

# CIShellJSON (.json, .cishellgraph.json, .cishelltable.json)

## CIShell JSON

The CIShell JSON spec defines a JavaScript Object Notation (JSON) format for tabular and graph data for use with CIShell and derived platforms. The primary use of this format is data transport to browser-based visualization clients, but this specification also defines suggested attributes of files saved in this format.

## Topologies

This format supports tabular and graph data in a common container, but only one of these topologies per object/file. The supported table format contains a single set of records; the supported graph format has a single set of nodes and a corresponding single set of edges. To help tools including CIShell and our web client visualizations know how to consume these objects, we provide topology metadata in the form of a top-level data member, MIME type (during transport), and file extension (Table 1).

	JSON element	MIME Type	File Extension
Graph	"topology": "graph"	application/org.cishell.graph+json	.cishellgraph.json
Table	"topology": "table"	application/org.cishell.table+json	.cishelltable.json

Table 1: Supported Topologies

## Structure

The basic structure is that of a JSON object containing top-level elements indicating *topology*, a *schema* containing structural members (allowing non-spec members, but not guaranteeing that this data will be read by a given client), and record or node-edge set containers (Table 2). Inside the *records*, *nodes*, and *edges* containers are *name* and *type* fields and schema and dataset members.

Name	Description	Parent(s)	Type	Optional
<i>topology</i>	Flag indicating type of data represented	(root)	String	N
<i>records</i>	Tabular row data	(root)	JSON object	N (for table topology)
<i>nodes</i>	Graph node data	(root)	JSON object	N (for graph topology)
<i>edges</i>	Graph edge data	(root)	JSON object	N (for graph topology)
<i>name</i>	Dataset name	*	String	Y
<i>type</i>	Dataset type (graph edge type: "directed" / "undirected")	*	String	Y
<i>schema</i>	Schema for data element in same container; at root level, it defines available members	(root), records, nodes, edges	JSON array of JSON object	N
<i>data</i>	Record/edge datasets	records, nodes, edges	JSON array of JSON object	N

Table 2: Base JSON object members

Within *schema* elements, a required set of *name* and *type* fields, and optional *default* value and *primarykey* fields define the data contained in records, nodes, and edges. For each object within the schema, an element with a name corresponding to the schema's *name* field should exist and contain data corresponding to the schema *type* (Table 3).

Name	Description	Parent(s)	Type	Optional
<i>name</i>	Field name	*.schema	string	N
<i>type</i>	Field data type	*.schema	string	N
<i>default</i>	Default value for field	*.schema	(default value for type)	Y
<i>primarykey</i>	Defines whether a field is/is part of a primary key	*.schema	boolean	Y
(column name, per schema)	Data element	*.data	(value for type)	N

Table 3: Schema and data member fields

Graph nodes and edges have a minimum required set of schema elements, as detailed in Table 4. Note: *id* is a suggested field for tabular data, but is not required; *primarykey* schema fields should be used when possible, whether *id* is present and conventionally named or not.

<i>Name</i>	<i>Description</i>	<i>Parent(s)</i>	<i>Type</i>	<i>Optional</i>
<i>id</i>	Record ID	nodes	int	N
<i>source</i>	Edge source node record ID	edges	int	N
<i>target</i>	Edge target node record ID	edges	int	N

Table 4: Required schema elements for graph members

## Multiple Datasets

This specification is not intended to handle multiple datasets or relational data other than node-edge groups. These could be added via top-level schema elements, but this is not a supported method. The preferred method of transferring a group of datasets would be to generate a corresponding group of CShell JSON objects.

## Sample Objects

The following are examples of objects that follow the above specification.

### Table:

```
{
  "name": "Sample Table",
  "topology": "table",
  "schema": [
    { "name": "records",
      "type": "records" }],
  "records": {
    "schema": [
      { "name": "id",
        "type": "int",
        "default": 0,
        "primarykey": true },
      { "name": "label",
        "type": "string",
        "default": "" }],
    "data": [ { "id": 0, "label": "Row 0" }, { "id": 1, "label": "Row 1" } ]
  }
}
```

### Graph:

```
{
  "name": "Sample Nodeset",
  "topology": "graph",
```

```
"schema": [
  { "name": "nodes",
    "type": "nodes" },
  { "name": "edges",
    "type": "edges" } ],

"nodes": {
  "schema": [
    { "name": "id",
      "type": "int",
      "default": 0,
      "primarykey": true },
    { "name": "label",
      "type": "string",
      "default": "" } ],
  "data": [ { "id": 0, "label": "Node 0" }, { "id": 1, "label": "Node 1" } ]
},
"edges": {
  "type": "undirected",
  "schema": [
    { "name": "source",
      "type": "int" },
    { "name": "target",
      "type": "int" } ],
  "data": [ { "source": 0, "target": 1 } ]
}
}
```