

# SQL Server JSON to Table and Table to JSON

 [red-gate.com/simple-talk/blogs/sql-server-json-to-table-and-table-to-json](http://red-gate.com/simple-talk/blogs/sql-server-json-to-table-and-table-to-json)

Phil Factor

March 26, 2013

## Articles by Phil Factor about JSON and SQL Server:

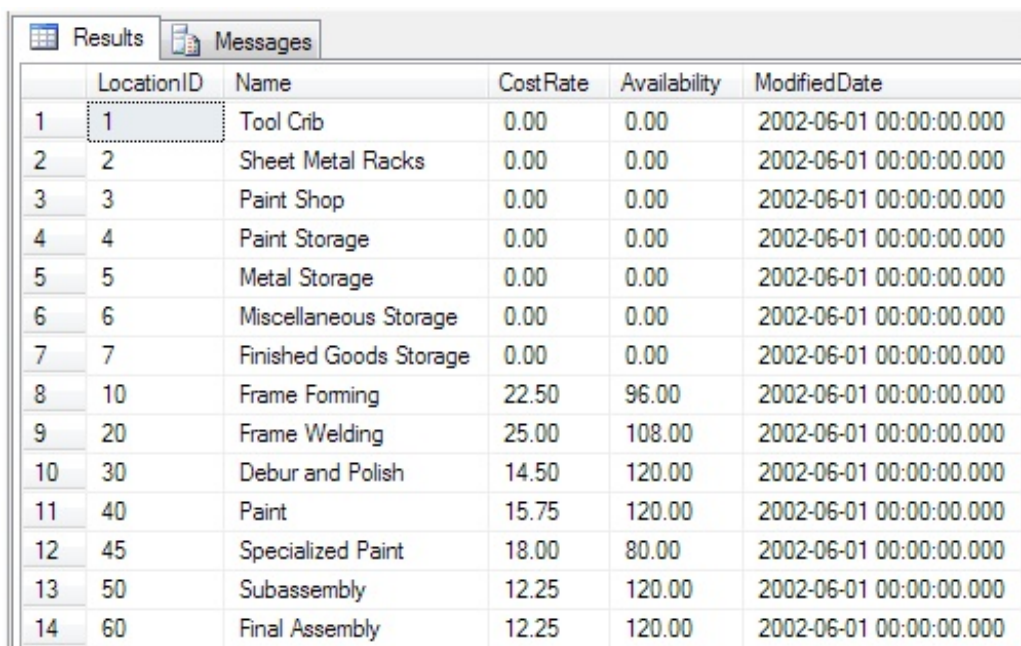
One of the surprises that I got from writing for Simple-Talk was the popularity of my article [Consuming JSON Strings in SQL Server](#). I hadn't really expected it to be so appreciated; in fact I was nervous about posting it at all. It came from a real requirement I had at the time, but I got interested in it in order to show how one could analyse hierarchical data documents iteratively in TSQL. Also, an anonymous troll on StackOverflow had told me it was impossible.

There were a few questions that I'd left unanswered. The first was how to read a JSON string as a table, and the other was how to produce a JSON document from a table. I ran out of space in the article and thought it was really the sort of thing that readers of the article would want to experiment with.

In this blog, I'll take the whole process around the circle. You'll notice that much of this is a bit cumbersome, but I only use short JSON Strings and haven't noticed a performance problem in doing this.

We start with a table.

- 1 Select \* from adventureworks.production.location



	LocationID	Name	CostRate	Availability	ModifiedDate
1	1	Tool Crib	0.00	0.00	2002-06-01 00:00:00.000
2	2	Sheet Metal Racks	0.00	0.00	2002-06-01 00:00:00.000
3	3	Paint Shop	0.00	0.00	2002-06-01 00:00:00.000
4	4	Paint Storage	0.00	0.00	2002-06-01 00:00:00.000
5	5	Metal Storage	0.00	0.00	2002-06-01 00:00:00.000
6	6	Miscellaneous Storage	0.00	0.00	2002-06-01 00:00:00.000
7	7	Finished Goods Storage	0.00	0.00	2002-06-01 00:00:00.000
8	10	Frame Forming	22.50	96.00	2002-06-01 00:00:00.000
9	20	Frame Welding	25.00	108.00	2002-06-01 00:00:00.000
10	30	Debur and Polish	14.50	120.00	2002-06-01 00:00:00.000
11	40	Paint	15.75	120.00	2002-06-01 00:00:00.000
12	45	Specialized Paint	18.00	80.00	2002-06-01 00:00:00.000
13	50	Subassembly	12.25	120.00	2002-06-01 00:00:00.000
14	60	Final Assembly	12.25	120.00	2002-06-01 00:00:00.000

And by a quick use of ...

```

1  Select '("+Convert(varchar(3),LocationID)+", "+ Name+",
    "+Convert(varchar(10),CostRate)
2      +", "+Convert(varchar(10),Availability)+",
3  "+Convert(varchar(10),ModifiedDate,126)+")',
    from adventureworks.production.location

```

We can convert it to a PowerShell array that we can use as if it were a table

```

1  @(("1", "Tool Crib", "0.00", "0.00", "2002-06-01"),
2  ("2", "Sheet Metal Racks", "0.00", "0.00", "2002-06-01"),
3  ("3", "Paint Shop", "0.00", "0.00", "2002-06-01"),
4  ("4", "Paint Storage", "0.00", "0.00", "2002-06-01"),
5  ("5", "Metal Storage", "0.00", "0.00", "2002-06-01"),
6  ("6", "Miscellaneous Storage", "0.00", "0.00", "2002-06-01"),
7  ("7", "Finished Goods Storage", "0.00", "0.00", "2002-06-01"),
8  ("10", "Frame Forming", "22.50", "96.00", "2002-06-01"),
9  ("20", "Frame Welding", "25.00", "108.00", "2002-06-01"),
10 ("30", "Debur and Polish", "14.50", "120.00", "2002-06-01"),
11 ("40", "Paint", "15.75", "120.00", "2002-06-01"),
12 ("45", "Specialized Paint", "18.00", "80.00", "2002-06-01"),
13 ("50", "Subassembly", "12.25", "120.00", "2002-06-01"),
14 ("60", "Final Assembly", "12.25", "120.00", "2002-06-01")
15 )|
16 Select @{ name = "LocationID"; Expression = { $_[0] } },
17   @{ name = "Name"; Expression = { $_[1] } },
18   @{ name = "CostRate"; Expression = { $_[2] } },
19   @{ name = "Availability"; Expression = { $_[3] } },
20   @{ name = "ModifiedDate"; Expression = { (Get-Date $_[4]).DateTime } } |
21   convertTo-json

```

... and this we can use to get the JSON version of the SQL Table, and from there to

JSON...

```
1
2  [
3    {
4      "CostRate": "0.0000",
5      "ModifiedDate": "06/01/1998 00:00:00",
6      "Name": "Tool Crib",
7      "Availability": "0.00",
8      "LocationID": "1"
9    },
10   {
11     "CostRate": "0.0000",
12     "ModifiedDate": "06/01/1998 00:00:00",
13     "Name": "Sheet Metal Racks",
14     "Availability": "0.00",
15     "LocationID": "2"
16   },
17   {
18     "CostRate": "0.0000",
19     "ModifiedDate": "06/01/1998 00:00:00",
20     "Name": "Paint Shop",
21     "Availability": "0.00",
22     "LocationID": "3"
23   },
24   {
25     "CostRate": "0.0000",
26     "ModifiedDate": "06/01/1998 00:00:00",
27     "Name": "Paint Storage",
28     "Availability": "0.00",
```

```
29     "LocationID": "4"
30 },
31 {
32     "CostRate": "0.0000",
33     "ModifiedDate": "06/01/1998 00:00:00",
34     "Name": "Metal Storage",
35     "Availability": "0.00",
36     "LocationID": "5"
37 },
38 {
39     "CostRate": "0.0000",
40     "ModifiedDate": "06/01/1998 00:00:00",
41     "Name": "Miscellaneous Storage",
42     "Availability": "0.00",
43     "LocationID": "6"
44 },
45 {
46     "CostRate": "0.0000",
47     "ModifiedDate": "06/01/1998 00:00:00",
48     "Name": "Finished Goods Storage",
49     "Availability": "0.00",
50     "LocationID": "7"
51 },
52 {
53     "CostRate": "22.5000",
54     "ModifiedDate": "06/01/1998 00:00:00",
55     "Name": "Frame Forming",
56     "Availability": "96.00",
57     "LocationID": "10"
```

```
58     },
59     {
60         "CostRate": "25.0000",
61         "ModifiedDate": "06/01/1998 00:00:00",
62         "Name": "Frame Welding",
63         "Availability": "108.00",
64         "LocationID": "20"
65     },
66     {
67         "CostRate": "14.5000",
68         "ModifiedDate": "06/01/1998 00:00:00",
69         "Name": "Debur and Polish",
70         "Availability": "120.00",
71         "LocationID": "30"
72     },
73     {
74         "CostRate": "15.7500",
75         "ModifiedDate": "06/01/1998 00:00:00",
76         "Name": "Paint",
77         "Availability": "120.00",
78         "LocationID": "40"
79     },
80     {
81         "CostRate": "18.0000",
82         "ModifiedDate": "06/01/1998 00:00:00",
83         "Name": "Specialized Paint",
84         "Availability": "80.00",
85         "LocationID": "45"
86     },
```

```

87     {
88         "CostRate": "12.2500",
89         "ModifiedDate": "06/01/1998 00:00:00",
90         "Name": "Subassembly",
91         "Availability": "120.00",
92         "LocationID": "50"
93     },
94     {
95         "CostRate": "12.2500",
96         "ModifiedDate": "06/01/1998 00:00:00",
97         "Name": "Final Assembly",
98         "Availability": "120.00",
99         "LocationID": "60"
100    }
101 ]

```

We could, of course, have got the table as a Datable by ADO.NET and converted that to JSON

```

1  $SourceTable = 'production.location'
2  $Sourceinstance = 'MyInstanceName'
3  $Sourcedatabase = 'Adventureworks'
4
5  $SourceConnectionString = "Data Source=$Sourceinstance;Initial
6  Catalog=$Sourcedatabase;Integrated Security=True"
7  $sql = "select * FROM $SourceTable"
8  $result = @()
9  try
10 {
11     $sourceConnection = New-Object
12     System.Data.SqlClient.SQLConnection($SourceConnectionString)

```

```

12     $sourceConnection.open()
13     $commandSourceData = New-Object
system.Data.SqlClient.SqlCommand($sql, $sourceConnection)
14
15     $reader = $commandSourceData.ExecuteReader()
16
17     $Counter = $Reader.FieldCount
18     while ($Reader.Read())
19     {
20         $tuple = @{ }
21         for ($i = 0; $i -lt $Counter; $i++)
22         {
23             $tuple."$($Reader.GetName($i))" = "$(if
24 ($Reader.GetFieldType($i).Name -eq 'DateTime')
25             { $Reader.GetDateTime($i) }
26             else { $Reader.GetValue($i) })";
27         }
28         $Result += $tuple
29     }
30     catch
31     {
32         $ex = $_.Exception
33         Write-Error "whilst opening source $Sourceinstance . $Sourcedatabase .
34 $SourceTable : $ex.Message"
35     }
36     finally
37     {
38         $reader.close()
39     }

```

Which produces the same JSON String. This is just a simple example: This will not cope

with a Varbinary binary, image, timestamp or rowversion field. You would need to convert it to a Base64 string

via **[System.Convert]::ToBase64String(\$Reader.GetValue(\$i)).**

To produce this directly from the table takes more effort. First you would need to install the library from my JSON article. You can convert the table to a hierarchy table.



```

1      create TABLE #hierarchy
2      (
3          element_id INT IDENTITY(1, 1) NOT NULL, /* internal surrogate primary
4          key gives the order of parsing and the list order */
5          parent_ID INT, /* if the element has a parent then it is in this column. The
6          document is the ultimate parent, so you can get the structure from recursing from
7          the document */
8          Object_ID INT, /* each list or object has an object id. This ties all elements
9          to a parent. Lists are treated as objects here */
10         NAME NVARCHAR(2000), /* the name of the object */
11         StringValue NVARCHAR(MAX) NOT NULL, /*the string representation of
12         the value of the element. */
13         ValueType VARCHAR(10) NOT null /* the declared type of the value
14         represented as a string in StringValue*/
15     )
16
17     ;With loc (Roworder,locationID, Name, CostRate, Availability, ModifiedDate)
18     as
19     (
20         Select ROW_NUMBER() OVER ( ORDER BY locationID) as RowOrder,
21         LocationID, Name, CostRate, Availability, ModifiedDate
22     from Adventureworks.production.location
23     )
24
25     INSERT INTO #Hierarchy
26     (parent_ID,Object_ID,NAME,StringValue,ValueType)
27     Select Roworder,null,'LocationID', convert(varchar(5),LocationID),'int'
28     from loc
29
30     union all Select Roworder,null,'Name', Name ,'string' from loc
31
32     union all Select Roworder,null,'CostRate', convert(varchar(10),CostRate)
33     ,'real' from loc

```

```
union all Select Roworder,null,'Availability', convert(varchar(10),Availability)
,'real' from loc
```

```
union all Select Roworder,null,'ModifiedDate',
Convert(varchar(10),ModifiedDate,126) ,'string' from loc
```

```
union all Select (Select count(*) from loc)+1, ROW_NUMBER() OVER (
ORDER BY locationID ), NULL,'1','object' from loc
```

```
union all Select null, (Select count(*) from loc)+1,'-',",','array'
```

from that point, it is easy to create the JSON string

```
1 DECLARE @MyHierarchy JSONHierarchy
2 INSERT INTO @myHierarchy
3     SELECT * from #hierarchy
4
5 SELECT dbo.ToJSON(@MyHierarchy)
```

And it won't surprise you that the string is more or less identical to the one we got in PowerShell.

Of course, reading a JSON string as a table is rather easier.

```
1 Create view TableOfJSONString as
2
3     Select parent_ID,
4         max(case when name='LocationID' then convert(int,StringValue) else
5         0 end) as LocationID,
6         max(case when name='Name' then convert(Varchar(50),StringValue)
7         else " end) as Name,
8         max(case when name='CostRate' then
9         convert(SmallMoney,StringValue) else 0 end) as CostRate,
10        max(case when name='Availability' then
11        convert(Decimal(8,2),StringValue) else 0 end) as Availability,
12        max(case when name='ModifiedDate' then
13        convert(DateTime,StringValue) else 0 end) as ModifiedDate
14    from dbo.parseJSON( '['
15
16        {
```

```
14         "LocationID": "1",
15         "Name": "Tool Crib",
16         "CostRate": "0.00",
17         "Availability": "0.00",
18         "ModifiedDate": "01 June 2002 00:00:00"
19     },
20     {
21         "LocationID": "2",
22         "Name": "Sheet Metal Racks",
23         "CostRate": "0.00",
24         "Availability": "0.00",
25         "ModifiedDate": "01 June 2002 00:00:00"
26     },
27     {
28         "LocationID": "3",
29         "Name": "Paint Shop",
30         "CostRate": "0.00",
31         "Availability": "0.00",
32         "ModifiedDate": "01 June 2002 00:00:00"
33     },
34     {
35         "LocationID": "4",
36         "Name": "Paint Storage",
37         "CostRate": "0.00",
38         "Availability": "0.00",
39         "ModifiedDate": "01 June 2002 00:00:00"
40     },
41     {
42         "LocationID": "5",
```

```
43         "Name": "Metal Storage",
44         "CostRate": "0.00",
45         "Availability": "0.00",
46         "ModifiedDate": "01 June 2002 00:00:00"
47     },
48     {
49         "LocationID": "6",
50         "Name": "Miscellaneous Storage",
51         "CostRate": "0.00",
52         "Availability": "0.00",
53         "ModifiedDate": "01 June 2002 00:00:00"
54     },
55     {
56         "LocationID": "7",
57         "Name": "Finished Goods Storage",
58         "CostRate": "0.00",
59         "Availability": "0.00",
60         "ModifiedDate": "01 June 2002 00:00:00"
61     },
62     {
63         "LocationID": "10",
64         "Name": "Frame Forming",
65         "CostRate": "22.50",
66         "Availability": "96.00",
67         "ModifiedDate": "01 June 2002 00:00:00"
68     },
69     {
70         "LocationID": "20",
71         "Name": "Frame Welding",
```

```
72         "CostRate": "25.00",
73         "Availability": "108.00",
74         "ModifiedDate": "01 June 2002 00:00:00"
75     },
76     {
77         "LocationID": "30",
78         "Name": "Debur and Polish",
79         "CostRate": "14.50",
80         "Availability": "120.00",
81         "ModifiedDate": "01 June 2002 00:00:00"
82     },
83     {
84         "LocationID": "40",
85         "Name": "Paint",
86         "CostRate": "15.75",
87         "Availability": "120.00",
88         "ModifiedDate": "01 June 2002 00:00:00"
89     },
90     {
91         "LocationID": "45",
92         "Name": "Specialized Paint",
93         "CostRate": "18.00",
94         "Availability": "80.00",
95         "ModifiedDate": "01 June 2002 00:00:00"
96     },
97     {
98         "LocationID": "50",
99         "Name": "Subassembly",
100        "CostRate": "12.25",
```

```

101         "Availability": "120.00",
102         "ModifiedDate": "01 June 2002 00:00:00"
103     },
104     {
105         "LocationID": "60",
106         "Name": "Final Assembly",
107         "CostRate": "12.25",
108         "Availability": "120.00",
109         "ModifiedDate": "01 June 2002 00:00:00"
110     }
111 ]
112 '
    )

    where ValueType = 'string'

    group by parent_ID

```

And we can just then use it in any SQL Expression like this

```
1    Select * from TableOfJSONString
```

	parent_ID	LocationID	Name	CostRate	Availability	ModifiedDate
1	1	1	Tool Crib	0.00	0.00	2002-06-01 00:00:00.000
2	2	2	Sheet Metal Racks	0.00	0.00	2002-06-01 00:00:00.000
3	3	3	Paint Shop	0.00	0.00	2002-06-01 00:00:00.000
4	4	4	Paint Storage	0.00	0.00	2002-06-01 00:00:00.000
5	5	5	Metal Storage	0.00	0.00	2002-06-01 00:00:00.000
6	6	6	Miscellaneous Storage	0.00	0.00	2002-06-01 00:00:00.000
7	7	7	Finished Goods Storage	0.00	0.00	2002-06-01 00:00:00.000
8	8	10	Frame Forming	22.50	96.00	2002-06-01 00:00:00.000
9	9	20	Frame Welding	25.00	108.00	2002-06-01 00:00:00.000
10	10	30	Debur and Polish	14.50	120.00	2002-06-01 00:00:00.000
11	11	40	Paint	15.75	120.00	2002-06-01 00:00:00.000
12	12	45	Specialized Paint	18.00	80.00	2002-06-01 00:00:00.000
13	13	50	Subassembly	12.25	120.00	2002-06-01 00:00:00.000
14	14	60	Final Assembly	12.25	120.00	2002-06-01 00:00:00.000

You'll notice that I've given you a 'parent\_ID' to give you the intrinsic order of the rows, since these things can be significant in a JSON document.

Of course, you can do some dynamic SQL to deal with any JSON String, but I don't like to do this since there is no guarantee that a JSON string represents table data. Also, when you are getting json strings from an application repeatedly, they tend to carry the same metadata.