

Detailed description of JSON syntax for diagram views

A diagram view can be described by a file in a specific JSON format.

To be able to design diagram views with JSON, you need to know the following information.

JSON syntax

The following pieces of JSON code are examples of name-value pairs that you can use to [create a diagram view](#).

Diagram section

These settings determine general display settings for the diagram.

```
"visitStrategy": "directed", "completeGraph"
```

- Determines which nodes and edges have to be traversed and displayed.
- This setting is optional.
- You must pick one of the following values:
 - **"directed"**: For the start node, DGC traverses the relations in all directions, and adds all nodes it found. For the nodes it just found, DGC traverses relations only in the same direction as how they got found: if a node was found by traversing an outgoing relation, DGC looks for outgoing relations, and vice versa: if the node was found by traversing an incoming relation, DGC looks only for incoming relations. This behaviour is similar to traversing a hierarchy. This is the default setting. For a refinement to this strategy, see also the setting for layoutRegion.
 - **"completeGraph"**: All nodes and edges related to the current asset are displayed irrespective of the direction in which they were found.
NOTE: be careful with using completeGraph, this setting can lead to very large diagrams.

```
"resultNodeUnicityStrategy": "MultipleNodesPerTermId", "singleNodePerTermId"
```

- Determines the number of times a node is displayed if it is found multiple times in the flow of the diagram.
- This setting is optional.
- You must pick one of the following values:
 - **"MultipleNodesPerTermId"**: Displays the result node multiple times. This is the default setting.
 - **"singleNodePerTermId"**: Displays the result node only once.
NOTE: If an asset is found more than once for the same viewnode, the result is always displayed only once. In that case, there is a loop for that node.

```
"showEdgeLabels": false, true
```

- Determines if the edge labels are displayed.
- This setting is optional.
- You must pick one of the following values:
 - **false**: The labels are not displayed. This is the default setting.
 - **true**: The labels are displayed.

```
"showLegend": false, true
```

- Determines if the diagram legend is displayed. The legend shows all asset types and complex relation types that occur in the diagram.
- This setting is optional.
- You must pick one of the following values:
 - **false**: The legend is not displayed.
 - **true**: The legend is displayed. This is the default setting.

```
"layout": "HierarchyLeftRight", "HierarchyTopBottom", "Circular", "SmartOrganic", "Radial",  
"Flow/Context"
```

- Determines the layout style of the diagram.
- This setting is optional.
- You must pick one of the following values:
 - **"HierarchyLeftRight"**: Nodes and edges are displayed in a flow from left to right. This is the default setting.
 - **"HierarchyTopBottom"**: Nodes and edges are displayed in a flow from top to bottom.
 - **"Circular"**: Nodes and edges are arranged in a radial tree.
 - **"SmartOrganic"**: Nodes and edges are distributed in a well-balanced manner, there are few edge crossings.
 - **"Radial"**: Nodes and edges are displayed with no overlaps, few edge crossings and few bends.
 - **"Flow/Context"**: Layout style for diagrams with a flow and a context region. See also layoutRegion in the nodes section.

Nodes and edges in the flow region are displayed mostly from left to right. The nodes and edges in the context region are displayed above the flow region.

An edge that begins or ends with a context node, is shown with less emphasis (thinner and in light grey) than an edge between two flow nodes.

If you specify this layout, keep in mind that for an edge between a flow node and a context node, the "from" node has to be in the flow region and the "to" node has to be in the context region.

"maxNodeLabelLength": 0, <positive integer number>

- Determines the length of the node labels (whether they should be truncated when they are too long).
- This setting is optional.
- You must provide 0 or a positive integer number as the value:
 - 0: Node labels are not truncated, they are displayed in full length.
 - The default setting is 50.

"maxEdgeLabelLength": 0, <positive integer number>

- Determines the length of the edge labels if they have to be truncated when they are too long.
- This setting is optional.
- You must provide 0 or a positive integer number as the value:
 - 0: Edge labels are not truncated, they are displayed in full length.
 - The default value is 30.


Nodes section

These settings determine the display settings for nodes. You must add a node for each asset type and complex relation type that you want to include in the diagram.

"id": "Business Term1"

- Determines the name of the node.
- This setting is mandatory.
- You can type any string here, but it must be unique in this view. For readability, we recommend you use the name of the asset type or complex relation type as the ID.
- You can refer to this node by using its ID in the "from" and "to" name/value pairs of the edges section.

"conceptTypeId": "00000000-0000-0000-0000-000000011001"

- Determines the resource ID of the asset type or the complex relation type.
- This setting is mandatory.
- You can have multiple nodes with the same conceptTypeId in one diagram view.
- The value has to be a valid resource ID.
- You can find the resource ID of an asset type by clicking **Settings > Types > Asset types** and then selecting a type. The URL of the asset type page contains the resource ID of the asset type. For example, if the URL to the asset type page of 'Business Asset' is <http://<your-DGC>/term/00000000-0000-0000-0000-000000031101#tbt-tabbar-content=attributes&tbt-tabbar-meta=comments>; the resource ID for 'Business Asset' is 00000000-0000-0000-0000-000000031101.
- You can also refer to the parent of an asset type.
- You can find the resource ID of a complex relation type by clicking **Settings > Attributes > Complex relation types** and then clicking  > **Columns > Resource Id**.

"label": "Term"

- Determines the name that is displayed on the node in the diagram design view. For a node that represents a complex relation type, the label is used in the result diagram as well.
- This setting is optional.
- You can provide any string as the value:
- If you do not specify a label, the name of the asset type or the complex relation type is used.

"layoutRegion": "flow", "context"

- Determines if the node is treated as a flow node or a context node. This influences the traversal strategy. An edge from a flow node to a context node is always included in the result diagram.
An edge between two flow nodes FN1 and FN2 is only included in the diagram if the edge from FN1 to FN2 is traversed in the same direction as the edge that brought FN1 into the diagram.
- This setting is optional.
- You must pick one of the following values:
 - "flow": The node is part of the flow. This is the default setting.
 - "context": The node is part of the context.

- When you select the "Flow/Context" layout for a diagram, the edges between flow nodes are rendered horizontally, mostly from left to right (the "from" node is to the left of the "to" node). The edges from flow to context nodes are rendered vertically, from bottom to top.

Edges section

These settings determine which directed relations should be traversed, and how they should be depicted on the diagram. Each edge represents a relation type between two nodes (asset types or complex relation types). You have to ensure that the diagram view is a *connected* graph: each node in the diagram view is reachable from any other node.




"from": "Business Term1"

- Determines which node is the head asset.
- This setting is mandatory.
- You have to fill in a node ID ("id") from the nodes section.

"to": "Table Column1"

- Determines which node is the tail asset.
- This setting is mandatory.
- You have to fill in a node ID ("id") from the nodes section.

"binaryFactTypeId": "00000000-0000-0000-0000-000000007038"

- Determines the relation type.
- This setting is mandatory.
- You have to fill in the resource ID of the relevant relation type.
You can copy and paste these resource IDs from the Settings UI:
 - You can find the resource ID of a relation type by clicking **Settings > Attributes > Relations** and then clicking  > **Columns > Resource Id**.
 - You can find the resource ID of a complex relation type by clicking **Settings > Attributes > Complex relation types** and then clicking  > **Columns > Resource Id**.
 - You can find the resource ID of all relation types that are part of a complex relation type by clicking **Settings > Attributes > Complex relation types** and then clicking  > **Columns > Relation Ids**.

"roleDirection": true, false

- Determines the direction of the edge; is it the role or co-role.
- This setting is mandatory.
- You must pick one of the following values:
 - true: The edge is traversed from head to tail.
 - false: The edge is traversed from tail to head.

"style": "arrow", "box", "reversed-box"

- Determines how edges are displayed.
- This setting is optional.
- You must pick one of the following values:
 - "arrow": The edge is represented by an arrow. This is the default setting.
 - "box": The edge is represented by a box of a head asset containing a box of a tail asset.
 - "reversed-box": The edge is presented as a box of a tail asset containing the box of a head asset.

"label": "Groups"

- Determines the name that is displayed on edges in the diagram.
- This setting is optional.
- If you do not specify a label, the role or co-role of the relation type from the operating model is used, in both the diagram view and the result diagram.
If "roleDirection" is true, DGC uses the role, if it is false, DGC uses the co-role.

Creating correct diagram views

JSON is a case-sensitive language. This means that you have to use the exact name/value pairs that we described here.

The JSON view needs to contain a Diagram section, a nodes section and an edges section. The order of the sections is irrelevant. Layout to improve readability (spaces, tabs, empty lines), is irrelevant.

DGC checks some aspects of the syntax of your JSON view constantly as you are typing.

If the JSON view is incorrect, DGC will signal this immediately and will not allow you to save the diagram view.

For example, if you omit the required comma between two name/value pairs, DGC will show an error message, and highlight the offending line.

Also, if you make a typo in a value, DGC will show an error, listing out all the allowed values.

DGC will perform some other syntax and semantic checks when you press the Save button.

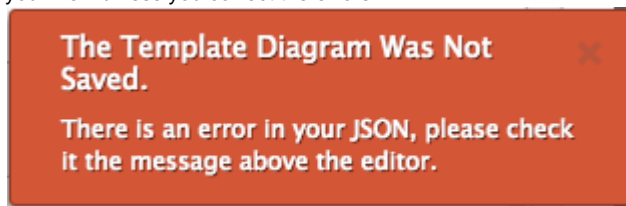
For example, as we pointed out earlier, the resulting graph needs to be connected: all nodes need to be reachable from all other nodes.

You can also create other name/value pairs, with a name that is not listed here. These name/value pairs are allowed but will be ignored.

You can use this to your advantage, for example to add comments to the JSON view, but be aware that if you make a typo in the name, DGC will not signal this, instead it will ignore that line.

Note: When you click the save button, DGC performs various checks:

1. Whether each resource ID (conceptTypeIdd or binaryFactTypeIdd) that you specified exists in the operating model.
If a resource ID does not exist, it is removed from the diagram view and a warning and an error are displayed. DGC does not save your view unless you correct the errors.



Traceability > Diagram

Technical Lineage < Table Columns flow through Field Mappings, with Table context

Technical Lineage

Table Columns flow through Field Mappings, with Table context

JSON

Diagram

'node.conceptTypeIdd' property is missing or is not a string. Id.: Table Column Index: 0

```
1 {
2   "showEdgeLabels": true,
3   "layout": "Flow/Context",
4   "nodes": [
```

2. Whether the diagram view is a connected graph.
A diagram view is a connected graph when every node can be reached from any other node by traversing the edges. If the diagram is not connected, a warning is displayed.
DGC does not save your view unless you correct the errors.

