How to validate an IBM MQ CCDT JSON file against the schema

https://www.ibm.com/support/pages/node/6964482

Date last updated: 21-Jun-2024

Angel Rivera
IBM MQ Support
https://www.ibm.com/products/mq/support
Find all the support you need for IBM MQ

+++ Questions +++

You are experimenting with the CCDT JSON text files to connect IBM MQ Client applications with queue managers.
You have successfully tested 2 files that have only 1 channel, and now you want to create another file that has 2 channels, but the 2nd channel does not seem to be found.

Now you want to know:

a) How to validate that the contents has a valid JSON format?

b) How to validate that the contents has the proper syntax for a CCDT file in JSON?

+++ Answers +++

++ a) For a general JSON syntax, see the following tutorial:

https://www.ibm.com/support/pages/node/6570329
Using cat and jq in Linux to validate an MQ CCDT file in JSON format

Short summary:
Use the web tool at https://jsonformatter.org/

++ b) The rest of this tutorial will focus on how to validate the contents for proper syntax for a CCDT using a schema.

+ Note for MQ 9.4 (no change to the schema):

The file ccdt_schema.json delivered in MQ 9.4 has the same contents as in MQ 9.3

+ Summary +

Visit the following web page, which has already the IBM MQ CCDT JSON schema from MQ 9.3.2 CD (which is the same as 9.3.5 CD and 9.4)

https://www.jsonschemavalidator.net/s/TgcFCL1A

The schema will be shown on the left side.

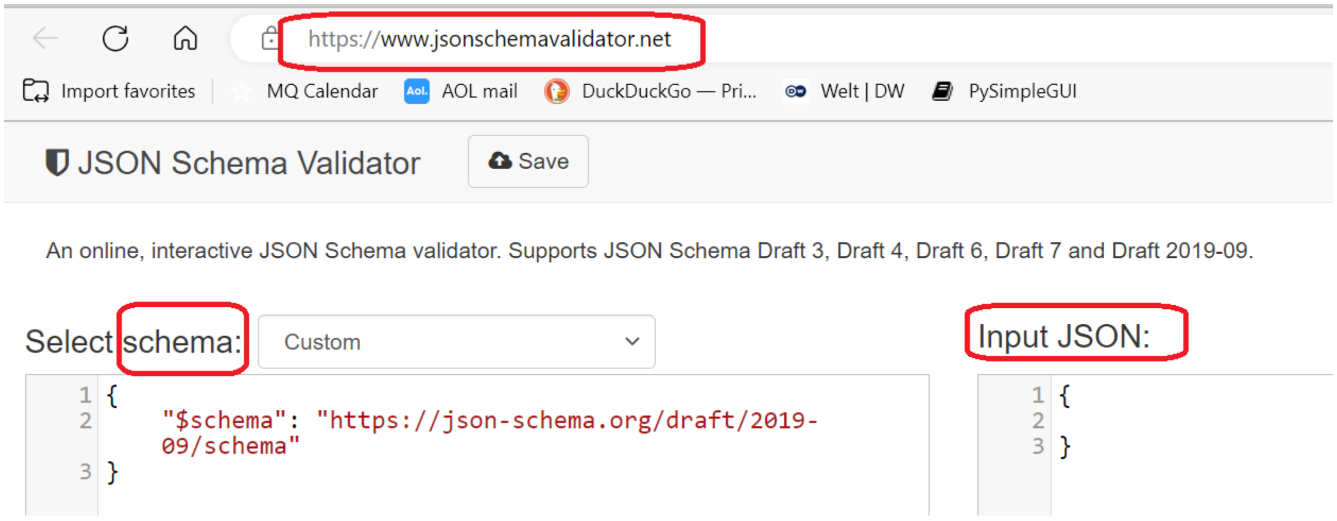
Then copy the entire contents of your CCDT JSON file and paste it on the right side.

+ Details

You can use the following web tool:

https://www.jsonschemavalidator.net/
An online, interactive JSON Schema validator.
Supports JSON Schema Draft 3, Draft 4, Draft 6, Draft 7 and Draft 2019-09.

You will see 2 panels:
- The left one is for the schema.
- The right one is for the Input JSON:



An online, interactive JSON Schema validator. Supports JSON Schema Draft 3, Draft 4, Draft 6, Draft 7 and Draft 2019-09.

Select schema: Custom

```
{
    "$schema": "https://json-schema.org/draft/2019-
    09/schema"
}
```

Input JSON:

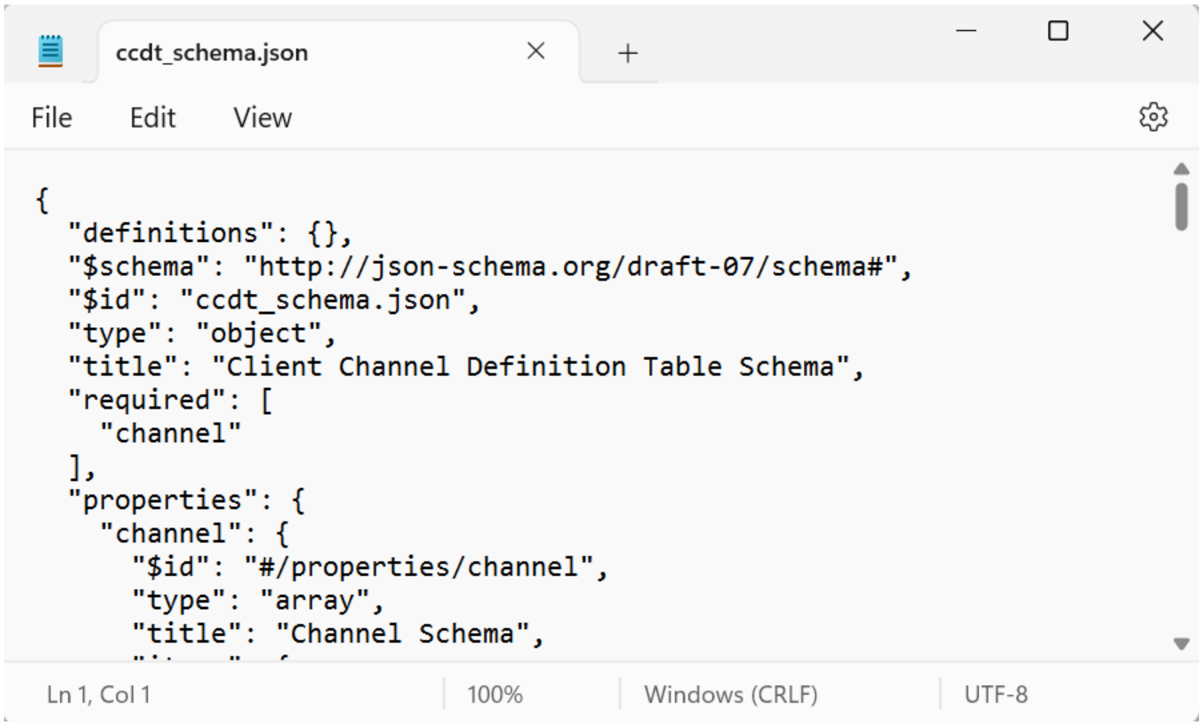
```
{
}
```

For the Schema, you will need to get the following file from your IBM MQ installation:

Linux:
    /opt/mqm/lib/**ccdt_schema.json**

Windows:
    C:\Program Files\IBM\MQ\bin\ccdt_schema.json

View the file in a text editor and copy the whole contents, such as with Windows Notepad:

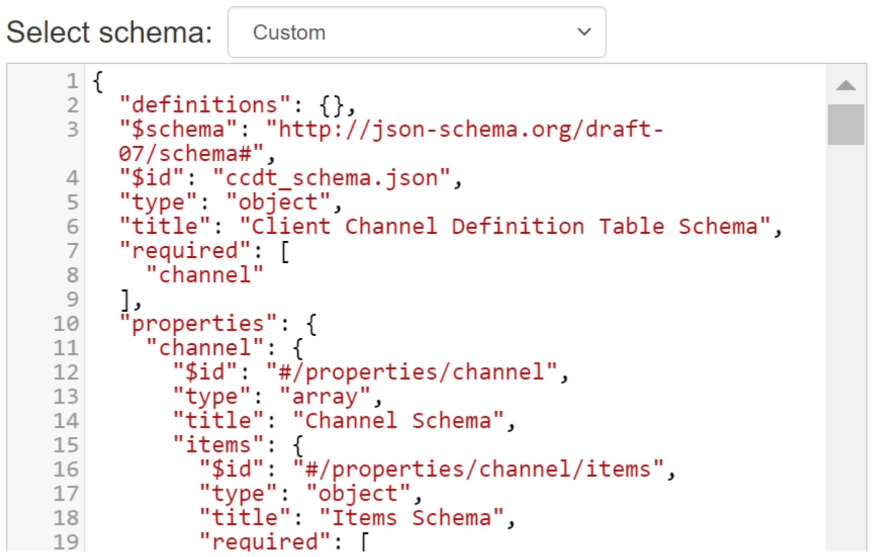


```
{
  "definitions": {},
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "ccdt_schema.json",
  "type": "object",
  "title": "Client Channel Definition Table Schema",
  "required": [
    "channel"
  ],
  "properties": {
    "channel": {
      "$id": "#/properties/channel",
      "type": "array",
      "title": "Channel Schema",
```

Select All the text and copy it.

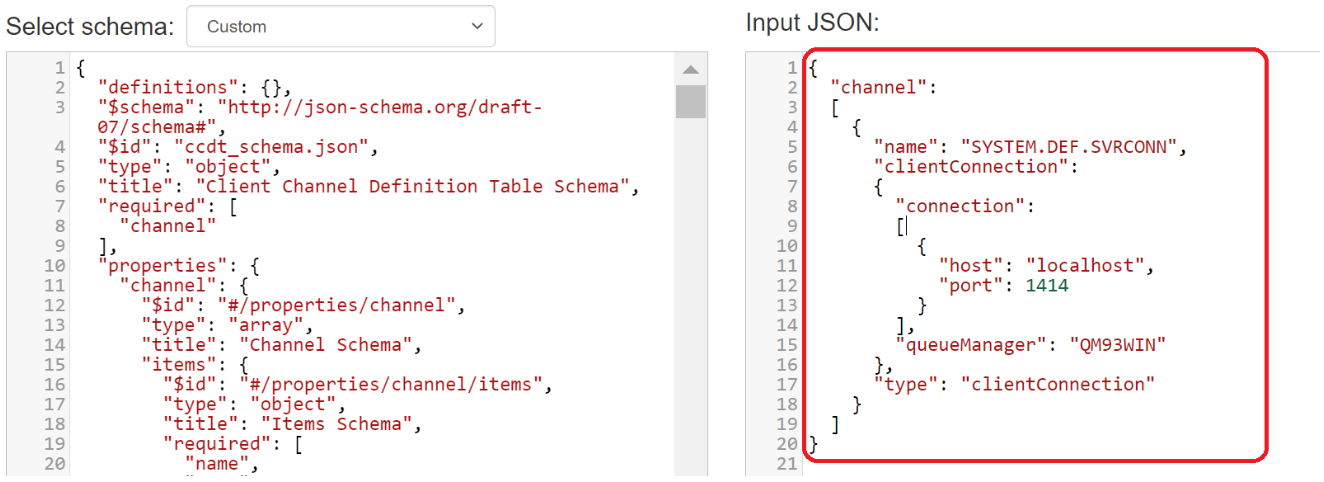
Then paste it in the left panel of the web page.
You need to completely replace the temporary few lines that were present in the left panel.

Now the left panel should look like this:


Select schema: Custom

```
{
  "definitions": {},
  "$schema": "http://json-schema.org/draft-
  07/schema#",
  "$id": "ccdt_schema.json",
  "type": "object",
  "title": "Client Channel Definition Table Schema",
  "required": [
    "channel"
  ],
  "properties": {
    "channel": {
      "$id": "#/properties/channel",
      "type": "array",
      "title": "Channel Schema",
      "items": {
        "$id": "#/properties/channel/items",
        "type": "object",
        "title": "Items Schema",
        "required": [
```

Select all the contents of your CCDT in JSON and paste it on the right panel.


Select schema: Custom

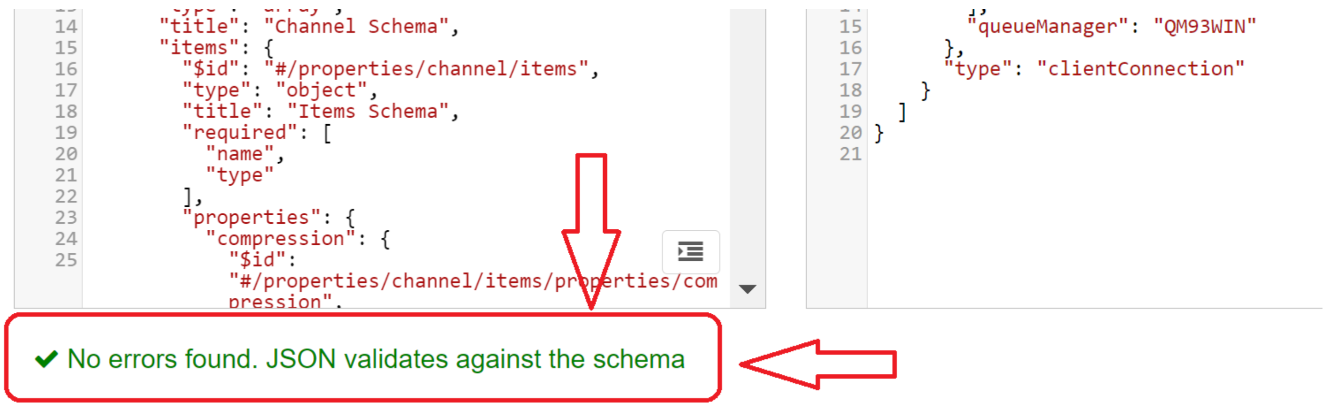
```
{
  "definitions": {},
  "$schema": "http://json-schema.org/draft-
  07/schema#",
  "$id": "ccdt_schema.json",
  "type": "object",
  "title": "Client Channel Definition Table Schema",
  "required": [
    "channel"
  ],
  "properties": {
    "channel": {
      "$id": "#/properties/channel",
      "type": "array",
      "title": "Channel Schema",
      "items": {
        "$id": "#/properties/channel/items",
        "type": "object",
        "title": "Items Schema",
        "required": [
          "name",
```

Input JSON:

```
{
  "channel":
  [
    {
      "name": "SYSTEM.DEF.SVRCONN",
      "clientConnection":
      {
        "connection":
        [
          {
            "host": "localhost",
            "port": 1414
          }
        ],
        "queueManager": "QM93WIN"
      },
      "type": "clientConnection"
    }
  ]
}
```

The text will be parsed and validated.

The result will be shown at the bottom of the screen:



```
    "title": "Channel Schema",
    "items": {
      "$id": "#/properties/channel/items",
      "type": "object",
      "title": "Items Schema",
      "required": [
        "name",
        "type"
      ],
      "properties": {
        "compression": {
          "$id":
          "#/properties/channel/items/properties/com
          pression",
```

```
        "queueManager": "QM93WIN"
      },
      "type": "clientConnection"
    }
  ]
}
```

✓ No errors found. JSON validates against the schema

++ Comments

The main reason that I wanted to find a way to validate the CCDT against the schema, is that I wrongly extrapolate how to construct a CCDT file with multiple channels.

For example, I created 2 separate CCDT JSON files with only 1 channel each:

File 1:

```
{    "channel":    [
    {        "name": "SYSTEM.DEF.SVRCONN",
      "clientConnection":        {
        "connection":          [
          {             "host": "localhost",   "port": 1414      }          ],
        "queueManager": "QM93WIN"       },
      "type": "clientConnection"     }    ]
}
```

And File2:

```
{    "channel":    [
    {        "name": "MY.SVRCONN",
      "clientConnection":        {
        "connection":          [
          {             "host": "localhost",   "port": 1415      }          ],
        "queueManager": "QM93REST"       },
      "type": "clientConnection"     }    ]
}
```

Each of the files worked fine with amqsputc as described in:

https://www.ibm.com/support/pages/node/6955535
Using a CCDT in JSON with SSL/TLS with the IBM MQ sample amqsputc

Then I decided to create a 3rd file with the contents of both:

File3 (named: ccdt-2-channels.json)

```
{   "channel":   [
    {       "name": "SYSTEM.DEF.SVRCONN",
      "clientConnection":       {
        "connection":         [
          {            "host": "localhost",  "port": 1414     }        ],
        "queueManager": "QM93WIN"      },
      "type": "clientConnection"    }   ]
}
{   "channel":   [
    {       "name": "MY.SVRCONN",
      "clientConnection":       {
        "connection":         [
          {            "host": "localhost",  "port": 1415     }        ],
        "queueManager": "QM93REST"      },
      "type": "clientConnection"    }   ]
}
```

… it did not work as expected.

The 1st channel in the list was found fine and it worked fine.

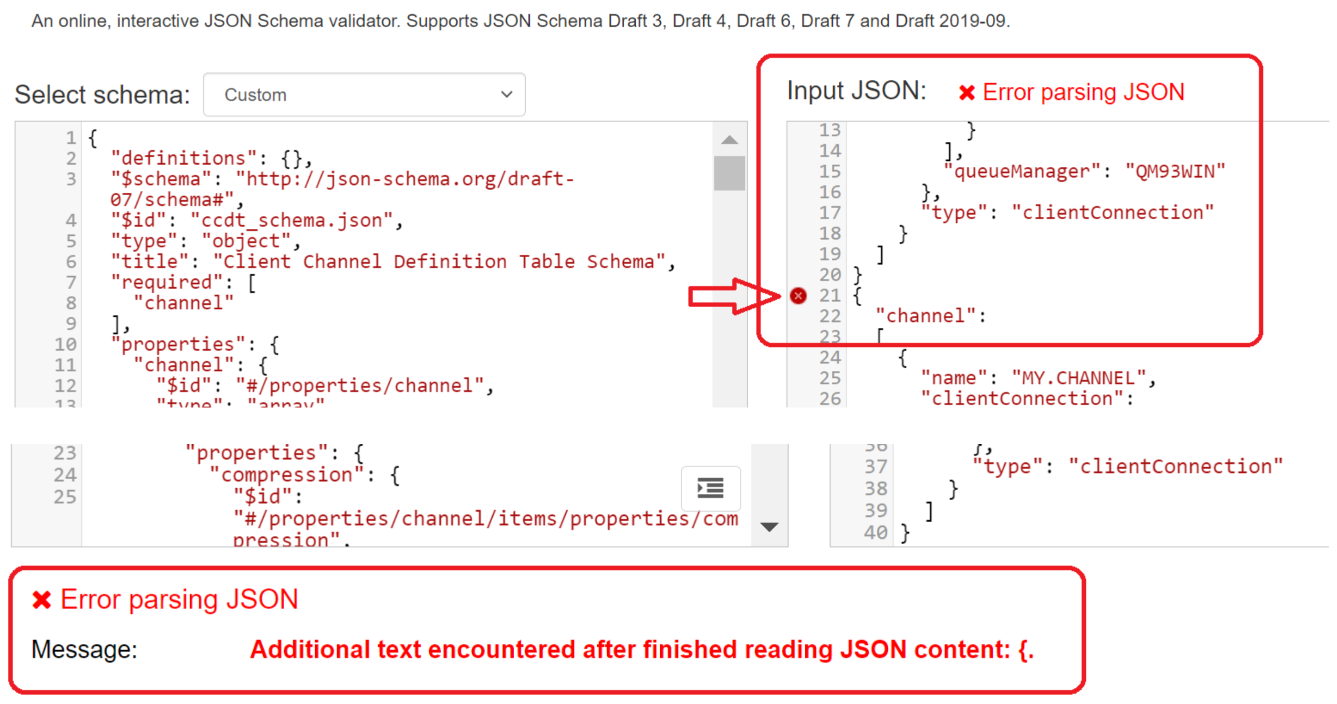
```
C:\> set MQCCDTURL=file:/c:/angel/coding/python/json/ccdt-2-channels.json

C:\>amqsputc Q1 QM93WIN
Sample AMQSPUT0 start
target queue is Q1
test
Sample AMQSPUT0 end
```

But the 2nd channel failed.

```
C:\angel\coding\python\json>amqsputc Q1 QM93REST
Sample AMQSPUT0 start
MQCONNX ended with reason code 2058!
```

Thus, I asked myself if that simple concatenation of 2 CCDT JSON files was correct.
Well, it turns out that it is NOT correct!

I used the JSON Schema Validator and I pasted the entire contents of the 3rd file (with the 2 channels, simple concatenation) and the validator reported a problem:



An online, interactive JSON Schema validator. Supports JSON Schema Draft 3, Draft 4, Draft 6, Draft 7 and Draft 2019-09.

Select schema: Custom

```
{
  "definitions": {},
  "$schema": "http://json-schema.org/draft-
07/schema#",
  "$id": "ccdt_schema.json",
  "type": "object",
  "title": "Client Channel Definition Table Schema",
  "required": [
    "channel"
  ],
  "properties": {
    "channel": {
      "$id": "#/properties/channel",
      "type": "array",
```

```
        "properties": {
          "compression": {
            "$id":
            "#/properties/channel/items/properties/com
pression",
```

Input JSON: ✖ Error parsing JSON

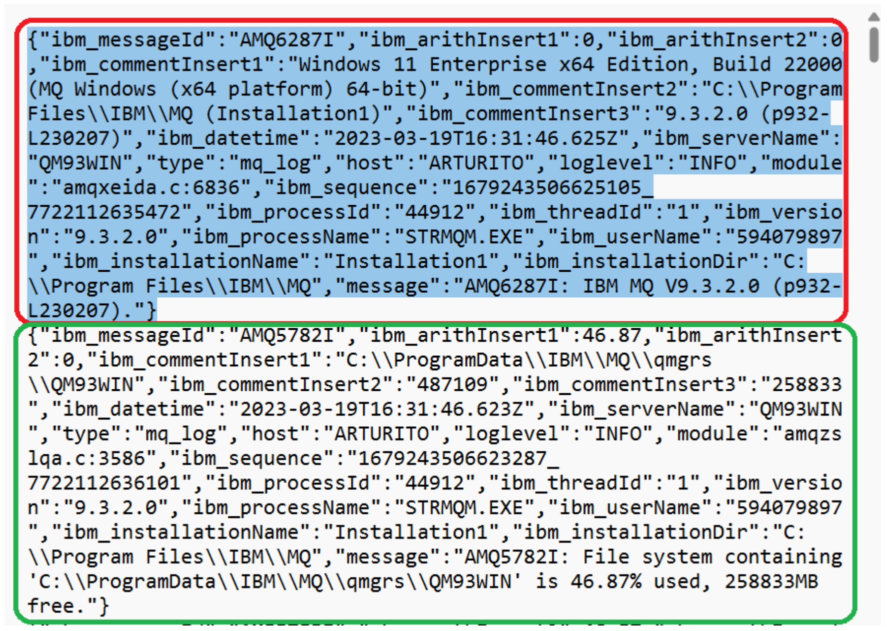
```
            }
          ],
          "queueManager": "QM93WIN"
        },
        "type": "clientConnection"
      }
    ]
}
{
  "channel":
  [
    {
      "name": "MY.CHANNEL",
      "clientConnection":
```

```
        },
        "type": "clientConnection"
      }
    ]
}
```

✖ Error parsing JSON

Message: **Additional text encountered after finished reading JSON content: {.**

The main reason that I thought that the simple concatenation would work ok, was because I experimented with the JSON format of the error logs for the queue manager.
In the output file, each entry is an individual JSON object, and it is concatenated at the bottom of the file, such as:



```
{"ibm_messageId":"AMQ6287I","ibm_arithInsert1":0,"ibm_arithInsert2":0
,"ibm_commentInsert1":"Windows 11 Enterprise x64 Edition, Build 22000
(MQ Windows (x64 platform) 64-bit)","ibm_commentInsert2":"C:\\Program
Files\\IBM\\MQ (Installation1)","ibm_commentInsert3":"9.3.2.0 (p932-
L230207)","ibm_datetime":"2023-03-19T16:31:46.625Z","ibm_serverName":
"QM93WIN","type":"mq_log","host":"ARTURITO","loglevel":"INFO","module
":"amqxeida.c:6836","ibm_sequence":"1679243506625105_
7722112635472","ibm_processId":"44912","ibm_threadId":"1","ibm_versio
n":"9.3.2.0","ibm_processName":"STRMQM.EXE","ibm_userName":"594079897
","ibm_installationName":"Installation1","ibm_installationDir":"C:
\\Program Files\\IBM\\MQ","message":"AMQ6287I: IBM MQ V9.3.2.0 (p932-
L230207)."}
{"ibm_messageId":"AMQ5782I","ibm_arithInsert1":46.87,"ibm_arithInsert
2":0,"ibm_commentInsert1":"C:\\ProgramData\\IBM\\MQ\\qmgrs
\\QM93WIN","ibm_commentInsert2":"487109","ibm_commentInsert3":"258833
","ibm_datetime":"2023-03-19T16:31:46.623Z","ibm_serverName":"QM93WIN
","type":"mq_log","host":"ARTURITO","loglevel":"INFO","module":"amqzs
lqa.c:3586","ibm_sequence":"1679243506623287_
7722112636101","ibm_processId":"44912","ibm_threadId":"1","ibm_versio
n":"9.3.2.0","ibm_processName":"STRMQM.EXE","ibm_userName":"594079897
","ibm_installationName":"Installation1","ibm_installationDir":"C:
\\Program Files\\IBM\\MQ","message":"AMQ5782I: File system containing
'C:\\ProgramData\\IBM\\MQ\\qmgrs\\QM93WIN' is 46.87% used, 258833MB
free."}
```

Like the saying … "if everything else fails, then read the instructions!" … I looked at the online manual and I found an example that has 2 channels:

https://www.ibm.com/docs/en/ibm-mq/9.3?topic=ccdt-json-examples
IBM MQ / 9.3
JSON CCDT examples
.
Define two channels with the same name
.
Each channel connects to two distinct queue managers:

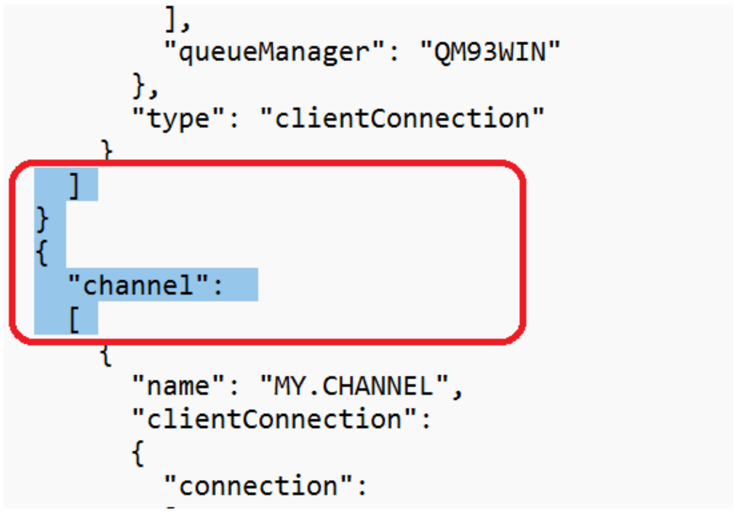
```
{
  "channel":
  [
    {
      "general":
      {
        "description": "First channel"
      },
      "name": "channel",
      "clientConnection":
      {
        "connection":
        [
          {
            "host": "localhost",
            "port": 1414
          }
        ],
        "queueManager": "QM1"
      },
      "type": "clientConnection"
    },
    {
      "general":
      {
        "description": "Second channel"
      },
      "name": "channel",
      "clientConnection":
      {
        "connection":
        [
          {
            "host": "localhost",
            "port": 1415
          }
        ],
        "queueManager": "QM2"
      },
      "type": "clientConnection"
    }
  ]
}
```

I noticed that there is ONLY one instance of the keyword: channel
And there is an array of 2 elements, one element per channel.

Thus, I cannot just do a simple concatenation of 2 files.
I have to merge the contents in order to imitate the sample from the online manual.

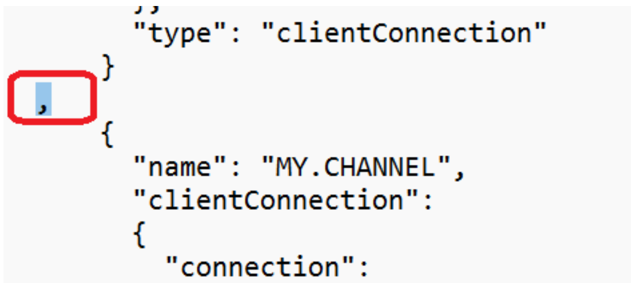
This is what I did to fix it:

I had to cut the following lines (bottom of 1st element and top of 2nd element)



```
            ],
            "queueManager": "QM93WIN"
          },
          "type": "clientConnection"
        }
      ]
    }
    {
      "channel":
      [
        {
          "name": "MY.CHANNEL",
          "clientConnection":
          {
            "connection":
```

And add a comma between the 2 curly brackets:

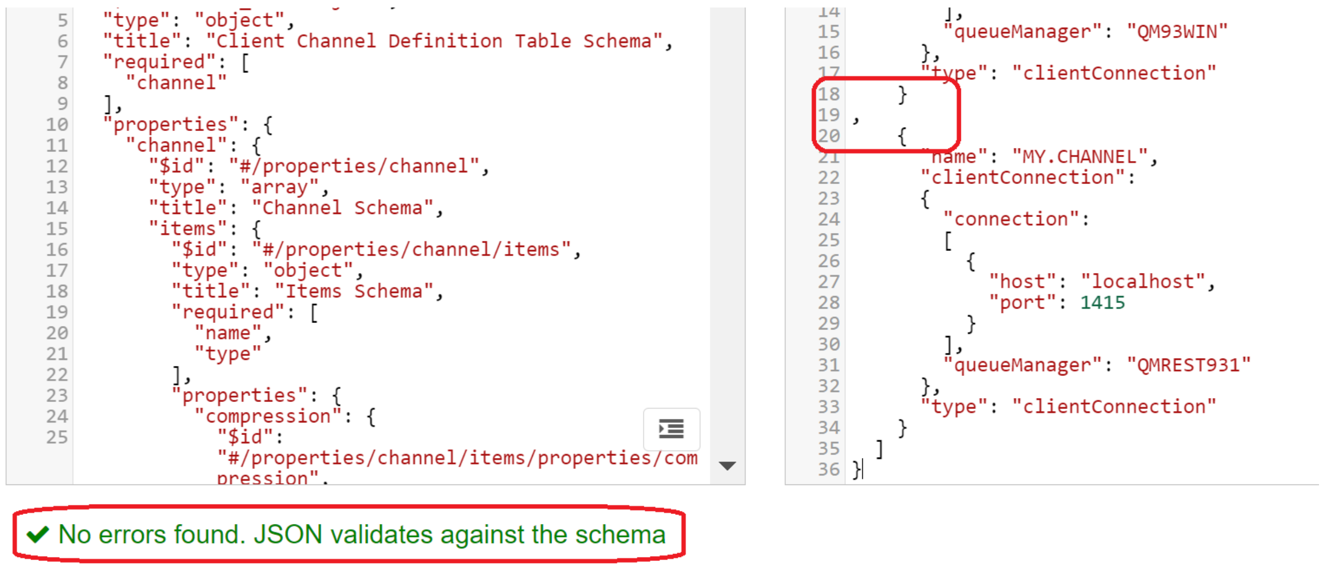


```
          "type": "clientConnection"
        }
    ,
        {
          "name": "MY.CHANNEL",
          "clientConnection":
          {
            "connection":
```

And now it worked fine when I used both channels:

```
C:\> set MQCCDTURL=file:/c:/angel/coding/python/json/ccdt-2-channels-localhost-ok.json

C:\> amqsputc Q1 QM93WIN
Sample AMQSPUT0 start
target queue is Q1
test
Sample AMQSPUT0 end

C:\> amqsputc Q1 QMREST931
Sample AMQSPUT0 start
target queue is Q1
test
Sample AMQSPUT0 end
```

To confirm the validation of the corrected file with the Schema, I pasted the corrected contents in the web page and the validation was successful!



```
"type": "object",
"title": "Client Channel Definition Table Schema",
"required": [
  "channel"
],
"properties": {
  "channel": {
    "$id": "#/properties/channel",
    "type": "array",
    "title": "Channel Schema",
    "items": {
      "$id": "#/properties/channel/items",
      "type": "object",
      "title": "Items Schema",
      "required": [
        "name",
        "type"
      ],
      "properties": {
        "compression": {
          "$id":
          "#/properties/channel/items/properties/com
          pression",
```

```
      ],
      "queueManager": "QM93WIN"
    },
    "type": "clientConnection"
  }
,
  {
    "name": "MY.CHANNEL",
    "clientConnection":
    {
      "connection":
      [
        {
          "host": "localhost",
          "port": 1415
        }
      ],
      "queueManager": "QMREST931"
    },
    "type": "clientConnection"
  }
 ]
}
```

✓ No errors found. JSON validates against the schema

+++ end +++