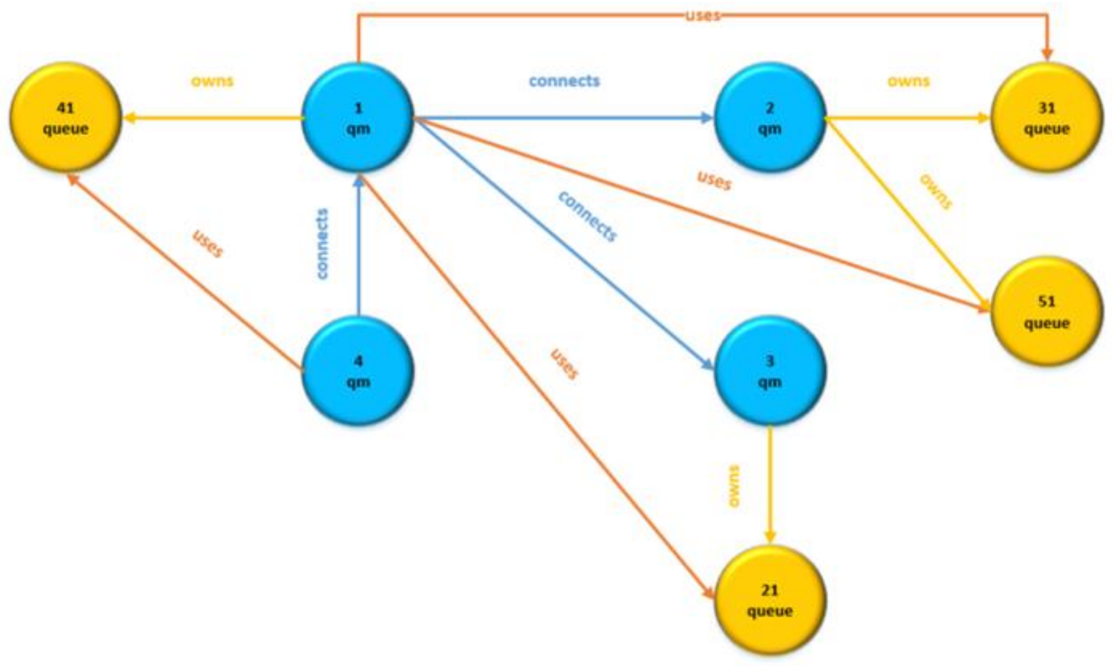
IBM MQ Cluster Tool

1. Overview

The IBM MQ Cluster tool provides a way to query and visualize your IBM MQ clusters using a graph. The tool is made of a set of Java applications that extracts IBM MQ cluster information and builds a directed property graph using the Apache TinkerPop™ computing framework and its in-memory TinkerGraph database. The graph can then be used with various open source or commercial tools to perform queries against the graph and to visualize the IBM MQ clusters.

The property graph model used is depicted below:



The property graph is comprised of the following nodes (vertices) and relationships (edges):

- *qm* nodes which describe cluster queue managers
- *queue* nodes which describe cluster queues
- *connects* relationships which describe connections between queue managers (i.e an application connected to a queue manager and putting messages on one or more cluster queues on other queue manager(s))
- *owns* relationships which describe the cluster queues owned by a queue manager

2. System Requirements

The IBM MQ Cluster Tool has the following system requirements:

- Linux or Windows workstation
- IBM MQ Client, any supported versions
  https://www-01.ibm.com/support/docview.wss?uid=swg24044791
- Java 8 JRE
- Apache TinkerPop™ and Gremlin Console, version 3.3.3 or later
  http://tinkerpop.apache.org/

  Note: downloading the Gremlin Console will automatically install the Apache TinkerPop™ computing framework that is required to use the TinkerGraph in-memory database.

3. Installation & Configuration

The IBM MQ Cluster Tool is installed and configured using the procedure below:

a) Ensure you have a Java 8 JRE installed
b) Ensure you have installed the Gremlin Console
c) Unzip ***IBM_MQ_Cluster_Tool_vx.x.zip*** to a directory of your choice

   The ZIP file contains the following files:

• com.ibm.xmq.cluster.jar	
• xmqcocls.cmd	Collect cluster information and build graph
• xmqcocls.sh	Collect cluster information and build graph
• xmqgrcls.cmd	Build graph from CSV cluster information
• xmqgrcls.sh	Build graph from CSV cluster information

d) Customize the scripts

   Depending on the platform you are planning to run the tool update the *.sh (Linux) or *.cmd (Windows) scripts as follows:

   - Update the ***MQTOOLS*** variable with the directory where the ***IBM MQ Cluster Tool*** was unzipped
   - Update the ***GREMLIN*** variable with the root directory for the ***Gremlin Console*** install

The IBM MQ Cluster Tool is now ready to be used.

4. Execution

The IBM MQ Cluster tool requires as input a colon (:) separated value file that contains the inventory of queue managers in the network along with information on how to connect to each queue manager using an MQ Client connection. Each record in the file identifies one queue manager, and the record should follow the format below:

```
Tag:QMName:ConnNameList:ChlName
```

Where:

- *Tag* can be any value used to identify the queue manager. For example, a business unit or a combination of things. Here are some examples:

  ```
  BKRT
  BKRT/App1
  BKRT/Insurance/App1
  ```

- *QMName* is the name of the queue manager
- *ConnNameList* is a comma separated list of one or more connection names that takes the following formats:

  ```
  Ip1(port1),…,ipN(portN)
  dns1(port1),…,dnsN(portN)
  ```

- *ChlName* is the name of the client (SVRCONN) channel to use

If connection authentication is required for the client connections, a user-id and password can be provided on the command line using the -u and -p options when executing the tools.

If SSL/TLS is required for the client connections, the ciphersuite can be provided on the command line using the -c option when executing the tools. The scripts can be modified to add a keystore and truststore on the Java command line using:

```
-Djavax.net.ssl.keyStore=<path and name of keystore>
-Djavax.net.ssl.keyStorePassword=<keystore password>
-Djavax.net.ssl.trustStore=<path and name of truststore>
-Djavax.net.ssl.trustStorePassword=<truststore password>
```

There are various scenarios that can be followed to run the tools and generate a graph that can subsequently be queried or visualized.

4.1. Scenario 1

A single queue manager inventory file is built and the tool to collect the IBM MQ cluster information for all queue managers can be run from a single location. In this instance, you can collect the cluster information and generate the graph in one pass.
In order to achieve the above, the following command can be used:

```
xmqcocls.cmd -f your_inventory.txt -g mygraph -q
```

The above command will create a graph and save it in file ***mygraph.graphml*** based on the queue manager inventory in file ***your_inventory.txt***. Additionally, three timestamped CSV files are created containing queue manager, cluster queue manager and cluster queue information.

To see all options for tool ***xmqcocls*** just run the tool without any options which will display help.

Once you have generated the graph you can skip to section 5.

4.2. Scenario 2

When a single queue manager inventory file cannot be built for any reasons, then ***xmqcocls*** can be run multiple times using different inventory files as shown below:

```
xmqcocls.cmd -f your_inventory_1.txt  -q
xmqcocls.cmd -f your_inventory_2.txt -q
xmqcocls.cmd -f your_inventory_N.txt -q
```

Three timestamped CSV files are created each time ***xmqcocls*** is executed:

qmdata-*timestamp*.csv
clusqmdata-*timestamp*.csv
clusqueuedata-*timestamp*.csv

Concatenate all *qmdata* into a single file, then concatenate all *clusqmdata* and *clusqueuedata* files into their own files.

The next is to generate the graph and save it using ***xmqgrcls***. In this example the graph is saved in file ***mygraph.graphml***.

```
xmqgrcls -g mygraph -m qmdata.csv -c clusqmdata.csv -q clusqueuedata.csv
```

5. Queries and Visualization

Once a graph has been generated and saved into a file in GRAPHML format, the graph can be queried and visualized using various open source tools. The two recommended tools are:

- Apache Gremlin to perform queries against the graph
  https://tinkerpop.apache.org/gremlin.html

- Cytoscape to visualize the graph
  https://cytoscape.org/

5.1. Apache Gremlin

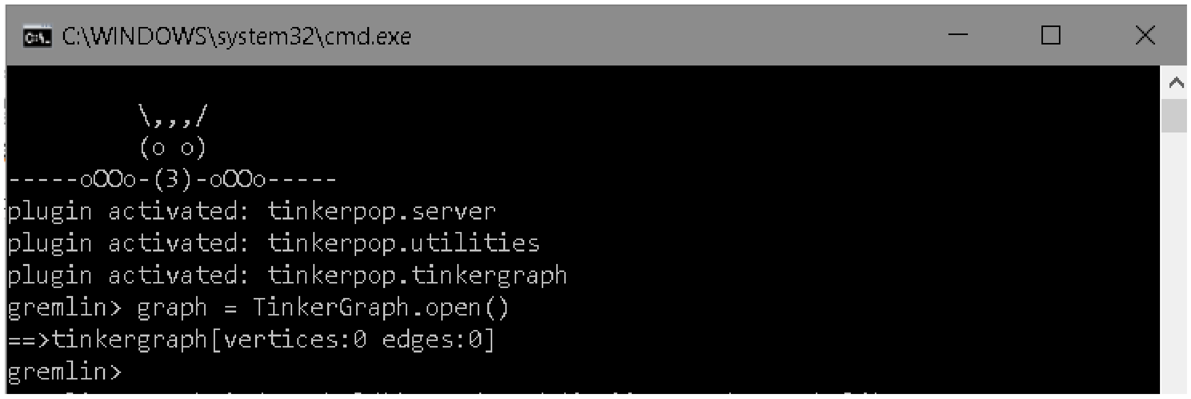
Apache Gremlin is the graph traversal language of Apache TinkerPop. It allows to succinctly express complex queries (traversals) on property graphs. A great tutorial for Apache Gremlin can be found in the following document:

http://kelvinlawrence.net/book/Gremlin-Graph-Guide.html

Though it is beyond the scope of this document to provide a complete tutorial on how to use Apache Gremlin, to get started and run basic queries against an MQ Cluster graph, the following steps can be followed.

a) Start the Gremlin console

Use the ***gremlin.sh*** or ***gremlin.cmd*** script in the ***bin*** directory of the Gremlin Console installation directory to start the console



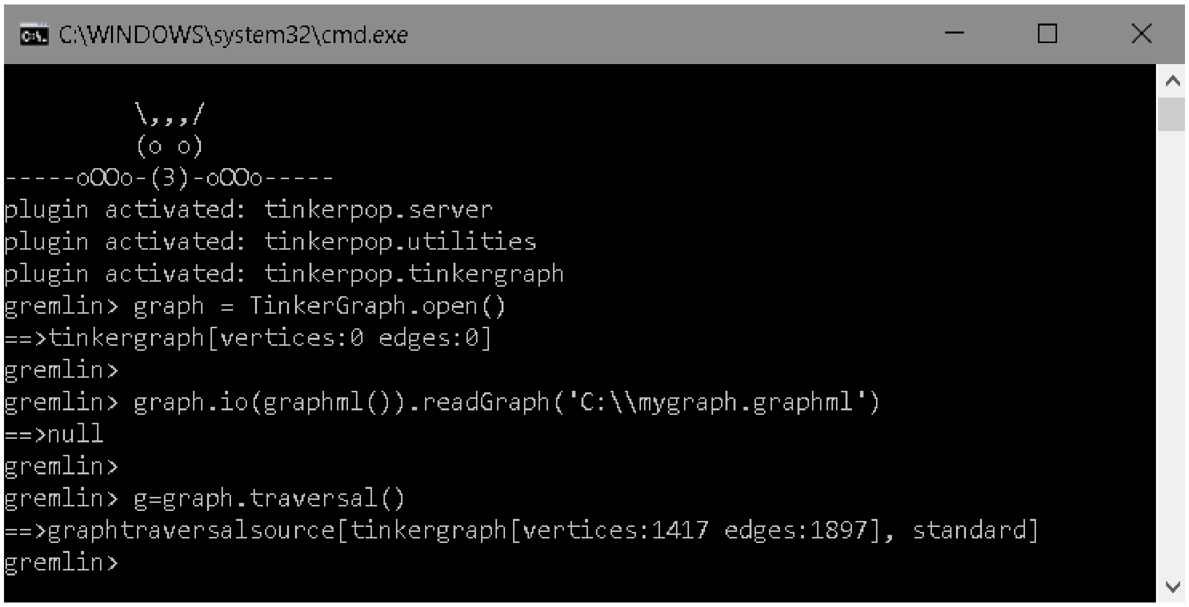
C:\WINDOWS\system32\cmd.exe

```
         \,,,/
         (o o)
-----oOOo-(3)-oOOo-----
plugin activated: tinkerpop.server
plugin activated: tinkerpop.utilities
plugin activated: tinkerpop.tinkergraph
gremlin> graph = TinkerGraph.open()
==>tinkergraph[vertices:0 edges:0]
gremlin>
```

b) Load the graph and create a traversal for queries

Create a TinkerGraph instance, load your graph and create a traversal object to run queries using the following commands

graph = TinkerGraph.open()
graph.io(graphml()).readGraph('c:\\mygraph.graphml')
g=graph.traversal()


C:\WINDOWS\system32\cmd.exe

```
         \,,,/
         (o o)
-----oOOo-(3)-oOOo-----
plugin activated: tinkerpop.server
plugin activated: tinkerpop.utilities
plugin activated: tinkerpop.tinkergraph
gremlin> graph = TinkerGraph.open()
==>tinkergraph[vertices:0 edges:0]
gremlin>
gremlin> graph.io(graphml()).readGraph('C:\\mygraph.graphml')
==>null
gremlin>
gremlin> g=graph.traversal()
==>graphtraversalsource[tinkergraph[vertices:1417 edges:1897], standard]
gremlin>
```

The output for this ***mygraph.graphml*** files shows that 1417 vertices (nodes, total of queue managers and cluster queues) have been loaded, and that 1897 edges (relationships, total of *connects, owns* and *uses* relationships).

Note: This ***mygraph.graphml*** can be found on GitHub in the ***Samples*** folder. This sample graph contains a single MQ cluster of 310 queue managers and a limited number of cluster queues across Linux and Windows queue managers at versions from 6.0 to 9.1.

c) Run some queries

With the graph loaded in the in-memory TinkerGraph data base, it is now time to build and run some queries.

- Display a queue manager information
  g.V().has('name','QM0079').valueMap(true).unfold()

- Display all queue managers having a specific tag along with queue manager name, version and platform
  g.V().has('tag','BWU').values('name','version','platform')

- Display number of queue managers having a specific tag
  g.V().has('tag','GTWT').count()

- Display number of queue managers per tag
  g.V().hasLabel(‘qm’).groupCount().by('tag')

- Display number of queue managers per version
  g.V().hasLabel('qm').groupCount().by('version')

- Display queue managers a queue manager is connecting to
  g.V().has('qm','name','QM0004').out().hasLabel('qm').values('name')

The above are just basic queries, but more complex queries can be built. An API is also provided by Apache Gremlin to run queries within applications.

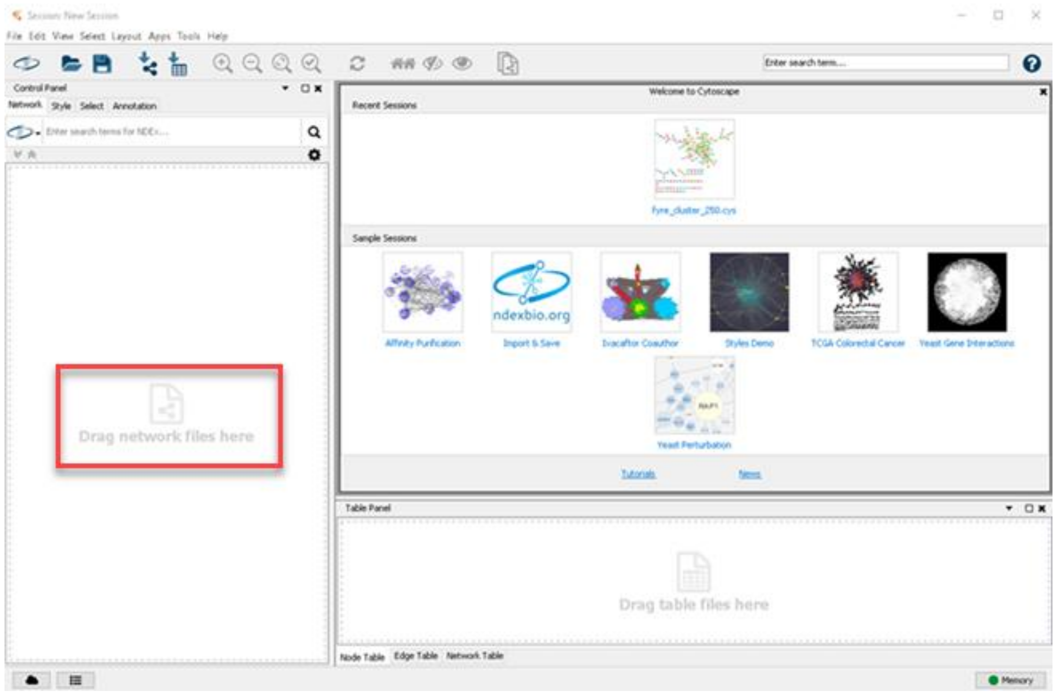
d) Exit the Gremlin Console
To exit the Gemlin Console, type the following command.
:exit

5.2. Cytoscape

Cytoscape is an open source software platform for visualizing complex networks or graphs. Cytoscape can be downloaded from https://cytoscape.org/.

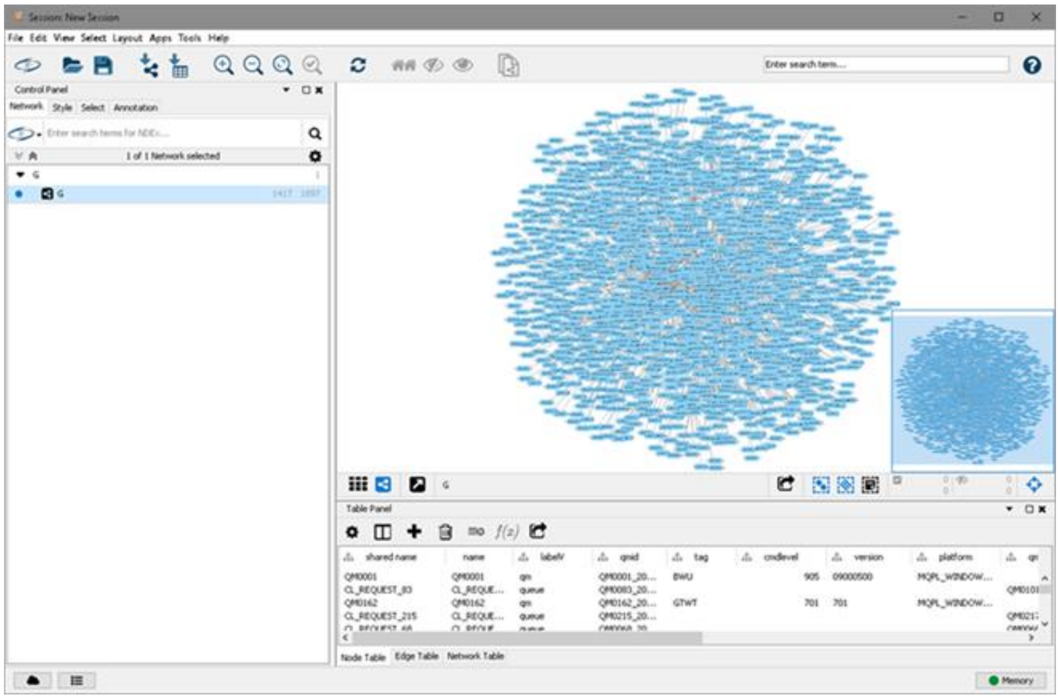
Though it is not possible to provide a full tutorial here, following the steps below will get you started.

a) Download, install and start Cytoscape



b) Load your graph into Cytospcape

To load your graph in Cytoscape, either drag the GRAPHML file in the window in red (above) or use the menu options: File > Import > Network from file… > then select your GRAPHML file.


Session: New Session
File Edit View Select Layout Apps Tools Help
Enter search term...
Control Panel
Network Style Select Annotation
Enter search terms for NDEx...
1 of 1 Network selected
Table Panel
Node Table Edge Table Network Table
Memory

shared name	name	labelV	qmid	tag	cmdlevel	version	platform	qm
QM0001	QM0001	qm	QM0001_20...	BWU	905	09000500	MQPL_WINDOW...	
CL_REQUEST_83	CL_REQUE...	queue	QM0083_20...					QM0101
QM0162	QM0162	qm	QM0162_20...	GTWT	701	701	MQPL_WINDOW...	
CL_REQUEST_215	CL_REQUE...	queue	QM0215_20...					QM021

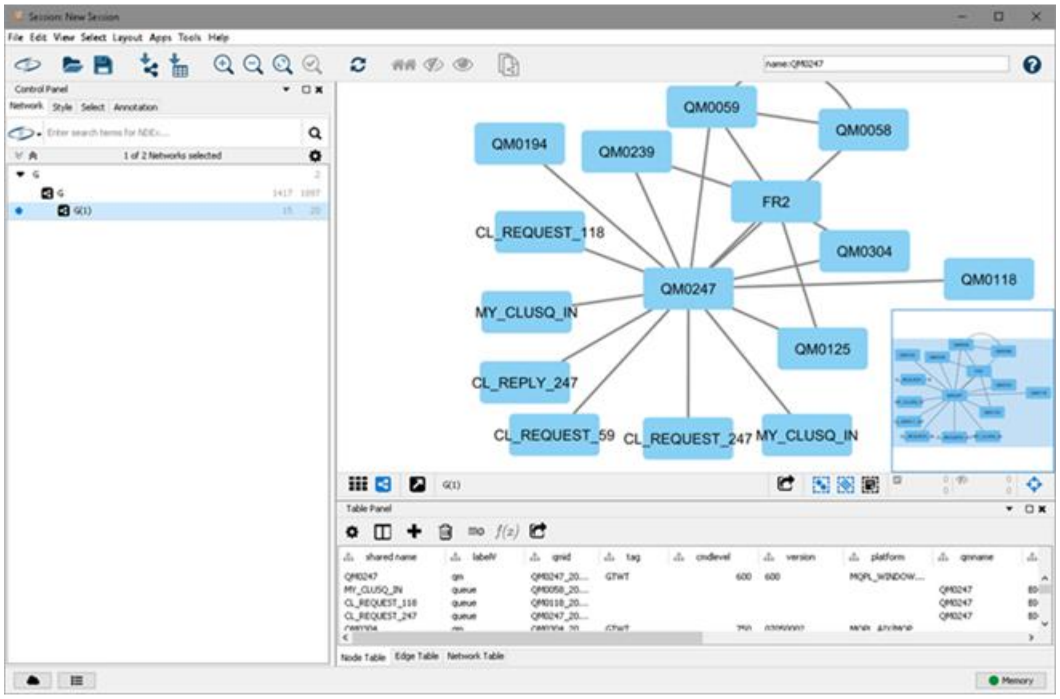
Depending on the size of your cluster (number of queue managers and cluster queues) the graph will look very dense. From that point on, the goal is to create sub-graphs from the main graph or from previously created sub-graphs based on some parameters.

c) Creating a sub-graph of all connections for a specific queue manager

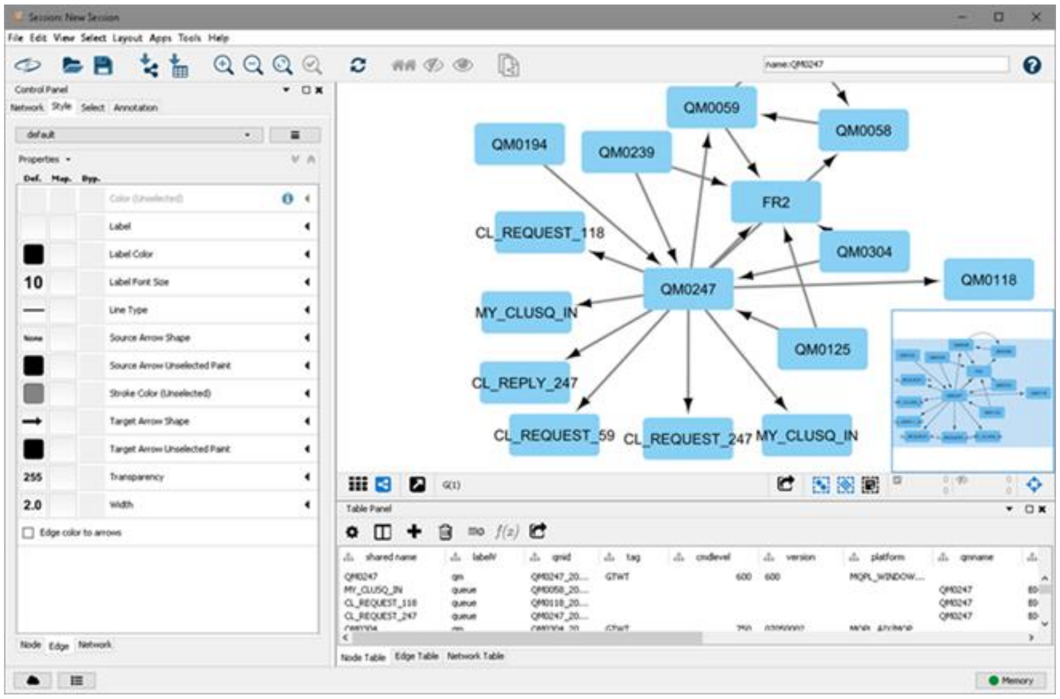
In the search bar at the top right, enter: *name:QM0247*, then press Enter.

Queue manager *QM0247* is highlighted in yellow. Then right click the queue manager and select: Select > Select First Neighbors (undirected). Then on the tool bar, click on the following icons (in order from left to right below):

A sub-graph is created as shown below.



On thing that is missing is the direction of the connection. To remediate that, click on the *Style* tab at the top, then at the bottom, click on the *Edge* tab. For *Target Arrow Shape*, click on ***None*** to display the options and select an arrow shape.

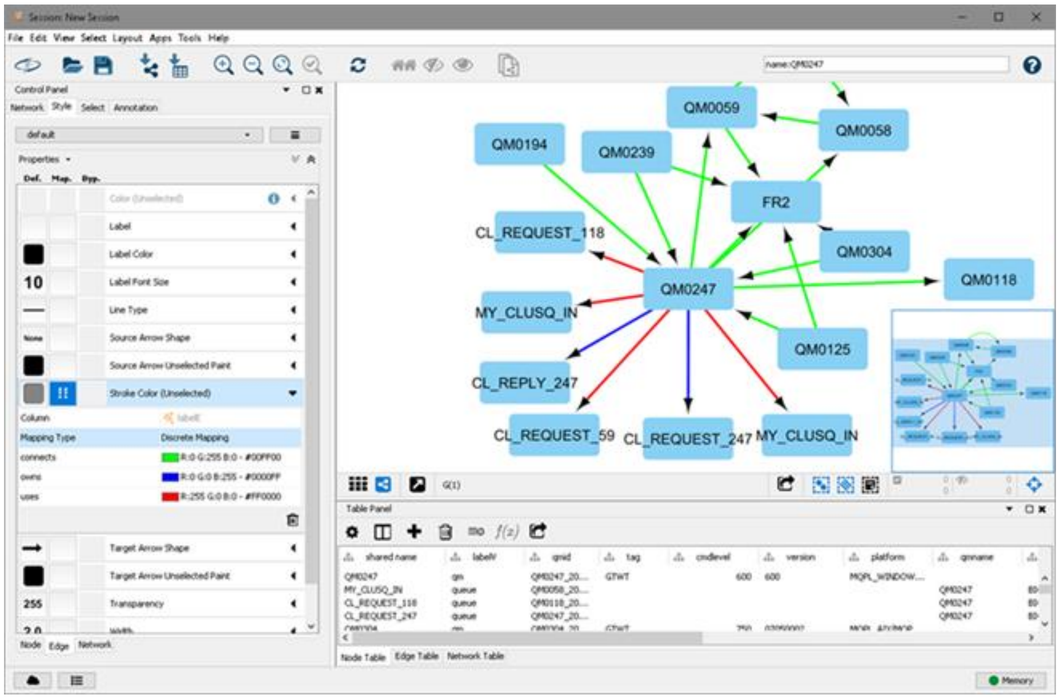


The graph now shows several things:

- The queue managers this queue manager is talking to
- The other queue managers using this queue manager owned cluster queues
- The cluster queues either owned by this queue manager or cluster queues residing on other queue managers that this queue manager is using

One additional step can be achieved to more clearly show what the relationships (connects, owns and uses). From the *Style* and *Edge* tabs (you should still be there), open the *Stroke Color (unselected)*, then specify the *LabelE* for the *Column* field. For *Mapping Type* select *Discrete Mapping*, then right click and select *Mapping Value Generators* > *Rainbow*.

Relationships are now color coded as shown in the picture below.



Stroke Color (Unselected)

Column	labelE
Mapping Type	Discrete Mapping
connects	🟩 R:0 G:255 B:0 - #00FF00
owns	🟦 R:0 G:0 B:255 - #0000FF
uses	🟥 R:255 G:0 B:0 - #FF0000

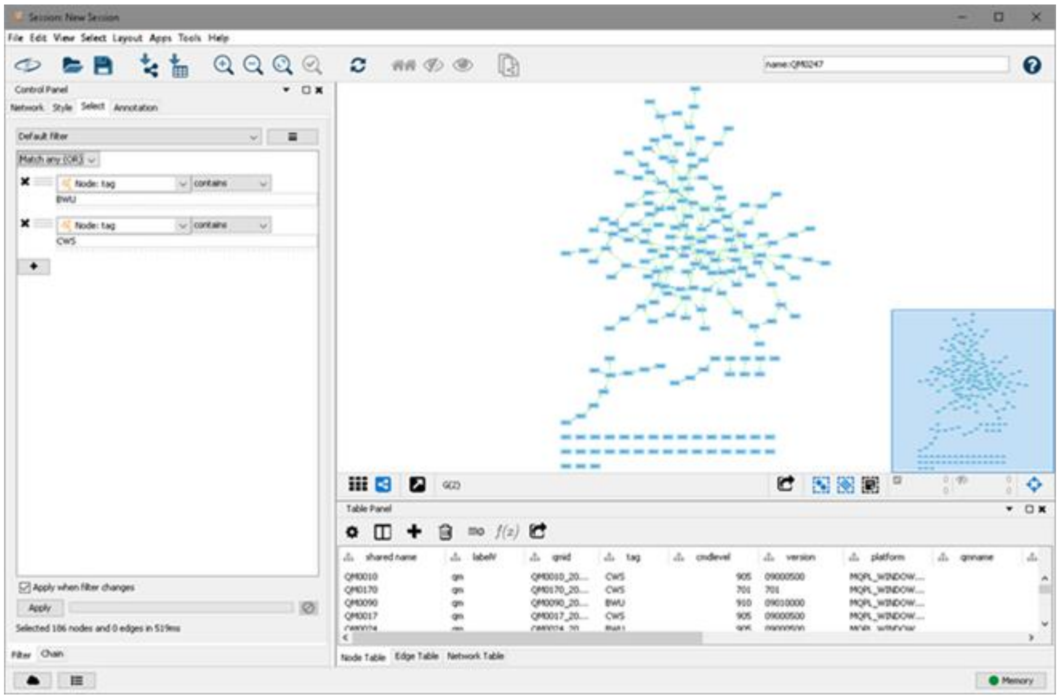
d) Visualize the connections between queue managers in different groups (tags)

The provided sample graph, ***mygraph.graphml***, contains queue managers split in three different groups (GTWT, CWS and BWU). If you wanted to understand the connections between all the queue managers in two of the groups, the following procedure can be followed.

Click on *Network* tab on top left and select the full graph (G). The sub-graph created in the steps earlier was named G(1), but these can be renamed if needed.

Click on the *Select* tab to create a filter. Click the + sign and select *Column Filter*. In the *Choose Column…* drop box, select column *Node:tag* and for the value type *BWU*. This will select/highlight (in yellow) all the queue managers that have a tag of BWU.

Click the + sign again to add another condition. Again, select *Colum Filter*, choose *Node:tag* and specify a value of CWS. Then change the conditional operator from AND to OR. Now, all queue managers having tags BWU or CWS are selected. It is time to click on the following two icons to generate a new sub-graph:


Session: New Session
File Edit View Select Layout Apps Tools Help
Control Panel
Network Style Select Annotation
Default filter
Match any (OR)
Node: tag
contains
Node: tag
contains
Apply when filter changes
Apply
Filter Chain
G(2)
Table Panel
Node Table Edge Table Network Table
Memory

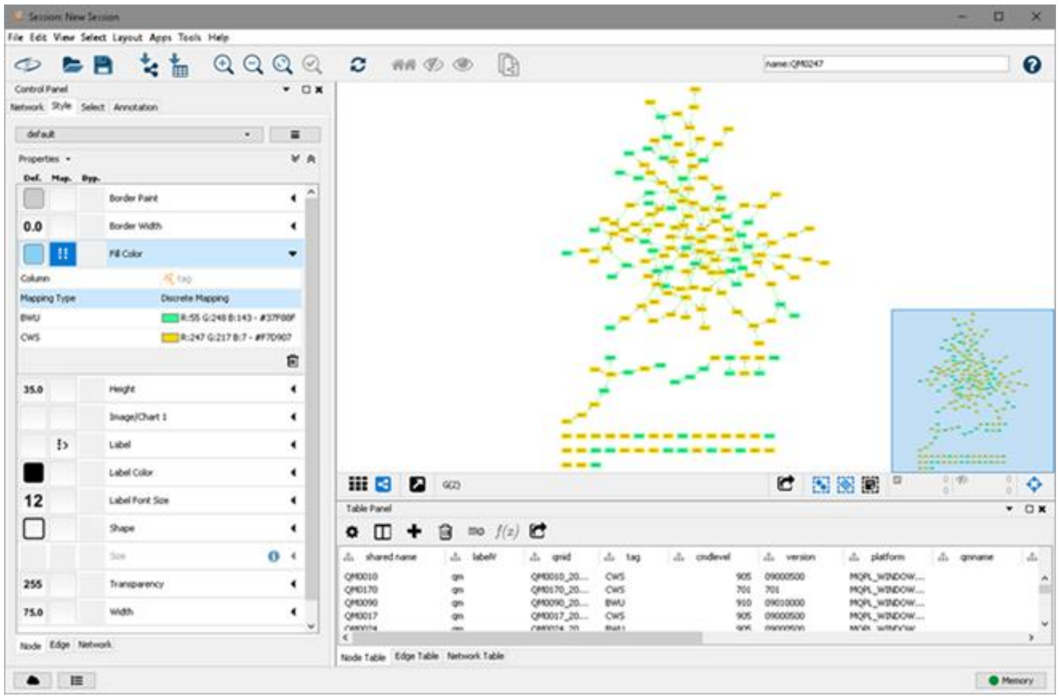
BWU

CWS

Selected 186 nodes and 0 edges in 519ms

shared name	labelV	qmid	tag	cmdlevel	version	platform	qmname
QM0010	qm	QM0010_20....	CWS	905	09000500	MQPL_WINDOW....	
QM0170	qm	QM0170_20....	CWS	701	701	MQPL_WINDOW....	
QM0090	qm	QM0090_20....	BWU	910	09010000	MQPL_WINDOW....	
QM0017	qm	QM0017_20....	CWS	905	09000500	MQPL_WINDOW....	

Now it would be nice to have the queue managers color coded by the tags. So, click on the *Style* tab, click on the *Node* tab (bottom left), then open *Fill Color*. For the *Column* field select *Tag* and for *Mapping Type* select *Discrete Value*. Then right click and select *Mapping Value Generators* > *Rainbow* or select your favorite colors!


Session: New Session
File Edit View Select Layout Apps Tools Help
name:QM0247
Control Panel
Network Style Select Annotation
default
Properties
Def. Map. Byp.
Node Edge Network
Table Panel
Node Table Edge Table Network Table
Memory

Def.	Map.	Property
		Border Paint
0.0		Border Width
		Fill Color
35.0		Height
		Image/Chart 1
		Label
		Label Color
12		Label Font Size
		Shape
		Size
255		Transparency
75.0		Width

Column	tag
Mapping Type	Discrete Mapping
BWU	🟩 R:55 G:248 B:143 - #37F88F
CWS	🟨 R:247 G:217 B:7 - #F7D907

shared name	labelV	qmid	tag	cmdlevel	version	platform	qmname
QM0010	qm	QM0010_20...	CWS	905	09000500	MQPL_WINDOW...	
QM0170	qm	QM0170_20...	CWS	701	701	MQPL_WINDOW...	
QM0090	qm	QM0090_20...	BWU	910	09010000	MQPL_WINDOW...	
QM0017	qm	QM0017_20...	CWS	905	09000500	MQPL_WINDOW...	

Cytoscape is a very feature-rich tool, the above examples just scratch the surface of what is possible. There are also many other fields in the generated graph that can be used as parameters to build sub-graphs using the same technics shown above. For example:

- Show all queue managers that are at on or more versions or less than a specific version
- Show all queue managers that are running on a specific platform
- Show all queue managers that have cluster channels in non-running status
- … and the list goes on… HAPPY DISCOVERY!

6. Appendix

6.1. CSV – Queue Manager Data

The following queue manager data is collected, used to build the ***qm*** nodes (vertices) in the graph and is also stored in a CSV file called qmdata-*timestamp*.csv.

The CSV record contains the following data:

- Tag
- Queue Manager Name
- Queue Manager ID
- Queue Manager Description
- Queue Manager Command Level
- Queue Manager Platform
- Queue Manager Version

6.2. CSV – Cluster Queue Manager Data

The following cluster queue manager data is collected, used to build the ***connects*** relationships (edges) in the graph and is also stored in a CSV file called clusqmdata-*timestamp*.csv.

The CSV record contains the following data:

- Source Queue Manager Name
- Tag of Target Queue Manager
- Target Queue Manager Name
- Target Queue Manager ID
- Target Queue Manager Type of Repository
- Cluster Name
- Channel Name
- Connection Name
- Cluster Channel Type
- SSL Client Authentication
- SSL Cipher
- SSL Peer
- Channel Status
- Name of Transmission Queue
- Target Queue Manager Version

6.3. CSV – Cluster Queue Data

The following cluster queue data is collected, used to build the ***owns/uses*** relationships (edges) in the graph and is also stored in a CSV file called clusqueuedata-*timestamp*.csv.

The CSV record contains the following data:

- Queue Manager Name using the Cluster Queue
- Cluster Queue Name
- Cluster Name
- Queue Manager Name owning the Cluster Queue
- Type of Cluster Queue
- Cluster Workload Priority
- Cluster Workload Rank
- Cluster Default Bind Option
- Queue Description
- PUT Enabled or Disabled
- Queue Manager ID owning the Cluster Queue