Remote access API

Created by Kohsuke Kawaguchi, last modified by Josh Soref on Nov 23, 2017

Jenkins provides machine-consumable remote access API to its functionalities. Currently it comes in three flavors:

1. XML
2. JSON with JSONP support
3. Python

Remote access API is offered in a REST-like style. That is, there is no single entry point for all features, and instead they are available under the `".../api/"` URL where `"..."` portion is the data that it acts on.

For example, if your Jenkins installation sits at http://ci.jruby.org/, visiting http://ci.jruby.org/api/ will show just the top-level API features available – primarily a listing of the configured jobs for this Jenkins instance.
Or if you want to access information about a particular build, e.g. http://ci.jruby.org/job/jruby-base/lastSuccessfulBuild/, then go to http://ci.jruby.org/job/jruby-base/lastSuccessfulBuild/api/ and you'll see the list of functionalities for that build.

The work on this front is ongoing, so if you find missing features, please file an issue.

What can you do with it?

Remote API can be used to do things like these:

1. retrieve information from Jenkins for programmatic consumption.
2. trigger a new build
3. create/copy jobs

Submitting jobs

Jobs without parameters

You merely need to perform an HTTP POST on `JENKINS_URL/job/JOBNAME/build?token=TOKEN` where TOKEN is set up in the job configuration.

Jobs with parameters

Also see Parameterized Build.

Simple example - sending "String Parameters":

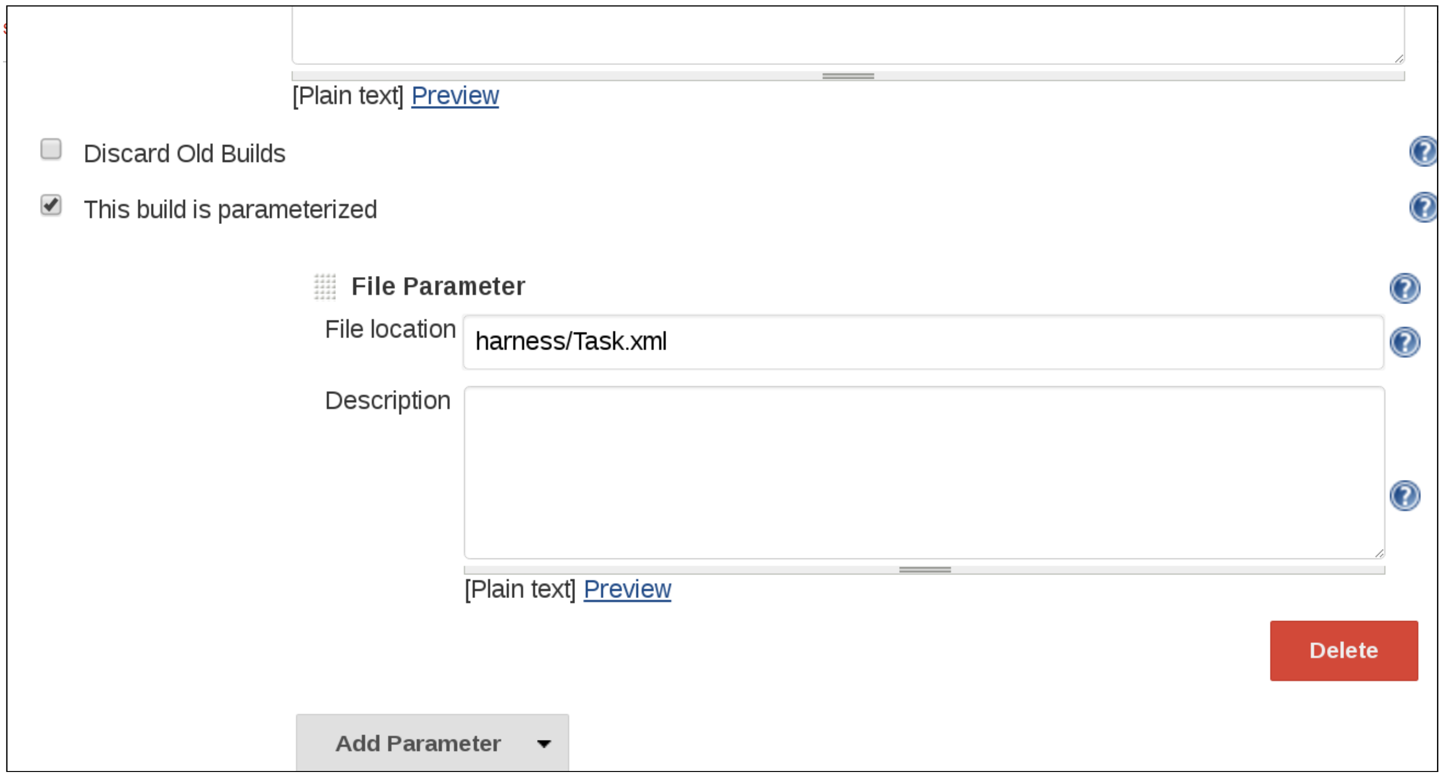
```
curl -X POST JENKINS_URL/job/JOB_NAME/build \
  --user USER:TOKEN \
  --data-urlencode json='{"parameter": [{"name":"id", "value":"123"}, {"name":"verbosity", "value":"high"}]}'
```

Another example - sending a "File Parameter":

```
url -X POST JENKINS_URL/job/JOB_NAME/build \
 --user USER:PASSWORD \
 --form file0=@PATH_TO_FILE \
 --form json='{"parameter": [{"name":"FILE_LOCATION_AS_SET_IN_JENKINS", "file":"file0"}]}'

.g.curl -X POST http://JENKINS_URL/job/JOB_NAME/build  --form file0=@/home/user/Desktop/sample.xml --form json='{"param
lease note, in this example, the symbol '@' is important to mention. Also, the path to the file is absolute path.
n order to make this command work, you need to configure your Jenkins job to take a file parameter and 'name' in this c

n above example, 'harness' is just a name of folder that may not exist in your workspace, you can write "name":"Task.xm
emember that name in this command should match with File location in file parameter in job configuration.
```


[Plain text] Preview

Discard Old Builds

This build is parameterized

File Parameter

File location: harness/Task.xml

Description:

[Plain text] Preview

Remote API and security

When your Jenkins is secured, you can use HTTP BASIC authentication to authenticate remote API requests. See Authenticating scripted clients for more details.

CSRF Protection

If your Jenkins uses the "Prevent Cross Site Request Forgery exploits" security option (which it should), when you make a `POST` request, you have to send a CSRF protection token as an HTTP request header.
For curl/wget you can obtain the header needed in the request from the URL `JENKINS_URL/crumbIssuer/api/xml` (or `.../api/json`). Something like this:

```
wget -q --auth-no-challenge --user USERNAME --password PASSWORD --output-document - \
'JENKINS_URL/crumbIssuer/api/xml?xpath=concat(//crumbRequestField,":",//crumb)'
```

This will print something like ".crumb:1234abcd", which you should add to the subsequent request.

Sample code

A simple client is available to demonstrate how you can invoke the XML from Java (Java source)

XPath selection

The XML API supports a selection by XPath by using the query parameter 'xpath'. This is convenient for extracting information in environments where XML manipulation is tedious (such as shell script.) See issue #626 for an example of how to use this.
See `.../api/` on your Jenkins server for more up-to-date details.

XPath exclusion

Similar to the 'xpath' query parameter above, you can use (possibly multiple) 'exclude' query patterns to exclude nodes from the resulting XML. All the nodes that match the specified XPath will be removed from the XML.
See .../api/ on your Jenkins server for more up-to-date details.

Depth control

Sometimes the remote API doesn't give you enough information in one call. For example, if you'd like to find out all the last successful build of a given view, you'd realize that the invocation to the remote API of the view won't give you this, and you'd have to recursively call the remote API of each project to find this out. The depth control, introduced in 1.167, solves this problem. To understand this feature, it's good to start with how the remote API works.

The data model that Jenkins maintains internally can be thought of as a big tree structure, and when you make a remote API call, you are getting a small subtree of it. The subtree is rooted at the object for which you made a remote API call, and the sub-tree is cut beyond certain depth to avoid returning too much data. You can adjust this cut-off behavior by specifying the depth query parameter. When you specify a positive depth value, the subtree cut-off happens that much later.

So the net result is, if you specify a bigger depth value, you'll see that the remote API will now return more data. Because of the algorithm, this works in such a way that the data returned by a bigger depth value includes all the data returned by smaller
depth value.

See `.../api/` on your Jenkins server for more up-to-date details.

Python API wrappers

- Query the test-results of a completed build
- Get objects representing the latest builds of a job
- Search for artifacts by simple criteria
- Block until jobs are complete
- Install artifacts to custom-specified directory structures
- username/password auth support for jenkins instances with auth turned on
- Ability to search for builds by subversion revision
- Ability to add/remove/query jenkins slaves

Ruby API wrappers

Jenkins API Client is an object oriented ruby wrapper project that consumes Jenkins's JSON API and aims at providing access to all remote API Jenkins provides. It is available as a Rubygem and can be useful to interact with the Job, Node, View, BuildQueue, and System related functionalities. Services currently offered include:

- Creating jobs by sending xml file or by specifying params as options with more customization options including source control, notifications, etc.
- Building jobs (with params), stopping builds, querying details of recent builds, obtaining build params, etc.
- Listing jobs available in Jenkins with job name filter, job status filter.
- Adding/removing downstream projects.
- Chaining jobs i.e given a list of projects each project is added as a downstream project to the previous one.
- Obtaining progressive console output.
- Username/password based authentication.
- Command Line Interface with a lot of options provided in the libraries.
- Creating, listing views.
- Adding jobs to views and removing jobs from views.
- Adding/removing jenkins slaves, querying details of slaves.
- Obtaining the tasks in build queue, and their age, cause, reason, ETA, ID, params and much more.
- Quiet down, cancel quiet down, safe restart, force restart, and wait till Jenkins becomes available after a restart.
- Ability to list installed/available plugins, obtain information about plugins, install/uninstall plugins and much more with plugins.

This project is in rapid development and new features are getting added every day. Watch the progress here.

Detecting Jenkins version

To check the version of Jenkins, load the top page (or, as of 1.483, any `.../api/*` page too) and check for the `X-Jenkins` response header. This contains the version number of Jenkins, like "1.404" This is also a good way to check if an URL is a Jenkins URL.

Discovering Jenkins on the network

Jenkins instances listen on UDP port 33848. Whenever a UDP packet is received, it will respond with a short XML fragment that shows the connection information. This XML has the following format:

```
<hudson>
  <version>1.380</version>             <!-- version of Jenkins -->
  <url>http://somewhere/jenkins/</url> <!-- HTTP URL. Not available if not configured -->
  <slave-port>12345</slave-port>       <!-- if TCP slave listener port is configured, its number -->
</hudson>
```

By using this, a client can use a UDP broadcast to try to discover nearby Jenkins instances. This is primarily useful for Swarm Plugin.

Like   Be the first to like this   No labels

34 Comments

Unknown User (alejandropg@autentia.com)

Spanish tutorial about how to launch a build from a Subversion's hook, using the remote API

http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=hudsonSubversionPush

Unknown User (muad.dune@gmail.com)

On the Parameterized Build page (http://wiki.jenkins-ci.org/display/HUDSON/Parameterized+Build) you can find other useful information about remote API feature:

e.g.: Remote build with multiple parameters: http://server/job/myjob/buildWithParameters?PARAM1=Value1&PARAM2=Value2

Unknown User (holger.horn1977@web.de)

I tried to trigger remotly by URL a job with multiple parameters like this :

http://server/job/myjob/buildWithParameters?PARAM1=Value1&PARAM2=Value2

exactly this is not working, what am I doing wrong?

is someone there who has tried it with success?

I tried also to send same request with JSON, like described above. no success too... I used this URL:

? Unknown Attachment   can someone help me? what am I doing wrong?

Joel Beaudoin

Is there any way to use this API to tell Hudson to "keep this build forever" for a particular build. I am working on some automation for publishing a particular build to an FTP site and this is one of the steps I would like performed. So far, I haven't been able to figure out a programmatic way to mark a given build as "keep" in Hudson. Ideas?

se Jen

Joel Beaudoin

After looking into this a bit closer, I figured out how to do this. A simple wget/curl of something like:

http://company.com/hudson/view/ViewNameHere/job/JobNameHere/nn/toggleLogKeep

or outside of a view like

http://company.com/hudson/job/JobNameHere/nn/toggleLogKeep

Replacing ViewNameHere and JobNameHere as appropriate and specifying the build number in place of 'nn'. Obvious after viewing the HTML, but it would be cool to have a page documenting this type of thing. Just an idea ... I sure don't have the time to do it 8-(

Joel Beaudoin

Here is a simple python example to do this toggling. Most of the logic is to deal with the 403 that Hudson returns to request authorization:

```
import urllib2
import base64

class HudsonToggleKeep:

    def __init__(self):
        self.username = 'username'
        self.password = 'password'
        self.server   = 'https://server.com'

        pass_mgr = urllib2.HTTPPasswordMgrWithDefaultRealm()
        pass_mgr.add_password(None, self.server, self.username, self.password)

        b_auth = urllib2.HTTPBasicAuthHandler(pass_mgr)
        d_auth = urllib2.HTTPDigestAuthHandler(pass_mgr)

        self.url_opener = urllib2.build_opener(b_auth, d_auth,
                                               self.HudsonFormLoginHandler(self))

        urllib2.install_opener(self.url_opener)


    class HudsonFormLoginHandler(urllib2.BaseHandler):
        def __init__(self, parent):
            self.p = parent

        def http_error_403(self, req, fp, code, msg, headers):
            for h in self.p.url_opener.handlers:
                if isinstance(h, self.p.HTTPOpenHandlerBasicAuthNoChallenge):
                    return

            self.p.url_opener.add_handler(
                self.p.HTTPOpenHandlerBasicAuthNoChallenge(self.p.username,
                                                           self.p.password))
            fp.close()
            return self.p.url_opener.open(req)

    class HTTPOpenHandlerBasicAuthNoChallenge(urllib2.BaseHandler):

        auth_header = 'Authorization'

        def __init__(self, username, password):
            raw = "%s:%s" % (username, password)
            self.auth = 'Basic %s' % base64.b64encode(raw).strip()

        def default_open(self, req):
            req.add_header(self.auth_header, self.auth)



if __name__ == "__main__":
    hp = HudsonToggleKeep()
    req = urllib2.Request("https://server.com/hudson/view/MyView/job/MyJob/15/toggleLogKeep")
    d = urllib2.urlopen(req).read()
```

se Jenkins

Jim Divine

Here's an easier way to do the forced basic authentication.

```
import urllib2
import base64

def urlopen(url, data=None):
    '''Open a URL using the urllib2 opener.'''
    request = urllib2.Request(url, data)
    base64string = base64.encodestring('%s:%s' % (JENKINS_LOGIN, JENKINS_PASSWD)).replace('\n',
    request.add_header("Authorization", "Basic %s" % base64string)
    response = urllib2.urlopen(request)
    return response
```

Then use this urlopen function instead of urllib2.urlopen.

Joel Beaudoin

Much simpler and solves a problem that I'm having with the method I was using previously. Thanks for sharing!

Danny Staple

My version of that is:

```
wget --no-proxy ${BUILD_URL}/toggleLogKeep --quiet --spider --http-user="<username>" --auth-no-challe
```

The --quiet means that it doesn't show progress etc, --spider means you don't get some HTML downloaded. The auth is as above.

The --no-proxy is needed to prevent proxies trying to cache this and causing weird behaviour. If you are the wrong side of the proxy from the jenkins/hudson box you'll need to leave that in of course.

Unknown User (hudson_luoxu)

python sample code of hudson remote api

```python
#!/usr/bin/env python

# this script is only demo purpose which is designed to get properties of job, queue, like
# nextBuildNumber. But note the logistics might not be correct

import urllib2

#call api of job 'git_plugin_test'
url="http://localhost:9001/job/git_plugin_test/api/python?depth=0"
response=urllib2.urlopen(url)
build_dict=eval(response.read())

#call api of job 'queue' of hudson (global but not specific for one job)
url="http://localhost:9001/queue/api/python?depth=0"
response=urllib2.urlopen(url)
queue_dict=eval(response.read())

print ''*40,'build dict',''*40
#print properties of job
for eachKey in build_dict:
    print eachKey,build_dict[eachKey]
    print

print ''*40,'queue dict',''*40
#look through items in queue and can be extended to forecast the job build
#number in for one item in queue
for index in range(1,len(queue_dict['items'])):
    print ''*40,'queue hash',''*40
    qi_action=queue_dict['items'][index]['actions']
    list_para=qi_action[0]['parameters']
    for index1 in range(0,len(list_para)):
        print list_para[index1]
        if list_para[index1]['name'] == 'SLEEP_TIME' and list_para[index1]['value'] == '62':
            print "OK"

#only valid when no more than one build found in queue
if build_dict['inQueue']:
    build_number=int(build_dict['nextBuildNumber']) + 1
else:
    build_number=int(build_dict['nextBuildNumber'])
print "Hudson Build URL:",build_dict['url']+str(build_number)
print "Current build tree:"+build_dict['builds'][0]['url']
```

James Coleman

python example extracting junit test results

urllib2 login using base64'd username/password in Auth header.

```
#!/usr/bin/env python

# get test result from jenkins job - jenkins http python API
# note: script can be called only after junit test results are loaded into job at end of test

import urllib2
import base64
import os

# URL of job
#url = os.environ['BUILD_URL']
#jobname = os.environ['JOB_NAME']
#url="%s/testReport/api/python" % (url)
baseurl="http://jenkinsserver:port"
jobname="jobname"
url="%s/job/%s/lastCompletedBuild/testReport/api/python" % (baseurl,jobname)
print "url: %s" % url

username="..."
password="..."
base64string = base64.encodestring('%s:%s' % (username, password)).replace('\n', '')
request = urllib2.Request(url)
request.add_header("Authorization", "Basic %s" % base64string)
result = urllib2.urlopen(request)
# job dict
j=eval(result.read())

# keys of job:
# failCount suites skipCount empty duration passCount _class testActions
print "TOTAL test count: pass:%d fail:%d skip:%d" % (j['passCount'],j['failCount'],j['skipCount'])

oldClassName=""
suites=j['suites']
for s in suites:
    for c in s['cases']:
        if c['className'] != oldClassName:
            print "TEST CLASS: %s" % c['className']
            oldClassName = c['className']
        skipped = ""
        if c['skipped']:
            skipped = " SKIP(%s)" % c['skippedMessage']
        print "     TEST RESULT: %s%s NAME: %s" % ( c['status'], skipped, c['name'] )
        if c['errorDetails']:
            print "      ERR: %s" % c['errorDetails']
        #if c['errorStackTrace']:
        #    from pprint import pprint
        #    pprint(c['errorStackTrace'],indent=8)

#keys e.g. of test case:
#           "testActions" : [],
#           "age" : 0,
#           "className" : "testfile.testclass",
#           "duration" : 31.570633,
#           "errorDetails" : None,
#           "errorStackTrace" : None,
#           "failedSince" : 0,
#           "name" : "test_001",
#           "skipped" : False,
#           "skippedMessage" : None,
#           "status" : "PASSED",
#           "stderr" : "stuff",
#           "stdout" : "loads of stuff"
```

Unknown User (boggybumblebee)

Is there any way to access the configuration of the Jobs, I tried the following, but with no success:

http://company.com/hudson/job/JOBNAME/configure/api/xml

This just yielded a 404, so either this functionality doesn't exist, or I have erred on the URL.

Joel Beaudoin

You can access a job's configuration via:

http://company.com/hudson/job/JOBNAME/config.xml

You can find out this and other cool things via:

http://company.com/hudson/job/JOBNAME/api/

Hope this helps,
Joel

Unknown User (boggybumblebee)

A big thanks Joel; that was exactly what I was after.

Victor Rodrigues

Could I get a JSON containing the jobs that are currently being executed?

I wish I could get build history through JSON api also. Is there any uri I'm missing?

Thanks!

Victor

Gstad Winkldorff

"Could I get a JSON containing the jobs that are currently being executed?"

I second that request.

Best regards.

Gstad

Scal Pa

Could someone show a working code sample on how to trigger a paramtrized job using JQuery?
I use no authentication of any kind at Jenkins level and there is no "Build Trigger" option enabled for the job. Here is the piece of code I try but I arways get back a status code of [0] with a failure:

```
var jqxhr = $.post(
        "http://servername:3333/job/jobname/buildWithParameters",
       { "PARAM_ID": "1", "PARAM_VALUE1": "1", "PARAM_VALUE2": "*firefox" },
        "html"
    )
    .success(function () { alert("success"); })
    .error(function (xhr, ajaxOptions, thrownError) { alert("Error\nxhr.status = \[" + xhr.status + "\]\n xhr.
    .complete(function () { alert("complete"); }
);
```

Can someone please confirm this is the correct way or point out errors/mistakes I'm doing with this remote api call?

Thanks!

Harish Kayarohanam

Is there a way to add jenkins plugin using Remote access API ?

User icon: jglick

Jesse Glick

See `/pluginManager/api/` for information on this.

Roger Myung

I'm able to successfully post a build using the instructions here. However, my script has difficulty telling whether it succeeded.

I get a HTTP/1.1 302 Found back.

Is there a way to get confirmation from Jenkins that the build was added to the queue? Even better, can I get the build number, and/or get notification when the build finishes?

Sanjoy Ghosh

I am running Hudson job remotely through separate application using remote access api. But how shall I get the build status of my request build as soon as the build is finished.

Matt Don

Has anyone been able to get authentication working with an API token using groovy HTTPBuilder? I know the example on this site shows groovy with HTTPclient, but I'm just wondering if its possible with HTTPBuilder. I can do basic pre-emptive authentication with username password, but I can't get it to work with the API token.

Jenkins

Nane Nare Hambardzumyan

is it possible ignore certificate during connection? i get _ssl.c:504: error:14077438:SSL routines:SSL23_GET_SERVER_HELLO:tlsv1 alert internal error during connection

Jenkins

Mahendra Singh

How can I update my Jenkins config using api:

Using Jenkins ver 1.5961

I am trying the below option:

```
curl -v -X POST --data-binary @myconfig.xml -u "userid:apiToken" http://localhost:8080/job/job_name/config.xml
```

Getting Exception:

```
Please check <a href="https://issues.jenkins-ci.org/">our bug tracker</a> to see if a similar problem has already been reported.
If it is already reported, please vote and put a comment on it to let us gauge the impact of the problem.
If you think this is a new issue, please file a new issue.
When you file an issue, make sure to add the entire stack trace, along with the version of Jenkins and relevant plugins.
<a href="http://jenkins-ci.org/content/mailing-lists">The users list</a> might be also useful in understanding what has happened.</p><h2>Stack trace</h2><pre
style="margin:2em; clear:both">java.io.IOException: Failed to persist config.xml
at hudson.model.AbstractItem.updateByXml(AbstractItem.java:648)
at hudson.model.AbstractItem.doConfigDotXml(AbstractItem.java:614)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
at java.lang.reflect.Method.invoke(Method.java:597)
at org.kohsuke.stapler.Function$InstanceFunction.invoke(Function.java:298)
at org.kohsuke.stapler.Function.bindAndInvoke(Function.java:161)
at org.kohsuke.stapler.Function.bindAndInvokeAndServeResponse(Function.java:96)
at org.kohsuke.stapler.MetaClass$1.doDispatch(MetaClass.java:121)
at org.kohsuke.stapler.NameBasedDispatcher.dispatch(NameBasedDispatcher.java:53)
at org.kohsuke.stapler.Stapler.tryInvoke(Stapler.java:746)
at org.kohsuke.stapler.Stapler.invoke(Stapler.java:876)
at org.kohsuke.stapler.MetaClass$6.doDispatch(MetaClass.java:249)
at org.kohsuke.stapler.NameBasedDispatcher.dispatch(NameBasedDispatcher.java:53)
at org.kohsuke.stapler.Stapler.tryInvoke(Stapler.java:746)
at org.kohsuke.stapler.Stapler.invoke(Stapler.java:876)
at org.kohsuke.stapler.MetaClass$6.doDispatch(MetaClass.java:249)
at org.kohsuke.stapler.NameBasedDispatcher.dispatch(NameBasedDispatcher.java:53)
at org.kohsuke.stapler.Stapler.tryInvoke(Stapler.java:746)
at org.kohsuke.stapler.Stapler.invoke(Stapler.java:876)
at org.kohsuke.stapler.Stapler.invoke(Stapler.java:649)
at org.kohsuke.stapler.Stapler.service(Stapler.java:238)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:717)
at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:290)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:206)
at hudson.util.PluginServletFilter$1.doFilter(PluginServletFilter.java:96)
at hudson.plugins.scm_sync_configuration.extensions.ScmSyncConfigurationFilter$1.call(ScmSyncConfigurationFilter.java:36)
at hudson.plugins.scm_sync_configuration.ScmSyncConfigurationDataProvider.provideRequestDuring(ScmSyncConfigurationDataProvider.java:103)
at hudson.plugins.scm_sync_configuration.extensions.ScmSyncConfigurationFilter.doFilter(ScmSyncConfigurationFilter.java:32)
at hudson.util.PluginServletFilter$1.doFilter(PluginServletFilter.java:99)
at net.bull.javamelody.MonitoringFilter.doFilter(MonitoringFilter.java:206)
at net.bull.javamelody.MonitoringFilter.doFilter(MonitoringFilter.java:179)
at net.bull.javamelody.PluginMonitoringFilter.doFilter(PluginMonitoringFilter.java:86)
at org.jvnet.hudson.plugins.monitoring.HudsonMonitoringFilter.doFilter(HudsonMonitoringFilter.java:84)
at hudson.util.PluginServletFilter$1.doFilter(PluginServletFilter.java:99)
at hudson.plugins.audit_trail.AuditTrailFilter.doFilter(AuditTrailFilter.java:66)
at hudson.util.PluginServletFilter$1.doFilter(PluginServletFilter.java:99)
at hudson.util.PluginServletFilter.doFilter(PluginServletFilter.java:88)
at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:235)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:206)
at hudson.security.csrf.CrumbFilter.doFilter(CrumbFilter.java:48)
at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:235)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:206)
at hudson.security.ChainedServletFilter$1.doFilter(ChainedServletFilter.java:84)
at hudson.security.UnwrapSecurityExceptionFilter.doFilter(UnwrapSecurityExceptionFilter.java:51)
at hudson.security.ChainedServletFilter$1.doFilter(ChainedServletFilter.java:87)
at jenkins.security.ExceptionTranslationFilter.doFilter(ExceptionTranslationFilter.java:117)
at hudson.security.ChainedServletFilter$1.doFilter(ChainedServletFilter.java:87)
at org.acegisecurity.providers.anonymous.AnonymousProcessingFilter.doFilter(AnonymousProcessingFilter.java:125)
at hudson.security.ChainedServletFilter$1.doFilter(ChainedServletFilter.java:87)
at org.acegisecurity.ui.rememberme.RememberMeProcessingFilter.doFilter(RememberMeProcessingFilter.java:142)
at hudson.security.ChainedServletFilter$1.doFilter(ChainedServletFilter.java:87)
at org.acegisecurity.ui.AbstractProcessingFilter.doFilter(AbstractProcessingFilter.java:271)
at hudson.security.ChainedServletFilter$1.doFilter(ChainedServletFilter.java:87)
at jenkins.security.BasicHeaderProcessor.success(BasicHeaderProcessor.java:140)
at jenkins.security.BasicHeaderProcessor.doFilter(BasicHeaderProcessor.java:82)
at hudson.security.ChainedServletFilter$1.doFilter(ChainedServletFilter.java:87)
at org.acegisecurity.context.HttpSessionContextIntegrationFilter.doFilter(HttpSessionContextIntegrationFilter.java:249)
at hudson.security.HttpSessionContextIntegrationFilter2.doFilter(HttpSessionContextIntegrationFilter2.java:67)
at hudson.security.ChainedServletFilter$1.doFilter(ChainedServletFilter.java:87)
at hudson.security.ChainedServletFilter.doFilter(ChainedServletFilter.java:76)
at hudson.security.HudsonFilter.doFilter(HudsonFilter.java:164)
at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:235)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:206)
at org.kohsuke.stapler.compression.CompressionFilter.doFilter(CompressionFilter.java:49)
at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:235)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:206)
at hudson.util.CharacterEncodingFilter.doFilter(CharacterEncodingFilter.java:81)
at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:235)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:206)
at org.kohsuke.stapler.DiagnosticThreadNameFilter.doFilter(DiagnosticThreadNameFilter.java:30)
at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:235)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:206)
at org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:233)
at org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:191)
at org.apache.catalina.authenticator.AuthenticatorBase.invoke(AuthenticatorBase.java:558)
at org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:127)
at org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:102)
at org.apache.catalina.core.StandardEngineValve.invoke(StandardEngineValve.java:109)
at org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:298)
at org.apache.coyote.http11.Http11Processor.process(Http11Processor.java:852)
at org.apache.coyote.http11.Http11Protocol$Http11ConnectionHandler.process(Http11Protocol.java:588)
at org.apache.tomcat.util.net.JIoEndpoint$Worker.run(JIoEndpoint.java:489)
at java.lang.Thread.run(Thread.java:619)
Caused by: java.lang.ClassNotFoundException: org.apache.xerces.parsers.SAXParser
at org.xml.sax.helpers.XMLReaderFactory.loadClass(XMLReaderFactory.java:189)
at org.xml.sax.helpers.XMLReaderFactory.createXMLReader(XMLReaderFactory.java:150)
at jenkins.util.xml.XMLUtils.safeTransform(XMLUtils.java:48)
at hudson.model.AbstractItem.updateByXml(AbstractItem.java:643)
... 84 more
```

se Jenkins

```
Caused by: java.lang.ClassNotFoundException: org.apache.xerces.parsers.SAXParser
at org.apache.catalina.loader.WebappClassLoader.loadClass(WebappClassLoader.java:1516)
at org.apache.catalina.loader.WebappClassLoader.loadClass(WebappClassLoader.java:1361)
at org.xml.sax.helpers.NewInstance.newInstance(NewInstance.java:49)
at org.xml.sax.helpers.XMLReaderFactory.loadClass(XMLReaderFactory.java:187)
... 87 more
```

Rick Oosterholt

I had a hard time getting jQuery/ajax and json to work.

Finnaly found the solution; See Stackoverflow

Shobha Deepak

Is there any way I can send file parameter do BuilWithParameters Jenkins API?

Currently, I am using curl command to send file which is working as expected but it is not showing up console log.

Any help will be much appreciated...

Thanks in Advance

Ankur Agarwal

I was able to send a file parameter to a Jenkins job and kick off a build. It worked as expected. However I could not get it to work using python requests.

> curl -v 'http://url/job/parameterized-test/build?token=buildtoken' -H "Jenkins-Crumb:8f4d0320c04619f6302301789b2b65ec" --form file0=@index.html --form json='{"parameter": [{"name":"files/abc.zip", "file":"file0"}]}'

The above works despite the fact that I am using /build api.

Shobha Deepak

How did you get Jenkins-Crumb & token?

Ankur Agarwal

I am hoping some one can give me working code on how to trigger a jenkins job that takes a file parameter using python requests. I tried this, but it does not work:

```
url = 'myurl/buildWithParameters'
data = {'Jenkins-Crumb': '8f4d0320c0982763098571789b2b65ec', 'token': 'mytoken'}
filepath='/some/file/path/file.log'
files = {'file': ('files/abc.zip', open(filepath, 'rb')) }
r = requests.post(url, data=data, files=files)
```

Kevin Navero

If in need of more examples, the following works:

```
curl -X POST http://jenkins.carson.nextest.com:8080/jenkins/job/Test_All_Accept/build?token=TOKEN \
       --data-urlencode json='{"parameter": [{"name":"FOO", "value":"foo"}, {"name":"BAR", "value":"bar"}]}' > foo.html

curl -X POST http://jenkins.carson.nextest.com:8080/jenkins/job/Test_All_Accept/build \
       --data token=TOKEN \
       --data-urlencode json='{"parameter": [{"name":"FOO", "value":"foo"}, {"name":"BAR", "value":"bar"}]}' > foo.html
```

Note that I redirect to a foo.html which helps with debugging if your curl command fails. Simply just open foo.html in a browser if you get an html dump.

Shobha Deepak

Curl command is working, but I cannot see the console log. It shows Java exception when the build is in progress.

This issue is not observed when I use Net::HTTP::Post for other jenkins jobs which doesn't have file has a parameter.

User icon: adibend

adi ben david

Hi,

I did a little coding and you can find it here: https://jenkinsmanagment.wordpress.com/

🙂

Regards, Adi

Mellisa Lee

Hi,

The jenkins I am using is Jenkins ver. 2.60.1, I just want to trigger the new created jenkins job to let jenkins clone the project branch(which contains Jenkinsfile) I want, how to do it?

The command I use is:

curl -X POST "http://localhost:8080/job/order-center-bis-dx/buildWithParameters" --user "$username:$password" --data-binary "tag_branch:CI" -H "$CRUMB"(tag_branch is I added in configuration as Git Parameter)

But I think the --data-binary I used was incorrect, I also tried --data-binary "branch:CI" or "ref:CI", every time I failed.

I want some advices.

Thanks

Jenkins User icon: adibend

adi ben david

Hi,

Want to know how to run your jenkins server from mobile?

You can find it here :

https://jenkinsmanagment.wordpress.com/

🙂

Regards, Adi