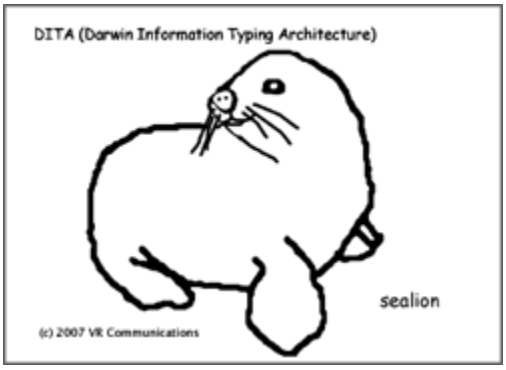
Bulkpub Module for Drupal 7



Richard H. (Dick) Johnson - Fifth Edition - 2014

Contents

Legal, Copyright iii

Chapter 1. Introduction 4

Chapter 2. Installing, configuring and using bulkpub 6

Chapter 3. bulkpub module APIs 8

addVocabularyTerm 8

contentTypeAllowed 9

delAllMenuLinks 10

deleteImage 10

deletePage 11

deletePages 12

editPage 13

getContentTypes 14

getFieldInfo 15

getNodePath 16

getVocabularies 16

getPage 17

newBook 18

newImage 19

newMenuLink 20

newPage 21

pruneVocabulary 22

updateOutline 23

Chapter 4. bulkpub module API data array reference 25

Index ..... a

Legal, Copyright

Keyword tags: Anna van Raaphorst, bulkpub module, Drupal, Richard Johnson, VR Communications, Inc.

Product, copyright, licensing

The **bulkpub module** (written by Richard H. (Dick) Johnson) is a web service that implements APIs for remote publishing on Drupal 7.

Website instantiations of the solution provide powerful, flexible, and highly extensible environments that can make use of a wide variety of applications, tools, platforms, content, and data.

Downloading the bulkpub module

Bulkpub is available for download at available on the Drupal.org website: bulkpub module project.

Fourth Edition, February 2013

The fourth edition of this documentation was published with release 1.7 of DITA Open Toolkit and supports the DITA 1.2 language standard.

Chapter 1. Introduction

Keyword tags: bulkpub module, introduction, VR Communications Web Portal Integration Scripts for the DITA/Drupal Environment, VR Communications web portal solution

Solution examples

Sample usage scenarios include the following:

- A commercial software or hardware organization that posts on the same website both structured product documentation produced by in-house technical writers along with crowdsourced product information created informally using the site's wiki authoring features
- An author posting the final review draft of her latest book on the same professional website that also includes her frequent blog posts
- A genealogist creating a family history website that includes genealogy data in GEDCOM format with blogs, trip journals, photo galleries, and videos contributed by other members of the family

Content participating in the solution

Solution content includes both files created externally to the CMS and content created internally using native CMS functionality.

The solution ensures that *both* natively produced and externally introduced content can take equal advantage of CMS platform functionality (for example, commenting, RSS feeds, search, views, and workflow).

External files would typically have these characteristics:

- Be produced by a single or limited number of authors
- Be published externally to multiple output targets (for example, PDF, ePub, and XHTML)
- Have a hierarchical structure
- Reference and link to one another
- Contain tabular material, examples, and images
- Contain metadata derived from keywords in the source files

Examples of such file formats might include:

- XHTML output produced from DITA source by DITA Open Toolkit or a similar application
- GEDCOM genealogy data rendered to XHTML
- ePub electronic book files

Internal files would typically be more informal in nature and be produced by a large number of authors. Example would be blog posts, announcements, slideshows, and videos.

Solution demonstration site

The following model website demonstrates several possible implementations of the solution:

- GEDCOM/Drupal solution: NewsFromNan.com

Implementation technologies

The key technical features used to produce the web portal solutions include the following:

- **A set of scripts** that drives the remote publishing process of the external content while preserving original relationships, links, media, and metadata.

  The Web Portal Integration Scripts (written by Richard H. (Dick) Johnson) drive the content integration process.

- **A web service** that implements APIs for remote publishing on a CMS platform.

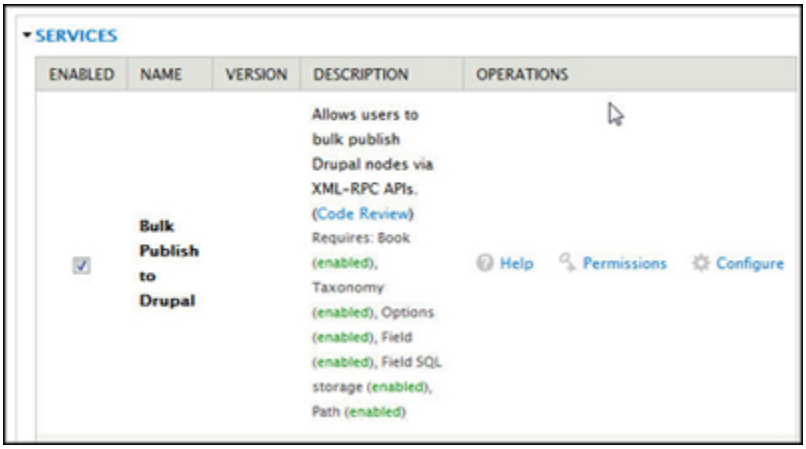
  The web service for the DITA/Drupal environment is the bulkpub module, which is the subject of this documentation. The bulkpub module is a general-purpose service that can be used with other integration scripts.>

Chapter 2. Installing, configuring and using bulkpub

Keyword tags: configuring, installing, usage

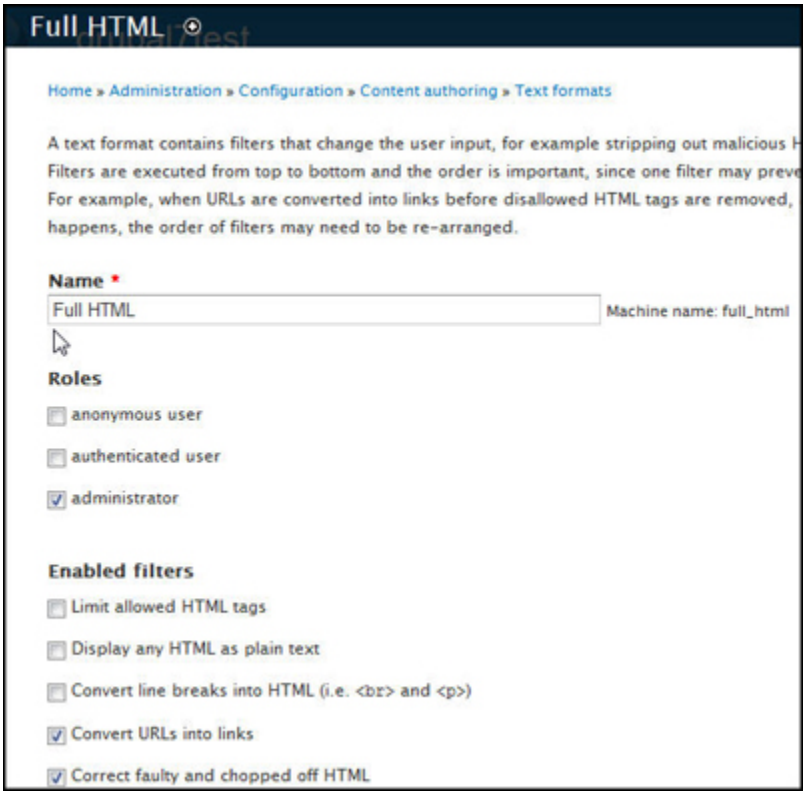
The bulkpub module is a Drupal 7 contributed module. Install the module files in `sites/all/modules/bulkpub`.

1. Enable the module at **Home > Administration > Modules**. The module will be found under the **Services** tab. Click the **ENABLED** checkbox for the module and then **Save configuration**.



2. Configure the module at **Home > Administration > Configuration > Web services** and select which content types can be used with the module and also the details of image filetypes and size limits.
3. Set the permissions for bulkpub at **Home > Administration > Modules**. The module will be found under the **Services** tab. Clink **Permissions** and set the permission for **Bulk Publish Drupal pages**.

4. Set the location for uploaded image files at **Home > Administration > Configuration > File System**.
5. If your uploaded files will contain image links, add <img> to the tags allowed in filteredHTML format or make full HTML the default format at **Home > Administration > Configuration > Text formats**.
6. If the content to be uploaded is XHTML format, you may want to disable filtering done by Drupal at **Home > Administration > Configuration > Content authoring > Text formats**.



7. Create any content types you plan to use at **Home > Administration > Structure > Content types**.
8. Create any Taxonomy vocabularies you will be using with content types at **Home > Administration > Structure > Taxonomy**.
9. If your content types will be in Book outlines, configure this at **Home > Administration > Structure > Content > Books**.

Chapter 3. bulkpub module APIs

Keyword tags: API (Application Programming Interface), XML-RPC (XML Remote Procedure Call)

The bulkpub module APIs are invoked via the XML-RPC protocol. Clients using this protocol open a connection to a site by creating a proxy object.

The following Python example shows how to do this:

```
siterpc = ""http://192.168.1.85/drupal/xmlrpc.php"

proxy = xmlrpc.client.ServerProxy(siterpc, allow_none=True)
```

Once communication with a remote site has been established, it is possible to invoke remote procedure calls to the site. The topics in this section describe the API calls that bulkpub supports.

addVocabularyTerm

Keyword tags: addVocabularyTerm, API (Application Programming Interface), XML-RPC (XML Remote Procedure Call)

Description

Add a term to a vocabulary.

```
proxy.bulkpub.addVocabularyTerm(user,password,vocab,term)
```

Parameters

string user

The site userid to be used

string password

The site password for the userid

string vocab

Name of the vocabulary to be updated

string term

Vocabulary term

Return values

Returns TRUE, or an error.

Python example

```
vocab = "tags"

term = "new term"

ret=proxy.bulkpub.addVocabularyTerm(user,password,vocab,term)
```

contentTypeAllowed

Keyword tags: API (Application Programming Interface), getContentTypes, XML-RPC (XML Remote Procedure Call)

Description

Check if content type can be in a book.

```
proxy.bulkpub.contentTypeAllowed(user,password,ctype)
```

Parameters

string user

The site userid to be used

string password

The site password for the userid

string ctype

Content type name

Return values

Returns TRUE if the type can be in a book, or FALSE, or an error.

Python example

```
ctype = "book"
```

```
    ret=proxy.bulkpub.contentTypeAllowed(user,password,ctype)
```

delAllMenuLinks

Keyword tags: API (Application Programming Interface), delAllMenuLinks, XML-RPC (XML Remote Procedure Call)

Description

Delete all links from a menu.

```
proxy.bulkpub.delAllMenuLinks(menuname,user,password)
```

Parameters

string menuname

A menu name

string user

The site userid to be used

string password

The site password for the userid

Return values

Returns TRUE, or an error.

Python example

```
mret = proxy.bulkpub.delAllMenuLinks(menuname, user, password)
```

deleteImage

Keyword tags: API (Application Programming Interface), deleteImage, XML-RPC (XML Remote Procedure Call)

Description

Delete an image file.

```
proxy.bulkpub.deleteImage(appid,user,password,fileurl)
```

Note:
This API assumes the image is an **unmanaged** file. It should not be used to delete managed image files that are described in the `file_managed` table.

Parameters

string appid

This parameter is currently ignored

string user

The site userid to be used

string password

The site password for the userid

string fileurl

Site url for the image to be deleted

Return values

Returns a struct with information about the image file created, or an error.

Python example

```
ret=proxy.bulkpub.deleteImage(user,password,fileurl)
```

deletePage

Keyword tags: API (Application Programming Interface), deletePage, XML-RPC (XML Remote Procedure Call)

Description

Delete a Drupal node.

```
proxy.bulkpub.deletePage(id,user,password,publish)
```

Parameters

string id

The node id

string user

The site userid to be used

string password

The site password for the userid

boolean publish

This parameter is currently ignored

Return values

Returns TRUE if the delete operation succeeded, or an error.

Python example

```
id = "23"

content = proxy.bulkpub.deletePage(id,user,password,True)
```

deletePages

Keyword tags: API (Application Programming Interface), deletePages, XML-RPC (XML Remote Procedure Call)

Description

Delete all Drupal nodes of a particular content type.

```
proxy.bulkpub.deletePages(ctype,user,password)
```

Parameters

string ctype

The content type name

string user

The site userid to be used

string password

The site password for the userid

Return values

Returns number of nodes deleted, or an error.

Python example

```
ctype = "mytype"
count = proxy.bulkpub.deletePages(ctype,user,password)
```

editPage

Keyword tags: API (Application Programming Interface), editPage, XML-RPC (XML Remote Procedure Call)

Description

Update an existing node with revised content.

```
proxy.bulkpub.editPage(id,user,password,data,publish)
```

Parameters

string id

The id of the node to be updated

string user

The site userid to be used

string password

The site password for the userid

struct data

An associative array containing the node content

boolean publish

A boolean value that determines whether the node should be published immediately

Return values

Returns the new node id as a string, or an error.

Python example

```
id = "23"
data = {"title":"my new node title"}
data["description"] = "hello world again"
data["alias"] = "hello/world"
publish = True
ret = proxy.bulkpub.editPage(id,user,password,data,publish)
```

getContentTypes

Keyword tags: API (Application Programming Interface), getContentTypes, XML-RPC (XML Remote Procedure Call)

Description

Get a list of site content types.

```
proxy.bulkpub.getContentTypes(user,password)
```

Parameters

string user

The site userid to be used

string password

The site password for the userid

Return values

Returns a list of content types, or an error.

Python example

```
ctype_list=proxy.bulkpub.getContentTypes(user,password)
```

getFieldInfo

Keyword tags: API (Application Programming Interface), getFieldInfo, XML-RPC (XML Remote Procedure Call)

Description

Get a list of fields for a content type.

```
proxy.bulkpub.getFieldInfo(user,password,contenttype)
```

Parameters

string user

The site userid to be used

string password

The site password for the userid

string contenttype

A content type

Return values

Returns an array of fieldname/fieldtype pairs, or an error.

Python example

```
ctype = "article"
ctype_info=proxy.bulkpub.getFieldInfo(user,password,ctype)
```

```
```

getNodePath

Keyword tags: API (Application Programming Interface), getNodePath, XML-RPC (XML Remote Procedure Call)

Description

Get the system node path for an alias.

```
proxy.bulkpub.getNodePath(alias,user,password)
```

Parameters

string alias

A node alias

string user

The site userid to be used

string password

The site password for the userid

Return values

Returns a system node path, or an error.

Python example

```


testalias = "testpage"


nodepath = proxy.bulkpub.getNodePath(testalias,user,password)


```

getVocabularies

Keyword tags: API (Application Programming Interface), getVocabularies, XML-RPC (XML Remote Procedure Call)

Description

Get a list of vocabulary names for the site.

```
proxy.bulkpub.getVocabularies(user,password)
```

Parameters

string user

The site userid to be used

string password

The site password for the userid

Return values

Returns a list of vocabulary names, or an error.

Python example

```
vocab_list=proxy.bulkpub.getVocabularies(user,password)
```

getPage

Keyword tags: API (Application Programming Interface), getPage, XML-RPC (XML Remote Procedure Call)

Description

Return the content of a Drupal node.

```
proxy.bulkpub.getPage(id,user,password)
```

Parameters

string id

The node id

string user

The site userid to be used

string password

The site password for the userid

Return values

Returns the node content as a struct, or an error.

Python example

```
id = "23"

data = proxy.bulkpub.getPage(id,user,password)
```

newBook

Keyword tags: API (Application Programming Interface), newBook, XML-RPC (XML Remote Procedure Call)

Description

Create a new book outline.

```
proxy.bulkpub.newBook(id,user,password)
```

Parameters

string id

First node id in the book

string user

The site userid to be used

string password

The site password for the userid

Return values

Updated node struct, or an error.

Python example

```
```

```
id = "123"

node=proxy.bulkpub.newBook(id,user,password)
```

newImage

Keyword tags: API (Application Programming Interface), newImage, XML-RPC (XML Remote Procedure Call)

Description

Create a new image file.

```
proxy.bulkpub.newImage(blogid,user,password,data)
```

Note:

The image is stored as an **unmanaged** Drupal image. This API is to be used to store images referenced in pages in a docset.

Parameters

string blogid

This parameter is currently ignored

string user

The site userid to be used

string password

The site password for the userid

struct data

An associative array containing encoded binary text for the image

Return values

Returns a struct with information about the image file created, or an error.

Python example

```

```

```
    # read in the file
    rawbits = fd.read()
    # convert it from binary to encoded text
    data['bits'] = xmlrpc.client.Binary(rawbits)
    # store the image
    ret = proxy.bulkpub.newImage("x",user,password,data)
    # return the image url
    return ret['url']
```

newMenuLink

Keyword tags: API (Application Programming Interface), newMenuLink, XML-RPC (XML Remote Procedure Call)

Description

Add a node menu link to a menu.

```
proxy.bulkpub.newMenuLink(ctype,user,password,data)
```

Parameters

string ctype

The content type to be used for the new node

string user

The site userid to be used

string password

The site password for the userid

struct data

An associative array containing menu link information

Return values

Returns the new menu link id as a string, or an error.

Python example

```
```

```
menuarg = {"name":menuname, "title":"testing"}

menuarg['node_path'] = nodepath

menuarg['description'] = "menu description"


nmret = proxy.bulkpub.newMenuLink(content_type,user,password,menuarg)
```

newPage

Keyword tags: API (Application Programming Interface), newPage, XML-RPC (XML Remote Procedure Call)

Description

Create a new node of a specific content type.

```
proxy.bulkpub.newPage(ctype,user,password,data,publish)
```

Parameters

string ctype

The content type to be used for the new node

string user

The site userid to be used

string password

The site password for the userid

struct data

An associative array containing the content to be published

boolean publish

A boolean value that determines whether the node should be published immediately

Return values

Returns the new node id as a string, or an error.

Python example

```
ctype = "page"
data = {"title":"my node title"}
data["description"] = "hello world"
data['terms'] = ["hello","world"]
data['vocabulary'] = "tags"
data['fields'] = {"birthmonth" : "Jan"}
publish = True
ret = proxy.bulkpub.newPage(ctype,user,password,data,publish)
```

pruneVocabulary

Keyword tags: API (Application Programming Interface), pruneVocabulary, XML-RPC (XML Remote Procedure Call)

Description

Prune orphan terms from a vocabulary. Orphans are terms that are not referenced by a by content node.

```
proxy.bulkpub.pruneVocabulary(user,password,vocab,switch)
```

Parameters

string user

The site userid to be used

string password

The site password for the userid

string vocab

Name of the vocabulary to be pruned

int switch

If switch>0, prune orphan terms, otherwise do not

Return values

Returns a list of orphan terms in the specified vocabulary, or an error

Python example

```
# get a list of what will be pruned
terms = proxy.bulkpub.pruneVocabulary(user,password,vocab,0)
norphan = len(terms)
for t in terms:
    print(t)
print(vocab,"has",norphan,"orphan terms")

# finally, prune orphan terms
if norphan>0:
print("pruning terms")
terms2 = proxy.bulkpub.pruneVocabulary(user,password,vocab,1)
```

updateOutline

Keyword tags: API (Application Programming Interface), updateOutline, XML-RPC (XML Remote Procedure Call)

Description

Update a book outline by linking a node to a parent node in a book outline

```
proxy.bulkpub.updateOutline(user,password,id,pid,weight)
```

Parameters

string user

The site userid to be used

string password

The site password for the userid

string id

node id

string pid

parent node id of a node in a book outline

int weight

weight of this parent-child link in the book outline

Return values

TRUE if the outline was successfully updated, or an error.

Python example

```
pid = "10"

id = "123"

weight = 1

ret=proxy.bulkpub.updateOutline(user,password,id,pid,weight)
```

Chapter 4. bulkpub module API data array reference

Keyword tags: array element reference

Some of bulkpub module APIs include a data structure as an input or output argument. The following table shows which structure key/value pairs are used as inputs (I) or outputs(O) for the newPage, editPage and getPage APIs.

key name	key description	newPage	editPage	getPage
format	full_html or filtered_html	I	I	O
title	node title	I	I	O
description	node body text	I	I	O
fields	field name/value pairs	I	I	O
vocabulary	vocabulary name for terms	I	I	O
terms	a list of terms	I	I	O
userid	userid who created node	O	O	O
postid	node number	O	O	O
status	published status as boolean			O
type	content type			O
link	node URL			O
comments	array of comments			O
dateCreated	date node created			O
alias	node alias	I	I	

Index

A

addVocabularyTerm
8

APIs (Application Programming Interfaces)
8

array element reference
25

B

bulkpub module
iii, 4

C

configuring bulkpub
6

D

delAllMenuLinks
10

deleteImage
10

deletePage
11

deletePages
12

DITA/Drupal integration scripts
4

Drupal
iii

E

editPage
13

G

getContentTypes
9, 14

getFieldInfo
15

getNodePath
16

getPage
17

getVocabularies
16

I

installing bulkpub
6

integration scripts
4

introduction
4

J

Johnson, Richard
iii

N

newBook
18

newImage
19

newMenuLink
20

newPage
21

P

pruneVocabulary
22

U

updateOutline
23

using bulkpub
6

V

van Raaphorst, Anna
iii

VR Communications Web Portal Integration Scripts for the DITA/Drupal Environment 4
VR Communications web portal solution 4
VR Communications, Inc. iii

W

web portal integration scripts 4
web portal solution 4

X

XML-RPC (XML Remote Procedure Call) 8