OpenTSDB接口使用说明及在巡检中的应用

笔记本： OpenTSDB
创建时间： 2018/8/20 18:18 **更新时间：** 2018/8/21 19:45
作者： yycmessi@163.com
URL： http://opentsdb.net/docs/build/html/api_http/config/filters.html

一、简介

OpenTSDB提供基于HTTP的接口，以实现与外部系统的集成。 几乎所有OpenTSDB功能都可通过API访问，例如查询时间序列数据，管理元数据和存储数据点。 在使用各个接口之前，需要阅读相关API文档。

目前OpenTSDB已经与很多开源的监控系统集成，这种情况下，对OpenTSDB的接口就相对简单些，只需要掌握一些相对固定的查询接口就可以实现来巡检、自定制可视化界面的功能。

但如果是自行开发可视化界面和监控报警功能，就需要深入掌握OpenTSDB接口的请求方法和返回内容了。

我的分享内容：

1）OpenTSDB接口概览
2）OpenTSDB常用接口详解
3）利用OpenTSDB实现巡检功能

二、OpenTSDB接口概览

OpenTSDB默认使用json序列号来显示请求和响应。

OpenTSDB目前无身份验证和访问控制系统，所以只能通过防火墙等方式来限制访问。

和常规的http标准响应一样，OpenTSDB的接口返回也会有响应代码200、204、301等，以及400等错误代码。如果是错误，返回内容会包括code、message、details、trace等信息用以显示错误描述。

通常情况下，OpenTSDB的接口是使用GET查询、POST更新及创建、PUT替换、DELETE删除，但为了方便开发，OpenTSDB的部分API也支持了通过GET、POST来完成PUT、DELETE操作，具体在各个API中会介绍。一般查询操作是通过GET字符串请求来查，但由于编码的复杂性，OpenTSDB的接口还支持POST+body content的形式进行访问。

OpenTSDB的接口内容包括：

- /s **GET、POST、PUT、DELETE** 访问本地系统上的静态文件，响应将是所请求文件的内容，并配置了适当的HTTP标头

```
http://localhost:4242/s/queryui.nocache.js
```

- /api/aggregators **GET、POST** 列出时间序列查询中使用的已实现聚合函数的名称

```
http://localhost:4242/api/aggregators
返回:
["min","sum","max","avg","dev"]
```

- /api/annotation **GET、POST、PUT、DELETE** 提供了添加，编辑或删除存储在OpenTSDB后端中的注释的方法

请求内容：

Name	Data Type	Required	Description	Default	QS	RW	Example
startTime	Integer	Required	Unix时间戳，以秒为单位，标记应记录注释事件的时间		start_time	RW	1369141261

endTime	Integer	Optional	事件的可选结束时间（如果已完成或已解决）	0	end_time	RW	1369141262
tsuid	String	Optional	如果注释与时间序列关联，则为TSUID。如果注释是针对全局事件，则此值可以为null或为空		tsuid	RW	000001000001000001
description	String	Optional	该事件的简要说明。由于这可能出现在GnuPlot图表上，描述应该非常短，理想情况下应少于25个字符。		description	RW	Network Outage
notes	String	Optional	关于该事件的详细说明		notes	RW	Switch #5 died and was replaced
custom	Map	Optional	用于存储自定义字段和值的键/值映射	null		RW	*See Below*

```
http://localhost:4242/api/annotation?start_time=1369141261&tsuid=000001000001000001
{
"startTime":"1369141261",
"tsuid":"000001000001000001",
"description": "Testing Annotations",
"notes": "These would be details about the event, the description is just a summary",
"custom": {
"owner": "jdoe",
"dept": "ops"
}}
返回:
{
"tsuid": "000001000001000001",
"description": "Testing Annotations",
"notes": "These would be details about the event, the description is just a summary",
"custom": {
"owner": "jdoe",
"dept": "ops"
},
"endTime": 0,
"startTime": 1369141261}
```

- /api/config **GET、POST** 返回有关TSD运行配置的信息。它是只读的，不能用于设置配置选项

  /api/config/filters 用法相似， 列出了TSD加载的各种过滤器以及有关如何使用它们的一些信息

```
http://localhost:4242/api/config
{
"tsd.search.elasticsearch.tsmeta_type": "tsmetadata",
"tsd.storage.flush_interval": "1000",
"tsd.network.tcp_no_delay": "true",
...省略
}
```

- /api/dropcaches **GET、POST** 清除OpenTSDB中缓存的内存数据。包括UID和名称的映射，映射内容包括metrics, tag names 及tag values

Note

此接口不会清除存储图形和其他文件的磁盘上临时缓存。

```
http://localhost:4242/api/dropcaches
{
"message": "Caches dropped",
"status": "200"
}
```

- /api/put **POST** 通过HTTP在OpenTSDB中存储数据，以替代Telnet接口。

为了节省带宽，该接口允许客户端在单个请求中存储多个数据点。数据点不必以任何方式相关。每个数据点都是单独处理的，一个数据的错误不会影响其他数据的存储。这意味着如果您的请求有100个数据点，其中1个有错误，则仍会写入99个数据点，其中一个将被拒绝。

API在处理完每个数据点之前不会返回。这意味着必须验证metric和tag/tag value，解析数据并将解析过的数据排队等待存储。如果put请求包含大量数据点，则API可能需要很长时间才能响应，特别是如果OpenTSDB必须将UID分配给tag名称或值。因此，限制每个请求的最大数据点数是个保证效率的好方法， 每个请求50个是一个相对较合适的配置。

Note

如果无法解析您提供的请求内容，此类JSON内容缺少引号或大括号，则将丢弃所有数据点。API将返回错误，其中包含有关错误的详细信息。

Note

如果`tsd.mode`设置为`ro`，则`/api/put`端点将不可用，并且所有呼叫都将返回404错误。

请求的body内容中添加以下内容来更改请求的响应：

Name	Data Type	Required	Description	Default	QS	RW	Example
summary	Present	Optional	是否返回摘要信息	false	summary		/api/put?summary
details	Present	Optional	是否返回详细信息	false	details		/api/put?details
sync	Boolean	Optional	是否在返回结果之前等待数据刷新到存储。	false	sync		/api/put?sync
sync_timeout	Integer	Optional	在返回错误之前等待数据刷新到存储的超时（以毫秒为单位）。发生超时时，使用该`details`标志将告知有多少数据点失败	0	sync_timeout		/api/put/?sync&sync_timeout=60000

			以及有多少数据点成功。`sync`还必须为此生效。值为0表示写入不会超时。				

请求的数据点内容：

Name	Data Type	Required	Description	Default	QS	RW	Example
metric	String	Required	存储的指标的名称			W	sys.cpu.nice
timestamp	Integer	Required	Unix时间戳，以秒或毫秒为单位。时间戳不得包含非数字字符。			W	1365465600
value	Integer, Float, String	Required	要记录此数据点的值。它可以引用或不引用，并且必须符合OpenTSDB值规则：../../user_guide / writing			W	42.5
tags	Map	Required	tag名称/tag值对的映射。必须至少提供一对。			W	{"host":"web01"}

单个数据点请求：

```
{
"metric": "sys.cpu.nice",
"timestamp": 1346846400,
"value": 18,
"tags": {
"host": "web01",
"dc": "lga"
}}
```

多个数据点请求：

```
[
{
...略
},
{
...略
}]
```

响应：

```
{
"failed": 1,
"success": 0}
```

明细响应：

```
{
"errors": [
{
"datapoint": {
"metric": "sys.cpu.nice",
"timestamp": 1365465600,
"value": "NaN",
"tags": {
"host": "web01"
}
},
"error": "Unable to parse value to a number"
}
],
"failed": 1,
"success": 0}
```

- /api/rollup **POST** 通过HTTP在OpenTSDB中存储汇总（ rollups ）和/或预聚合数据。

汇总和预聚合值是/api/put接口的扩展，带有三个附加字段。下面列出了所有字段：

Name	Data Type	Required	Description	Default	QS	RW	Example
metric	String	Required	存储的指标的名称			W	sys.cpu.nice
timestamp	Integer	Required	Unix时间戳，以秒或毫秒为单位。时间戳不得包含非数字字符。			W	1365465600
value	Integer, Float, String	Required	要记录此数据点的值。它可以引用或不引用，并且必须符合OpenTSDB值规则：../../ user_guide / writing			W	42.5
tags	Map	Required	标签名称/标签值对的映射。必须至少提供一对。			W	{"host":"web01"}
interval	String	Optional*	反映**汇总**值代表的时间跨度的时间间隔。间隔包括`<amount>` `<unit>`类似于下采样器或相对查询时间戳。例如，`5h`对于5小时的数据，`30m`对于30分钟的数据。			W	1h
aggregator	String	Optional*	用于生成**汇总**值的聚合函数。必须与提供的TSDB聚合器匹配。			W	SUM
groupByAggregator	String	Optional*	用于生成**预聚合**值**的聚合**函数。必须与提供的TSDB聚合器匹配。		W	COUNT	

```
单个数据点
{
"metric": "sys.cpu.nice",
"timestamp": 1346846400,
"value": 18,
"tags": {
"host": "web01",
"dc": "lga"
},
"interval": "1h",
"aggregator": "SUM",
```

```
"groupByAggregator": "SUM"}
多个数据点
[
{
...略
},
{
...略
}]
```
响应与/api/put相同

- /api/histogram **POST** 通过HTTP在OpenTSDB中存储直方图数据

与put接口一样，该接口也可以一次发送多个数据点。除了发送的body字段略有不同，其他的都与put接口类似。body内容如下：

Name	Data Type	Required	Description	Default	QS	RW	Example
metric	String	Required	指标的名称			W	sys.cpu.nice
timestamp	Integer	Required	Unix时间戳，以秒或毫秒为单位。时间戳不得包含非数字字符。			W	1365465600
id	Integer	Optional	在编写默认简单分块直方图以外的直方图或草图时，必须将此值设置为`tsd.core.histograms.config`配置设置中定义的正确直方图编解码器的ID 。该值必须介于0到255之间。给定时，`value`必须设置。			w	1
value	String	Optional	基础64编码要存储的直方图或草图的二进制数据。**注意**在写入二进制数据时，还必须提供ID以匹配正确的编解码器。			W	AgMIGo=
buckets	Map	Optional	存储桶下限和上限（以逗号分隔）的映射作为具有整数计数器存储桶值的键。详情如下。			W	{"0,1.75":12,"1.75,3.5":16}
underflow	Integer	Optional	测量计数低于最低桶下限。默认值为零。			W	0
overflow	Integer	Optional	测量计数高于最高桶上限。默认值为零。			W	0
tags	Map	Required	标签名称/标签值对的映射。必须至少提供一对。			W	{"host":"web01"}

单个数据点
```
{
"metric": "sys.cpu.nice",
"timestamp": 1356998400,
```

```
"overflow": 1,
"underflow": 0,
"buckets": {
"0,1.75": 12,
"1.75,3.5": 16
},
"tags": {
"host": "web01",
"dc": "lga"
}}
```


多个数据点


```
[
{
...略
},
{
...略
}]
```


响应与/api/put相同

- /api/query **GET、POST、DELETE** 是API中最有用的接口，`/api/query`能够以确定的格式从存储系统中序列化地提取数据。

请求的接口内容：

- /api/query/exp
- /api/query/gexp
- /api/query/last

可以使用DELETE动词删除查询的数据。`tsd.http.query.allow_delete`必须启用配置参数才能允许删除。删除的数据将在查询结果中返回。第二次执行查询应该返回空结果。

Warning

删除数据是永久性的。还要注意，删除时，可能会删除开始和结束时间边界之外的某些数据，因为数据是按小时存储的。

请求字段：

Name	Data Type	Required	Description	Default	QS	RW	Example
start	String, Integer	Required	查询的开始时间。这可以是相对或绝对时间戳。有关详细信息，请参阅查询或读取数据。		start		1h-ago
end	String, Integer	Optional	查询的结束时间。如果未提供，TSD将假定服务器上的本地系统时间。这可以是相对或绝对时间戳。有关详细信息，请参阅查询或读取数据。	*current time*	end		1s-ago
queries	Array	Required	用于选择要返回的时间序列的一个或多个子查询。这些		m or tsuids		*See below*

			可以是度量`m`或TSUID `tsuids`查询				
noAnnotations	Boolean	Optional	是否返回带有查询的注释。默认设置是返回请求的时间跨度的注释，但此标志可以禁用返回。这会影响本地和全局注释和覆盖`globalAnnotations`	false	no_annotations		false
globalAnnotations	Boolean	Optional	查询是否应检索所请求的时间跨度的全局注释	false	global_annotations		true
msResolution (or ms)	Boolean	Optional	是否以毫秒或秒为单位输出数据点时间戳。建议使用msResolution标志。如果未提供此标志且一秒内有多个数据点，则将使用查询的聚合函数对这些数据点进行下采样。	false	ms		true
showTSUIDs	Boolean	Optional	是否在结果中输出与时间序列关联的TSUID。如果将多个时间序列聚合到一个集合中，则将以排序的方式返回多个TSUID	false	show_tsuids		true
showSummary *(2.2)*	Boolean	Optional	是否在结果中显示查询周围的计时摘要。这将在地图中创建另一个与数据点对象不同的对象。请参阅查询详细信息和统计信息	false	show_summary		true
showStats *(2.2)*	Boolean	Optional	是否在结果中显示有关查询的详细计时。这将在地图中创建另一个与数据点对象不同的对象。请参阅查询详细信息和统计信息	false	show_stats		true
showQuery *(2.2)*	Boolean	Optional	是否使用查询结果返回原始子查询。如果请求包含许多	false	show_query		true

			子查询，那么这是确定哪些结果属于哪个子查询的好方法。请注意，在*通配符或通配符查询的情况下，这会产生大量重复输出。				
delete	Boolean	Optional	可以使用POST传递给JSON以删除与给定查询匹配的任何数据点。	false		W	true
timezone *(2.3)*	String	Optional	基于日历的下采样的可选时区。必须是TSD服务器上安装的JRE支持的有效时区数据库名称。	UTC	timezone		Asia/Kabul
useCalendar *(2.3)*	Boolean	Optional	是否使用基于给定时区的日历来进行下采样间隔	false			true

查询至少需要一个子查询，这是一种选择应该在结果集中包含哪些时间序列的方法。有两种类型：

- Metric Query - 提供metric的全名以及可选的tag列表。这被优化用于将多个时间序列聚合成一个结果。
- TSUID Query - 共享公共metric的一个或多个TSUID的列表。这针对获取不需要聚合的单个时间序列进行了优化。

查询可以包括多个子查询以及两种类型的任何混合。通过body content提交查询时，如果提供了TSUID列表，则将忽略该特定子查询的度量标准和标记。

每个子查询可以检索单个或一组时间序列数据，对每个集合执行聚合或分组计算。每个子查询的字段包括：

Name	Data Type	Required	Description	Default	Example
aggregator	String	Required	要使用的聚合函数的名称。请参见/ api / aggregators		sum
metric	String	Required	存储在系统中的metric的名称		sys.cpu.0
rate	Boolean	Optional	在返回之前是否应将数据转换为增量。如果度量标准是一个连续递增的计数器，并且您想要查看数据点之间的变化率，这将非常有用。	false	true
rateOptions	Map	Optional	单调增加处理选项	*See below*	*See below*
downsample	String	Optional	可选的下采样功能，用于减少返回的数据量。	*See below*	5m-avg
tags	Map	Optional	要深入查看特定时间序列或按标记对结果进行分组，请提供与查询字符串格式相同的一个或多个映射值。标签在2.2中转换为过滤器。请参阅以下有关转化的说明。请注意，如果未指定任何标记，系统中的所有指标都将汇总到结果中。*在2.2中弃用*		*See Below*

filters *(2.2)*	List	Optional	发出结果中过滤时间序列。请注意，如果未指定过滤器，则给定度量标准的所有时间序列都将聚合到结果中。		*See Below*
explicitTags *(2.3)*	Boolean	Optional	仅返回包含过滤器中提供的tag的序列	false	true
percentiles *(2.4)*	List	Optional	获取度量的直方图数据，并计算数据上给定的百分位数列表。百分位数是从0到100的浮点值。更多详细信息如下。		[99.9, 95.0, 75.0]

在查询字符串中选择速率选项时，必须将选项括在花括号中。例如：`m=sum:rate{counter,,1000}:if.octets.in`。如果您希望使用默认值`counterMax`但希望提供一个`resetValue`，则必须添加两个逗号，如上例所示。`rateOptions`对象中的其他字段包括以下内容：

Name	Data Type	Required	Description	Default	Example
counter	Boolean	Optional	基础数据是否是可以翻转的单调递增计数器	false	true
counterMax	Integer	Optional	一个正整数，表示计数器的最大值。	Java Long.MaxValue	65535
resetValue	Integer	Optional	一个可选值，当超出该值时，将导致聚合器返回`0`而不是计算的速率。经常重置数据源以避免虚假尖峰时很有用。	0	65000
dropResets	Boolean	Optional	是否简单地丢弃翻转或重置数据点。	false	true

下采样的示例

```
<interval><units>-<aggregator>[c][-<fill policy>]
1h-sum30m-avg-nan24h-max-zero1dc-sum0all-sum
```

fileters过滤器使用

有关TSD中加载的过滤器列表，请参阅/ api / config / filters。

过滤器可用于url请求和POST格式的查询。允许在同一标记键上使用多个过滤器，并且在处理时，它们将*进行AND运算*，例如，如果我们有两个过滤器`host=literal_or(web01)`并且`host=literal_or(web02)`查询将始终返回空。如果同一个标记键包含两个或更多个过滤器，而另一个没有，则另一个不启用，则对于该标记键上的所有过滤器，group by实际上将为true。与过滤器相关的POST查询字段包括：

Name	Data Type	Required	Description	Default	Example
type	String	Required	要调用的过滤器的名称。请参见 /api/config/filters		regexp
tagk	String	Required	过滤的tag		host
filter	String	Required	过滤器表达式，取决于所使用的过滤器		web.*.mysite.com
groupBy	Boolean	Optional	是否按过滤器匹配的每个值对结果进行分组。默认情况下，与筛选器匹配的所有值都将聚合到一个系列中。	false	true

POST查询`tags`映射中的值和*按*URI大括号查询的*组*将自动转换为过滤器，以提供与现有系统的向后兼容性。自动转换包括：

Example	Description

`<tagk>=*`	Wildcard filter, effectively makes sure the tag key is present in the series 通配符过滤器，有效地确保标记键存在于系列中
`<tagk>=value`	区分大小写的字符OR过滤器
`<tagk>=value1|value2|valueN`	区分大小写的字符OR过滤器
`<tagk>=va*`	不区分大小写的通配符过滤器。带有任何其他字符串的星号（星号）现在变为通配符过滤器快捷方式

query with filters

```
{
"start": 1356998400,
"end": 1356998460,
"queries": [
{
"aggregator": "sum",
"metric": "sys.cpu.0",
"rate": "true",
"filters": [
{
"type":"wildcard",
"tagk":"host",
"filter":"*",
"groupBy":true
},
{
"type":"literal_or",
"tagk":"dc",
"filter":"lga|lga1|lga2",
"groupBy":false
}
]
},
{
"aggregator": "sum",
"tsuids": [
"000001000002000042",
"000001000002000043"
]
}
]}
```

响应字段

Name	Description
metric	查询的metric名称
tags	仅当结果针对单个时间系列时，才会返回标记列表。如果聚合了结果，则此值可能为null或空映射
aggregatedTags	如果结果集中包含多个时间序列，即它们已聚合，则会显示在所有时间序列中共同找到的标记名称列表。
dps	聚合器处理后检索的数据点。每个数据点由时间戳和值组成，格式由序列化程序确定。
annotations	如果查询在请求的时间跨度内检索了时间序列的注释，则它们将在此组中返回。每个时间序列的注释将合并为一个集合并按其排序`start_time`。聚合器函数不会影响注释，将为跨度返回所有注释。
globalAnnotations	如果用户请求，查询将在时间跨度内扫描全局注释，并在此组中返回结果

- /api/search **GET、POST** 此接口提供了搜索OpenTSDB元数据的基本方法

`tsdb-meta`启用后，可以对表执行查找。可选地，可以安装搜索插件以从诸如弹性搜索的外部搜索索引服务发送和检索信息。每个搜索插件都可以实现此接口的各个部分，并以一致的格式返回数据。搜索和返回的对象类型取决于所选的接口。

Note

如果未配置或启用插件，则除`api/search/lookup`以外的接口将返回异常。

搜索的API

- /api/search/lookup
- /api/search/tsmeta - TSMETA Response
- /api/search/tsmeta_summary - TSMETA_SUMMARY Response
- /api/search/tsuids - TSUIDS Response
- /api/search/uidmeta - UIDMETA Response
- /api/search/annotation - Annotation Response

请求参数：

Name	Data Type	Required	Description	Default	QS	RW	Example
query	String	Optional	基于请求的查询传递给搜索引擎。这将由引擎或插件解析以执行实际搜索。允许值取决于插件。忽略lookup。		query		name:sys.cpu.*
limit	Integer	Optional	限制每个查询返回的结果数，以便不覆盖TSD或搜索引擎。允许值取决于插件。忽略lookup。	25	limit		100
startIndex	Integer	Optional	与`limit`值结合使用以分页结果。允许值取决于插件。忽略lookup。	0	start_index		42
metric	String	Optional	用于lookup查询的度量标准或通配符的名称	*	metric		tsd.hbase.rpcs
tags	Array	Optional	一个或多个键/值对象，具有用于lookup查询的标记名称和/或标记值。请参见 /api/search/lookup		tags		See /api/search/lookup

响应字段：

Name	Data Type	Description	Example
type	String	提交的查询类型，即调用的接口。将是上面列出的接口之一。	TSMETA
query	String	提交的查询请求。可能会被插件更改	name:sys.cpu.*

limit	Integer	结果集中返回的最大项目数。请注意，返回的实际数字可能小于限制。	25
startIndex	Integer	查询中提供的当前结果集的起始索引	0
metric	String	用于lookup的metric	
tags	Array	用于lookup查询的标记对列表。可能是一个空列表。	[ ]
time	Integer	完成查询所花费的时间（以毫秒为单位）	120
totalResults	Integer	查询匹配的结果总数	1024
results	Array	结果集。格式取决于请求的接口。	*See Below*

请求示例：
字符串形式：http://localhost:4242/api/search/tsmeta?query=name:*&limit=3&start_index=0
data形式：

```
{
"query": "name:*",
"limit": 4,
"startIndex": 5}
```

响应：


```
{
"type": "TSMETA",
"query": "name:*",
"metric": "*",
"tags": [],
"limit": 2,
"time": 675,
"results": [
{
"tsuid": "0000150000070010D0",
"metric": {
"uid": "000015",
"type": "METRIC",
"name": "app.apache.connections",
"description": "",
"notes": "",
"created": 1362655264,
"custom": null,
"displayName": ""
},
"tags": [
{
"uid": "000007",
"type": "TAGK",
"name": "fqdn",
"description": "",
"notes": "",
"created": 1362655264,
"custom": null,
"displayName": ""
},
{
"uid": "0010D0",
"type": "TAGV",
"name": "web01.mysite.com",
"description": "",
"notes": "",
"created": 1362720007,
"custom": null,
"displayName": ""
}
],
```

```
"description": "",
"notes": "",
"created": 1362740528,
"units": "",
"retention": 0,
"max": 0,
"min": 0,
"displayName": "",
"dataType": "",
"lastReceived": 0,
"totalDatapoints": 0
},
{
"tsuid": "0000150000070010D5",
"metric": {
"uid": "000015",
"type": "METRIC",
"name": "app.apache.connections",
"description": "",
"notes": "",
"created": 1362655264,
"custom": null,
"displayName": ""
},
"tags": [
{
"uid": "000007",
"type": "TAGK",
"name": "fqdn",
"description": "",
"notes": "",
"created": 1362655264,
"custom": null,
"displayName": ""
},
{
"uid": "0010D5",
"type": "TAGV",
"name": "web02.mysite.com",
"description": "",
"notes": "",
"created": 1362720007,
"custom": null,
"displayName": ""
}
],
"description": "",
"notes": "",
"created": 1362882263,
"units": "",
"retention": 0,
"max": 0,
"min": 0,
"displayName": "",
"dataType": "",
"lastReceived": 0,
"totalDatapoints": 0
}
],
"startIndex": 0,
"totalResults": 9688066}
```


其他几个接口在此就不进行详细的介绍了。

- /api/serializers **GET、POST** 列出了正在运行的TSD加载的序列化程序插件。给出的信息包括名称，实现的方法，内容类型和方法。

响应的字段：

Field Name	Data Type	Description	Example
serializer	String	序列化程序的名称，适用于查询字符串`serializer=<serializer_name>`参数	xml
formatters	Array<String>	序列化程序实现的一组方法或端点，用于转换响应数据。这些通常映射到端点，例如`/api/suggest`映射到`Suggest`。如果序列化程序未实现某种方法，则默认格式化程序将响应。每个名称也以支持的API版本结束，例如，`V1`将支持版本1 API调用。	"Error","Suggest"
parsers	Array<String>	序列化程序实现的一组方法或端点，用于解析HTTP请求正文中的用户输入。这些通常映射到端点，例如/api/suggest将映射到Suggest。如果序列化程序未实现解析器，则将使用默认序列化程序。每个名称也以支持的API版本结束，例如，V1将支持版本1 API调用。	"Suggest","Put"

```
响应：
[
{
"formatters": [
"SuggestV1",
"SerializersV1",
"ErrorV1",
"ErrorV1",
"NotFoundV1"
],
"serializer": "json",
"parsers": [
"SuggestV1"
],
"class": "net.opentsdb.tsd.HttpJsonSerializer",
"response_content_type": "application/json; charset=UTF-8",
"request_content_type": "application/json"
}]
```

- /api/stats **GET、POST** 查看正在运行的TSD的统计信息列表。子接口返回有关其他TSD组件的详细信息，例如JVM，线程状态或存储客户端。所有统计数据都是只读的。
- /api/stats/jvm
- /api/stats/query
- /api/stats/region_clients
- /api/stats/threads

请求：
http://localhost:4242/api/stats

```
响应：
[
{
"metric": "tsd.connectionmgr.connections",
"timestamp": 1369350222,
"value": "1",
"tags": {
"host": "wtdb-1-4"
}
},
{
"metric": "tsd.connectionmgr.exceptions",
"timestamp": 1369350222,
```

```
"value": "0",
"tags": {
"host": "wtdb-1-4"
}
},
{
"metric": "tsd.rpc.received",
"timestamp": 1369350222,
"value": "0",
"tags": {
"host": "wtdb-1-4",
"type": "telnet"
}
}]
```


- /api/suggest **GET、POST** 提供了一种实现"自动完成"调用的方法，当用户在GUI中键入请求时，可以重复访问该调用。

它不提供全文搜索或通配符，而是简单地匹配查询中传递的整个请求，用于存储数据的第一个字符。例如，传递查询`type=metrics&q=sys`将返回系统中排名前25位的指标`sys`。匹配区分大小写，因此`sys`不匹配`System.CPU`。结果按字母顺序排序。

响应字段：

Name	Data Type	Required	Description	Default	QS	RW	Example
type	String	Required	要自动完成的数据类型。必须是以下之一：metrics，tagk或tagv		type		metrics
q	String	Optional	要匹配给定类型的字符串		q		web
max	Integer	Optional	要返回的最大建议结果数。必须大于0	25	max		10


```
请求:
http://localhost:4242/api/suggest?type=metrics&q=sys&max=10
{
"type":"metrics",
"q":"sys",
"max":10}
响应:
[
"sys.cpu.0.nice",
"sys.cpu.0.system",
"sys.cpu.0.user",
"sys.cpu.1.nice",
"sys.cpu.1.system",
"sys.cpu.1.user"]
```


- /api/tree **GET、POST、PUT、DELETE** 用于在分层结构中组织时间序列的元数据，用于类似于典型文件系统的浏览，提供了多个子接口来进行操作
  - /api/tree/branch
  - /api/tree/collisions
  - /api/tree/notmatched
  - /api/tree/rule
  - /api/tree/rules
  - /api/tree/test

该/tree接口允许创建或修改树的定义。树定义包括通过此接口可访问的配置和元数据，以及可通过/tree/rule或访问的规则集/tree/rules。

请求字段：

Name	Data Type	Required	Description	Default	QS	RW	Example
treeId	Integer	Required*	用于获取或修改特定树。*创建新树时，tree不得出现该值。		treeid	RO	1
name	String	Required*	树的简短描述性名称。*仅在创建树时需要。		name	RW	Network Infrastructure
description	String	Optional	对树的更长的描述		description	RW	Tree containing all network gear
notes	String	Optional	关于树的详细说明		notes	RW	
strictMatch	Boolean	Optional	如果时间序列不符合一个或多个规则级别，则是否应将其包括在树中。	false	strict_match	RW	true
enabled	Boolean	Optional	是否应该通过树处理TSMeta。默认设置为，false以便您可以在构建分支之前设置规则并测试某些对象。	false	enabled	RW	true
storeFailures	Boolean	Optional	是否应记录碰撞和“不匹配”的TSUID。这可以创建非常宽的行。	false	store_failures	RW	true
definition	Boolean	Optional	仅在DELETE使用树时使用，如果此标志设置为true，则将删除整个树定义以及所有分支，冲突和未匹配的条目	false	definition		true

响应字段：

Name	Data Type	Description	Example
rules	Map	为树组织的规则的地图或字典，由level和组织order。如果尚未定义规则，则值为null	*See Examples*
created	Integer	最初创建树时的Unix时间戳（以秒为单位）。	1350425579

GET 方法 如果没有树ID将返回所有在系统中配置的树列表。结果将包括每个树的配置规则。如果尚未配置树，则列表将为空。

POST、PUT 方法 使用POST或PUT方法，您可以创建新树或编辑现有树的大多数字段。新树需要name值以及ID和需要修改的任何字段。成功的请求将返回修改后的树对象。

DELETE 方法 使用该`DELETE`方法将仅删除存储中给定树的冲突，而不是匹配的条目和分支。此接口启动删除。由于删除可能需要一些时间，因此如果删除完成，端点将返回成功的204响应而不包含数据。如果找不到树，它将返回404.如果要删除树定义本身，可以`defintion`在查询字符串中提供值为的标志，`true`并且树和规则定义也将被删除。

Warning

此方法无法撤消。执行后，除非TSD关闭，否则清除将继续运行。

- /api/uid 此接口提供用于管理UID及其关联数据的实用程序，提供了3个子接口
  - /api/uid/assign
  - /api/uid/tsmeta
  - /api/uid/uidmeta

- /api/version **GET、POST** 此接口用于返回OpenTSDB运行版本的信息。

```
请求：
http://localhost:4242/api/version
响应：
{
"timestamp": "1362712695",
"host": "localhost",
"repo": "/opt/opentsdb/build",
"full_revision": "11c5eefd79f0c800b703ebd29c10e7f924c01572",
"short_revision": "11c5eef",
"user": "localuser",
"repo_status": "MODIFIED",
"version": "2.0.0"
}
```

三、常用接口使用

192.168.180.128:16002是opentsdb的地址

```
*1、查询所有的metric*

    http://192.168.180.128:16002/api/suggest?max=1000&q=&type=metrics
    返回
    ["test.delete","test.delete2"]

*2、查询全量tag*

    http://192.168.180.128:16002/api/suggest?max=1000&q=&type=tagk
    ["host","idc"]

*3、查询全量tagv*

    http://192.168.180.128:16002/api/suggest?max=1000&q=&type=tagv
    ["0.0.0.1","0.0.0.2","test1"]

*4、指定metric，查全量tagk*

    http://192.168.180.128:16002/api/search/lookup?limit=1000&m=test.delete

    得到的结果是这个metric下所有的tag、tagv，需要处理之后才会得到全量tagk

    {"type":"LOOKUP","metric":"test.delete","tags":[],"limit":1000,"time":14.0,"results":
[{"tsuid":"C66AC10000010000010000002000002","metric":"test.delete","tags":
{"host":"0.0.0.1","idc":"test1"}}],"startIndex":0,"totalResults":1}

*5、指定metric，指定tagk，查全量tagv*

    http://192.168.180.128:16002/api/search/lookup?limit=3000&m=test.delete{idc=*}
```

```
    {"type":"LOOKUP","metric":"test.delete","tags":
[{"key":"idc","value":"*"}],"limit":3000,"time":13.0,"results":
[{"tsuid":"C66AC10000010000010000002000002","metric":"test.delete","tags":
{"host":"0.0.0.1","idc":"test1"}}],"startIndex":0,"totalResults":1}

*6、指定metric，限定一个tagk=takv，查另外一个tagk下的所有tagv*

    (如：test.delete，查指定idc=test1条件下,host的所有值)
    http://192.168.180.128:16002/api/search/lookup?limit=3000&m=test.delete%7Bhost%3D*,idc%3Dtest1%7D

    {"type":"LOOKUP","metric":"test.delete","tags":[{"key":"host","value":"*"},
{"key":"idc","value":"test1"}],"limit":3000,"time":4.0,"results":
[{"tsuid":"C66AC10000010000010000002000002","metric":"test.delete","tags":
{"host":"0.0.0.1","idc":"test1"}}],"startIndex":0,"totalResults":1}
```

四、利用OpenTSDB进行开发巡检程序

功能1，巡检cmdb中没有添加监控的主机：
　通过/api/search/lookup?limit=3000&m=monitor.alive{endpoint=*}接口查所有监控主机，与cmdb中的主机进行比对，得出未添加监控的主机列表。
功能2，巡检cmdb中没有添加监控的数据库实例
　通过/api/search/lookup?limit=3000&m=mysql.alive{endpoint=*}接口查询所有监控主机及数据库端口号，与cmdb中的数据库实例进行对比，得出未添加监控的实例列表。
功能3，巡检指定时间内指定metric无数据上报的主机：

```
  #使用 /api/query接口
  #ipList为遍历的ip列表
   tsdbQuery = {
      'start': starttime,
      'end': endtime,
      'queries': [{
         'aggregator':'avg',
         'downsample':'1h-avg',
         'metric': ‘monitor.alive’,
         'rate': 'false',
         'filters': [{
            'type': 'literal_or',
            'tagk': 'endpoint',
            'filter': "|".join(ipList),
            'groupBy': 'true'
         }]
      }]
   }
取出无数据的endpoint，这些endpoint则为采集数据失败的主机
```

功能4，巡检cpu、磁盘、内存全天平均使用率top 10的主机列表：
　同功能3的接口，只是metric改为对于cpu、磁盘、内存的metric即可。

功能n，使用/api/query出来的监控数据可以做各种应用，具体看需求。