structWSF: A Framework for Collaboration Networks

by Mike Bergman - Thursday, July 02, 2009

http://www.mkbergman.com/497/structwsf-a-framework-for-collaboration-networks/

An Innovative, Distributed, Scalable Design with Dataset Access Rights

structWSF is a platform-independent Web services framework for accessing and exposing structured RDF data. Its central organizing perspective is that of the *dataset*. These datasets contain instance records, with the structural relationships amongst the data and their attributes and concepts defined via separate ontologies (schema with accompanying vocabularies).

The structWSF middleware framework is generally RESTful in design and is based on HTTP and Web protocols and open standards, conforming to what is known as a Web-oriented architecture. The initial structWSF framework comes packaged with a baseline set of about a dozen Web services in CRUD, browse, search and export and import. All Web services are exposed via APIs and SPARQL endpoints. It also has direct interfaces to the Virtuoso RDF triple store and the Solr faceted, full-text search engine.

This post follows the release of the alpha version of the open source structWSF code on the OpenStructs Web site. It is available for download under Apache 2 license.

But, Wait! There's More!

These baseline capabilities are useful enough. But there is another foundation to structWSF that is quite innovative and exciting: Its explicit design to support collaboration networks. It is this aspect that is the focus of this current article.

The collaboration design is a result of the needs of the Bibliographic Knowledge Network (BibKN or BKN) [1]. BibKN has as one of its express purposes creating a network of collaborators in math and statistics, ranging from the individual researcher to departments and universities and various virtual organizations (VOs) representing different communities of interest. Moreover, this nucleus of researchers also has external collaborators ranging from major publishers to software and service providers of various sizes from around the globe.

Thus, one key requirement of the BKN project was to design an infrastructure responsive to this broad spectrum of interests, locations and organizations. And, besides questions of varying scale, locale and distribution, there was also the need to combine public and private data. In some cases, initial work products need to be kept within its sponsoring groups before being made public. Sometimes external publishers want to segregate network members by whether they are already paid subscribers or not. And, most importantly, the project had a mandate to create an easy and open framework for encouraging incipient collaborators and curators to add and take ownership of new datasets.

Boiled down, these requirements represent a completely fluid spectrum of scales, access rights, virtual groups and distributed locations. These requirements were daunting indeed to establish a workable and responsive framework. But, what has resulted from this mandate -- **structWSF** -- is a generalized solution that has applicability to collaboration within ***any*** knowledge network.

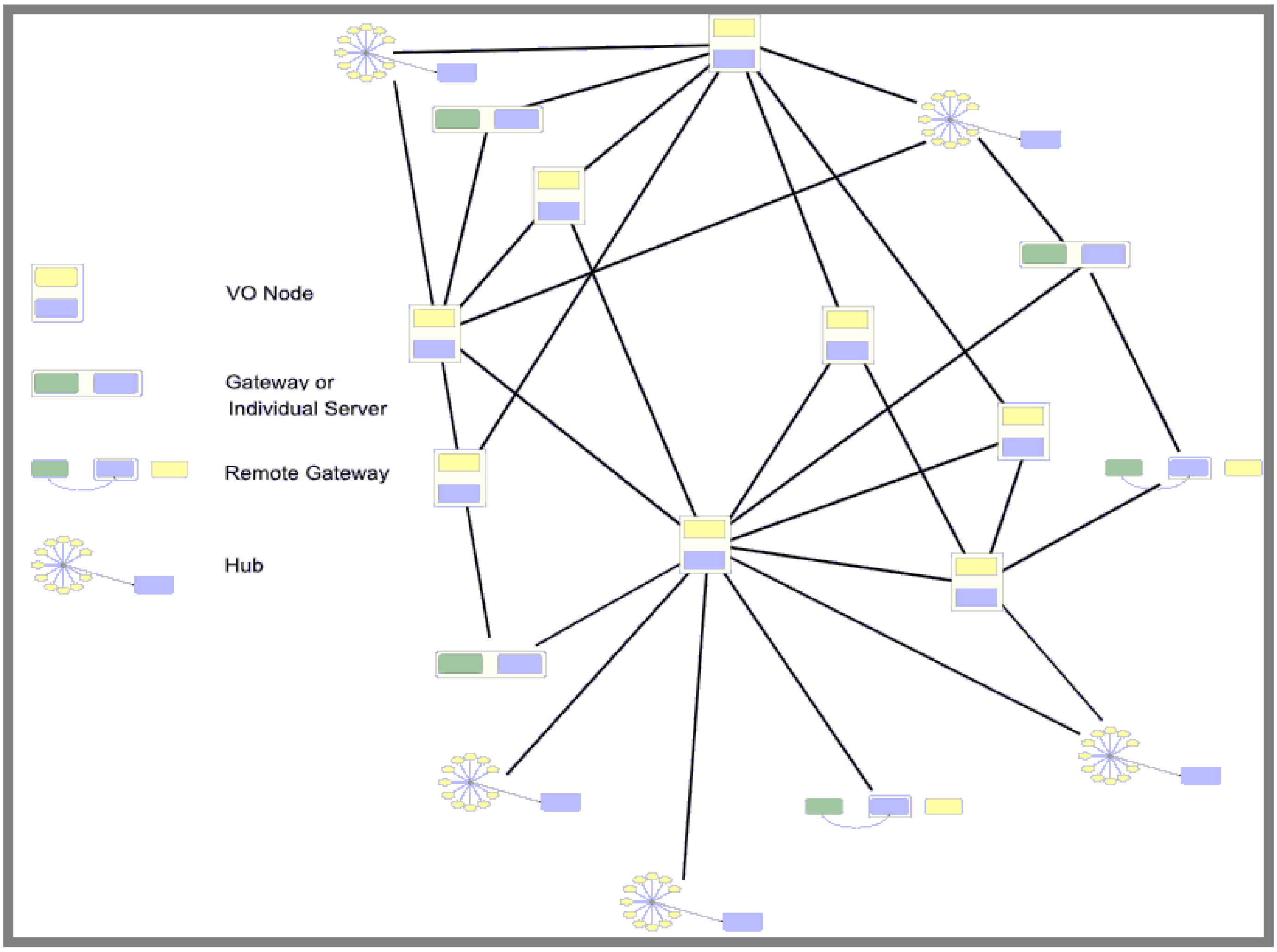
Four Exemplar Deployment Modes

BibKN anticipates and is to include four exemplar types of participants on the network (or "nodes', which are not to be confused with the different meaning of node in Drupal):

- **VO Nodes** -- these "virtual organization" nodes are the main collaboration portals and are generally based on a content management system (CMS) [2]. VO nodes may also be publishers or consumers of datasets
- **Gateways** -- connections to existing external content in the native data formats of the publisher, which are converted and made available to the network in BKN-compliant forms [3]
- **Hubs** -- aggregate suppliers of useful datasets in BKN-compliant data formats, most often BibJSON [4], and
- **Individual** dataset contributors and clients, generally located on a desktop machine.

Each of these nodes exposes its data to the rest of the network via a **structWSF** Web services framework. Each **structWSF** installation provides an access point and endpoint to the network. Through these installations, data is converted to "canonical" form for use by other nodes on the network with common tools and services provided.

In conceptual, form, then, the network can be represented as follows:



Each node has a structWSF instance, the common network denominator, shown in blue.

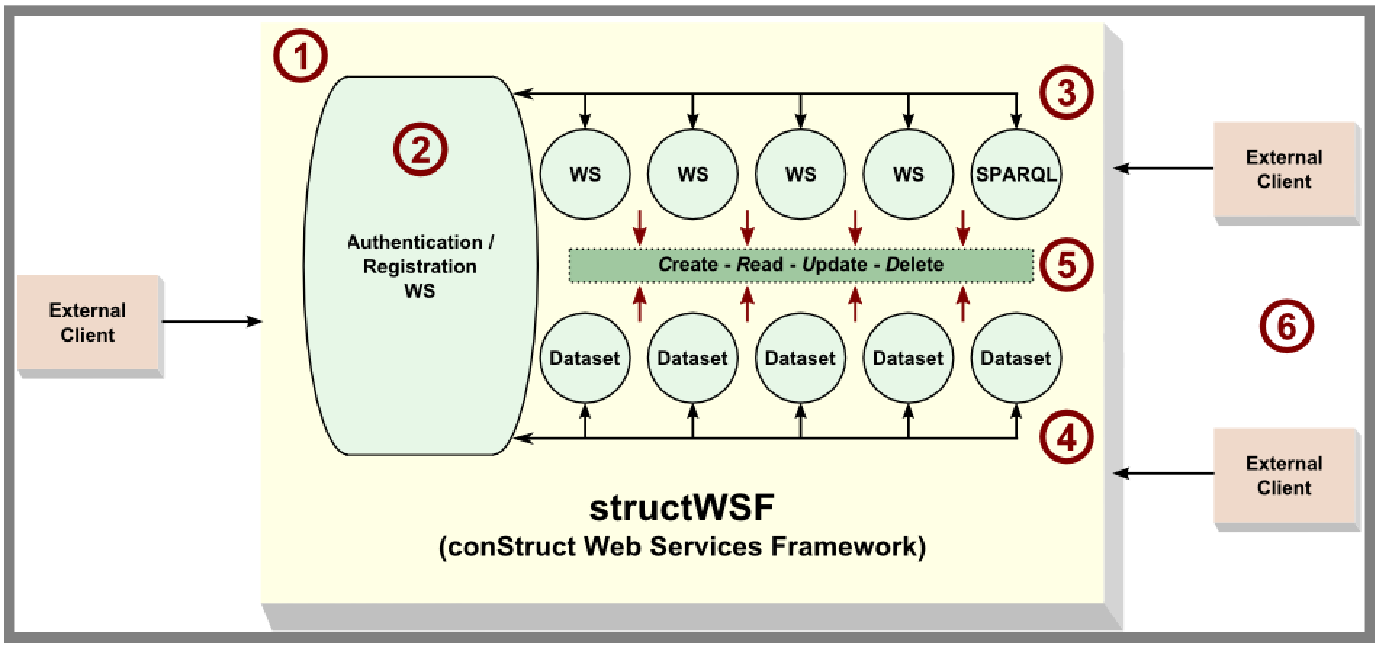
A key aspect of each structWSF installation is dataset registration and access authorization. Only users with proper authorization may access or exercise certain privileges such as write or updates for a given dataset.

The other core Web services provided with structWSF are the CRUD functional services (*create* - *read* - *update* - *delete*), import and export, browse and search, and a basic templating system [see **(3)** in the next figure]. These are viewed as core services for any structured dataset. The current alpha release supports CSV, TSV, RDF/XML, RDF/N3 and XML, with JSON forthcoming shortly. (**UPDATE:** Now provided.)

Rights: The Intersection of Web Service, Dataset, Group, Role and CRUD

The controlling Web service in structWSF is the Authentication/Registration WS [see **(2)** in the figure below]. The current alpha version of structWSF uses registered IP addresses as the basis to grant access and privileges to datasets and functional Web services. Later versions will be expanded to include other authentication methods such as OpenID, keys (*a la* Amazon EC2), foaf+ssl or oauth. A secure channel (HTTPS, SSH) could also be included.

A simple but elegant system guides access and use rights. First, every Web service is characterized as to whether it supports one or more of the CRUD actions. Second, each user is characterized as to whether they first have access rights to a dataset and, if they do, which of the CRUD permissions they have [see **(4, 5)**]. We can thus characterize the access and use protocol simply as A + CRUD.



Thereafter, a mapping of dataset access and CRUD rights (see below) determines whether users see a given dataset and what Web services ("tools") are presented to them and how they might manipulate that data. When expressed in standard user interfaces this leads to a simple contextual display of datasets and tools. For example, under standard search or browse activities the user would only see results sets drawn from the datasets for which they have access. Similarly, users only see the tools that their CRUD rights allow.

At the Web service layer, these access values are part of the GET request. The system, however, is designed to more often be driven by user and group management at the CMS level via a lightweight plug-in or module layer.

Because a CMS may employ its own access system and protocols, the potential combinations can become quite large. Let's take for an example a VO node in the BibKN scenario which layers Drupal (via the **conStruct** modules) over the **structWSF** framework. By including the additional third-party contributed Drupal module of Organic Groups, we also now add an entire dimension of group access to the standard roles access in the base Drupal [5]. So, in this scenario, we theoretically have these potential access and rights combinations:

- By ***dataset***
- By ***Web service*** (tool) and whether that tool can potentially support *create*, *read* (access), *update* or *delete* [*CRUD*] operations
- By ***user role*** (for example, *administrator*, *owner*, *curator*, *contributor*, *unregistered*)
- By ***group*** (for example, *SuperWhizBangs*, *SortOfOKs*, *Clueless*, *RockStars*).

Since the group and user role categories can be quite extensive, the combinatorial result of these options can also be quite large.

Nonetheless, as a general proposition, these access and rights dimensions can capture most any reasonable use case.

Patterned Profiles Aid Management

One way to ease the management of these choices at the UI level is to create a series of access patterns or templates -- called ***profiles*** -- to which a newly registered dataset can be assigned. While the Drupal site owner could go in and change or tweak any of the individual assignments, the use of such profiles simplify the steps needed for the majority of newly registered datasets (Pareto assumption).

For instance, consider these possible profile patterns:

- **Profile: Public** (standard) -- this profile is for a dataset intended for broad public access
- **Profile: Registered** -- this profile is for datasets that are limited to registered users of a VO node (possibly as a way to prevent spam or to encourage membership or participation)
- **Profile: Curated** -- this profile is where a specific group or groups (which themselves can be flexibly determined and assigned) has curation rights for the dataset, or
- **Profile: Internal** -- this profile is for internal (private) datasets where only a specific group or groups may access or modify. In some instances, an internal dataset might be the profile type while the dataset is under development, with the profile shifting to a broader access category once completed.

We can now expand this concept for a given dataset by adding the dimension of user type or category. Four categories of users can illustrate this user dimension:

- **O**

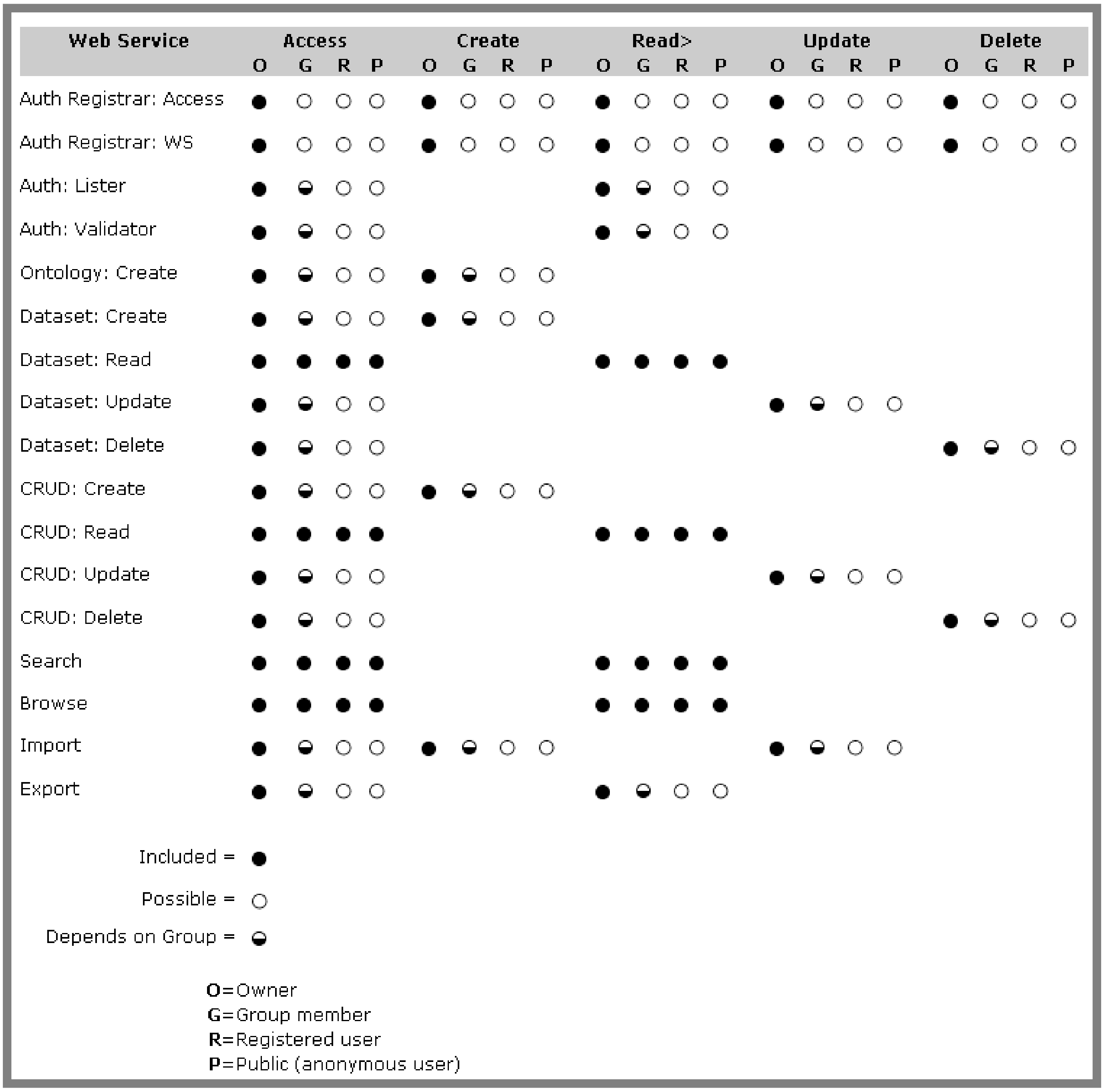
= Owner (the original registrar of the dataset; often possibly the "owner" of the VO node, but not necessarily so)
- **G** = Group member (a registered user who is a member of a specific group)
- **R** = Registered user (an authorized VO node user with a Drupal login and password)
- **P** = Public (anonymous user)

(Of course, with a multitude of groups, there are potentially many more than four categories of users.)

To illustrate how we can collapse this combinatorial space into something more manageable, let's look at what one of the profile cases noted above -- that is the **Public** profile -- can now be expressed as a pattern or template. In this example, the **Public** profile means that owners and some groups may curate the data, but everyone can see and access the data. Also note that export is a special case, which could warrant a sub-profile.

We also need to relate this **Public** profile to a specific dataset. For this dataset, we can characterize our "possible" assignments as described above as to whether a specific user category (**O**, **G**, **R** and **P** as noted above) has available a given function (open dot), gets permission rights to that function by virtue of the assigned profile (solid dot), or whether that function may also be limited to a specific group or groups (half-filled dot) or not.

Thus, we can now see this example profile matrix for the **Public** profile for an example dataset with respect to the available structWSF Web services:



Web Service	Access				Create				Read>				Update				Delete			
	O	G	R	P	O	G	R	P	O	G	R	P	O	G	R	P	O	G	R	P
Auth Registrar: Access	●	○	○	○	●	○	○	○	●	○	○	○	●	○	○	○	●	○	○	○
Auth Registrar: WS	●	○	○	○	●	○	○	○	●	○	○	○	●	○	○	○	●	○	○	○
Auth: Lister	●	◒	○	○					●	◒	○	○								
Auth: Validator	●	◒	○	○					●	◒	○	○								
Ontology: Create	●	◒	○	○	●	◒	○	○												
Dataset: Create	●	◒	○	○	●	◒	○	○												
Dataset: Read	●	●	●	●					●	●	●	●								
Dataset: Update	●	◒	○	○									●	◒	○	○				
Dataset: Delete	●	◒	○	○													●	◒	○	○
CRUD: Create	●	◒	○	○	●	◒	○	○												
CRUD: Read	●	●	●	●					●	●	●	●								
CRUD: Update	●	◒	○	○									●	◒	○	○				
CRUD: Delete	●	◒	○	○													●	◒	○	○
Search	●	●	●	●					●	●	●	●								
Browse	●	●	●	●					●	●	●	●								
Import	●	◒	○	○	●	◒	○	○					●	◒	○	○				
Export	●	◒	○	○					●	◒	○	○								

Included = ●

Possible = ○

Depends on Group = ◒

O=Owner
G=Group member
R=Registered user
P=Public (anonymous user)

Note, of course, that these options and categories and assignments are purely arbitrary for our illustrative discussion. Your own needs and circumstances may vary wildly from this example.

Matrices such as this seem complex, but that is why profiles can collapse and simplify the potential assignments into a manageable number of discrete options. The relevant question, with a quick answer, is for you to assemble profiles responsive to your own specific circumstances.

And, of course, if your pre-packaged profiles need to be tweaked or adjusted for a particular circumstance, the CMS enables all assignments to be accessed in individual detail.

A Powerful Vision

Via this design, knowledge and collaboration networks can be deployed that support an unlimited number of configurations and options, all in a scalable, Web-accessible manner. The data that is accessed is automatically expressed as linked data. This same framework can be layered over *in situ* existing data assets to provide data federation and interoperable functionality, all responsive to standard enterprise concerns regarding data access, rights and permissions.

This is not science fiction, and this is not complex. When combined with its data mixing and conversion potentials [3], we can now see emerging a general framework that enables access and interoperability to virtually any data source and for virtually any purpose, with permissions and rights built in, anywhere and everywhere across the Web.

These are exciting prospects that were not possible until Web-oriented architectures with structured RDF data came to the fore. There are no longer any barriers to the powerful vision of complete data access and interoperability without disrupting existing assets.

And the mere thought of that, is, disruptive, indeed.

Note: The alpha version of **structWSF** and its related **conStruct** modules are somewhat raw or incomplete in some ways. A few of the functions expressed in this posting have not yet been released in these code bases.

---

[1] BibKN is a project to develop a suite of tools and services to encourage formation of virtual organizations in scientific communities of various types. The project started in September 2008 with funding by the NSF Cyber-enabled Discovery and Innovation (CDI) Program. The major participating organizations are the American Institute of Mathematics (AIM), Harvard University, Stanford University and the University of California, Berkeley. Research support to BibKN has come in part from NSF Award 0835851.

[2] **structWSF** is actually combined with the **conStruct** structured content system and Drupal for the delivery of the VO nodes.

[3] See the earlier posting on, structWSF: A Framework for Data Mixing, for discussion about **structWSF** data formats.

[4] BibJSON is the standard, human-readable and editable data exchange format used within the BKN project. It has a standard attribute vocabulary geared to bibliographic material and is based on the JSON (JavaScript Object Notation) data notation.

[5] Though the specifics may differ, including the modules and add-ins, other leading CMS systems provide similar functionality.