# Mock Server

V 1.2.1 by Allen Chen

Index

[Mock Server 1](#_Toc410288983)

[Functionality 2](#_Toc410288984)

[Theory 2](#_Toc410288985)

[CONDITION 2](#_Toc410288986)

[Usage 2](#_Toc410288987)

[Set Main Rule 2](#_Toc410288988)

[Set response file 5](#_Toc410288989)

[Rule file and response file modification 6](#_Toc410288990)

[HTTPS support 6](#_Toc410288991)

[limitation 6](#_Toc410288992)

[work with Fiddler 7](#_Toc410288993)

[Mock for unexist domain 7](#_Toc410288994)

## Functionality

Define a rule that tells mock server which kind of http request should be mocked. The mock server will return predefined HTTP response for it. HTTPS supported.

## Theory

## CONDITION

Local 9999 port is not occupied by other programs.

Win 7+

## Usage

### Set Main Rule

You can find Rules.txt under the root directory of the program and set rules in it. The main rule file defines which request should be mocked and which response should be returned for it.

#### Priority

Mock Server checks rules from the top down, when it sees current request matches a rule, it will return the response.

#### Grammar

##### exact match

Sample(Note it’s tab instead of white spaces between url and file path) :

https://www.jd.com/hello?a=2 2.txt

For the request whose url is <https://www.jd.com/hello?a=2> it will return the content defined in the 2.txt file under the Response folder.

##### Match start

Sample(Note it’s tab instead of white spaces between url and file path) :

[startwith]https://www.jd.com/hello 2.txt

Any request that starts with <https://www.jd.com/hello> will return the content defined in the 2.txt file under the Response folder.

##### match contains

Sample(Note it’s tab instead of white spaces between url and file path) :

[contains] hello 2.txt

Any request whose url contains hello will return the content defined in the 2.txt file under the Response folder. When you use match contains, please put it down if possible, in case the mock server mocks response of other request unexpectedly due to priority.

##### Append Body contains

Sample(Note it’s tab instead of white spaces between url and file path) :

[startwith]http://www.thisisfakedomain123.com[contains]<funcId>10003 2.txt

Any request whose url starts with<http://www.thisisfakedomain123.com> and body contains <funcId>10003 will return the content defined in the 2.txt file under the Response folder.

##### Two file mapping

Sample(Note it’s tab instead of white spaces between url and file path) :

https://www.jd.com/hello h.txt,b.txt

For the request whose url is <https://www.jd.com/hello> the response will be:

HTTP header：content defined in the h.txt under the Response folder

HTTP Body: content defined in the b.txt under the Response folder

##### Verb Filter

Sample(Note it’s tab instead of white spaces between url and file path) :

http://www.baidu.com/ 1.txt GET

For the request whose url is <http://www.baidu.com/> and the request VERB is GET, return content defined in the 1.txt under the Response folder

##### simulate network delay

Sample(Note it’s tab instead of white spaces between url and file path) :

http://www.baidu.com/ 1.txt +20s

http://www.baidu.com/ +20s

http://www.baidu.com/ +200ms

You can append network delay at the end of a single rule. Support second and millisecond.

##### REPLACE RESPONSE content

Sample(Note it’s tab instead of white spaces between url and file path) :

[startwith]https://www.jd.com/ [replaceStr]JD.COM[with]jingdong

Any request that starts with <https://www.jd.com/> whose response content will replace JD.COM with jingdong.

[startwith]https://www.jd.com/ [replaceRegex] <meta([^<]\*)charset=([^<]\*)[\"'] [with]jingdong

Any request that starts with <https://www.jd.com/> whose response content will replace any mathing of the regex expressions with jingdong

##### Enable request/response converter

This function is used for cases that the request/response is handled in a special way in your system. A typical case is, the request and response is encrypted, but you expect to set rule for this request and want to make sure only the request body contains some text, the response is tampered. Besides, you want to set the tampered response in plain text. In such cases, you can use this function.

To use this function, in Mock Server’s App.Config set this to true:

<add key="EnableConvertService" value="true"/>

When it’s set to true, you can specify whether to convert request/response only by:

<add key="ConvertMode" value="Both"/>

The supported options are:

* Both
* ReqOnly
* RespOnly

To set converter in rule, add [convert] at the end of a single rule (but before the network delay part if there is) . Sample:

[startwith]http://www.thisisfakedomain123.com[contains]<funcId>10003 s.txt [convert]

This rule means, for any request that starts with <http://www.thisisfakedomain123.com>, the mock server will call request converter first to convert original request body, then checks whether the converted body string contains <funcId>10003, if it does, the mock server will get the content defined in s.txt first and call response converter to get the converted response and return the converted response.

To use this function, you also need a RESTful Service and start that service when the mock is in use

**Specification of the request converter：**

In Mock Server’s A pp.Config you can set the interface url of your request convert service. Sample:

<add key="ConvertReqServiceUrl" value="http://localhost:18989/ConvertSvc/ConvertReq"/>

The request converter should accept POST request, convert the posted body and return the converted string.

**Specification of the response converter：**

In Mock Server’s A pp.Config you can set the interface url of your response convert service. Sample:

<add key="ConvertRespServiceUrl" value="http://localhost:18989/ConvertSvc/ConvertRespEncrypt"/>

The response converter should accept POST request, convert the posted body and return the converted string.

##### Comment

Sample(Note it’s tab instead of white spaces between url and file path) :

#http://www.baidu.com/ 1.txt

‘#’ can be used for comment if you put it at the beginning of a single rule. You can also add comments in this way

### Set response file

All response files need to be put under the response folder

#### single file

In Single file mapping mode, a single file need to contain both http header and body. The format is the raw data of HTTP. A simple way to get that is to use fiddler to check the raw data of http response. Sample：

HTTP/1.1 200 OK

Cache-Control: private

Content-Length: 15445

Content-Type: text/html; charset=utf-8

Expires: Tue, 10 Jun 2014 08:16:09 GMT

Vary: Accept-Encoding

Server: BWS/1.1 Microsoft-HTTPAPI/2.0

X-Powered-By: HPHP

Date: Tue, 10 Jun 2014 08:15:53 GMT

haha

The first row mainly contains http response code. A break line breaks the http header and body.

#### Two file

Like single file mode. The difference is the header and body are stored in separate files. This mode is more suitable for image response mock.

### Rule file and response file modification

#### Rule file：

Modification in rule.txt will take effect immediately after saving file.

#### response files

Modification in response files will take effect immediately after saving file.

### HTTPS support

Refer to Fiddler’s HTTPS documentation if you cannot get it work.

### limitation

Content-Length header and the HTTP code description cannot be tampered at this moment.

### work with Fiddler

When you need to use Fiddler to see the tampered response, please follow below steps:

1. Start Mock Server
2. Start Fiddler。Make sure the Fiddler option’s Connections->Act as system proxy on startup is unchecked.
3. Make sure your app uses Fiddler as the HTTP proxy
4. Make sure the Fiddler option’s Gateway->Manual Proxy Configuration is set to：http://localhost:9999。9999 is the default port of the Mock Server。

### Mock for unexist domain

Mock Server will start a HTTP server on port 80. You can change local hosts file to map 127.0.0.1 for unexist domain