OData ***Compact*** JSON Format v4.0

Abstract

The Open Data Protocol (OData) is a set of specifications for representing and interacting with structured content. It comprises of both core specifications and a JSON format specification specific to OData.

This document describes a compact JSON format for OData which can act as a lossless alternative JSON format for representing OData responses. Its prime aim is to minimize the, uncompressed, size of the responses returned by the service.

Table of Contents

Introduction

The OData protocol is comprised of a set of specifications for representing and interacting with structured content. Next to the core specification it defines representations for the OData requests and responses of the core protocol using the JavaScript Object Notation (JSON), see RFC7159. See Related work section for more information about these specifications.

This document defines a lossless, alternative, JSON format for OData responses.

Processing of large volumes of JSON has proven to be expensive, largely due to its verbosity. And while compression on the wire takes most of that inefficiency away during transport, the client consuming the, uncompressed, response will still end up dealing with large volumes of JSON.

The aproach taken in this compact format, inspired by traditional rowset interfaces and newer ideas behind recursive JSON, is to use, whenever possible, arrays for collections of properties that repeat. The great thing about JSON arrays is that the order of items in the array is retained whereas the order of properties in an object, effectively a map, is not.

OData's services already use the OData Common Schema Definition Language (CSDL) to specify all the, structured, types supported by the service. This format, unless otherwise explicitly specified, takes advantage of the order of the properties as specified in the metadata, using it to specify a 'default' order in which the properties will be returned in this compact JSON format. The $select and $expand query options continue to be used to shape the result in which case the odata.context annotation returned in the response will specify which properties are returned as well as the order in which the included properties are returned.

This format was intended to be a lossless alternative for the OData JSON Format. Since positions in an array are used to represent values for properties, objects are used in cases where additional information, in the form of annotations, need to be included in relationship to such property. The actual value, if such a value is to be included, is in those cases represented by a value property, following the pattern of the root object in the response.

Terminology

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC2119 and updated by RFC8174 when, and only when, they appear in all capitals, as shown here.

Related work

This specification is related to:

- OData Version 4.0, a multi-part Work Product which includes:
  - OData Version 4.0 Part 1: Protocol
  - OData Version 4.0 Part 2: URL Conventions

    - OData Version 4.0 Part 3: Common Schema Definition Language (CSDL)
    - OData Version 4.0 ABNF Construction Rules and Test Cases
- OData JSON Format Version 4.0

Compact JSON Format Design

JSON, as described in RFC7159, define a text format for serializing structured data. Objects are serialized as an unordered collection of name/value pairs.

JSON does not define any semantics around the name/value pairs that make up an object, nor does it define an extensibility mechanism for adding control information to a payload.

The OData JSON Format extends JSON for the purpose of defining representations for OData requests and responses using a JSON format.

This ***compact*** JSON format defines an alternative representation for responses to the OData JSON Format, heavily taking advantage of arrays and the fact that arrays in JSON have a guaranteed order.

The main difference between this ***compact*** JSON format and the OData JSON Format is that in the ***compact*** JSON format structured types are represented by a JSON array as opposed to a JSON object. The order in which the properties of such a structured type is represented in such array can be derived from the odata.context annotation in combination with the, in the odata.context referenced, metadata for the service. Unlike the OData JSON Format, as size and position of the items in the JSON arrays matters, the ***compact*** JSON format MUST return a value for every property considered to be reflected in the response and MUST NOT return any additional properties.

This document defines the specifics for the ***compact*** JSON format where it deviates from the OData JSON Format and as such is not intended to be a full OData format specification. Please see the OData JSON Format for additional details.

Requesting the ***Compact*** JSON Format

The ***compact*** JSON format can be requested using the $format query option in the request URL with the media type application/json followed by the compact=true format parameter (read: application/json;compact=true), optionally followed by other format parameters.

Alternatively, this format can be requested using the Accept header with the media type application/json followed by the compact=true format parameter, optionally followed by other format parameters.

If specified, $format overrides any value specified in the Accept header.

The OData JSON Format furthermore specifies additional format parameters which are presumed to have the same meaning with this ***compact*** JSON format. The following format parameters are however implied to have default values that cannot be overwritten:

- metadata (odata.metadata), is presumed to be always none
- streaming (odata.streaming), is presumed to be always true

Services SHOULD advertise the supported media types by annotating the entity container with the term Capabilities.SupportedFormats defined in OData Capabilities Vocabulary, listing all available formats and combinations of supported format parameters, including the compact=true format parameter if the ***compact*** JSON format is supported by the service.

Compact JSON Format Characteristics

This section describes the specific characteristics of the ***compact*** JSON format and how information returned in a reponse is represented in JSON. The ***compact*** JSON format MUST NOT be used for requests, only for responses generated by a service when, and only when, explicitedly requested to apply the compact format by means of the compact format parameter as describe above.

Header Content-Type

Responses with a JSON message body, formated using the ***compact*** JSON format, MUST have a Content-Type header value of application-json and MUST include the compact parameter with a value of true.

Responses MUST include the metadata parameter to specify the amount of metadata included in the response with a value of none.

Responses MUST include the IEEE754Compatible parameter if Edm.Int64 and Edm.Decimal numbers are represented as strings.

Responses MUST add the streaming parameter with a value of true. The ***compact*** JSON format by design guarantees the order in which data is returned in the response and as such also follows all the rules specified in the Payload Ordering Constraints as defined in the OData JSON Format.

Message Body

Like in the OData JSON Format each message body is represented as a single JSON object.

This object MUST contain the odata.context, might contain additional annotations and a name/value pair of which the name MUST be value and the

value containing the representation of the entity, entity reference, complex type instance, primitive value, collection of entities, collection of entity references, collection of complex values or collection of primitive values, if the request warrents such a value to be returned.

Client libraries MUST retain the order of objects within an array in JSON responses.

Service Document

Requests for the service document are not supported using the ***compact*** json format. Request for the service document should include a content type in the accept header that can be serviced using the OData JSON Format.

Annotations

Annotations, when included in the response, should be wrapped in a JSON object. Annotations applying to the root value being returned, irrespective of the fact if that value represents an entity, an entity reference, a complex type instance, aprimitive value, acollection of entities, a collection of entity references, a collection of complex values or a collection of primitive values, should be returned in the root object. Any annotations applying to a property or navigation property, should be wrapped in a JSON object containing such annotation(s) and, if so required, a name/value pair of which the name MUST be value and the value containing the representing of the property or navigation property.

Note that this follows the same pattern then extensions to the JSON format proposed for OData v4.01.

Control Information: odata.context

The context annotation returns the context URL (see OData Protocol) for the payload. This URL can be absolute or relative.

Even though a response to a request using the ***compact*** JSON format is considered to have the metadata parameter for the format set to none, the context annotation is nevertheless included as it specifies vital information to the client for understanding the contents of the response.

Entity

The values for the properties of an entity are serialized as a JSON array. Each property to be transmitted is represented by just it's value, as such the order in which these property values appear in the array is significant. The value for each property is formatted as appropriate for the type of the property.

The order in which the values for the transmitted properties are being returned MUST be the same as the order in which the structural properties and navigation properties of the Entity Type have been specified in the the metadata for the service. For Entity Types that have a base type the properties of the base type are presumed to come before any of the properties of the sub type.

An entity in a payload may be the complete entity or a partial entity (see $select system query option in OData Protocol). The context returned in every response using the odata.context annotation can be used to find out the actual list of properties returned in the payload.

The service MUST NOT include any additional properties over and above those specified by the metadata or explicitly requested by means of the $select and/or $expand query option, in which case the 'known' additional and/or dynamic properties can be included in the response.

Complex Value

The values for the properties of a complex type are serialized as a JSON array. Each property to be transmitted is represented by just it's value, as such the order in which these property values appear in the array is significant. The value for each property is formatted as appropriate for the type of the property.

The order in which the values for the transmitted properties are being returned MUST be the same as the order in which the structural properties and navigation properties for the Complex Type have been specified in the metadata for the service. For Complex Types that have a base type the properties of the base type are presumed to come before any of the properties of the sub type.

A complex value in a payload may contain all the properties that make up the complex type or a subset of the complex type's properties (see $select system query option in OData Protocol). The context returned in every response using the odata.context annotation can be used to find out the actual list of properties returned in the payload.

The service MUST NOT include any additional properties over and above those specified by the metadata or explicitly requested by means of the $select and/or $expand query option, in which case the 'known' additional and/or dynamic properties can be included in the response.

Open Types

Open Types allow clients to add properties dynamically to instances of an Entity Type or Complex Type if such type is tagged as being an Open Type (see OData Common Schema Definition Language (CSDL)).

A service MUST only include values for such dynamic properties if, and only if, they have been explicitly requested by means of the $select query option. The service will respond by including such dynamic property in the list of properties projected for the type, as reflected in the context represented in the odata.context annotation.

The service on the other hand MUST include a value for every dynamic property reported to be included in the response, or a null value if such property does not exist on instance being projected.

Delta Payload

Delta payloads are not supported in the ***compact*** JSON format. Please use OData JSON Format for deltas.

Appendix A: Examples

Example 1: Retrieving a single property of a 'Cube' from the Cubes entity set

```
// GET ~/Cubes('plan_BudgetPlan')?$format=application/json;odata.metadata=none
{
    "@odata.context": "$metadata#Cubes/$entity",
    "Name": "plan_BudgetPlan",
    "Rules": null,
    "DrillthroughRules": null,
    "LastSchemaUpdate": "2018-01-31T00:00:02.701Z",
    "LastDataUpdate": "2018-01-31T00:00:02.700Z",
    "Attributes": {
        "Caption": "Basis Budget"
    }
}
```

```
// GET ~/Cubes('plan_BudgetPlan')?$format=application/json;compact=true
{
    "@odata.context": "$metadata#Cubes/$entity",
    "value": [
        "plan_BudgetPlan",
        null,
        null,
        "2018-01-31T00:00:02.701Z",
        "2018-01-31T00:00:02.700Z",
        [
            "Basis Budget"
        ]
    ]
}
```

Example 2: Retrieve the names for the 'Cubes' from the Cubes entity set

```
// GET ~/Cubes?$select=Name?$format=application/json;odata.metadata=none
{
    "@odata.context": "$metadata#Cubes(Name)",
    "value": [
        {
            "Name": "plan_BudgetPlan"
        },
        {
            "Name": "plan_BudgetPlanLineItem"
        },
        {
            "Name": "plan_Control"
        },
        {
            "Name": "plan_ExchangeRate"
        },
        {
            "Name": "plan_Report"
        }
    ]
}
```

```
// GET ~/Cubes$select=Name?$format=application/json;compact=true
{
    "@odata.context": "$metadata#Cubes(Name)",
    "value": [
        [
            "plan_BudgetPlan"
        ],
```

```
        [
            "plan_BudgetPlanLineItem"
        ],
        [
            "plan_Control"
        ],
        [
            "plan_ExchangeRate"
        ],
        [
            "plan_Report"
        ]
    ]
}
```

Example 3: Retrieve the names and attributes, an open type, from the 'Views' navigation property of a Cube. Note that Views is a collection of 'View' and that 'View' is a base type of the types of the instances in the actual collection.

```
// GET ~/Cubes('plan_BudgetPlan')/Views?
$select=Name,Attributes/Caption,Attributes/Foo&$format=application/json;odata.metadata=none
{
    "@odata.context": "../$metadata#Cubes('plan_BudgetPlan')/Views(Name,Attributes/Caption,Attributes/Foo)",
    "value": [
        {
            "@odata.type": "#ibm.tm1.api.v1.NativeView",
            "Name": "budget_placeholder",
            "Attributes": {
                "Caption": "budget_placeholder"
            }
        },
        {
            "@odata.type": "#ibm.tm1.api.v1.NativeView",
            "Name": "budget_placeholder_all",
            "Attributes": {
                "Caption": "budget_placeholder_all"
            }
        }
    ]
}
```

```
// GET ~/Cubes('plan_BudgetPlan')/Views?
$select=Name,Attributes/Caption,Attributes/Foo&$format=application/json;compact=true
{
    "error": {
        "code": "278",
        "message": "Compact JSON content type can not be used to formulate a response to this request."
    }
}
```

Note that trying to use the ***abstract*** JSON format causes an invalid request being returned with an error message stating that the format is not supported for such request.

Example 4: Retrieve the names and attributes, an open type, from the 'Views' navigation property of a Cube after casting the navigation property to one of the 'View' sub types.

```
// GET ~/Cubes('plan_BudgetPlan')/Views/tm1.NativeView?
$select=Name,Attributes/Caption,Attributes/Foo&$format=application/json;odata.metadata=none
{
    "@odata.context":
"../../$metadata#Cubes('plan_BudgetPlan')/Views/ibm.tm1.api.v1.NativeView(Name,Attributes/Caption,Attributes/Fo
o)",
    "value": [
        {
            "Name": "budget_placeholder",
            "Attributes": {
                "Caption": "budget_placeholder"
            }
        },
        {
            "Name": "budget_placeholder_all",
```

```
            "Attributes": {
                "Caption": "budget_placeholder_all"
            }
        }
    ]
}
```

Note that compared to the previous example, due to the type case, the odata.type annotation is no longer being returned as all entities are of the same type.

```
// GET ~/Cubes('plan_BudgetPlan')/Views/tm1.NativeView?
$select=Name,Attributes/Caption,Attributes/Foo&$format=application/json;compact=true
{
    "@odata.context":
"../../$metadata#Cubes('plan_BudgetPlan')/Views/ibm.tm1.api.v1.NativeView(Name,Attributes/Caption,Attributes/Fo
o)",
    "value": [
        [
            "budget_placeholder",
            [
                "budget_placeholder",
                null
            ]
        ],
        [
            "budget_placeholder_all",
            [
                "budget_placeholder_all",
                null
            ]
        ]
    ]
}
```

Example 5: Retrieve the names and the dimension count, using $count on the Dimensions navigation property, for the 'Cubes' from the Cubes entity set

```
// GET ~/Cubes?$select=Name&$expand=Dimensions/$count&$format=application/json;odata.metadata=none
{
    "@odata.context": "$metadata#Cubes(Name,Dimensions)",
    "value": [
        {
            "Name": "plan_BudgetPlan",
            "Dimensions@odata.count": 7
        },
        {
            "Name": "plan_BudgetPlanLineItem",
            "Dimensions@odata.count": 7
        },
        {
            "Name": "plan_Control",
            "Dimensions@odata.count": 2
        },
        {
            "Name": "plan_ExchangeRate",
            "Dimensions@odata.count": 3
        },
        {
            "Name": "plan_Report",
            "Dimensions@odata.count": 6
        }
    ]
}
```

```
// GET ~/Cubes?$select=Name&$expand=Dimensions/$count&$format=application/json;compact=true
{
    "@odata.context": "$metadata#Cubes(Name,Dimensions)",
    "value": [
        [
            "plan_BudgetPlan",
```

```
            {
                "@odata.count": 7
            }
        ],
        [
            "plan_BudgetPlanLineItem",
            {
                "@odata.count": 7
            }
        ],
        [
            "plan_Control",
            {
                "@odata.count": 2
            }
        ],
        [
            "plan_ExchangeRate",
            {
                "@odata.count": 3
            }
        ],
        [
            "plan_Report",
            {
                "@odata.count": 6
            }
        ]
    ]
}
```

Note that the object takes the place of the Dimensions property to wrap, in this case the odata.count, annotation.

Example 6: Retrieve the names of the 'Cubes' and the names of the dimensions from the Dimensions navigation property from the Cubes entity set

```
// GET ~/Cubes?$select=Name&$expand=Dimensions($select=Name)&$format=application/json;odata.metadata=none
{
    "@odata.context": "$metadata#Cubes(Name,Dimensions(Name))",
    "value": [
        {
            "Name": "plan_BudgetPlan",
            "Dimensions": [
                {
                    "Name": "plan_version"
                },
                {
                    "Name": "plan_business_unit"
                },
                {
                    "Name": "plan_department"
                },
                {
                    "Name": "plan_chart_of_accounts"
                },
                {
                    "Name": "plan_exchange_rates"
                },
                {
                    "Name": "plan_source"
                },
                {
                    "Name": "plan_time"
                }
            ]
        },
        {
            "Name": "plan_BudgetPlanLineItem",
            "Dimensions": [
                {
                    "Name": "plan_version"
                },
                {
                    "Name": "plan_business_unit"
                },
```

```
                {
                    "Name": "plan_department"
                },
                {
                    "Name": "plan_chart_of_accounts"
                },
                {
                    "Name": "plan_exchange_rates"
                },
                {
                    "Name": "plan_lines"
                },
                {
                    "Name": "plan_time"
                }
            ]
        }
    ]
}
```

```
// GET ~/Cubes?$select=Name&$expand=Dimensions($select=Name)&$format=application/json;compact=true
{
    "@odata.context": "$metadata#Cubes(Name,Dimensions(Name))",
    "value": [
        [
            "plan_BudgetPlan",
            [
                [
                    "plan_version"
                ],
                [
                    "plan_business_unit"
                ],
                [
                    "plan_department"
                ],
                [
                    "plan_chart_of_accounts"
                ],
                [
                    "plan_exchange_rates"
                ],
                [
                    "plan_source"
                ],
                [
                    "plan_time"
                ]
            ]
        ],
        [
            "plan_BudgetPlanLineItem",
            [
                [
                    "plan_version"
                ],
                [
                    "plan_business_unit"
                ],
                [
                    "plan_department"
                ],
                [
                    "plan_chart_of_accounts"
                ],
                [
                    "plan_exchange_rates"
                ],
                [
                    "plan_source"
                ],
                [
                    "plan_time"
                ]           ]
```

```
        ]
    ]
}
```

Example 7: Retrieve the names of the 'Cubes', the dimension count and the names of the dimensions from the Dimensions navigation property from the Cubes entity set

```
// GET ~/Cubes?
$select=Name&$expand=Dimensions($select=Name,$count)&$format=application/json;odata.metadata=none
{
    "@odata.context": "$metadata#Cubes(Name,Dimensions(Name))",
    "value": [
        {
            "Name": "plan_BudgetPlan",
            "Dimensions@odata.count": 7,
            "Dimensions": [
                {
                    "Name": "plan_version"
                },
                {
                    "Name": "plan_business_unit"
                },
                {
                    "Name": "plan_department"
                },
                {
                    "Name": "plan_chart_of_accounts"
                },
                {
                    "Name": "plan_exchange_rates"
                },
                {
                    "Name": "plan_source"
                },
                {
                    "Name": "plan_time"
                }
            ]
        },
        {
            "Name": "plan_BudgetPlanLineItem",
            "Dimensions@odata.count": 7,
            "Dimensions": [
                {
                    "Name": "plan_version"
                },
                {
                    "Name": "plan_business_unit"
                },
                {
                    "Name": "plan_department"
                },
                {
                    "Name": "plan_chart_of_accounts"
                },
                {
                    "Name": "plan_exchange_rates"
                },
                {
                    "Name": "plan_lines"
                },
                {
                    "Name": "plan_time"
                }
            ]
        }
    ]
}
```

```
// GET ~/Cubes?$select=Name&$expand=Dimensions($select=Name,$count)&$format=application/json;compact=true
{
    "@odata.context": "$metadata#Cubes(Name,Dimensions(Name))",
```

```
    "value": [
        [
            "plan_BudgetPlan",
            {
                "@odata.count": 7,
                "value": [
                    [
                        "plan_version"
                    ],
                    [
                        "plan_business_unit"
                    ],
                    [
                        "plan_department"
                    ],
                    [
                        "plan_chart_of_accounts"
                    ],
                    [
                        "plan_exchange_rates"
                    ],
                    [
                        "plan_source"
                    ],
                    [
                        "plan_time"
                    ]
                ]
            }
        ],
        [
            "plan_BudgetPlanLineItem",
            {
                "@odata.count": 7,
                "value": [
                    [
                        "plan_version"
                    ],
                    [
                        "plan_business_unit"
                    ],
                    [
                        "plan_department"
                    ],
                    [
                        "plan_chart_of_accounts"
                    ],
                    [
                        "plan_exchange_rates"
                    ],
                    [
                        "plan_source"
                    ],
                    [
                        "plan_time"
                    ]
                ]
            }
        ]
    ]
}
```


Appendix B: Revision History

Version	Date	Editor(s)	Notes
Working Draft 01	2018-01-29	Hubert Heijkers	Initial version