Odata

OData is a query language that allows developers to interact with data using RESTful web services.

<https://www.odata.org/> - Open Odata

<https://sap.github.io/cloud-sdk/docs/js/overview> - SCP Odata lib

<https://www.odata.org/documentation/>

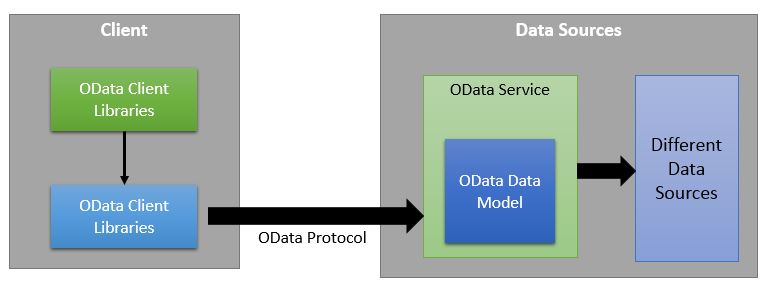
OData (Open Data Protocol) is an ISO/IEC approved, OASIS standard that defines a set of best practices for building and consuming REST APIs.

 Open Data Protocol (OData) is an open [protocol](https://en.wikipedia.org/wiki/Protocol_(computing)) that allows the creation and consumption of queryable and interoperable [Web service](https://en.wikipedia.org/wiki/Web_service) [APIs](https://en.wikipedia.org/wiki/Application_programming_interface) in a standard way. . [Microsoft](https://en.wikipedia.org/wiki/Microsoft) initiated OData in 2007.[[](https://en.wikipedia.org/wiki/Open_Data_Protocol#cite_note-1)

Technical Committee (TC).- After initial development by Microsoft, OData became a standardized protocol of the OASIS OData

OData architecture

OData is built on the REST (Representational State Transfer) architecture, which is based on the HTTP protocol.



**OData Protocol:**

OData protocol lets a client make requests to and get responses from an OData service.

OData protocol is a set of RESTful interactions.

OData protocol**is just like HTTP** and used to do CRUD operation by using OData.**It transfers the data in form of XML or JSON.**

**HTTP methods in OData:**

OData also offer more than just exposing content, it offers full CRUD support by using the different HTTP methods:

* GET: Gets one or many entries.
* POST: Create a new entry.
* PUT: Update an existing entry
* DELETE: Remove an entry.

Any platform that provides support for HTTP and XML is enough to form HTTP requests to interact with OData service.

Odata client library

* Microsoft .NET Framework 3.51: the**WCF Data Services framework**is available as a separate download for .NET 3.x.
* Java:**odata4j** (including Java on an Android phone) supports the OData protocol.
* JavaScript:**OpenUI5** library maintained by **SAP**

**ODATA V1**

Also known as A2X (SOAP) services, these can be used to integrate with non-SAP systems with or without middleware.

Odata has there own data type

Example .

|  |  |
| --- | --- |
| **Primitive Type** | **JSON Serialization Format** |
| Edm.Binary | Base64 encoded value of an EDM.Binary value represented as a JSON string |
| Edm.Boolean | true | false |
| Edm.Byte | Literal form of Edm.Byte as used in URIs formatted as a JSON string |
| Edm.DateTime | "/Date(<ticks>["+" | "-" <offset>)/"<ticks> = number of milliseconds since midnight Jan 1, 1970<offset> = number of minutes to add or subtract |
| Edm.Decimal | Literal form of Edm.Decimal as used in URIs formatted as a JSON string |
| Edm.Double | Literal form of Edm.Double as used in URIs formatted as a JSON string |
| Edm.Guid | Literal form of Edm.Guid as used in URIs formatted as a JSON string |

<https://www.odata.org/documentation/odata-version-2-0/json-format/>

OData v2 supports two pieces of collection-level metadata

ODATA V2

This version uses the non-standard MERGE HTTP method for updates with Edit semantics. After 2020, only OData v2 (REST) will be active to be used for such integrations with non-SAP systems where SAP doesn't provide standard integration.

Odata V1 and V2 diff

1. Representing Collections of Entries
2. [Collections](https://www.odata.org/documentation/odata-version-2-0/terminology) represent a set of [Entries](https://www.odata.org/documentation/odata-version-2-0/terminology). In OData v1, Collections are represented as an array of objects, with one object for each Entry within the Collection. For example, a collection of Entries would be represented as shown below. The format of each object in the array is described in the [Representing Entries](https://www.odata.org/documentation/odata-version-2-0/json-format#RepresentingEntries) section. In OData v2, Collections are still represented as arrays, however to enable representing Collection-level metadata, the array of objects representing the set of Entries is included as the value of a "results" name/value pair.

OData V1: {

"d" : [

{ ... },

{ ... },

{ ... },

]

}

OData V2: {

"d" : {

"results": [

{ ... },

{ ... },

{ ... }

]

}

}

1. OData v2 supports two pieces of collection-level metadata: an Entry count (the count of the number of entities in the Collection) and "next links" in the case when a [partial listing](https://www.odata.org/documentation/odata-version-2-0/terminology) of the Collection of Entries is being represented

Example .

OData V2: {

"d" : {

"results": [ {

"\_\_metadata": {

"uri": "https://services.odata.org/OData/OData.svc/Categories(0)",

"type": "DataServiceProviderDemo.Category" }, "ID": 0, "Name": "Food",

"Products": {

"\_\_deferred": {

"uri": "https://services.odata.org/OData/OData.svc/Categories(0)/Products"

}

}

},

{ */\* another Category Entry \*/* },

{ */\* another Category Entry \*/* },

{ */\* another Category Entry \*/* }

],

"\_\_count": "3",

"\_\_next": "https://services.odata.org/OData/OData.svc$skiptoken=12"

}

}

1. Representing Entries

In OData v1, Entries are represented as JSON objects with all the [properties](https://www.odata.org/documentation/odata-version-2-0/terminology) of the Entry represented as name/value pairs of the object

But

OData v2 represents Entries the same way as V1. An optional "\_\_metadata" name/value pair is the only pair that should be included on the object that does not directly represent a Property of the Entry being represented. This name/value pair is not data, but instead, by convention defined in this document, specifies the metadata for the Entry being represented

|  |  |  |
| --- | --- | --- |
| **"\_\_metadata" object name/value pairs** | **Optional** | **Description** |
| **uri** | No | The canonical URI identifying the Entry being represented |
| **type** | Yes\* | The name of the [EntityType](https://www.odata.org/documentation/odata-version-2-0/terminology) in the data model of the OData service that describes the Entry being represented.\*This name/value pair must be included if the Entry being represented is part of a type hierarchy and is not the base type in the hierarchy. |
| **etag** | Yes | The [concurrency token](https://www.odata.org/documentation/odata-version-2-0/terminology) associated with the Entry being represented |
| **edit\_ media, media\_src, media\_etag, content\_type** | -- | These name/value pairs only apply if the Entry is a Media Link Entry as described by the [Representing Media Link Entries section](https://www.odata.org/documentation/odata-version-2-0/json-format#RepresentingMediaLinkEntries). |

|  |  |  |
| --- | --- | --- |
| **"\_\_metadata" object name/value pairs for MLEs** | **Optional** | **Description** |
| **media\_src** | No | The URI to use when retrieving the Media Resource |
| **edit\_ media** | Yes | The URI to use when editing the Media Resource |
| **media\_etag** | Yes | The concurrency token for the Media Resource |
| **content\_type** | No | The IANA media type of the Media Resource (ex. image/jpeg ) |

OData V1: {

"d" : {

"\_\_metadata": {

"uri": "https://services.odata.org/OData/OData.svc/Categories(0)",

"type": "DataServiceProviderDemo.Category"

},

"ID": 0,

"Name": "Food",

"Products": {

"\_\_deferred": {

"uri": "https://services.odata.org/OData/OData.svc/Categories(0)/Products"

}

}

}

}

OData V2: {

"d" : {

"\_\_metadata": {

"uri": "https://services.odata.org/OData/OData.svc/Categories(0)",

"type": "DataServiceProviderDemo.Category"

},

"ID": 0,

"Name": "Food",

"Products": {

"\_\_deferred": {

"uri": "https://services.odata.org/OData/OData.svc/Categories(0)/Products"

}

}

}

}

1. Representing Complex Types Properties
2. OData V2 Response Payload: {
3. "d" : {
4. "results": {
5. "\_\_metadata": {
6. "type": "DataServiceProviderDemo.Address"
7. },
8. "Street": "NE 228th",
9. "City": "Sammamish",
10. "State": "WA",
11. "ZipCode": "98074",
12. "Country": "USA"
13. }
14. }
15. }

4. Representing Links

OData V1 Response Payload: {

"d" : [

{ "uri": "https://services.odata.org/OData/OData.svc/Products(0)" },

{ "uri": "https://services.odata.org/OData/OData.svc/Products(7)" },

{ "uri": "https://services.odata.org/OData/OData.svc/Products(8)" }

]

}

OData V2 Response Payload: {

"d" : {

"results": [

{ "uri": "https://services.odata.org/OData/OData.svc/Products(0)" },

{ "uri": "https://services.odata.org/OData/OData.svc/Products(7)" },

{ "uri": "https://services.odata.org/OData/OData.svc/Products(8)" }

],

"\_\_count": "3"

}

}

Support multiple parameter

* $Select
* $Expand
* $Filter
* $OrderBy
* $Top
* $Skip

<https://docs.prospect365.com/en/articles/2455275-getting-started-with-the-odata-api>

ODATA V3

OData v3 adds support for actions, functions, collection values, navigation properties on derived types, and stream properties.

It also introduces a completely new serialization format for JSON

ODATA V4

This version was released in 2014 and incorporates many lessons learned from extensive use of v2. It has many features not included in OData Version 2.0. For example, in OData 4, the standard HTTP PATCH method is used to update an entity with Edit semantics. $count has been enhanced to be used with $filter, $expand and $orderby options

The OData V4 model supports the following:

* Read access
* Updating properties of OData entities (in entity sets and contained entities) via two-way-binding
* Deleting entities
* Operation (function and action) execution
* Grouping data requests in a batch request
* Server-side sorting and filtering