X-BSCW: An XML-RPC API to BSCW

Updated to BSCW version 5.0

Fraunhofer FIT and OrbiTeam Software

Bonn, Germany
May, 2012

Contents

X-BSCW Application Programming Interface: What for?

X-BSCW, the Application Programming Interface (API) to the BSCW system, is meant as an interface for application programmers. It is **not a development environment** of BSCW. Because BSCW is implemented in Python, http://python.org/, it can quite easily be modified or enhanced by directly changing BSCW's source code (provided some internal knowledge of BSCW, of course :-). BSCW developers, please ask the distributor of BSCW, OrbiTeam Ltd., http://www.orbiteam.de/, for BSCW development information and BSCW source code.

Via the X-BSCW API, a BSCW server offers most of the end-user actions as via the HTML interface which normally is presented in a web browser. The API may be used to offer of BSCW services by other end-user programs or in batch scripts, or to customize BSCW and strip down BSCW's functionality. It is the purpose of X-BSCW to provide access to the core features of BSCW to programs and to people not willing to use the HTML interface of BSCW; no internal knowledge of BSCW is required. It is also a well-defined interface to other clients of a BSCW server: various internal and external projects, mobile devices, third-party services etc.

It should always be kept in mind that X-BSCW is just an alternative way to access BSCW with basically the same functionality and the same restrictions as the normal HTML user interface.

Why XML-RPC?

The API to BSCW is intended to be simple, easy to understand and to use, platform independent, Internetbased, public domain. That's why we decided to use XML-RPC as the base protocol for our API.

XML-RPC, http://www.xmlrpc.com/, is a wide-spread de facto standard which has been stable for many years now. It is a very simple, but flexible RPC protocol which uses XML to encode the data units. There are numerous open source XML-RPC libraries of good quality for virtually all platforms and all major programming languages, including Python and Java, which makes XML-RPC ideal for the purpose of a simple, easy-to-use and ubiquitous API to BSCW.

XML-RPC also is the starting point of the SOAP protocol (Simple Object Access Protocol, http://www.w3.org/standards/techs/soap) which for simple tasks seems to be some overkill. However, there also is a SOAP-based Web Services API to BSCW, see BSCW package “soap” in your BSCW installation directory.

X-BSCW: XML-RPC API to BSCW

Features

X-BSCW provides (almost) all operations which a registered user may invoke in BSCW's normal Web user interface. The obvious exceptions are operations which refer to the HTML interface itself, e.g. hiding/showing columns, folding out of description texts, manipulating fonts or colors. Otherwise, all operations which are feasible at the HTML interface of BSCW are available in XBSCW as well. To be aware of what is feasible in X-BSCW, it is always a good idea to open a Web browser, log into BSCW and have a look at what operations are available. This may change according the logged-on user's particular data and her access rights. See section on reference documentation for a full list of operations.

Architecture

The BSCW Shared Workspace system is an extension of a standard Web server through the server CGI Application Programming Interface. A *BSCW server* (Web server with the BSCW extension) manages a number of shared workspaces, i.e. repositories for shared information, accessible to members of a group using a simple user name and password scheme. In general a BSCW server will manage workspaces for different groups, and users may be members of several workspaces (e.g. one workspace corresponding to each project a user is involved in).

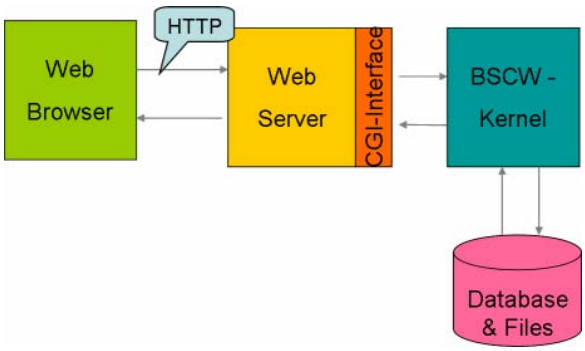

Figure 1: Web access to BSCW server

BSCW's functionality can be accessed either via an ordinary Web browser or via the XML-RPC API using an XML-RPC client. The XML-RPC interface basically provides the same functions as the human-operable Web interface. Where the latter returns calls by rendering HTML output which is eventually displayed by a Web browser, the first returns with an XML-RPC response. Both the normal HTTP/HTML and the XML-RPC interface to BSCW are accessible via the same port (usually port 80).

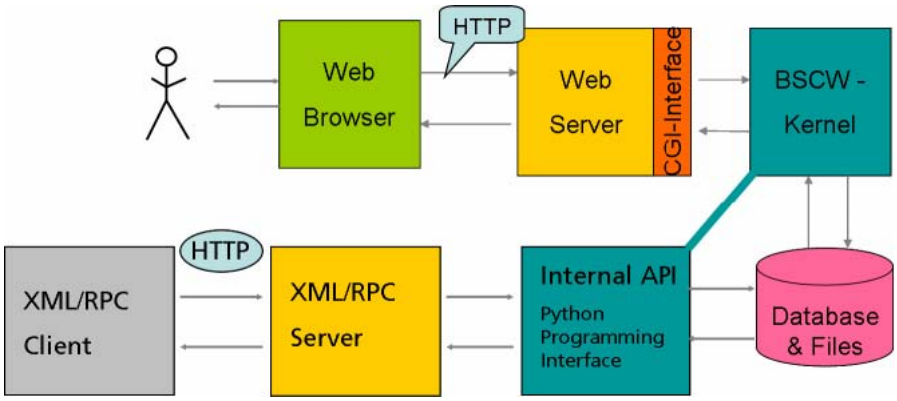

Figure 2: Access to BSCW via XML-RPC and via the normal Web interface

Authentication

It should be noted that functions invoked via XML-RPC are access controlled: XML-RPC clients have to provide authentication as fully registered BSCW user (user name and password) and operations will be subject to access control in BSCW. Some XML-RPC libraries, however, do not provide basic authentication. For most of them, simple hacks to send the HTTP authentication header should be no problem.

BSCW Actions and Their Corresponding XML-RPC Methods

Most kernel actions (aka operations) in BSCW as of date of this document are listed below. The code is an excerpt of the configuration module `bs_action_config.py` which defines all actions for a BSCW server. It lists all actions available in BSCW including the access view and all object classes for which the action is defined. For example:

```
Action('adddocument', view_change, 'Home', 'Bag', 'Folder')
```

means that the Action named *adddocument* is contained in access view *view_change* and is defined for object classes “Home”, “Bag”, “Folder”. Therefore, in principle *adddocument* can be executed on a user’s *Home* area, in her *Bag* (the user’s clipboard in BSCW) or on any *Folder*. Whether in fact adddocument may be executed depends on the concrete access rights of the object (i.e. the Home, Bag or Folder to which a document shall be uploaded): the user whose credentials are used to upload the document must have the right to *adddocument* – if in doubt, have a look at the *Folder’s* info page.

In general, each BSCW action below corresponds to an X-BSCW method.

Although obvious user interface manipulation actions have been removed here, still not all actions in the list below need to be available in the X-BSCW API. Other methods again may not have a corresponding BSCW action, e.g. get_attributes or get_access_rights. Consult http://www.bscw.de/api/ for an up-to-date list of X-BSCW methods.

```
###########################################################################
# All BSCW actions must be defined here.
# Action(name, view, Classes ...)
#
###########################################################################


Action('addbmfolder', None, 'Bookmarks', 'BookmarkFolder')
Action('addct', view_change, 'Contact', 'ContactList', 'AddressBook')
Action('addctlist', view_change, 'Home', 'Folder', 'AddressBook')
Action('ctpref', view_change, 'ContactList', 'AddressBook')
Action('adddocument', view_change, 'Home', 'Bag', 'Folder')
def _chk_addtmpdocument(obj, act=None):
    from bscw.core.cl_request import theRequest as request
    return obj is request.user.waste
Action('addtmpdocument', None)
addtmpdocument.set_function(_chk_addtmpdocument)
Action('addfolder', view_change, 'Home', 'Bag', 'Folder', 'Communities')
Action('addmember', view_share, 'WSGroup', 'Folder', 'Notes', 'UserGroup',
    'Contact')
Action('addaddress', None, 'AddressBook', 'ContactList')
Action('addAddr', None, 'User', 'MailAddress', 'Contact', 'ContactList')
Action('addrole', view_share_ext, 'Folder', 'Document', 'URL',
    'Notes', 'Appointment')
Action('addtempl', view_change, 'Home', 'Bag', 'Folder')
Action('addtempls', view_change, 'Home', 'Bag', 'Folder')
Action('editrole', view_share_ext, 'Folder', 'Document',
    'URL', 'Notes', 'Appointment', 'UserGroup')
Action('chrole', view_share_ext, 'Folder', 'Document',
    'URL', 'Notes', 'Appointment')
Action('freeze', view_creator | view_share_ext, 'Note', 'Document')
Action('unfreeze', view_creator | view_share_ext, 'Note', 'Document')
Action('addurl', view_change, 'Home', 'Bag', 'Bookmarks',
    'BookmarkFolder', 'Folder')
Action('archive', None)
Action('change_pwd', view_user, 'User')

Action('charset', view_change)
Action('chbanner', view_change, 'Folder', 'ContactList', 'Search',
    'WSCalendar')
Action('checkout', view_change, 'Document')
Action('checkin', None)
Action('chtags_EditTags', view_change, 'Folder', 'RssFolder', 'BookmarkFolder',
    'Contact', 'Document', 'URL', 'User', 'Note', 'BlogNote', 'Notes', 'Search',
    'Appointment', 'WSCalendar', 'DocSet', 'ContactList')
chtags_EditTags.opcode = 'chtags.EditTags'
Action('chtags_OnOtherUser', None, 'User')
chtags_OnOtherUser.opcode = 'chtags.EditTags&act=chtags_OnOtherUser'
```

```
Action('steallock', None)
Action('viewlock', None)
Action('chtype', view_change, 'Document')
Action('contents', None)
Action('convert', None)
Action('copy', view_get, 'BookmarkFolder', 'Folder', 'RssFolder', 'Contact',
    'ContactList', 'Document', 'URL', 'Note', 'Notes', 'Search', 'Appointment')
Action('cut', view_edit, 'BookmarkFolder', 'Folder', 'RssFolder', 'Contact',
    'ContactList', 'Document', 'URL', 'Note', 'Notes', 'Search', 'Appointment')
Action('deleteMem', view_share, 'WSGroup', 'AddressBook', 'UserGroup')
Action('deleteObj', view_change_ext,
    'Home', 'Bag', 'Bookmarks', 'BookmarkFolder', 'Folder', 'Contact',
    'ContactList', 'Notes', 'Noteboard', 'LockBag', 'UserCalendar',
    'TaskList', 'WSCalendar', 'AddressBook')
Action('deleteToken', view_share_ext, 'Document')
Action('destroy', None)
Action('edit', view_change, 'Document')
Action('edit_prefs', view_user, 'User')
Action('export', view_get, 'Folder', 'Document')
Action('editdescription', view_change, 'BookmarkFolder', 'Folder', 'RssFolder',
   'Contact', 'ContactList', 'Document', 'URL', 'Notes', 'Search',
   'WSCalendar')
Action('editdetails', view_user, 'User')
Action('editmetadata', view_change, 'Folder', 'RssFolder', 'BookmarkFolder', 'Contact', 'Document',
'URL', 'Appointment', 'ContactList')
Action('metaexp', view_get, 'ALL')
Action('editurl', view_change, 'URL')
Action('extract', None)
Action('get', view_get, 'ALL')
Action('info', view_get_ext, 'ALL')
Action('invite', view_user, 'MailAddress')
Action('lastlog', info, 'User')
Action('mail_to', view_user, 'User', 'MailAddress', 'WSGroup',
    'Contact', 'UserGroup')
Action('link', view_share_ext, 'Folder', 'Contact', 'ContactList',
    'Document', 'URL', 'Notes', 'Search', 'WSCalendar', 'Appointment')
Action('mailaccess', view_share_ext, 'Folder')
Action('pubaccess', view_share_ext, 'Folder', 'Notes', 'Appointment',
       'ContactList', 'WSCalendar')
Action('extpubaccess', None)
extpubaccess.set_function(_no)
Action('publish', view_share, 'Folder', 'Document', 'URL', 'Note',
    'Notes', 'WSCalendar')
Action('chgrpkey', view_share_ext, 'Folder', 'WSGroup')
chgrpkey.set_function(_chk)
Action('rate', view_change, 'RssFolder', 'Document', 'URL', 'Note')
Action('rename', view_change, 'BookmarkFolder', 'Folder', 'RssFolder',
    'ContactList', 'Document', 'URL', 'Notes', 'Search', 'WSCalendar',
    'UserGroup')
Action('replace', view_change, 'Document')
Action('resubmit', view_change, 'Document')
Action('undelete', view_waste, 'Waste')
Action('verifyurl', view_change, 'URL')
Action('uploadurl', None, 'URL')
Action('drop', None)
from_cbd = drop
Action('OAuth', info, 'OAuthConsumer')
#RssFolder
Action('rss.addrss', 0, 'Folder', 'Home', 'Bag')
Action('rss.chrss', 0, 'RssFolder')
#Search
Action('addSearch', view_change, 'Folder', 'Home', 'Bag')
Action('saveSearch', None, 'Search')
Action('editQuery', view_change, 'Search')
Action('applyQuery', view_change, 'Search')
Action('moreHits', view_change, 'Search')
```

```
saveurl = copy

#Versioning
Action('firstversion', view_change, 'Document')
Action('chvinfo', view_change, 'Document')
Action('forget', view_change, 'Document')
Action('branch', view_change, 'Document') # KDa: IMHO this should be view_get
Action('revise', view_change, 'Document')
Action('autovers', view_owner, 'Folder', 'Home', 'Document')

Action('put_vcard', view_change, 'ContactList', 'AddressBook')
Action('get_vcard', view_get, 'Contact', 'User',
               'AddressBook', 'ContactList', 'UserGroup', 'WSGroup')

#Notes and Noteboards, Attachments
Action('addnote', view_change, 'Notes', 'Noteboard')
Action('addnotes', view_change, 'Home', 'Bag', 'Folder')
Action('editnote', view_edit, 'Note')
Action('replynote', None)
Action('nextnote', None)
Action('prevnote', None)
Action('attachnote', view_change, 'Contact', 'Document', 'URL', 'Appointment')
Action('attachdoc', view_edit, 'Contact', 'Note','Appointment', 'Attachments')
Action('attachurl', view_edit, 'Contact', 'Note','Appointment', 'Attachments')
Action('cutattachment', view_edit, 'Document', 'URL',  'Note')
Action('delattachment', view_edit, 'Attachments')

#Calendear / Appointment stuff
Action('addcal', view_change, 'Folder')
Action('add2cal', view_change, 'UserCalendar', 'WSCalendar')
Action('putical', view_change, 'UserCalendar', 'WSCalendar')
Action('editapt', view_edit, 'Appointment')
Action('chpart', view_edit, 'Appointment')
Action('chreminder', view_get_ext, 'Appointment')
Action('getical', None, 'UserCalendar', 'WSCalendar', 'Appointment')

Action('send_fax', None)
Action('send_email', None)

Action('bouncedMailAddr', None, 'MailAddress')
Action('pendingMailAddr', None, 'MailAddress')
Action('allocateMailAddr', None, 'MailAddress')
Action('destroyUser', None, 'User', 'MailAddress')
Action('renameUser', None, 'User', 'MailAddress')

# history
Action('history', info, 'ALL')

# Communities

Action('addcommunity', view_share_ext, 'Folder', 'WSGroup')
Action('newcommunity', view_share_ext, 'Home')
Action('join_community', 0, 'UserGroup')
Action('set_subscribe', view_share_ext, 'UserGroup')
Action('mail_managers', None)

Action('subgroup', None)

Action('htmlcompare', None)
Action('textcompare', None)

##############################################################################
# End of Action definitions
##############################################################################
```

Packages in BSCW

Packages are optional code which are part of the BSCW downloaded software, but which may or may not be enabled on your server (ask your system administrator if in doubt). Only if a package is enabled on the BSCW server, the X-BSCW methods of that package are available via XML-RPC. See http://www.bscw.de/api/ for a full list of X-BSCW methods including package methods.

Object Identifiers

BSCW stores and manages objects by unique identifiers which never change during the lifetime of an object and which are never assigned twice. Object identifiers consist of integers of any length and in X-BSCW are encoded as strings, e.g. "1234567890123456789". Alternatively, you may prefix BSCW object identifiers by "bs_", e.g. "bs_1234567890123456789". This is to embed X-BSCW into a multi-service environment, which (thankfully) is out of the scope of this document.

If there is no prefix in calls to X-BSCW, it defaults to "bs_".

String Encoding

In XML-RPC, the data type *string* originally denotes an ASCII string. As this certainly does not suffice for most applications, many XML-RPC servers and clients interpret the *string* data type to also comprise other encodings. Basically the XML-RPC server and its clients have to agree on acceptable encodings. Most XML-RPC libraries use the UTF-8 standard for transfer. In X-BSCW only UTF-8 encoded strings will be accepted by both clients and servers.

XML-RPC Fault Codes

When an X-BSCW call fails, an error will be raised and sent as XML-RPC Fault back to the client. Possible Fault messages are given below (see module `core/bs_xmlrpc_msg.py` in your BSCW installation directory):

```
#
# module bs_xmlrpc_msg.py
#
"""Error messages for the BSCW (XML-RPC) XAPI"""

DEFAULT_HANDLER = 'service'

# method call
call = DEFAULT_HANDLER + '.%s%s\n'

# fault code
fault = 'BSCW XML-RPC fault code: %d\n'

# error messages -- {fault code: error msg}

err_msg = {
      10000: 'No such method: %s',
      10001: 'Bad number of parameters in %s: expected %d, got %d',
      10003: 'Missing or null parameter value in %s: parameter %s',
      10004: 'No handler specified',
      10005: 'No such handler: %s',
      10010: 'Internal error: %s',
      10011: 'Cannot commit request',
```

```
      10101: 'Bad object id: %s',
      10102: 'Bad user id(s): %s',
      10103: 'Object is not %s: %s',
      10104: 'Bad parameter: %s',
      10105: 'Bad meta data: %s',
      10106: 'Bad container id: %s',
      10107: 'Object id (%s) does not refer to BSCW Artifact',

      10200: 'No permission: %s',
      10201: 'Bad user agent: %s',

      15100: 'Bad mail address: %s',
      15101: 'Invalid mail address: %s',
      15102: 'Bad user name: %s',
      15103: 'Bad password: %s',

      19000: 'Unregistered service id: %s',
      19001: 'Service id already registered: %s',
      19002: 'No %s subscription for service id: %s',
}

# mnemonics

nosuchmethod=10000
badparnumber=10001
nohandler=10004
nosuchhandler=10005
internalerror=10010
nullvalue=10003
```

Good to Know

Attributes of Main BSCW Object Classes

The following Python code will return the attributes of a concrete BSCW object (a folder named “API tests”) and then will list the default attribute names of a number of BSCW object classes, though not all. The code can be found in your BSCW installation: `etc/src-aux/api_clients/get_attributes.py` or in our online reference http://www.bscw.de/api/clients/ .

```
#
# module get_attributes.py
#
# return attributes of BSCW object
# return default attributes of a set of BSCW classes
#
USER_PASSWD = 'my_user_name:my_password'
SERVER_URL = 'http://my_server.orbiteam.de/bscw/bscw.cgi'
OBJ_ID = '123456789'

ATTRS = [
    '__class__',
    '__id__',
    'name',
    'type',
    'url_link',
    'duration',
    'ratings'
    'ctime',
```

```
    'presence',
    'idletime',
    'addresses',
    'ac_mb_roles',
    'ac_own_roles',
    'ac_own_role_actions',
    'l_dicts',
    'owner',
    'owners',
    'creator',
    'size',
    'content_size',
    'list_size',
    'is_shared',
    'has_sub_container',
    '__ws_group__',
    'contractors',
    'duration',
    'end',
    'fields',
    'last_requestor',
    'start',
    'status',
    'task_is_active',
]

def main():
    print SERVER_URL
    srv = XMLRPC_Server(SERVER_URL, USER_PASSWD, verbose=0)

    print OBJ_ID
    result = srv.get_attributes(OBJ_ID, ATTRS, 2, True )
    print str(result)
    dump(result)

    classes = [
        'bscw.core.cl_artifact.Artifact',
        'bscw.core.cl_community.Communities',
        'bscw.core.cl_contact.ContactList',
        'bscw.core.cl_contact.AddressBook',
        'bscw.core.cl_contact.Contact',
        'bscw.core.cl_folder.Folder',
        'bscw.core.cl_document.Document',
        'bscw.core.cl_user.User',
        'bscw.core.cl_user.Home',
        'bscw.core.cl_user.Bag',
        'bscw.core.cl_user.Waste',
        'bscw.core.cl_url.URL',
        'bscw.core.cl_note.Notes',
        'bscw.core.cl_note.Note',
        'bscw.core.cl_usergroup.UserGroup',
        'bscw.core.cl_calendar.WSCalendar',
        'bscw.core.cl_calendar.UserCalendar',
        'bscw.core.cl_tasklist.TaskList',
# packages
        'bscw.core.blog.cl_blog.Blog',
        'bscw.core.poll.cl_poll.Poll',
        'bscw.core.portal.cl_portal.WSPortal',
        'bscw.core.Tasks.cl_wbtask.WBTask',
    ]

    for cl in classes:
        print "====================="
        print "attributes of class: " + cl
        print "====================="
        result = srv.get_attributenames(cl)
```

```
        dump(result)
    return
main()
```

Most Important API Methods

Even if there are many dozens of X-BSCW methods, the most important are probably:

Retrieving the entire X-BSCW function definition

```
get_api
```

Adding objects to the BSCW database:

```
addfolder
adddocument
addnote
addurl
...
```

Access rights

```
addmember
addrole
editrole
chrole
pubaccess
get_access_rights
```

Accessing and manipulating data

```
get_attributes
set_attributes
get_document
get_events
```

Sample X-BSCW Clients

X-BSCW clients can be found in your BSCW installation: `etc/src-aux/api_clients/`, e.g. `get_attributes.py` (see section on attributes above).
`get_events.py`, printed below, will return with a list of events on an object. You may use (but need not) `xmlrpc_srv.py` to create an XML-RPC server instance and `dump.py` for formatted output.

Another set of fairly comprehensive API tests is performed by `apix_test.py` which uses `apix_util.py` and `apix_call.py`.

```
#
# module xmlrpc_srv.py
#
# configure a xmlrpclib server object for BSCW
#
def XMLRPC_Server(uri, user_passwd=None, encoding=None,
                  verbose=0, allow_none=0, use_datetime=0):
    import urllib, xmlrpclib

    if user_passwd:

        uri_type = urllib.splittype(uri)[0]
        if uri_type == "https":
            BaseTransport = xmlrpclib.SafeTransport
        else:
            BaseTransport = xmlrpclib.Transport

        class AuthorizedTransport(BaseTransport):
            def __init__(self, use_datetime=0, user_passwd=''):
                import base64, string
                BaseTransport.__init__(self, use_datetime=use_datetime)
                self.auth = string.strip(base64.encodestring(user_passwd))

            def send_host(self, connection, host):
                BaseTransport.send_host(self, connection, host)
                if self.auth:
                    connection.putheader('Authorization',
                        'Basic %s'%self.auth)

        ServerTransport = AuthorizedTransport(use_datetime, user_passwd)

    else:

        ServerTransport = None

    return xmlrpclib.Server(uri, ServerTransport,
        encoding=encoding, verbose=verbose, allow_none=allow_none)



#
# module get_events.py
#
# return a list of events on a BSCW object
#
from xmlrpc_srv import XMLRPC_Server
from dump import dump

USER_PASSWD = '<userid>:<passwd>'

SERVER_URL = 'http://my_server.orbiteam.de/bscw/bscw.cgi'
OBJ_ID = '1234567890'

ATTRS = [
       'name',
       'home',
       '__id__',
       '__class__',
]

def main():
       print SERVER_URL
       srv = XMLRPC_Server(SERVER_URL, USER_PASSWD, verbose=0)

       result = srv.get_events(OBJ_ID, 'show_history', True)
       print "================== show_history"
```

```
        dump(result)

        result = srv.get_events(OBJ_ID, 'show_touches', True)
        print "================== show_touches"
        dump(result)

        result = srv.get_events(OBJ_ID, 'show_readers', True)
        print "================== show_readers"
        dump(result)

        result = srv.get_events(OBJ_ID, 'show_all', True)
        print "================== show_all"
        dump(result)

        return

main()
```

Online Reference Documentation

The complete list of X-BSCW methods can be found at BSCW's web site: http://www.bscw.de/api (XML-RPC methods). The pages are continuously updated and should serve as the exclusive reference to the X-BSCW. Python's documentation package Epydoc is used for documentation. This document can be found here: http://www.bscw.de/api/doc/X-BSCW.pdf.

In your own BSCW server installation, you will find all relevant documentation here: `doc/devel/.`