Document Identifier: DSP0285

Date: 2023-04-04

Version: 1.1.0

RedPath Specification

Supersedes: 1.0.1

Document Class: Normative

Document Status: Published

Document Language: en-US

CONTENTS

Foreword

The RedPath Specification was prepared by DMTF's Redfish Forum.

DMTF is a not-for-profit association of industry members that promotes enterprise and systems management and interoperability. For information about DMTF, see DMTF.

Acknowledgments

DMTF acknowledges the following individuals for their contributions to this document:

- Patrick Boyd — Dell Technologies
- Derek Chan — Google LLC
- Rahul Kapoor — Google LLC

Introduction

RedPath is a string syntax to allow a client to specify a path to resources and properties in the Redfish data model.

A RedPath interpreter is a Redfish client that is responsible for executing a sufficiently optimal query strategy for a given RedPath string. The *Redfish Specification* defines optional query parameters that might be available on a Redfish service for optimizing the sequence of client requests. A RedPath interpreter implementation determines which query parameters are suitable against a particular Redfish service.

This specification provides the RedPath syntax definition.

1 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

- DMTF DSP0266, *Redfish Specification*, https://www.dmtf.org/sites/default/files/standards/documents/DSP0266_1.15.1.pdf
- *XML Path Language (XPath) Version 1.0*, https://www.w3.org/TR/1999/REC-xpath-19991116/
- *ECMA-404, The JSON data interchange syntax, 2nd edition*, https://www.ecma-international.org/wp-content/uploads/ECMA-404_2nd_edition_december_2017.pdf

2 Terms, definitions, symbols, and abbreviated terms

Some terms and phrases in this document have specific meanings beyond their typical English meanings. This clause defines those terms and phrases.

The terms "shall" ("required"), "shall not", "should" ("recommended"), "should not" ("not recommended"), "may", "need not" ("not required"), "can" and "cannot" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, Clause 7. The terms in parenthesis are alternatives for the preceding term, for use in exceptional cases when the preceding term cannot be used for linguistic reasons. Note that ISO/IEC Directives, Part 2, Clause 7 specifies additional alternatives. Occurrences of such additional alternatives shall be interpreted in their normal English meaning.

The terms "clause", "subclause", "paragraph", and "annex" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, Clause 6.

The terms "normative" and "informative" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, Clause 3. In this document, clauses, subclauses, or annexes labeled "(informative)" do not contain normative content. Notes and examples are always informative elements.

The term "deprecated" in this document is to be interpreted as material that is not recommended for use in new development efforts. Existing and new implementations may use this material, but they should move to the favored approach. Deprecated material may be implemented in order to achieve backwards compatibility. Deprecated material should contain references to the last published version that included the deprecated material as normative material and to a description of the favored approach. Deprecated material may be removed from the next major version of the specification.

3 RedPath query language

RedPath is a query language based on XPath 1.0. This language treats the entire Redfish service as though it were a single JSON document. When an implementation encounters an `@odata.id` property, it shall retrieve the resource specified by the `@odata.id` property if it is needed to satisfy the expression.

Table 1 contains example expressions of RedPath queries. For the construction rules of queries, see the RedPath language constructs clause.

Table 1 — Example expressions of RedPath queries

Expression	Description
`"nodename"`	Selects the JSON entity with the name `"nodename"`.
`/`	Selects from the root node.
`["index"]`	Selects the index number JSON entity from an array or object. For arrays, the index is 0-based.
`[last()]`	Selects the last index number JSON entity from an array or object.
`["nodename"]`	Selects all the elements from an array or object that contain a property named `"nodename"`.
`[!"nodename"]`	Selects all the elements from an array or object that does not contain a property named `"nodename"`.
`["name"="value"]`	Selects all the elements from an array or object where the property `"name"` is equal to `"value"`.
`["name"<"value"]`	Selects all the elements from an array or object where the property `"name"` is less than `"value"`.
`["name"<="value"]`	Selects all the elements from an array or object where the property `"name"` is less than or equal to `"value"`.
`["name">"value"]`	Selects all the elements from an array or object where the property `"name"` is greater than `"value"`.
`["name">="value"]`	Selects all the elements from an array or object where the property `"name"` is greater than or equal to `"value"`.
`["name"!="value"]`	Selects all the elements from an array or object where the property `"name"` does not equal `"value"`.
`["name1">="value1" and "name2">="value2"]`	Selects all the elements from an array or object where the property `"name1"` is greater than or equal to `"value1"` and the property `"name2"` is greater than or equal to `"value2"`.

Expression	Description
`["name1">="value1" or "name2">="value2"]`	Selects all the elements from an array or object where the property `"name1"` is greater than or equal to `"value1"` or the property `"name2"` is greater than or equal to `"value2"`.
`["nodename1">"nodename2"]`	Selects all the elements from an array or object where the value of property named `"nodename1"` is greater than the value of property named `"nodename2"`. This node comparison applies to all relational operators discussed above.
`[*]`	Selects all the elements from an array or object.
`["node"."child"]`	Selects all the elements from an array or object that contain a property named `"node"` that contains `"child"`.

The following are example RedPath queries using the expressions shown in Table 1:

- `/Chassis[0]` : Returns the first `Chassis` resource.
- `/Chassis[SKU=1234]` : Returns all `Chassis` resources whose `SKU` property equals `1234`.
- `/Chassis[*]/Sensors[Reading>Critical.UpperThreshold]` : Returns all `Sensors` resources whose `Reading` property is greater than `Critical` `UpperThreshold` property.
- `/Systems[Storage]` : Returns all the `ComputerSystem` resources that contain a `Storage` property.
- `/Systems[!Storage]` : Returns all the `ComputerSystem` resources that do not contain a `Storage` property.
- `/Systems[*]` : Returns all the `ComputerSystem` resources.
- `/Systems[*]/Processors[*]` : Returns all `Processor` resources from all `ComputerSystem` resources.
- `/SessionService/Sessions[last()]` : Returns the last `Session` resource.
- `/Chassis[Location.Info]` : Returns all the `Chassis` resources that contain a `Location` property with an `Info` property.
- `/Systems[Status.Health=OK]` : Returns all `ComputerSystem` resources whose `Health` property inside `Status` equals `OK`.
- `/Systems[Status.Health=OK]/Memory[CapacityMiB>1024]` : Returns all `Memory` resources whose `CapacityMiB` property is greater than `1024` from all `ComputerSystem` resources whose `Health` equals `OK`.
- `/Chassis[*]/Sensors[Reading and Critical.UpperThreshold]` : Returns all `Sensors` resources that contain both the `Reading` property and `UpperThreshold` property.
- `/Chassis[*]/Sensors[Reading or Critical.UpperThreshold]` : Returns all `Sensors` resources that contain the `Reading` property or `UpperThreshold` property or both.
- `/Chassis[Actions.#Chassis\.Reset.@Redfish\.ActionInfo]` : Returns all `Chassis` resources that contain property `@Redfish.ActionInfo` inside `#Chassis.Reset` object that resides within the `Actions` object. In this example, there are `.` characters in the node names that are escaped with `\` characters.

3.1 RedPath language constructs

RedPath interpreters shall implement all constructs defined in this clause.

RedPath is defined to be an absolute location path relative to the `/redfish/v1` node. A relative path is a sequence of steps.

```
[1] RedPath ::== '/' RelativePath

[2] RelativePath ::== Step
                    | RelativePath '/' Step
```

Each step selects a set of child nodes from the current node. The step can select a single specific child node or provide a predicate that describes the criteria for selecting child nodes.

```
[3] Step ::== NodeName
            | NodeName '[' Predicate ']'
```

If `NodeName` in the step expression resolves to an array, the selection criteria in predicate expression shall apply to nodes within that array. Predicates shall apply to objects when `NodeName` resolves to Redfish resource collection to select a subset of collection members. For example, a Chassis instance within a collection can be queried using RedPath `/Chassis/Members[7]` where predicate `[7]` applies to `Members` array or the same instance can be queried using RedPath `/Chassis[7]` where predicate `[7]` applies to `Chassis` resource collection.

```
[4] Predicate ::== Index
                 | '*'
                 | 'last()'
                 | BooleanExpr
                 | NodeName '.' NodeName
```

`BooleanExpr` identifies expressions that evaluate to one of the two values: `true` or `false`.

```
[5] BooleanExpr ::== NodeName
                   | ValueComparison
                   | NodeName LogicalOp BooleanExpr
                   | '!' BooleanExpr
```

> **Note:** `NodeName` as a `BooleanExpr` is an implicit `Contains()` function that instructs a RedPath interpreter to check if the Redfish object contains `NodeName` and evaluates to `true` or `false`.

```
[6] ValueComparison ::== NodeName RelationalOp NodeValue
```

The node name is a qualified name referencing a Redfish property name. Its syntax matches valid syntax for a Redfish property name as defined in the *Redfish Specification*.

> **Note:** Some Redfish properties contain a `.` character in the property name, which can conflict with the predicate expression syntax. This specification allows escaping `.` character using backslash `\`. For example, to filter nodes that contain `@odata.id`, a predicate expression can be `[@odata\.id]`.

```
[7] NodeName ::== RedfishPropertyNameSyntax
```

The node value permits any valid JSON value as defined by the *JSON syntax*.

```
[8] NodeValue ::== JsonValueSyntax

[9] RelationalOp ::== '=' | '<' | '<=' | '>' | '>=' | '!='

[10] LogicalOp ::== 'and' | 'or'
```

The order of precedence, from highest to lowest, is as follows:

- `()`
- `<=`, `<`, `>=`, `>`
- `=`, `!=`
- `and`
- `or`

All operators are left associative and parenthesis can override operator precedence where applicable.

4 RedPath interpreter design tenets

RedPath is a declarative programming language designed to interoperate with any conformant interpreter implementation that abstracts a Redfish service as a single json document. Interpreters that implement RedPath shall support interacting with Redfish services that do not support query parameters.

5 ANNEX A (informative) Change log

Version	Date	Description
1.1.0	2023-04-04	Added logical operators in predicate expression.
		Added order of precedence for operators.
		Added logical operations between Redfish properties.
		Added character escaping within predicate expression.
		Added `BooleanExpr` and `ValueComparison` expressions in RedPath syntax.
		Corrected description of predicate expressions to clarify that predicate applies to all elements of array and not an object.
1.0.1	2022-12-07	Corrected the usage of indexing an array to be 0-based.
1.0.0	2022-10-12	Initial release.