**OSLC Query literals syntax & semantics proposal**

# Motivation

The OSLC query 2.0 spec mandated strict adherence to SPARQL semantics for literals. There are several problems with this that make compliance overly burdensome and inappropriate. For example, it would require that a server using an RDB for persistence, also create an RDF dataset and use SPARQL to query that dataset. Very few, current implementations do that, and hence most are non-compliant. We want the OSLC query spec to be easily implementable by servers regardless of how they persist data, and whether they have an open type system that allows any property to be added to a resource without being restricted by a type system, or have a finite type system in which properties are enumerated and must be defined in the type system before custom properties are added to resources.

# Overview

The proposal is to describe literal values and operators so that operations that are useful and easily implementable are defined with MUST, and where persistent-dependent semantics are either described by MAY or omitted.

# Proposal

The proposal mirrors the Literal Value Types in <https://tools.oasis-open.org/version-control/browse/wsvn/oslc-core/trunk/specs/resource-shape.html> with the addition of oslc:Resource.

|  |  |  |
| --- | --- | --- |
| **Value type in shape** | **Permitted literal values** | **Operators** |
| rdf:XMLLiteral | An XMLLiteral, such as **“abc”^^rdf:XMLLiteral**, MUST be supported. A server MAY treat the literal in the same way as a plain literal with the data type removed such as **“abc”**.  A plain string literal, such as **“abc”** MUST be supported. A server MAY treat the literal in the same way as an XML literal string such as **“abc”^^rdf:XMLLiteral**.  An xsd:string typed literal, such as “abc”^^xsd:string, MUST be supported. A server MAY treat the literal in the same way as an XML literal string such as **“abc”^^rdf:XMLLiteral**.  Servers MAY validate the string for adherence to XMLLiteral compliance. | The operators **=**, **!=**, and **in** MUST be supported.  Servers MUST implement one of the following semantics:   * case sensitive string compare * case insensitive string compare * pattern match semantics where **%** means match zero or more characters, and **\_** means match any single character.   A server may choose which of the above to apply based on whether the string contains a **%** or **\_** character.  Other operators MAY be supported and MAY have implementation-dependent semantics.  A server MAY report such usage as unsupported. |
| xsd:boolean | The values **true** and **false** MUST be supported.  The values **“true”^^xsd:boolean** and **“false”^^xsd:boolean** MUST be supported and MUST have the same meaning as true and false respectively.  The values **“true”** and **“false”** MAY be supported, and if so, MUST have the same meaning as true and false respectively.  A language tagged literal, such as **“true”@en** MAY be supported and have implementation and/or locale dependent behavior. | The operators **=**, **!=**, and **in** MUST be supported.  Other operators MAY be supported and MAY have implementation-dependent semantics. |
| xsd:dateTime | Values that are conformant with an xsd:dateTime, as described in <https://www.w3.org/TR/2004/REC-xmlschema-2-20041028/#dt-dateTime>, such as **“2018-01-30T12:25:00”^^xsd:dateTime**, MUST be supported.  Plain literals whose string value conforms to an xsd:dateTime, such as **“2018-01-30T12:25:00”**, MAY be supported, and if so, MUST have the same meaning as an xsd:dateTime typed literal such as **“2018-01-30T12:25:00”^^xsd:dateTime”**. | The operators **=**, **!=**, **<**, **>**, **<=**, **>=**, and **in** MUST be supported, and MUST have the semantics of a date/time comparison. |
| xsd:decimal | Decimal literals with or without a decimal point, such as **42** and **42.0**, MUST be supported if the server supports decimal property values.  An xsd:decimal typed literal, such as **“42”^^xsd:decimal** or **“42.0”^^xsd:decimal**, MUST be supported.  A plain string with a decimal value, such as **“42”** or **“42.0”**, MAY be supported, and if so, MUST have the same meaning as the xsd:decimal typed literal. | The operators **=**, **!=**, **<**, **>**, **<=**, **>=**, and **in** MUST be supported, and MUST have the semantics of a decimal comparison. |
| xsd:double | Decimal literals with or without a decimal point, such as **42** and **42.0**, MUST be supported if the server supports double precision floating point property values.  An xsd:double typed literal, such as **“42”^^xsd:double** or **“42.0”^^xsd:double**, MUST be supported if the server supports double precision floating point property values.  A plain string with a decimal value, such as **“42”** or **“42.0”**, MAY be supported and if so, MUST have the same meaning as the xsd:double typed literal. | The operators **=**, **!=**, **<**, **>**, **<=**, **>=**, and **in** MUST be supported, and MUST have the semantics of a double comparison. |
| xsd:float | Decimal literals with or without a decimal point, such as **42** and **42.0**, MUST be supported if the server supports single precision floating point property values.  An xsd:float typed literal, such as **“42”^^xsd:float** or **“42.0”^^xsd:float**, MUST be supported if the server supports single precision floating point property values.  A plain string with a decimal value, such as **“42”** or **“42.0”**, MAY be supported and if so, MUST have the same meaning as the xsd:float typed literal. | The operators **=**, **!=**, **<**, **>**, **<=**, **>=**, and **in** MUST be supported, and MUST have the semantics of a float comparison. |
| xsd:integer | Decimal literals without a decimal point, such as **42**, MUST be supported if the server supports integer property values.  An xsd:integer typed literal, such as **“42”^^xsd:integer**, MUST be supported and treated as an integer value if the server supports integer property values.  An xsd:decimal typed literal without a decimal point, such as **“42”^^xsd:decimal**, MAY be supported and if so, MUST have the same meaning as the xsd:integer typed literal.  Strings such as **“42”**, **“42”@en**, and **“42”^^xsd:string** MAY be supported as an integer value of the string. | The operators **=**, **!=**, **<**, **>**, **<=**, **>=**, and **in** MUST be supported, and MUST have the semantics of an integer comparison. |
| xsd:string | A plain string literal, such as “abc” MUST be supported.  An xsd:string typed literal, such as “abc”^^xsd:string, MUST be supported. A server MUST treat the literal the same way as a plain string literal such as **“abc”**.  An XMLLiteral, such as **“abc”^^rdf:XMLLiteral**, MUST be supported. A server MAY treat the literal in the same way as a plain literal string such as **“abc”**.  A string with a language tag, such as **“abc”@en**, MAY be supported. The string MAY be treated as a plain string, or it MAY be treated as a language specific string that only matches a string with an identical value and language tag. | The operators **=**, **!=**, and **in** MUST be supported.  Servers MUST implement one of the following semantics:   * case sensitive string compare * case insensitive string compare * pattern match semantics where **%** means match zero or more characters, and **\_** means match any single character.   A server may choose which of the above to apply based on whether the string contains a **%** or **\_** character.  Servers MAY choose which semantic to use based on the property being compared.  Other operators MAY be supported and MAY have implementation-dependent semantics, or a server MAY report such usage as unsupported. |
| oslc:Resource | A *uri\_ref\_esc* such as **<http://example.com>** MUST be supported.  A *PrefixedName* such as **oslc:Zero-or-many**, MAY be supported. If supported, the value MUST be treated as equivalent to its corresponding *uri\_ref\_esc*.  Should the example use a URI that needs to be escaped? | The operators **=**, **!=**, and **in** MUST be supported.  A server MUST implement this as a case-sensitive URI string comparison.  Other operators MAY be supported and MAY have implementation-dependent semantics, or a server MAY report such usage as unsupported. |