SPARQL 1.1 Federated Query

W3C Recommendation 21 March 2013

This version:
http://www.w3.org/TR/2013/REC-sparql11-federated-query-20130321/
Latest version:
http://www.w3.org/TR/sparql11-federated-query/
Previous version:
http://www.w3.org/TR/2012/PR-sparql11-federated-query-20121108/
Editors:
Eric Prud'hommeaux, W3C <eric@w3.org>
Carlos Buil-Aranda, Ontology Engineering Group, UPM, Spain; currently at Universidad Pontificia Católica de Chile
Contributors:
Andy Seaborne, The Apache Software Foundation
Axel Polleres, Siemens AG <axel.polleres@siemens.com>
Lee Feigenbaum, Cambridge Semantics <lee@thefigtrees.net>
Gregory Todd Williams, Rensselaer Polytechnic Institute <greg@evilfunhouse.com>

Please refer to the **errata** for this document, which may include some normative corrections.

See also **translations**.

Abstract

RDF is a directed, labeled graph data format for representing information in the Web. SPARQL can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middleware. This specification defines the syntax and semantics of SPARQL 1.1 Federated Query extension for executing queries distributed over different SPARQL endpoints. The `SERVICE` keyword extends SPARQL 1.1 to support queries that merge data distributed across the Web.

Status of this Document

May Be Superseded

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at http://www.w3.org/TR/.

Set of Documents

This document is one of eleven SPARQL 1.1 Recommendations produced by the SPARQL Working Group:

1. SPARQL 1.1 Overview
2. SPARQL 1.1 Query Language
3. SPARQL 1.1 Update
4. SPARQL1.1 Service Description
5. SPARQL 1.1 Federated Query (this document)
6. SPARQL 1.1 Query Results JSON Format
7. SPARQL 1.1 Query Results CSV and TSV Formats
8. SPARQL Query Results XML Format (Second Edition)
9. SPARQL 1.1 Entailment Regimes
10. SPARQL 1.1 Protocol
11. SPARQL 1.1 Graph Store HTTP Protocol

No Substantive Changes

There have been no substantive changes to this document since the previous version. Minor editorial changes, if any, are detailed in the change log and visible in the color-coded diff.

Please Send Comments

Please send any comments to public-rdf-dawg-comments@w3.org (public archive). Although work on this document by the SPARQL Working Group is complete, comments may be addressed in the errata or in future revisions. Open discussion is welcome at public-sparql-dev@w3.org (public archive).

Endorsed By W3C

This document has been reviewed by W3C Members, by software developers, and by other W3C groups and interested parties, and is endorsed by the Director as a W3C Recommendation. It is a stable document and may be used as reference material or cited from another document. W3C's role in making the Recommendation is to draw attention to the specification and to promote its widespread deployment. This enhances the functionality and interoperability of the Web.

Patents

This document was produced by a group operating under the 5 February 2004 W3C Patent Policy. W3C maintains a public list of any patent disclosures made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) must disclose the information in accordance with section 6 of the W3C Patent Policy.

Table of Contents

Appendices

---

1 Introduction

The growing number of SPARQL query services offer data consumers an opportunity to merge data distributed across the Web. This specification defines the syntax and semantics of the `SERVICE` extension to the SPARQL 1.1 Query Language. This extension allows a query author to direct a portion of a query to a particular SPARQL endpoint. Results are returned to the federated query processor and are combined with results from the rest of the query.

1.1 Document Conventions

1.1.1 Namespaces

This document uses the same namespaces as from the SPARQL 1.1 Query document.

1.1.2 Result Descriptions

Result sets are illustrated in tabular form as in the SPARQL 1.1 Query document.

x	y	z
"Alice"	<http://example/a>	

A 'binding' is a pair (variable, RDF term). There are three variables: `x`, `y` and `z` (shown as column headers). Each solution is shown as one row in the body of the table. Here, there is a single solution, in which variable `x` is bound to `"Alice"`, variable `y` is bound to `http://example/a`, and variable `z` is not bound to an RDF term. Variables are not required to be bound in a solution.

1.1.3 Terminology

The following terms are defined in SPARQL 1.1 Query Language [SQRY] and reused in this document:

- IRI (corresponds to the Concepts and Abstract Syntax term `RDF URI reference`)
- Solution Mapping
- Solution Sequence

2 SPARQL 1.1 Federated Query Extension

The `SERVICE` keyword instructs a federated query processor to invoke a portion of a SPARQL query against a remote SPARQL endpoint. This section presents examples of how to use the `SERVICE` keyword. The following sections define the syntax and semantics of this extension.

2.1 Simple query to a remote SPARQL endpoint

This example shows how to query a remote SPARQL endpoint and join the returned data with the data from the local RDF Dataset. Consider a query to find the names of the people we know. Data about the names of various people is available at the `http://people.example.org/sparql` endpoint:

```
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
@prefix : <http://example.org/> .

:people15  foaf:name     "Alice" .
:people16  foaf:name     "Bob" .
:people17  foaf:name     "Charles" .
:people18  foaf:name     "Daisy" .
```

and one wants to combine with a local FOAF file `http://example.org/myfoaf.rdf` that contains the single triple:

```
<http://example.org/myfoaf/I> <http://xmlns.com/foaf/0.1/knows>  <http://example.org/people15> .
```

Query:

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
SELECT ?name
FROM <http://example.org/myfoaf.rdf>
WHERE
{
  <http://example.org/myfoaf/I> foaf:knows ?person .
  SERVICE <http://people.example.org/sparql> {
    ?person foaf:name ?name . }
}
```

This query, on the data above, has one solution:

Query Result:

name
"Alice"

2.2 SPARQL query with OPTIONAL to two remote SPARQL endpoints

Imagine we want to query people and optionally obtain their interests and the names of people they know. Imagine for instance, two endpoints containing data about people:

Data in the default graph at remote SPARQL endpoint: `http://people.example.org/sparql`

```
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
@prefix : <http://example.org/> .

:people15  foaf:name     "Alice" .
:people16  foaf:name     "Bob" .
:people17  foaf:name     "Charles" .
:people17  foaf:interest     <http://www.w3.org/2001/sw/rdb2rdf/> .
```

and data in the default graph the remote SPARQL endpoint: `http://people2.example.org/sparql`

```
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
@prefix : <http://example.org/> .

:people15  foaf:knows    :people18 .
:people18  foaf:name     "Mike" .
:people17  foaf:knows    :people19 .
:people19  foaf:name     "Daisy" .
```

Query:

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
SELECT ?person ?interest ?known
WHERE
{
  SERVICE <http://people.example.org/sparql> {
    ?person foaf:name ?name .
    OPTIONAL {
      ?person foaf:interest ?interest .
      SERVICE <http://people2.example.org/sparql> {
        ?person foaf:knows ?known . } }
  }
}
```

This query, on the data above, has three solutions:

Query Result:

person	interest	known
"Alice"		
"Bob"		
"Charles"	<http://www.w3.org/2001/sw/rdb2rdf/>	<http://example.org/people19>

Notice that in the query above there is a nested SERVICE in the OPTIONAL clause. This query requires the SPARQL query service at `http://people.example.org/sparql` to support basic federated query.

2.3 Service Execution Failure

The execution of a SERVICE pattern may fail due to several reasons: the remote service may be down, the service IRI may not be dereferenceable, or the endpoint may return an error to the query. Normally, under such circumstances the invoked query containing a SERVICE pattern fails as a whole. Queries may explicitly allow failed SERVICE requests with the use of the SILENT keyword. The SILENT keyword indicates that errors encountered while accessing a remote SPARQL endpoint should be ignored while processing the query. The failed SERVICE clause is treated as if it had a result of a single solution with no bindings.

In the following query the SILENT keyword is present. If the remote SPARQL endpoint is not available because the SPARQL endpoint does not exist, it is down or it is not accessible the query will return a solution sequence of one empty solution mapping. If the SILENT keyword is not present, the query will stop and return the error.

Data in `<http://people.example.org/sparql>` endpoint:

```
<http://example.org/people15>  <http://xmlns.com/foaf/0.1/name>     "Charles" .
```

Query:

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE
{
  SERVICE SILENT <http://people.example.org/sparql> {
    <http://example.org/people15> foaf:name ?name . }
}
```

Query result if an error occurs while querying the remote SPARQL endpoint:

name

2.4 Interplay of SERVICE and VALUES (Informative)

SPARQL 1.1 Query includes the VALUES clause (VALUES), which can be used to provide an unordered solution sequence that is joined with the results of the query evaluation. Implementers of SPARQL 1.1 Federated Query may use the VALUES clause to constrain the results received from a remote endpoint based on solution bindings from evaluating other parts of the query.

The following example shows how SERVICE and VALUES can work together. Suppose a query that asks for all instances of foaf:Person in the default graph and also their known people in the remote endpoint `http://example.org/sparql`:

Data in the default graph:

```
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
@prefix : <http://example.org/> .

:a a foaf:Person  ;
   foaf:name     "Alan" ;
   foaf:mbox;     "alan@example.org" .
:b a foaf:Person  ;
   foaf:name     "Bob" ;
   foaf:mbox     "bob@example.org" .
```

and data in the default graph the remote SPARQL endpoint `http://example.org/sparql`:

```
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
@prefix : <http://example.org/> .

:a  foaf:knows     :b .
:b  foaf:knows     :c .
:c  foaf:knows     :a .
:a  foaf:interest  "SPARQL 1.1 Basic Federated Query" .
:b  foaf:interest  "SPARQL 1.1 Query" .
:c  foaf:interest  "RDB2RDF Direct mapping" .
```

Query:

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
SELECT ?s
{
  ?s a foaf:Person .
  SERVICE <http://example.org/sparql> {?s foaf:knows ?o }
}
```

When the original query is executed naively, with an unconstrained service call the endpoint may return more results than necessary. It may also happen that the SPARQL endpoint will not return all of them. Many existing SPARQL endpoints have restrictions in the number of results they return and may miss the ones matching subjects `?s` from the local default graph. Thus, an implementation of a query planner for federated queries may decide to decompose the query into two queries instead, where first the bindings from the local default graph are evaluated:

Query:

```
PREFIX : <http://example.org/>
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
SELECT ?s
{
  ?s a foaf:Person
}
```

This query, on the data above, has two solutions:

Query Result:

s
<http://example.org/a>
<http://example.org/b>

Next, dispatch to the remote endpoint <http://example.org/sparql> a constrained query with the solutions for `?s`:

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
PREFIX : <http://example.org/>
SELECT * {?s foaf:knows ?o } VALUES (?s) { (:a) (:b) }
```

The query process involving SERVICE limits the data returned to the data it needs for the overall query:

Query:

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
SELECT ?s ?o
{
  ?s a foaf:Person
  SERVICE <http://example.org/sparql> {?s foaf:knows ?o }
}
```

This query, on the data above using VALUES, has the expected two solutions to the overall query:

Query Result:

s	o
<http://example.org/a>	<http://example.org/b>
<http://example.org/b>	<http://example.org/c>

3 SPARQL 1.1 Simple Federation Extension: semantics

3.1 Translation to the SPARQL Algebra

The SERVICE extension is defined as an additional type of GroupGraphPattern, with an accompanying addition to SPARQL Query 1.1's Transform (syntax form):

> If the form is `GroupGraphPattern`
>
> > From the Translate Graph Patterns section of [SPARQL 1.1 Query Language] we extend the transformation of GroupGraphPattern to define the transformation of SERVICE patterns:

```
Let FS := the empty set
Let G := the empty pattern, Z, a basic graph pattern which is the empty set.
Let SilentOp := boolean, indicating SERVICE error behavior.

For each element E in the GroupGraphPattern
    If E is of the form FILTER(expr)
        FS := FS ∪ {expr}
        End

    If E is of the form OPTIONAL{P}
        Let A := Transform(P)
        If A is of the form Filter(F, A2)
            G := LeftJoin(G, A2, F)
        Else
            G := LeftJoin(G, A, true)
            End
        End

    If E is of the form MINUS{P}
        G := Minus(G, Transform(P))
        End

    If E is of the form BIND(expr AS var)
        G := Extend(G, var, expr)
        End

    If E is any other form
        Let A := Transform(E)
        G := Join(G, A)
        End

    If E is of the form SERVICE [SILENT] IRI {P}
        Let G := Join(G, Service(IRI, Transform(P), SilentOp))
        End

   End

If FS is not empty:
  Let X := Conjunction of expressions in FS
  G := Filter(X, G)

The result is G.
```

3.2 SPARQL 1.1 Simple Federation Extension Algebra

The evaluation of SERVICE is defined in terms of the SPARQL Results [RESULTS] returned by a SPARQL Protocol [SPROT] execution of the nested graph pattern:

Definition: Evaluation of a Service Pattern

Let

- iri be an IRI,
- Ω_0 the solution set with one empty solution, and
- SilentOp be a boolean variable to indicate that SERVICE execution should ignore errors when true.

then:

eval(D(G), Service(IRI,P,SilentOp)) = Invocation(iri, P, SilentOp)

where: Invocation(IRI, P, SilentOp) is

- the multiset of solution mappings corresponding to the results of executing query `SELECT * WHERE Q` against the service endpoint with IRI iri where Q is the serialization of P in SPARQL syntax, in case of a successful service invocation according to the SPARQL protocol, and otherwise
- Ω_0. in case SilentOp is true, and otherwise
- error.

3.2.1 SERVICE Examples

In the folowing section we introduce two examples showing the evaluation of SERVICE patterns in the SPARQL algebra:

Example: a SERVICE graph pattern in a series of joins:

```
... WHERE { { ?s :p1 ?v1 } SERVICE <srvc> {?s :p2 ?v2 } { ?s :p3 ?v2 } }

Join( Service( <srvc>,
               BGP( ?s :p2 ?v2 ), false ),
      BGP( ?s :p3 ?v2 ) )
```

Example: a SERVICE SILENT graph pattern in a series of joins:

```
... WHERE { { ?s :p1 ?v1 } SERVICE SILENT <srvc> {?s :p2 ?v2 } { ?s :p3 ?v2 } }

Join( Service( <srvc>,
               BGP( ?s :p2 ?v2 ), true ),
      BGP( ?s :p3 ?v2 ) )
```

4 SERVICE Variables (Informative)

In the this section we do not present official evaluation semantics for the SPARQL pattern SERVICE VAR. We only provide indications about how the evaluation of the SPARQL pattern SERVICE VAR can be evaluated.

A variable used in place of a service IRI indicates that the service call for any solution depends on that variable's binding in that solution. For instance, the default graph may contain data about which services contain data about project endpoints. We assume the following data on various projects that contains information about SPARQL endpoints where data about these projects (using the DOAP vocabulary) can be queried from:

```
@prefix void:    <http://rdfs.org/ns/void#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix doap: <http://usefulinc.com/ns/doap#> .

[] dc:subject "Querying RDF" ;
   void:sparqlEndpoint <http://projects1.example.org/sparql> .
[] dc:subject "Querying RDF remotely" ;
   void:sparqlEndpoint <http://projects2.example.org/sparql> .
[] dc:subject "Updating RDF remotely"  ;
   void:sparqlEndpoint <http://projects3.example.org/sparql> .
```

Data in the default graph at remote SPARQL endpoint http://projects2.example.org/sparql:

```
_:project1  doap:name    "Query remote RDF Data" .
_:project1  doap:created "2011-02-12"^^xsd:date .
_:project2  doap:name    "Querying multiple SPARQL endpoints" .
_:project2  doap:created "2011-02-13"^^xsd:date .
```

Data in the default graph at remote SPARQL endpoint http://projects3.example.org/sparql:

```
_:project3  doap:name    "Update remote RDF Data" .
_:project3  doap:created "2011-02-14"^^xsd:date .
```

We now want to query the project names of projects on the subject "remote":

```
PREFIX  void: <http://rdfs.org/ns/void#>
PREFIX  dc:   <http://purl.org/dc/elements/1.1/>
PREFIX  doap: <http://usefulinc.com/ns/doap#>

SELECT ?service ?projectName
WHERE {
  # Find the service with subject "remote".
  ?p dc:subject ?projectSubject ;
     void:sparqlEndpoint ?service .
     FILTER regex(?projectSubject, "remote")

  # Query that service projects.
  SERVICE ?service {
     ?project  doap:name ?projectName . }
}
```

In the following table we present the intuitive solutions for this query with the data above:

Query Result:

service	title
<http://projects2.example.org/sparql>	"Query remote RDF Data"
<http://projects2.example.org/sparql>	"Querying multiple SPARQL endpoints"
<http://projects3.example.org/sparql>	"Update remote RDF Data"

A SERVICE clause involving a variable can be executed as a series of separate invocations of SPARQL query services. The results of each invocation are combined using union.

The query engine must determine the possible target SPARQL query services. The exact mechanism for doing this is not defined in this document. Execution order may also be used to determine the list of services to to be tried. The example above suggests a specific order of execution: evaluating the basic graph pattern and filter outside the SERVICE block first will yield bindings for ?service which may then be used to evaluate the SERVICE block:

```
?p dc:subject ?projectSubject ;
   void:sparqlEndpoint ?service
   FILTER regex(?projectSubject, "remote")
```

Once ?service has been evaluated it is possible to execute SERVICE for each value of ?service:

```
SERVICE ?service {
?project doap:name ?projectName . }
```

Note that blank nodes are unique to any document which serializes them. Also, SERVICE calls depend on the SPARQL Protocol [SPROT] which transfers serialized RDF documents making blank nodes unique between service calls.

5 Conformance

See section 4 SPARQL 1.1 Federated Query Grammar regarding conformance of SPARQL Query strings that include the SPARQL 1.1 Federated Query Extensions. See section 3.1 Definition of SERVICE for conformance of query results for the SERVICE keyword.

This specification is intended for use in conjunction with the SPARQL 1.1 Query Language. See that specification for its conformance criteria.

6 Security Considerations (Informative)

SPARQL queries using SERVICE imply that a URI will be dereferenced, and that the result will be incorporated into a working data set. All of the security issues of SPARQL Protocol 1.1 [SPROT] Section 3.1 SPARQL 1.1 Query [SQRY] Section 21, and Uniform Resource Identifier (URI): Generic Syntax [RFC3986] Section 7 should be considered.

A References

A.1 Normative References

[SQRY]
SPARQL 1.1 Query Language, S. Harris, A. Seaborne, Editors, W3C Recommendation, 21 March 2013, http://www.w3.org/TR/2013/REC-sparql11-query-20130321. Latest version available at http://www.w3.org/TR/sparql11-query.

[SPROT]
SPARQL 1.1 Protocol, L. Feigenbaum, G. Williams, K. Clark, E. Torres, Editors, W3C Recommendation, 21 March 2013, http://www.w3.org/TR/2013/REC-sparql11-protocol-20130321. Latest version available at http://www.w3.org/TR/sparql11-protocol.

[CHARMOD]
Character Model for the World Wide Web 1.0: Fundamentals, R. Ishida, F. Yergeau, M. J. Düst, M. Wolf, T. Texin, Editors, W3C Recommendation, 15 February 2005, http://www.w3.org/TR/2005/REC-charmod-20050215/ . Latest version available at http://www.w3.org/TR/charmod/ .

[RFC3629]
RFC 3629 UTF-8, a transformation format of ISO 10646, F. Yergeau November 2003

[RFC3986]
RFC 3986 Uniform Resource Identifier (URI): Generic Syntax, T. Berners-Lee, R. Fielding, L. Masinter January 2005

[RFC3987]
RFC 3987, "Internationalized Resource Identifiers (IRIs)", M. Dürst , M. Suignard
[UNICODE]
The Unicode Standard, Version 4. ISBN 0-321-18578-1, as updated from time to time by the publication of new versions. The latest version of Unicode and additional information on versions of the standard and of the Unicode Character Database is available at http://www.unicode.org/unicode/standard/versions/.
[XML11]
Extensible Markup Language (XML) 1.1, J. Cowan, J. Paoli, E. Maler, C. M. Sperberg-McQueen, F. Yergeau, T. Bray, Editors, W3C Recommendation, 4 February 2004, http://www.w3.org/TR/2004/REC-xml11-20040204/ . Latest version available at http://www.w3.org/TR/xml11/ .
[BCP47]
Best Common Practice 47, P. V. Biron, A. Malhotra, Editors, W3C Recommendation, 28 October 2004, http://www.rfc-editor.org/rfc/bcp/bcp47.txt .

A.2 Other References

[RESULTS]
SPARQL Query Results XML Format (Second Edition), D. Beckett, J. Broekstra, Editors, W3C Recommendation, 21 March 2013, http://www.w3.org/TR/2013/REC-rdf-sparql-XMLres-20130321. Latest version available at http://www.w3.org/TR/rdf-sparql-XMLres.
[TURTLE]
Turtle: Terse RDF Triple Language, E Prud'hommeaux, G Carothers, Editors, W3C Candidate Recommendation, 19 February 2013, http://www.w3.org/TR/2013/CR-turtle-20130219/. Latest version available at http://www.w3.org/TR/turtle/.

B Acknowledgements

The SPARQL 1.1 Federated Query document is a product of the whole of the W3C SPARQL Working Group, and our thanks for discussions, comments and reviews go to all present and past members.

In addition, we have had comments and discussions with many people through the working group comments list. All comments go to making a better document. Carlos would also like to particularly thank Jorge Pérez, Oscar Corcho and Marcelo Arenas for their discussions on the syntax and semantics of the Federated query extension.

C CVS History (Last Call and after)

Change Log

Changes since Proposed Recommendation

- None

Changes since Last Call

- Updated references, fix DOAP URL
- Changed the word "BINDINGS" to "VALUES" to match change in Query Specification.