

OASIS Committee Note

Repeatable Requests Version 1.0

Committee Note Draft 01 /
Public Review Draft 01

24 October 2019

This stage:

<https://docs.oasis-open.org/odata/repeatable-requests/v1.0/cnprd01/repeatable-requests-v1.0-cnprd01.docx> (Authoritative)

<https://docs.oasis-open.org/odata/repeatable-requests/v1.0/cnprd01/repeatable-requests-v1.0-cnprd01.html>

<https://docs.oasis-open.org/odata/repeatable-requests/v1.0/cnprd01/repeatable-requests-v1.0-cnprd01.pdf>

Previous stage:

N/A

Latest stage:

<https://docs.oasis-open.org/odata/repeatable-requests/v1.0/repeatable-requests-v1.0.docx> (Authoritative)

<https://docs.oasis-open.org/odata/repeatable-requests/v1.0/repeatable-requests-v1.0.html>

<https://docs.oasis-open.org/odata/repeatable-requests/v1.0/repeatable-requests-v1.0.pdf>

Technical Committee:

[OASIS Open Data Protocol (OData) TC](https://www.oasis-open.org/committees/odata/)

Chairs:

Ralf Handl (ralf.handl@sap.com), [SAP SE](http://www.sap.com/)

Mike Pizzo (mikep@microsoft.com), [Microsoft](http://www.microsoft.com/)

Editors:

Evan Ireland (evan.ireland@sap.com), [SAP SE](http://www.sap.com/)

Matt Borges (matt.borges@sap.com), [SAP SE](http://www.sap.com/)

Related work:

This document is related to:

* Org.OData.Repeatability.V1.xml. <https://oasis-tcs.github.io/odata-vocabularies/vocabularies/Org.OData.Repeatability.V1.xml>.
* *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 JSON Format Version 4.01*. Latest version: <https://docs.oasis-open.org/odata/odata-json-format/v4.01/odata-json-format-v4.01.html>.

Abstract:

This document describes a method to provide the ability to retry unsafe (i.e. POST, PUT, PATCH, DELETE) requests without incurring unintended side-effects. This specification can be applied to any HTTP based protocol.

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 stage" 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](https://www.oasis-open.org/committees/comments/index.php?wg_abbrev=odata)" 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:

[RepeatableRequests-v1.0]

*Repeatable Requests Version 1.0*. Edited by Evan Ireland and Matt Borges. 24 October 2019. OASIS Committee Note Draft 01 / Public Review Draft 01. <https://docs.oasis-open.org/odata/repeatable-requests/v1.0/cnprd01/repeatable-requests-v1.0-cnprd01.html>. Latest stage: <https://docs.oasis-open.org/odata/repeatable-requests/v1.0/repeatable-requests-v1.0.html>.

Notices

Copyright © OASIS Open 2019. All Rights Reserved.

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full [Policy](https://www.oasis-open.org/policies-guidelines/ipr) may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Table of Contents

[1 Introduction 5](#_Toc22895296)

[1.1 References (non-normative) 6](#_Toc22895297)

[2 Example Scenarios 7](#_Toc22895298)

[2.1 Insert a new Item 7](#_Toc22895299)

[2.2 Invoke an Action 8](#_Toc22895300)

[3 Proposal 10](#_Toc22895301)

[3.1 New Request Headers 10](#_Toc22895302)

[3.1.1 Repeatability-Request-ID 10](#_Toc22895303)

[3.1.2 Repeatability-First-Sent 10](#_Toc22895304)

[3.1.3 Repeatability-Client-ID 10](#_Toc22895305)

[3.2 New Response Header 10](#_Toc22895306)

[3.2.1 Repeatability-Result 10](#_Toc22895307)

[3.2.1.1 accepted 10](#_Toc22895308)

[3.2.1.2 rejected 11](#_Toc22895309)

[3.3 Client Behavior 11](#_Toc22895310)

[3.4 Server Behavior 11](#_Toc22895311)

[3.5 Repeatability Deletion 12](#_Toc22895312)

[3.6 Notes 13](#_Toc22895313)

[4 Incorporation into OData 14](#_Toc22895314)

[4.1 Support 14](#_Toc22895315)

[4.2 Discovery 14](#_Toc22895316)

[4.3 Response Payload 14](#_Toc22895317)

[4.4 Batch Requests 15](#_Toc22895318)

[Appendix A. Acknowledgments 16](#_Toc22895319)

[Appendix B. Revision History 17](#_Toc22895320)

1. [Introduction](#sec_Introduction)

HTTP is an inherently unreliable protocol. If connection or other issues prevent the client from receiving a response, the client is left in doubt as to whether the request was processed by the server. For safe HTTP requests as defined in [**[RFC7231]**](#RFC7231) section 4.2 (for example, GET) the client can simply re-try the request, but for operations that change state (for example, inserting a new resource or invoking a side-effecting service operation such as PlaceOrder or TransferFunds) re-issuing the request may result in an undesired state (for example, two orders placed, or double the amount of funds transferred).



Figure 1: Lost requests and responses without Repeatability

As the sender does not receive responses to requests 2a and 2b, it creates three orders instead of the intended two orders.

This document proposes a simple approach that lets the receiver recognize repeated requests, so it can echo a stored response for an already received and processed request without processing the request a second time:



Figure 2: Lost requests and responses with Repeatability

## [References (non-normative)](#sec_Referencesnonnormative)

[Idempotency] SAP Gateway Foundation (SAP\_GWFND) – Defining Settings for Idempotent Services, <https://help.sap.com/viewer/68bf513362174d54b58cddec28794093/7.5.14/en-US/09f82651c294256ee10000000a445394.html>

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

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

[OData-VocRep] OData Vocabularies Version 4.0: Repeatability Vocabulary.
See link in "[Related work](#RelatedWork)" section on cover page.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique Identifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <https://www.rfc-editor.org/info/rfc4122>.

[RFC7231] Fielding, R., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <https://www.rfc-editor.org/info/rfc7231>.

1. [Example Scenarios](#sec_ExampleScenarios)

## [Insert a new Item](#sec_InsertanewItem)

Adding a new item to a collection is a POST request to the collection. To safeguard against a lost response the client adds repeatability headers:

Example 1: Create a new Order

POST /service/Orders

Content-Type: application/json

**Repeatability-Request-ID: 112a3a3e-f94c-4f56-b49b-5aab3d97e5b7**

**Repeatability-First-Sent: Tue, 26 Mar 2019 16:06:51 GMT**

{

 "CustomerID": "ALFKI",

 "OrderLines": [

 {

 "ProductID": "tomatoes-red-cherry",

 "Quantity": 5,

 "Unit": "kg",

 },

 {

 "ProductID": "grapejuice-merlot",

 "Quantity": 2,

 "Unit": "l",

 }

 ]

}

The client does not receive a response, so it simply sends the request again:

Example 2: Repeat: Create a new Order

POST /service/Orders

Content-Type: application/json

**Repeatability-Request-ID: 112a3a3e-f94c-4f56-b49b-5aab3d97e5b7**

**Repeatability-First-Sent: Tue, 26 Mar 2019 16:06:51 GMT**

{

 "CustomerID": "ALFKI",

 "OrderLines": [

 {

 "ProductID": "tomatoes-red-cherry",

 "Quantity": 5,

 "Unit": "kg",

 },

 {

 "ProductID": "grapejuice-merlot",

 "Quantity": 2,

 "Unit": "l",

 }

 ]

}

This time the client receives a response:

Example 3: Response to Create a new Order

HTTP/1.1 200 OK

Content-Type: application/json

Location: http://host/service/Orders/4711

**Repeatability-Result: accepted**

{

 "OrderID": 4711,

 "CustomerID": "ALFKI",

 "OrderLines": [

 {

 "ProductID": "tomatoes-red-cherry",

 "Quantity": 5,

 "Unit": "kg",

 },

 {

 "ProductID": "grapejuice-merlot",

 "Quantity": 2,

 "Unit": "l",

 }

 ]

}

The Repeatability-Result response header tells the client that it need not worry: the new order was created exactly once.

## [Invoke an Action](#sec_InvokeanAction)

Sometime later the client wants to place an exact clone of a recent order

Example 4: Clone an existing Order

POST /service/Orders/4711/Clone

Content-Type: application/json

**Repeatability-Request-ID: a47a83d9-be50-46aa-ab2a-55f18f4fbc64**

**Repeatability-First-Sent: Mon, 01 Apr 2019 06:22:03 GMT**

{}

The client does not receive a response, so it simply sends the request again. This time the client receives a response:

Example 5: Response to Clone an existing Order

HTTP/1.1 204 No Content

Location: http://host/service/Orders/4712

**Repeatability-Result: accepted**

The Repeatability-Result response header tells the client that it need not worry: the new order was cloned exactly once.

1. [Proposal](#sec_Proposal)

Two new request headers and one response header are added to facilitate the ability to retry requests without incurring unintended side effects.

Another optional header is added to facilitate the ability for a server to eagerly cleanup tracking information that it may use for the implementation of repeatable requests (rather than keeping such information for a possibly extended retention period).

## [New Request Headers](#sec_NewRequestHeaders)

### [Repeatability-Request-ID](#sec_RepeatabilityRequestID)

An opaque string representing a client-generated, globally unique for all time, identifier for the request. Servers must accommodate the 36-character hexadecimal case-insensitive encoding of a UUID (GUID), as defined in [**[RFC4122]**](#RFC4122). It is recommended for security purposes to use version 4 (random) UUIDs as defined in [**[RFC4122]**](#RFC4122) section 4.1.3. Support for other forms of unique identifiers is optional.

If specified, the client directs that the request must be repeatable; that is, that the client can make the request multiple times with the same Repeatability-Request-ID and [Repeatability-First-Sent](#sec_RepeatabilityFirstSent) header values and get back an appropriate response without the server executing the request multiple times. Servers aware of repeatability but unable to fulfill this direction for this request type must not execute the request and instead return 501 Not Implemented.

### [Repeatability-First-Sent](#sec_RepeatabilityFirstSent)

The date and time at which the request was first created, expressed using the IMF-fixdate form of HTTP-date as defined in [**[RFC7231]**](#RFC7231).

Repeatability-First-Sent allows the server to determine if the request is within its currently tracked window of time for repeatability. If Repeatability-First-Sent is within the server’s window and the request has not been seen previously, the server can safely execute it. If it is not in the window of currently tracked requests, the server cannot guarantee the request was not already executed and so returns an error. Without using Repeatability-First-Sent, if/when the server cleans up tracking information, the server could receive a [Repeatability-Request-ID](#sec_RepeatabilityRequestID) that it has already executed but no longer has any tracking data for and so the server would incorrectly execute the request again.

### [Repeatability-Client-ID](#sec_RepeatabilityClientID)

Optional. An opaque string representing a client-generated, globally unique for all time, identifier for the instance of the client application that issued the request. Servers, if they do not ignore this header, must accommodate the 36-character hexadecimal case-insensitive encoding of a UUID (GUID), as defined in [**[RFC4122]**](#RFC4122). It is recommended for security purposes to use version 4 (random) UUIDs as defined in [**[RFC4122]**](#RFC4122) section 4.1.3. Support for other forms of unique identifiers is optional.

Repeatability-Client-ID , if provided by the client, may be used by the server to support bulk [Repeatability Deletion](#sec_RepeatabilityDeletion).

## [New Response Header](#sec_NewResponseHeader)

### [Repeatability-Result](#sec_RepeatabilityResult)

One of the following string values:

#### [accepted](#sec_accepted)

The request was accepted, and the server guarantees that the server state reflects a single execution of the operation.

The response returns the success or failure state of the operation as first executed by the server and reflects either the current state of the system or the state as it existed when the request was first received.

#### [rejected](#sec_rejected)

The request was rejected because the combination of Repeatability-First-Sent and Repeatability-Request-ID were invalid or because the Repeatability-First-Sent value was outside the range of values held by the server.

The server returns 412 Precondition Failed without attempting to execute the request.

The server state is the same as if the request were never received.

## [Client Behavior](#sec_ClientBehavior)

In order to issue a repeatable request, the client first creates a UUID (GUID) and encodes that as a string. It sets that as the string value of the Repeatability-Request-ID header and sets the Repeatability-First-Sent header to the current date-time value.

The client may also include a Repeatability-Client-ID header allowing the server to associate with the client any tracking information that it may use in support of repeatable requests.

If the request fails to return, for example, due to connection issues, the client can re-execute the same command with the same Repeatability-Request-ID and Repeatability-First-Sent headers (and Repeatability-Client-ID, if it was specified previously).

If the request returns with a Repeatability-Result value of accepted then the client knows that the request has been executed in a repeatable manner and consumes the results.

If the request returns with Repeatability-Result value of rejected then the client knows that the creation time is beyond the window of requests that the server has stored and it cannot safely retry the operation, or some other error has occurred (for example, the Repeatability-Request-ID and Repeatability-First-Sent values were inconsistent with each other or with a previous request).

If the request returns without a Repeatability-Result header, then the client has to assume that the request did not reach a server that knows about repeatable requests and therefore the usual mechanism to determine request outcome should be used, for example by checking the response status code. It is possible in this case that requests might be executed multiple times.

If the request returns with HTTP response code 501 Not Implemented with a Repeatability-Result value of accepted, it implies the service knows about repeatability but there is something wrong with the request.

If the request returns with HTTP response code 501 Not Implemented with a Repeatability-Result value of rejected, it implies that the service does know about repeatability.

## [Server Behavior](#sec_ServerBehavior)

When a server receives a request with a valid, non-null Repeatability-First-Sent value:

If the server is aware of this repeatability specification but does not support repeatable execution of the request it must return 501 Not Implemented.

If the Repeatability-Request-ID value is missing, it must return 400 Bad Request with a Repeatability-Result of rejected.

If the Repeatability-First-Sent value is before the earliest remembered Repeatability-Request-ID, or this request cannot be reliably executed for some other implementation-specific reason, the server must return 412 Precondition Failed with a Repeatability-Result value of rejected. Otherwise:

If the server has not seen the Repeatability-Request-ID since its earliest remembered Repeatability-Request-ID (if any), and the Repeatability-First-Sent value is within its window of remembered Repeatability-Request-ID values, then it must execute the request and return the result with Repeatability-Result header value of accepted and record the Repeatability-Request-ID.

The server should return an error 400 Bad Request along with a Repeatability-Result value of rejected if Repeatability-First-Sent is non-null and

* the request verb, URI, or header fields other than Date are different from that of the original request, or
* the same Repeatability-Request-ID is within the window of remembered Repeatability-Request-ID values but has a different Repeatability-First-Sent.

If the server has seen the Repeatability-Request-ID, it may return an error 400 Bad Request along with a Repeatability-Result header value of rejected if the request body was different from that of the original Repeatability-Request-ID.

If the server has seen the Repeatability-Request-ID and the request matches the previous request to the extent validated by the server, the server must return a response with a Repeatability-Result value of accepted that is either:

* the same response code and body as was generated (if any) when the original request with that Repeatability-Request-ID was processed, or
* the response code and response body resulting from re-executing the request if the response code was 4xx or 5xx, i.e. a client error or an internal server error.

In order to permit the server to optimize the storage of response bodies, the client and server may wish to negotiate the amount of content that will be returned in an initial response and any subsequent repeated response. The mechanism for such response content negotiation may depend on the protocol used.

Whether a server is considered to have *seen* a previous request should be transactionally consistent with the mutating effects of the request. For example, a server is not required to remember a previous request whose effects were rolled back due to a failure, since the client could reissue such a request without any possibility for duplication of the effects.

## [Repeatability Deletion](#sec_RepeatabilityDeletion)

In some situations, such as when using occasionally-connected mobile devices, clients may expect the server to offer a significant retention period (e.g. 50 days) for remembered repeatable requests. In such situations, the server’s storage system may be burdened by the retention requirements, so it is valuable to offer clients a way to signal that certain remembered repeatable requests may be forgotten (deleted) by the server even before the retention period has expired. Some clients may be able to acknowledge that they have received all responses to all outstanding requests. Bulk deletion of all the tracking information for repeatable requests from a particular Repeatability-Client-ID may enable a significant performance boost for the server.

If a server supports deletion of remembered requests by Repeatability-Request-ID, then the recommended HTTP request method is DELETE and the recommended URL pattern is “$RepeatableRequestWithRequestID/*<Repeatability-Request-ID>*”. The HTTP response status should be 204 No Content, even if no such request was found.

If a server supports deletion of remembered requests by Repeatability-Client-ID, then the recommended HTTP request method is DELETE and the recommended URL pattern is “$RepeatableRequestsWithClientID/*<Repeatability-Client-ID>*”*.* The HTTP response status should be 204 No Content, even if no such requests were found.

Note: supporting deletion by Repeatability-Client-ID does not mean that the server needs to record information about client instances separately from its set of remembered repeatable requests. For example, it might be achieved simply with an extra (indexed) storage column in the storage table used to track repeatable requests.

## [Notes](#sec_Notes)

Servers may support repeatability on POST, PUT, PATCH and DELETE.

* Repeatability on POST ensures that the operation is executed, or the insert is performed no more than once.
* Repeatability on PUT or PATCH is different from use of an ETag in that repeated PUT or PATCH operations to the same resource will return success (or fail), possibly including a payload, versus a concurrency violation.
* Repeatability on DELETE is different from use of an ETag in that repeated DELETE operations to the same resource will return success (or fail) rather than 404 Not Found.

For some clients, it is important for a repeated request to return with success if the original request actually succeeded, rather than a failure due to a conflict detected on the repeated execution.

Servers must ignore Repeatability-Request-ID and Repeatability-First-Sent for GET and HEAD requests.

1. [Incorporation into OData](#sec_IncorporationintoOData)

The proposal for repeatable requests is valid outside of OData. The following sections describe the use of repeatable requests within OData.

## [Support](#sec_Support)

OData services are not required to support Repeatability. Clients must rely on external means (e.g. capabilities) in order to know whether the server supports repeatability.

## [Discovery](#sec_Discovery)

Services supporting repeatability should annotate the entity container, entity sets, singletons, action imports, or actions in the service metadata with the term Repeatability.Supported defined in the Repeatability vocabulary, see [**[OData-VocRep]**](#VocRep).

Services supporting repeatability deletion by Repeatability-Request-ID and/or Repeatability-Client-ID should annotate the entity container with the terms Repeatability.DeleteWithRequestIDSupported and/or Repeatability.DeleteWithClientIDSupported.

If lower-level elements such as individual entity sets do not support repeatability, then they can opt out of repeatability using a lower-level override of the Repeatability.Supported term.

Example 6: Service that supports Repeatability and Repeatability Deletion

<edmx:Edmx xmlns:edmx="http://docs.oasis-open.org/odata/ns/edmx" Version="4.0">

 <edmx:Reference Uri="https://oasis-tcs.github.io/odata-vocabularies/vocabularies/Org.OData.Repeatability.V1.xml">

 <edmx:Include Alias="Repeatability" Namespace="Org.OData.Repeatability.V1" />

 </edmx:Reference>

 <Schema xmlns="http://docs.oasis-open.org/odata/ns/edm" Namespace="MySchema">

 …

 <EntityContainer Name="MyContainer">

 …

 <Annotation Term="Repeatability.Supported" />

 <Annotation Term="Repeatability.DeleteWithRequestIDSupported" />

 <Annotation Term="Repeatability.DeleteWithClientIDSupported" />

 <EntitySet Name="MyEntitySet" EntityType="MySchema.MyEntity">

 <Annotation Term="Repeatability.Supported" Bool="false" />

 </EntitySet>

 </EntityContainer>

 </Schema>

<edmx:Edmx>

Services may support repeatability without the use of annotations in the service metadata.

## [Response Payload](#sec_Preferences)

The client may optionally use system query options $select and/or $expand in the request URL to force the service return a payload containing the minimal information required by the client, as compared to what it would ordinarily return without the use of system query options. Note that @Core.ContentID is always returned in the response payload if it was specified in the request body.

If the client sends a repeatable request containing a data modification operation for an entity, and the client does not include $select or $expand in the request URL, the server may choose to return 204 No Content even if it would ordinarily return status code 200 or 201 for a non-repeatable request.

The above paragraphs allow the service to minimize the tracking information that it stores in support of repeatable requests.

## [Batch Requests](#sec_BatchRequests)

Services may support repeatability for individual requests within a batch request, as well as for individual requests within a change set or atomicity group within a batch request.

Individual requests within a batch may have a mix of Repeatability-Request-ID and Repeatability-First-Sent values. In this case, each individual response within the batch response has the appropriate Repeatability-Result (or not) according to the corresponding request.

Repeatable request headers cannot be applied to change sets or atomicity groups directly because there is no way to specify headers for an atomicity group in JSON batch requests, see [**[OData-JSON]**](#ODataJSONRef). A client makes a change set or atomicity group repeatable by specifying the same Repeatability-Request-ID and Repeatability-First-Sent values for all requests in the change set or atomicity group. The client must retry the entire change set or atomicity group as a unit if it is repeatable; individual operations within the change set or atomicity group must not be retried.

There is no correlation between the repeatability of a request and the repeatability of any of its dependent requests. That is, a repeatable request may be retried without retrying any of its dependent requests.

Repeatability cannot be applied to batch requests themselves because a single Repeatability-Request-ID on the batch request is not sufficient for uniquely identifying the individual requests within the batch request, and because repeatability implies transactional atomicity which cannot be guaranteed for a batch request containing multiple change sets, some of which may succeed (commit) and some of which may fail (rollback). Therefore, if a server receives a batch request with either a Repeatability-Request-ID or a Repeatability-First-Sent value, it must not execute any requests within the batch and respond with 400 Bad Request.

1. [Acknowledgments](#sec_Acknowledgments)

The contributions of the OASIS OData Technical Committee members, enumerated in [**[OData‑Protocol]**](#odataProtocol), are gratefully acknowledged.

1. [Revision History](#sec_RevisionHistory)

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Editor** | **Changes Made** |
| Working Draft 01 | 2013-06-25 | Mike Pizzo, Ralf Handl | Initial version |
| Committee Note Draft 01 | 2019-10-17 | Matt Borges, Evan Ireland | Aligned header namesAdded Repeatability DeletionClarified client and server behavior with regard to errorsClarified what servers are required to store and return for repeated requests and how the client and server can negotiate this for OData. |