

Development and QA, TBL

Page Contents

- 1 Available Web Services
- 2 Run all DASH Test Cases

Available Web Services

EDG exposes common operations as public for use in Web Services, such as

- SPIN Templates from .spin. files
- SPARQLMotion scripts from .sms. files
- SWP Services from .ui. files.

CRUD operation are also exposed through SPARQL endpoint which can be called via a REST API. TopBraid also includes support for GraphQL services and mutations since version 6.0. This support is limited to RDF data that is under control of the TopBraid EDG (teamwork) framework, and in particular to only certain asset collections (which can be recognized by their GraphQL link on the Export tab). Find out more here <https://www.topquadrant.com/technology/graphql/>

Using the provided IDE, TopBraid Composer, additional template web services can be developed if needed. For more information on Web Services with TopBraid see <http://www.topquadrant.com/2015/07/24/web-services-and-topquadrant-products/> and <http://www.topquadrant.com/2013/07/01/creating-web-services-with-the-topbraid-platform/>. TopBraid includes a facility for auto-generating API documentation. This means that any API that has been developed and deployed is automatically fully documented when marked as public. This includes pre-built TopQuadrant provided APIs as well as any customer developed APIs.

Authentication applied to EDG will also apply to the web service call. Please see [Server Installation and Integration#WebServiceAuthentication](#) for more information.

Selecting the Available Web Services displays a page that lists web services available on this server. Selecting the checkbox next to any of these names displays documentation below the list about how to call that web service.

Select categories:

↓ Select all

- AutoClassifier Web Services (7 services)**
APIs for training AutoClassifier models, starting and monitoring AutoClassifier batch jobs, and getting tag recommendations on individual pieces of content.
- Compliance Reports (3 services)**
APIs to display the differences between a given set of input resources and a given base graph. This can be used to (periodically) check the compliance of data used by external applications against reference datasets managed by TopBraid.
- Crosswalk Modules (4 services)**
APIs, implemented as SPIN query templates, that encapsulate frequently needed queries against Crosswalks.
- Data Generator Example Installer (1 services)**
API for generating a set of asset collections based on random data.
- Reference Datasets (3 services)**
APIs for accessing codes within the Reference Datasets.
- SPIN Dynamic Ranges Library (2 services)**
APIs for getting all subjects or all objects that are two steps away from the resource provided as ?this argument via the property ?firstProperty and then ?secondProperty.
- SPIN Standard Library (3 services)**
A collection of generally useful SPARQL functions (expressed as SPIN functions), and SPIN templates. Also provides a top-level classification of functions, and definitions of the standard SPARQL functions.
- SWA SPIN Modules (3 services)**
A collection of SPIN functions and auxiliary definitions to support building interactive web applications.
- TBL Admin (5 services)**
A collection of APIs for performing various administrative tasks such as refreshing indexes, deleting asset collections and exporting data
- Teamwork SPIN Modules (2 services)**
A collection of APIs implemented as SPIN modules that provide generic utility services operating on Teamwork (TCH) graphs.
- Teamwork SPIN Templates (3 services)**
A collection of APIs implemented as SPIN templates that provide generic utility services operating on Teamwork (TCH) graphs.
- Teamwork UI Elements and Modules (6 services)**
A collection of APIs for programmatically creating and cloning asset collections and for finding resources within asset collections based on their full or partial URIs.
- Teamwork Workflows Support (1 services)**
API for starting a workflow.
- TopBraid SKOS Templates (32 services)**
A collection of APIs implemented as SPIN templates that are designed for retrieving information about taxonomy concepts. These APIs encapsulate typical SKOS-based query patterns such as finding all broader parents of a concept provided as an argument.

AutoClassifier Web Services

APIs for training AutoClassifier models, starting and monitoring AutoClassifier batch jobs, and getting tag recommendations on individual pieces of content.

Graph URI: <http://evn.topbrailive.org/tagger/autoclassifier-services>

acsrv:AutoClassifyContent (ui:Service)

A service that runs the AutoClassifier on a snippet of content provided in the service call. Delivers results in JSON. The `_base` parameter is ignored.

Each recommended concept has the following fields:

- `concept`: The concept's URI
- `label`: The concept's label (literal with the configuration's language settings; only available if a vocabulary graph can be established)
- `probability`: The concept's probability (0..1)

Arguments

autotagger:configuration (rdfs:Resource): The AutoClassifier configuration resource (often a content tag set resource), a resource in the configuration graph described with <code>tagger:xxx</code> properties
arg:configurationGraph (rdfs:Resource): [Optional] Graph containing the configuration resource. If unspecified, assume that the configuration resource itself is a named graph containing its own description.
arg:text (xsd:string): The text to auto-classify
autotagger:trainingID (xsd:string): [Optional] The ID of a training result (see <code>StartTraining</code> service) to be employed by the AutoClassifier; if not specified, use the URI of the configuration resource
autotagger:vocabularyGraph (rdfs:Resource): [Optional] Named graph of the concept vocabulary; will be used to retrieve labels for recommended concepts. Defaults to <code>tagger:objectGraph</code> of configuration resource.

Service Syntax

```
http://.../smp?  
_viewClass=acsrv:AutoClassifyContent&_base=...&configuration=...&configurationGraph=...&text=...&trainingID=...&vocabularyGraph=...
```

Service Body (Prototype)

```
<acsrv:AutoClassifyImpl ui:args="*" />
```

For template services, the following values can either be supplied in the HTTP request's Accept header or as the value of the `_format` argument:

Response type	Format name	Arguments accepted
application/sparql-results+xml	SPARQL XML Results	xml
application/sparql-results+json	SPARQL JSON Results	json
application/sparql-results+json-simple	Simple JSON Format	json-simple
text/csv	Comma separated values	csv

text/tab-separated-values	Tab separated values	tsv
application/rdf+xml	RDF/XML	application/rdf+xml
text/turtle	Turtle	text/turtle

If you do not explicitly specify the response type, defaults will be used. The default serialization of SELECT result sets is JSON, following the SPARQL protocol. When you call a service from a web browser's address bar, the mime type will typically be requested as something that includes the string "xml", and in this case it will produce SPARQL XML Results. CONSTRUCT queries are returned in Turtle format by default.

The value of the `_base` argument can be a full graph URI or a short graph name. To define short graph names, go to the Server Administration -> Server Configuration Parameters page. In the section named "URI Parameters" define an instance of "Short graph names".

To call a saved SPARQL query outside the EDG application, navigate to the Export tab -> Export using Saved SPARQL Query and copy the Service URL.

Run all DASH Test Cases

TBL can run unit tests defined as instances of `dash:TestCase`. Typically, such test cases are created with *TopBraid Composer*, which has a menu item for creating new test case files. See <http://datashapes.org/testcases.html> for details.