of tags. GraphSpace includes an interface to search for networks that match one or more tags. Tags, which organize graphs, provide a system that is orthogonal to groups, which organize collaborators.

Note: Tags provide a level of organization that are orthogonal to groups. Groups are intended to organize collaborating users of GraphSpace while tags are used to collect graphs.

CHAPTER 12

Programmers Guide

12.1 What is GraphSpace REST API ?

The GraphSpace REST API provides endpoints for entities such as graphs, layouts, and groups that allow developers to interact with the GraphSpace website remotely by sending and receiving JSON objects. This API enables developers to create, read, and update GraphSpace content from client-side JavaScript or from applications written in any language. After a network is uploaded, the API allows the network owner to modify, delete, and share it. The API also allows a user to access several group management features available through the web interface. For example, a user can create or remove a group, add or remove members, obtain a list of groups he or she belongs to, and get information such as the membership on a group.

```
Note: In order to fully utilize the features of GraphSpace REST API, you must have an␣
↪account on GraphSpace.
```

12.1.1 Why use GraphSpace REST API ?

The GraphSpace REST API makes it easier than ever to use GraphSpace in new and exciting ways, such as creating external applications on top of GraphSpace. For example,

- Create a Cytoscape plugin which will allow users to transfer networks between GraphSpace and Cytoscape.
- Users can also automate the way they upload graphs to GraphSpace.

The scope of what can be done with the GraphSpace REST API is only limited by our imagination. Overall, if a user want a structured, extensible, and simple way to get data in and out of GraphSpace over HTTP, they should probably use the GraphSpace REST API.

12.1.2 Base URL

All URLs referenced in the documentation have the following base:

> http://www.graphspace.org/api/v1/

12.1.3 API Reference

12.2 HTTP Status Codes

The GraphSpace API attempts to return appropriate HTTP status codes for every request.

12.2.1 Success Codes

- ***200:*** Your request has succeeded.
- ***201:*** Your request has been fulfilled and resulted in a new resource being created.

12.2.2 Error Codes

- ***400:*** Bad Request! The server cannot or will not process your request due to an apparent client error (e.g., malformed request syntax, too large size, invalid request message framing, or deceptive request routing).
- ***401:*** Unauthenticated! Either your authentication token is missing or invalid, or you are not allowed to access the content provided by the requested URL.
- ***403:*** Unauthorized! You are not authorized to access this resource, create an account and contact resource's owner for permission to access this resource.
- ***405:*** Method Not Allowed! Your request method is not supported by the resource. For example, using GET on a form which requires data to be presented via POST, or using PUT on a read-only resource.
- ***1000:*** User with the provided email id already exists!
- ***1003:*** Your Username or Password is not recognized.
- ***1006:*** `is_public` is required to be set to True when `owner_email` and `member_email` are not provided.
- ***1007:*** You are not authorized to access private graphs created by other users.
- ***1008:*** You are not allowed to create a graph for other users.
- ***1009:*** Your graph ID is missing.
- ***1010:*** You are not authorized to access groups you aren't part of. Set `owner_email` or `member_email` to your email.
- ***1011:*** You are not allowed to create a group for other users.
- ***1012:*** You are not authorized to access layouts which are not shared. Set `owner_email` to your email or `is_shared` to 1.
- ***1013:*** Cannot create the layout with your provided owner email.
- ***1014:*** Layout with the provided name already exists.

12.3 graphspace-python

The GraphSpace software also includes a simple yet powerful Python library called `graphspace python` that allows a user to rapidly construct a network, add nodes and edges, modify their visual styles, and then upload the network, all within tens of lines of code. Moreover, the user need not know the details of the REST API to use this module. It is very easy to integrate this library into a user's software pipeline.

12.3.1 Installation

Install graphspace_python from PyPI using:

```
pip install graphspace_python
```

12.3.2 Usage

Please refer to `graphspace_python` package's documentation to learn how to use it.

12.4 GraphSpace REST APIs using the Postman app

12.4.1 This documentation is based on Sandeep Mahapatra's blog post in the 2017 GSoc.

```
Note: In order to fully utilize the features of GraphSpace REST API, you must␣
↪have an account on GraphSpace.
```

Postman is a Google Chrome app for interacting with HTTP APIs. It provides a friendly GUI for constructing requests and reading responses. Postman makes it easy to test, develop and document APIs by allowing users to quickly put together both simple and complex HTTP requests.

12.4.2 Postman Installation

Postman is available as a native app (recommended) for Mac / Windows / Linux, and as a Chrome App. The Postman Chrome app can only run on the Chrome browser. To use the Postman Chrome app, you need to:

- Install Google Chrome: Install Chrome.
- If you already have Chrome installed, head over to Postman's page on the Chrome Webstore, and click 'Add to Chrome'.
- After the download is complete, launch the app.

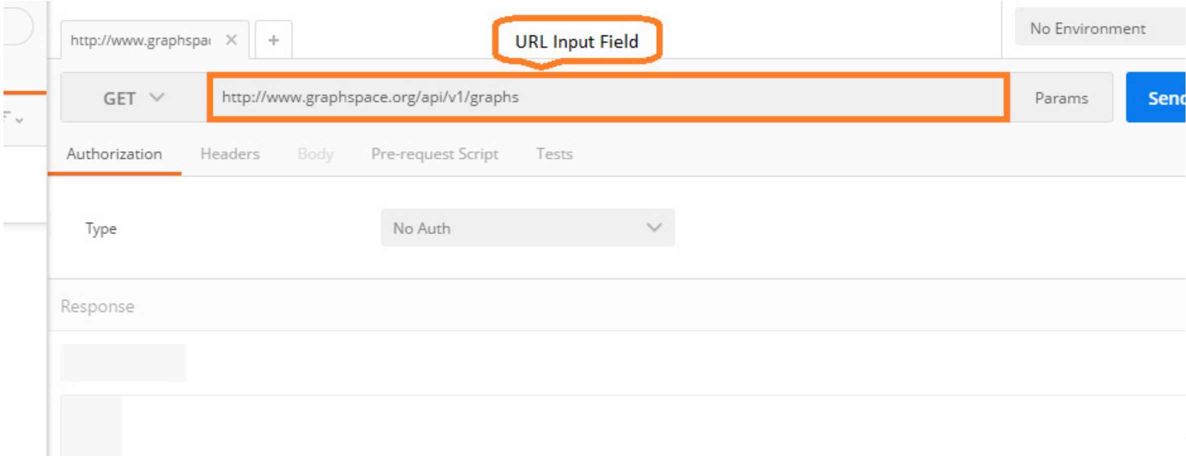
12.4.3 Using Postman for GraphSpace REST API

The GraphSpace REST APIs have the base URL http://www.graphspace.org/api/v1/. There are many endpoints defined under this base URL (the documentation of which can be found here), but to learn and understand the usage of GraphSpace REST APIs through Postman, we would be considering only the /graphs endpoint for GET and POST request.

- The GET /graphs request fetches a list of graphs from GraphSpace matching the query parameters.
- The POST /graphs request creates a graph in GraphSpace.

12.4.4 GET /graphs

- The URL is the first thing that we would be setting for a request. We will set the URL to http://www.graphspace.org/api/v1/graphs.

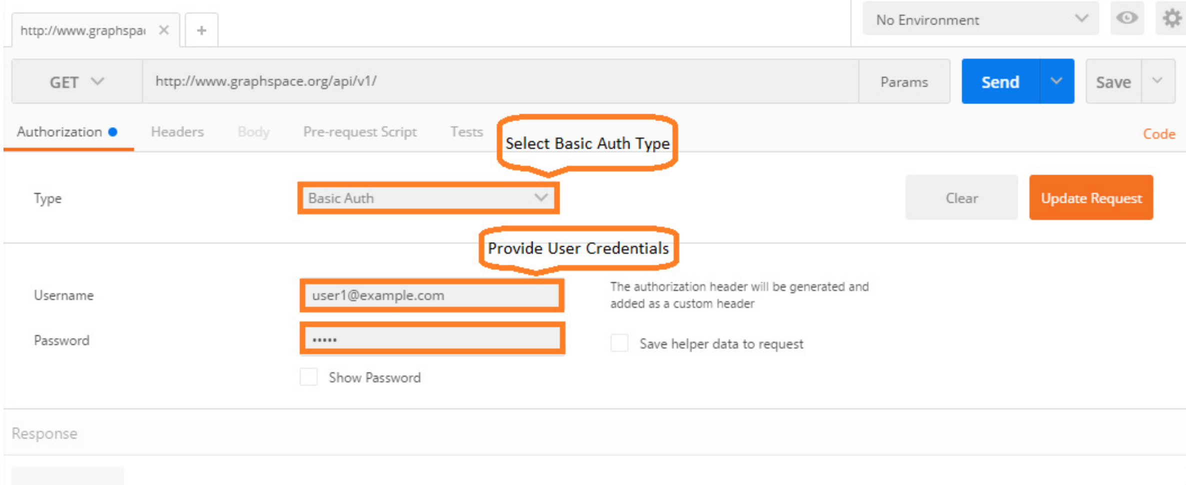


GET http://www.graphspace.org/api/v1/graphs

Type: No Auth

Rest API get

- Provide Authorization: Select 'Basic Auth' from Authorization type drop-down. Enter the username and password and click on 'Update Request'.



GET http://www.graphspace.org/api/v1/

Type: Basic Auth

Username: user1@example.com

Password: •••••

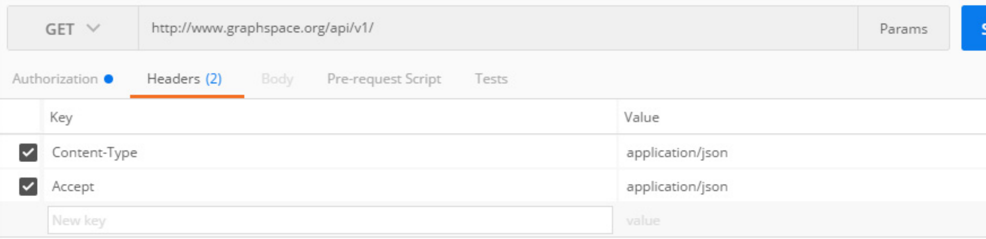
□ Show Password

The authorization header will be generated and added as a custom header

□ Save helper data to request

Rest API get 2

- Set Header: Add the following key value pairs, `Content-Type:application/json` and `Accept:application/json`.



GET http://www.graphspace.org/api/v1/

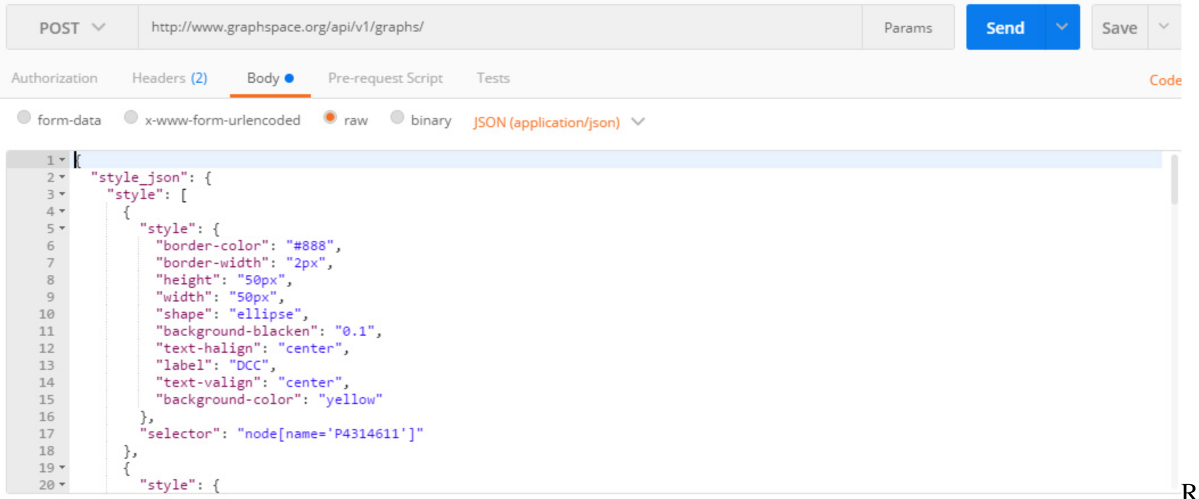
	Key	Value
☑	Content-Type	application/json
☑	Accept	application/json
	New key	value

API get 3

- Select Method: Changing the method is straightforward. Just select the method from the select control. We will use GET method here.
- Add URL Params: Clicking on the URL Params button will open up the key-value editor for entering URL parameters. The details of the URL Params for /graphs endpoint can be found in the documentation.
- Click on the Send button to the send the request. A list of graphs matching the query parameters will be received in the response.

12.4.5 POST /graphs

- The initial steps of setting URL, Authorization and Header are performed.
- Change Method to POST.
- Set Request Body: Click on Body to open the request body editor. Select raw request from the choices and JSON(application/json) from the drop-down. Enter the json data for the graph to be created in the editor. The details regarding the properties of the json graph body can be found in the documentation.



POST http://www.graphspace.org/api/v1/graphs/

```
{
  "style_json": {
    "style": [
      {
        "style": {
          "border-color": "#888",
          "border-width": "2px",
          "height": "50px",
          "width": "50px",
          "shape": "ellipse",
          "background-blacken": "0.1",
          "text-halign": "center",
          "label": "DCC",
          "text-valign": "center",
          "background-color": "yellow"
        },
        "selector": "node[name='P4314611']"
      },
      {
        "style": {
```

REST API post 1

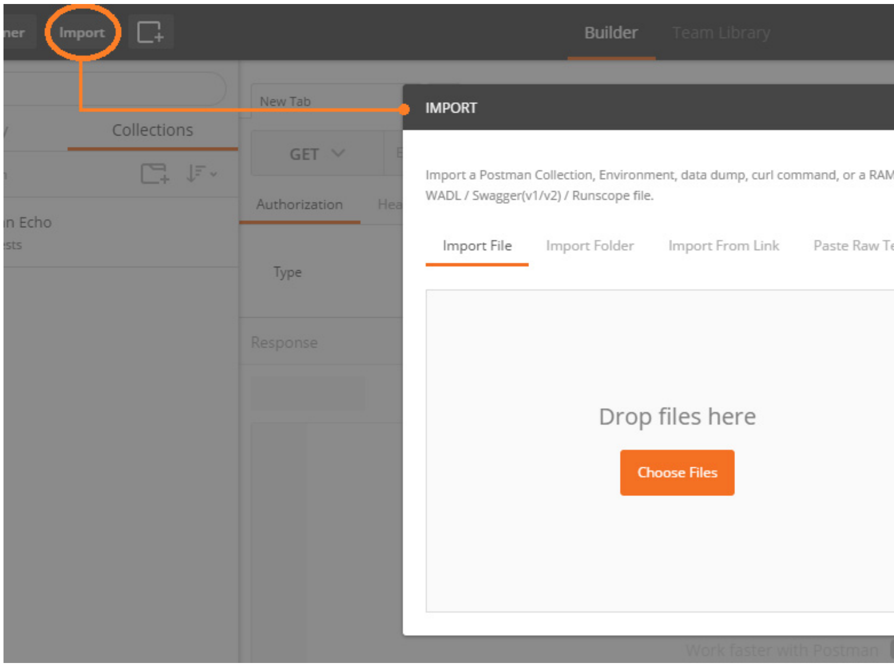
- Click on the Send button to the send the request. A new graph object will be created and returned in the response.

12.4.6 Postman Collection

A collection lets you group individual requests together. These requests can be further organized into folders to accurately mirror our API. Requests can also store sample responses when saved in a collection. You can add metadata like name and description too so that all the information that a developer needs to use your API is available easily. Collections can be exported as JSON files. Exporting a collection also saves the Authorization details. Hence, it is advised to remove the Authorization details from the Header before exporting.

For quick use of the GraphSpace REST APIs or if you are stuck somewhere and you want reference, you can download the collection of the APIs here. The collection has details regarding the API endpoints like params and body properties. Importing steps:

- Click Import button in the top menu.



Import a Postman Collection, Environment, data dump, curl command, or a RAM
WADL / Swagger(v1/v2) / Runscope file.

- Choose the Import File in the pop up window.
  man collection
- Provide the Authorization details for the imported requests (as Authorization details have been removed for security concern)

CHAPTER 13

Hosting GraphSpace

13.1 Steps to deploy GraphSpace on a server

We recommended downloading the latest release. Alternatively, you can download the latest code directly from the master branch. Then follow the steps to run GraphSpace on Apache. These include steps to install the correct version of python, setup postgreSQL, virtualenv, bower, ElasticSearch, etc.

> **Note**: If you have an existing deployment of GraphSpace, be careful when running alembic migration scripts. Understand how to use alembic and alembic commands. Migrations are not needed if you are deploying GraphSpace for the first time. `alembic upgrade head` will run migrations if the current *head* is the downgrade version of the latest migration script.

Useful Links

- How to configure Apache to run Django website
- How to use Django with Apache and mod-wsgi
- How to run migrations with Alembic

CHAPTER 14

Release Notes

14.1 GraphSpace 2.0

14.1.1 Highlights

- We have added features in the layout editor that allow a user to select nodes and edges based on color and shape in order to edit their visual properties. This functionality is available in conjunction with the features that allowed a user to arrange the positions of the nodes in a variety of shapes.
- We have brought the JSON format accepted by GraphSpace in line with Cytoscape.js (this JSON format is also compatible with Cytoscape).
  - We separate the structure of the network (nodes and edges) from the description of the visual styles of the nodes and edges.
  - Cytoscape.js supports a network-level `data` section in the JSON file specifying the network structure. We allow users to include network-specific attributes in this section, e.g., `PMID`, `authors`, and `organism`.
  - Cytoscape.js supports a `data` section within each node. Currently, GraphSpace recognizes an attribute called \aliases" in this section, through which the network creator can specify a list of aliases for the node.
  - GraphSpace now supports CSS-based Cytoscape.js JSON files for specifying the style of the graph. GraphSpace users can import these files either in the `Upload` section or the `Layout Editor` section.
  - We have deprecated support for the JSON format supported only by GraphSpace.
- We have streamlined the search interface and made it more efficient. We have added support to search for networks that match a specific attribute (key-value pair) in addition to nodes. The search also includes the \aliases" attribute of nodes. We have disabled support for `Exact` searches: GraphSpace now supports only case-insensitive and substring searches.
- Concomitant with these changes, we now allow Cytoscape users to export their visually-coded networks from Cytoscape and to import them directly into GraphSpace.
- We have added support to allow a group owner to invite another person to a group via a signup link.

- To facilitate bulk uploads of networks to GraphSpace, we have implemented the Graphspace python module that a user can install from PyPI.

14.2 GraphSpace 1.1

14.2.1 Bug Fixes

- Fixed bug reported when trying to use the Forgot Password functionality.

14.3 GraphSpace 1.0

14.3.1 Highlights

- This is the first public release of GraphSpace.