OASIS Committee Note

What's New in OData Version 4.01

Committee Note 03

21 June 2019

This version:
https://docs.oasis-open.org/odata/new-in-odata/v4.01/cn03/new-in-odata-v4.01-cn03.docx (Authoritative)
https://docs.oasis-open.org/odata/new-in-odata/v4.01/cn03/new-in-odata-v4.01-cn03.html
https://docs.oasis-open.org/odata/new-in-odata/v4.01/cn03/new-in-odata-v4.01-cn03.pdf

Previous version:
http://docs.oasis-open.org/odata/new-in-odata/v4.01/cn02/new-in-odata-v4.01-cn02.docx (Authoritative)
http://docs.oasis-open.org/odata/new-in-odata/v4.01/cn02/new-in-odata-v4.01-cn02.html
http://docs.oasis-open.org/odata/new-in-odata/v4.01/cn02/new-in-odata-v4.01-cn02.pdf

Latest version:
https://docs.oasis-open.org/odata/new-in-odata/v4.01/new-in-odata-v4.01.docx (Authoritative)
https://docs.oasis-open.org/odata/new-in-odata/v4.01/new-in-odata-v4.01.html
https://docs.oasis-open.org/odata/new-in-odata/v4.01/new-in-odata-v4.01.pdf

Technical Committee:
OASIS Open Data Protocol (OData) TC

Chairs:
Ralf Handl (ralf.handl@sap.com), SAP SE
Mike Pizzo (mikep@microsoft.com), Microsoft

Editors:
Ralf Handl (ralf.handl@sap.com), SAP SE
Hubert Heijkers (hubert.heijkers@nl.ibm.com), IBM
Michael Pizzo (mikep@microsoft.com), Microsoft
Martin Zurmuehl (martin.zurmuehl@sap.com), SAP SE

Related work:
This document is related to:

- *OData Version 4.01. Part 1: Protocol*. Latest version. https://docs.oasis-open.org/odata/odata/v4.01/odata-v4.01-part1-protocol.html.
- *OData Version 4.01. Part 2: URL Conventions*. Latest version. https://docs.oasis-open.org/odata/odata/v4.01/odata-v4.01-part2-url-conventions.html.
- *OData JSON Format Version 4.01*. Edited by Ralf Handl, Michael Pizzo, and Mark Biamonte. Latest version: https://docs.oasis-open.org/odata/odata-json-format/v4.01/odata-json-format-v4.01.html.
- *OData Common Schema Definition Language (CSDL) JSON Representation Version 4.01*. Edited by Michael Pizzo, Ralf Handl, and Martin Zurmuehl. Latest version: https://docs.oasis-open.org/odata/odata-csdl-json/v4.01/odata-csdl-json-v4.01.html.
- *OData Common Schema Definition Language (CSDL) XML Representation Version 4.01*. Edited by Michael Pizzo, Ralf Handl, and Martin Zurmuehl. Latest version: https://docs.oasis-open.org/odata/odata-csdl-xml/v4.01/odata-csdl-xml-v4.01.html.

Abstract:

This document describes the shape of, and motivation behind, the changes in OData Version 4.01 compared to its predecessor version 4.0.

Status:

This is a Non-Standards Track Work Product. The patent provisions of the OASIS IPR Policy do not apply.

This document was last revised or approved by the OASIS Open Data Protocol (OData) TC on the above date. The level of approval is also listed above. Check the "Latest version" location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Technical Committee (TC) are listed at https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=odata#technical.

TC members should send comments on this document to the TC's email list. Others should send comments to the TC's public comment list, after subscribing to it by following the instructions at the "Send A Comment" button on the TC's web page at https://www.oasis-open.org/committees/odata/.

Citation format:

When referencing this document the following citation format should be used:

[New-in-OData-v4.01]

What's New in OData Version 4.01. Edited by Ralf Handl, Michael Pizzo, Stefan Hagen , and Martin Zurmuehl. 21 June 2019. OASIS Committee Note 03. https://docs.oasis-open.org/odata/new-in-odata/v4.01/cn03/new-in-odata-v4.01-cn03.html. Latest version: https://docs.oasis-open.org/odata/new-in-odata/v4.01/new-in-odata-v4.01.html.

Notices

Table of Contents

1 Introduction

The Open Data Protocol (OData) Version 4.0 was first published in 2013 and has since been implemented in numerous products and tools.

OData Version 4.0 has proven to be very suited for building inherently consistent REST APIs.

OData Version 4.01 adds various new features and removes a few restrictions. These changes can be categorized into:

- Extended Query Language
- Simplified Syntax
- Simplified Payloads
- Easier partial adoption of OData in existing REST APIs

OData 4.01 is highly compatible, incremental release over OData 4.0. A compliant 4.01 OData Service fully supports OData 4.0 clients. New OData 4.01 query features and simplified syntax can be supported as compatible extensions to OData 4.0 syntax. Content negotiation is facilitated through the ODataVersion header to ensure OData 4.0 clients don't receive unexpected constructs in response payloads.

This document follows the structure of the specification documents and cross-references related changes. It is non-normative, so the key words “MUST”, “SHOULD”, “MAY”, and “NOT” are avoided and readers must not assume it states any requirements.

1.1 References (non-normative)

[OData-Aggregation] *OData Extension for Data Aggregation Version 4.0*. Edited by Ralf Handl, Hubert Heijkers, Gerald Krause, Michael Pizzo, and Martin Zurmuehl. 04 November 2015. OASIS Committee Specification 02. http://docs.oasis-open.org/odata/odata-data-aggregation-ext/v4.0/cs02/odata-data-aggregation-ext-v4.0-cs02.html. Latest version: http://docs.oasis-open.org/odata/odata-data-aggregation-ext/v4.0/odata-data-aggregation-ext-v4.0.html.

[OData-CSDLJSON] *OData Common Schema Definition Language (CSDL) JSON Representation Version 4.01.* See link in "Related work" section on cover page.

[OData-CSDLXML] *OData Common Schema Definition Language (CSDL) XML Representation Version 4.01.* See link in "Related work" section on cover page.

[OData-JSON] *OData JSON Format Version 4.01.*
See link in "Related work" section on cover page.

[OData-Protocol] *OData Version 4.01 Part 1: Protocol.*
See link in "Related work" section on cover page.

[OData-URL] *OData Version 4.01 Part 2: URL Conventions.*
See link in "Related work" section on cover page.

2 OData Part 1: Protocol

2.1 New: Default Namespaces

The syntax of version 4.0 avoids ambiguity by referencing types, actions, functions, and all other model elements via a qualified name, i.e. prefixed with the namespace of the model element. Although shortcut aliases can be used in place of namespaces to qualify a model element, this namespace qualification still results in somewhat verbose URLs and payloads.

Version 4.01 allows defining default namespaces, enabling references to model elements in those default namespaces to omit the qualifier. Precedence rules are introduced to resolve potential ambiguity.

2.2 New: Schema Versioning

Version 4.0 allows evolving OData services by adding optional constructs; breaking changes require publishing a new service with a different service root URL. This has been proven problematic for some providers as the service root URL tends to creep into derived documents and rolling out a new service root URL can be challenging.

Version 4.01 allows breaking changes if (and only if) the service supports schema version negotiation, i.e. the old and new schema can both be served from the same service root URL.

2.3 New: Header `Isolation` without `OData-` prefix

The header `OData-Isolation` isn't that OData-specific, so version 4.01 accepts it without the prefix as `Isolation`.

2.4 New: Preference `omit-values`

Version 4.01 allows omitting null or, alternatively, default values from response payloads. This significantly reduces the message size for sparsely populated entities with many properties.

2.5 New: Response Header `AsyncResult`

Version 4.0 requires that the final response from a status monitor resource for an asynchronously processed request uses content-type `application/http`; i.e., the response is a wrapper around the actual response to the original asynchronous request. This is considered unnecessary burden by some clients, and it is only necessary for cases where the original request results in an error.

Version 4.01 allows a status monitor resource to return the unwrapped response to the original request; the response code for the final response is 200 (to signal the end of asynchronous processing), and the response code for the original request is conveyed through the `AsyncResult` response header.

2.6 Pruned: Metadata Service

Exposing an OData service's metadata itself as an OData service, while useful, is separate from the more concise metadata description format(s), so we removed it from the CSDL specification. We have not published a 4.01 version of the Metadata as a Service specification so, while services may still implement a metadata service that represents OData 4.0 concepts, we don't guarantee that the additional features of a version 4.01 service can be represented in a version 4.0 metadata service.

2.7 Improved: Case-Insensitive System Query Options without `$` prefix

Version 4.01 allows omitting the `$` prefix for system query options and allows any casing, so clients may request `Products?OrderBy=Price`. And yes, you are still allowed to prefix system query options with a dollar (`$`) sign ☺.

2.8 New: System Query Option `$compute`

Version 4.01 allows requesting computed properties on-the-fly with `$compute`. This is the counterpart to the `compute()` transformation defined in **[OData-Aggregation]**.

2.9 New: Indexing into Ordered Collections and Positional Insert

Version 4.0 treated collections as atomic, unordered bags of stuff and didn't allow access to individual items in a collection.

Version 4.01 introduces the concept of an ordered collection and allows zero-based indexing into an ordered collection for reading, updating, and inserting items.

2.10 New: Deep Update

Version 4.0 allowed "deep insert" of a new entity together with new related entities, e.g. creation of an order together with its order items.

Version 4.01 adds "deep update" of an existing entity, where the payload of the update can contain nested entities for creating new or updating existing related entities. The nested entities can be represented as a full set of related entities, or as a delta, i.e. only specifying new, changed, and removed entities.

2.11 New: Set-Based Operations

Similar to SQL's `UPDATE ... WHERE ...` and `DELETE ... WHERE ...` version 4.01 allows `PATCH` and `DELETE` requests to `/$each` member of a collection, optionally limiting the operation to a subset of the collection identified via the new `/$filter(...)` path segment.

Both `/$filter(...)` and `/$each` can be used with POST requests to invoke actions.

Collections of entities can additionally be modified using `PATCH` requests with a delta payload. So you can e.g. get a delta from one service and use it to modify another, similarly structured, service.

2.12 New: `$expand` and `$select` with `POST` and `PATCH`

When modifying a single instance, the response shape can now be specified with `$expand` and `$select` combined with `Prefer:return=representation`.

2.13 New: Invoking Functions with Implicit Parameter Aliases

Functions as the last path segment can now be invoked without appending parentheses and named parameters. Instead implicit parameter aliases can be used to provide parameter values.

2.14 New: Optional Parameters

Non-binding action and function parameters can be marked as optional and omitted from requests, subject to overload resolution rules.

2.15 New: Referencing an ETag in a Batch Request

Version 4.01 allows conditional requests within a batch request that reference the ETag returned in a previous operation in the same batch request.

2.16 New: Referencing across Change Sets in a Batch Request

Version 4.01 allows `content-id` referencing to requests in other change sets, or to requests not nested within a change set.

2.17 New: Referencing Nested Inserted Entities

Version 4.01 allows `content-id` referencing for nested entities within deep insert requests. The content-id of the nested entity can be provided with the `Core.ContentID` instance annotation within the nested entity.

2.18 New: Referencing Values in Response Bodies

Version 4.01 allows `content-id` referencing into JSON response bodies, using standard OData URL syntax to address a specific part of the response.

2.19 New: Continue-on-Error for Collection requests

Version 4.01 defines new uses of the `Continue-on-Error` preference which, when applied, allows the service to continue executing the request despite errors encountered while processing the request.

3 OData Part 2: URL Conventions

3.1 New: Alternate Keys

In addition to the (primary) key, entities now may have alternate keys that can be used in URLs in the parentheses-style key syntax, specifying the property names of an alternate key.

3.2 New: Key-as-Segment Convention

In addition to the rather recognizable parentheses-style key convention specific for OData, version 4.01 also supports the new key-as-segment convention which has become the typical style for specifying individual resources by key in REST APIs.

3.3 New: Addressing Operations without Namespace or Alias

Actions or functions defined in a default namespace (see section 2.1) can now be used in URLs without namespace- or alias-qualification, so long as the operation name doesn't collide with property or navigation property names.

3.4 New: Addressing a Member of an Ordered Collection

Version 4.01 introduces the concept of an ordered collection of primitive or complex values and allows zero-based indexing into an ordered collection for reading, updating, and inserting items.

3.5 Improved: Case-Insensitive Operators and Functions

The logical and arithmetic operators, as well as the built-in functions that can be used in `$filter` and `$orderby`, are case-insensitive in version 4.01.

3.6 New: Annotation values in expressions

Instance annotation values can now be used in `$compute`, `$expand`, `$filter`, `$orderby`, and `$select`.

3.7 Improved: `EQ` Comparison

The comparison operator `eq` allows comparing ordered collections, unordered collections, collections of entities, and single entities, and supports comparing nullable single-valued navigation paths to `null`.

3.8 Improved: `cast` Function

Version 4.0 allows casting primitive values to string values using their literal representation used in payloads. Version 4.01 allows the inverse operation: casting a string value that is a literal representation of a primitive value to that primitive value. This includes the WKT format for `Geo` types, and datetime values without explicit time zone to `Edm.DateTimeOffset`.

Version 4.01 also allows casting between enumeration values and their underlying numeric values.

3.9 Pruned: Structural Casting to Arbitrary Structured Types

Version 4.0 optionally allows structural casting of entities and complex instances to arbitrary structured types by casting identically named properties. This is deprecated with version 4.01; structured types can only be cast to a direct or indirect base type. In addition, a type cast segment can be used to treat instances of one type as a type outside the hierarchy where that treatment is meaningful to the service (for example, to treat an automobile as a connected device, without requiring that automobile derive from connected device).

3.10 Improved: Enumeration and Duration Literals without Prefix

Enumeration literals no longer need to be prefixed with the type name, and duration literals no longer need the prefix `duration`.

3.11 New: `in` Operator

The new `in` operator checks whether the left operand is a member of the right operand, which must be a collection. A static collection can be specified as a parentheses-enclosed list of comma-separated primitive values. This latter form is syntactic sugar equivalent to several `EQ` expressions joined by `OR`.

The left operand can be an array of properties, in which case the right operand must be an array of arrays:

```
$filter=[FirstName,LastName] in [["John","Doe"],["Jane","Smith"]]
```

3.12 New: `divby` Operator

The new `divby` operator promotes both operands to decimals and always results in a decimal.

3.13 New: `hassubset` and `hassubsequence` Collection Functions

The `hassubset` function checks whether the first collection has the second collection as a subset. The `hassubsequence` function in addition checks whether the items appear in the same order.

3.14 Improved: Collection Overloads for Functions `concat`, `contains`, `endswith`, `indexof`, `length`, `startswith`, and `substring`

These functions now allow collections as their parameters.

3.15 New: `matchesPattern` Function

This new function checks whether a string matches a regular expression.

3.16 Improved: `substring` with Negative Start Index

The `substring` function now allows a negative start index (second argument) to start N characters before the end of the string.

3.17 New: `case` Function

This new function evaluates a list of conditions and returns one of multiple possible results.

3.18 Improved: `/$count`

In `$expand`, the path suffix `/$count` can now be followed by parentheses containing a `$filter` or `$search` query option to return only the count of matching instances.

3.19 New: `$expand` of Stream Properties and Media Resources

Version 4.01 allows inlining stream properties as base64url-encoded values.

3.20 Improved: System Query Option `$search`

Version 4.01 allows `$search` on collections of any type, not only on collections of entities.

Version 4.01 also allows digits, dots, and commas in search terms, e.g. `$search=3.14`. as well as non-whitespace characters in search words, and allows single-quote delimited, unbalanced search expressions to be sent to the server.

3.21 Improved: System Query Option `$select`

Version 4.01 extends `$select` similar to `$expand` to allow filtering, searching, sorting and paging on collection-valued properties as well as computing, selecting and expanding properties of complex properties.

To allow filtering and sorting on collections of primitive properties the symbol `$this` was added.

3.22 New: System Query Option `$compute`

Computed properties can be requested on-the-fly with `$compute`. This is the counterpart to the `compute()` transformation defined in **[OData-Aggregation]**.

3.23 New: System Query Option `$index`

Version 4.01 introduces the concept of an ordered collection of primitive or complex values. The new system query option `$index` allows positional insert of an item into an ordered collection.

3.24 New: `/$query`

In resource paths the path suffix `/$query` can be used together with the HTTP verb POST to execute what is logically a GET request and transport overlong query options in the request body.

4 Pruned: OData Part 3: Common Schema Definition Language (CSDL)

Version 4.0 only specifies an XML representation of the Common Schema Definition Language (CSDL), and Part 3 of version 4.0 describes both the meta-model of OData metadata documents and its XML representation.

Version 4.01 adds an alternative JSON representation of CSDL, so the OData TC has decided to discontinue Part 3 and instead incorporate the meta-model specification in both the CSDL JSON document **[OData-CSDLJSON]** and the CSDL XML document **[OData-CSDLXML]**.

5 New: OData Common Schema Definition Language XML Representation

This new document describes the XML representation of OData metadata documents. It replaces the former Part 3: CSDL. The following sections describe changes introduced by OData Version 4.01.

5.1 New: Decimal with Floating Scale

The type `Edm.Decimal` now allows floating scale and the special values `-INF`, `INF`, and `NaN` to support `DECFLOAT`.

5.2 New: Built-in Abstract Type `Edm.Untyped`

Non-key properties typed with `Edm.Untyped` can contain any valid type or collection of types, recursively. This is an "escape hatch" for situations where the OData type system is too restrictive to model the payload. Use sparingly and with care as you lose all benefits of OData metadata for these "untyped" parts – no types, no metadata annotations, no predictability.

5.3 New: Built-in Types for Terms: `Edm.AnyPropertyPath` and `Edm.ModelElementPath`

The `Edm.AnyPropertyPath` is a generalization of `Edm.PropertyPath` and `Edm.NavigationPropertyPath` that can be used in term definitions for situations that do not differentiate between structural and navigation properties.

The `Edm.ModelElementPath` allows referencing any model element.

5.4 Improved: Key Properties

Key properties of an entity type can now be part of a related entity type provided the navigation path to the "remote" key property consists only of single-valued, non-nullable segments. And of course, these key properties need an alias.

5.5 New: Key-Less Entity Types

Non-abstract entity types need not specify a key. These can be used e.g. in singletons or single-valued containment navigation properties. Entity types used within an entity set or collection-valued containment navigation property must still define a set of key properties.

5.6 Improved: Inheritance

Derived structured types can now redefine the type of a complex property or navigation property to a more specific subtype.

5.7 Improved: Referential Constraints

Referential constraints can specify navigation properties and complex properties in addition to primitive properties.

5.8 Improved: `Unicode` Facet

The `Unicode` facet can be used for defining terms, parameters, and return types. Somehow that got lost in version 4.0, and only properties and type definitions explicitly allowed it.

5.9 New: Terms applying to Collections

Terms can now be marked as applicable to any collection of entities (e.g. collection-valued navigation properties), not just to entity sets.

5.10 New: Annotations targeting Action/Function Overloads

External targeting of annotations is now also possible for individual overloads of actions and functions, their parameters and return types. In addition three of actions and functions, with the restriction that these annotations apply to all overloads. There's still no way to externally target an individual overload.

5.11 New: Inheriting Annotations

Annotations are propagated along inheritance hierarchies: annotations on a base type are also valid on a derived type unless applied again with a different value.

5.12 New: Absolute Paths in Annotations

In version 4.0 path expressions in annotations are relative to the annotated model element.

Version 4.01 allows absolute paths starting with a forward slash followed by the qualified name of a model element.

5.13 New: Key and Index Path Segments in Annotations

Path expressions in annotations can now contain key segments for drilling into collections of entities, and index segments for drilling into ordered collections of complex or primitive values.

5.14 New: Annotation Expressions `Has`, `In`, `Add`, `Sub`, `Neg`, `Mul`, `Div`, `DivBy`, and `Mod`

These comparison and arithmetic annotation expressions are the counterpart to the equally named operators in URL expressions.

5.15 New: Client-Side Function `odata.matchesPattern`

Annotation expressions can now check whether a string matches a regular expression.

5.16 New: All URL Functions as Client-Side Functions

Annotation expressions can now use all functions defined in **[**OData-URL**]** with the same semantics.

5.17 Pruned: Metadata Service

Exposing an OData service's metadata itself as an OData service, while useful, is separate from the more concise metadata description format(s), so we removed it from the CSDL specification. We have not published a 4.01 version of the Metadata as a Service specification so, while services may still implement a *metadata service*, we don't guarantee that the additional features of a version 4.01 service can be represented in a version 4.0 *metadata service*.

6 New: OData Common Schema Definition Language JSON Representation

This new document describes a JSON representation of an OData metadata document. The representation-independent parts of this document are identical to its twin document for the XML representation **[OData-CSDLXML]**.

7 Pruned: OData Atom Format

The Atom format has very little adoption. It's going to be deprecated and hasn't been updated to reflect the new 4.01 features.

8 OData JSON Format

8.1 Improved: Exponential Notation for Decimals

Version 4.0 only allowed exponential notation for `Edm.Decimal` values when combined with the format parameter `ExponentialDecimals=true`.

Version 4.01 always allows exponential notation for decimals, as this is considered normal in JSON messages.

8.2 Improved: Control Information without prefix `odata.`

Control information represented in JSON as payload annotations in the `odata` namespace can now be specified without qualifier, i.e. `@count`, `@nextLink`, `@context`.

8.3 Improved: `@type` for Built-In Primitive Types

The value of the `@type` control information no longer needs the prefix `#` for built-in primitive types.

8.4 New: Inlined JSON Streams

Inlined streams that are annotated as having an acceptable media type of `application/json` are represented as JSON and not as base64-encoded strings.

8.5 New: Simplified representation of Delta with Expand

Expanded entities can now be nested in delta responses, similar to their representation in non-delta messages. The nested representation is either a collection of all related entities (in case the underlying data store doesn't track changes of the corresponding relationship), or a collection of added, changed, and deleted entities (but no added or deleted links).

8.6 Improved: Representation of Deleted Entities

Version 4.01 represents deleted entities similar to added and changed entities and marks them with an `@removed` annotation. The deleted entities contain at least their key properties or `@id` and can contain additional properties.

8.7 Improved: Representation of Deleted Links

Deleted links for single-valued navigation paths no longer need to specify the id of the link target as this can be derived from the link source and the navigation path.

8.8 New: Advertise Actions/Functions on Collection-Valued Properties

Bound actions and functions with a collection-valued binding parameter can now be advertised on collection-valued properties or in responses representing a collection.

8.9 New: Advertise Non-Availability of Actions/Functions

Action/function advertisements in payloads now allow the value `null` to indicate that this action or function is not available on the current instance or collection.

8.10 Improved: Invoke Parameterless Actions with Empty Request Body

Invoking a parameterless action in version 4.0 requires sending an empty object in the request body. This empty object can be omitted in version 4.01, and the request body can be empty, saving two bytes.

8.11 New: JSON Batch Requests and Responses

Version 4.0 only defined a multipart format for describing batch requests and responses, which was somewhat hard to implement.

Version 4.01 introduces an alternative JSON format for batch requests and responses, so clients can now use off-the-shelf JSON libraries to compose batch requests and consume batch responses. Combined with the CSDL JSON representation **[OData-CSDLJSON]** this allows pure JSON communication with OData 4.01 services.

The new JSON Batch format allows more flexible dependencies between requests, including expressions for executing a dependent request based on return codes or even values from the response bodies of preceding requests.

8.12 New: Error Information within a Success Response

A version 4.01 response may include well-formed error information within a success response. The error information, represented by newly introduced exception annotations, may represent primitive values that are in error and/or structured values that are in error if the client has specified the `continue-on-error` preference.

8.13 New: Annotating Primitive Values within a Collection

A new `collectionAnnotations` control information was introduced to enable annotating primitive values within collections.

Appendix A. Acknowledgments

The contributions of the OASIS OData Technical Committee members, enumerated in **[OData-Protocol]**, are gratefully acknowledged.

Appendix B. Revision History

Revision	Date	Editor	Changes Made
WD01	2016-12-16	Ralf Handl	Initial version
CN01	2017-06-08	Mike Pizzo, Ralf Handl	Incorporated changes since OData Version 4.01 CSD01
CN02	2017-09-28	Mike Pizzo, Ralf Handl	Incorporated changes since OData Version 4.01 CSD02
CN03	2019-06-21	Mike Pizzo, Ralf Handl, Hubert Heijkers	Incorporated changes since OData Version 4.01 CS01