ウェブフック導入ガイド

ウェブフックの概要 2

表記法 2

機能の概要 2

設定手順 3

サブスクリプションの設定 7

Yang モデル 9

ウェブフック ステータスの確認 9

Logstash および Python Flask 用の SSL 証明書の生成 10

Python Flask で使用する証明書を生成するには、次の追加コマンドを実行 10

Python を使用した Linux ウェブフック サーバ 12

Elastic（ELK）スタックの Kibana を使用したウェブフック データの表示 14

まとめ 19

改訂：2018 年 6 月 20 日

ウェブフックの概要

このドキュメントでは、ウェブフックの機能とその導入についての一般的なガイドラインを示します。このドキュメントの目的は以下のとおりです。

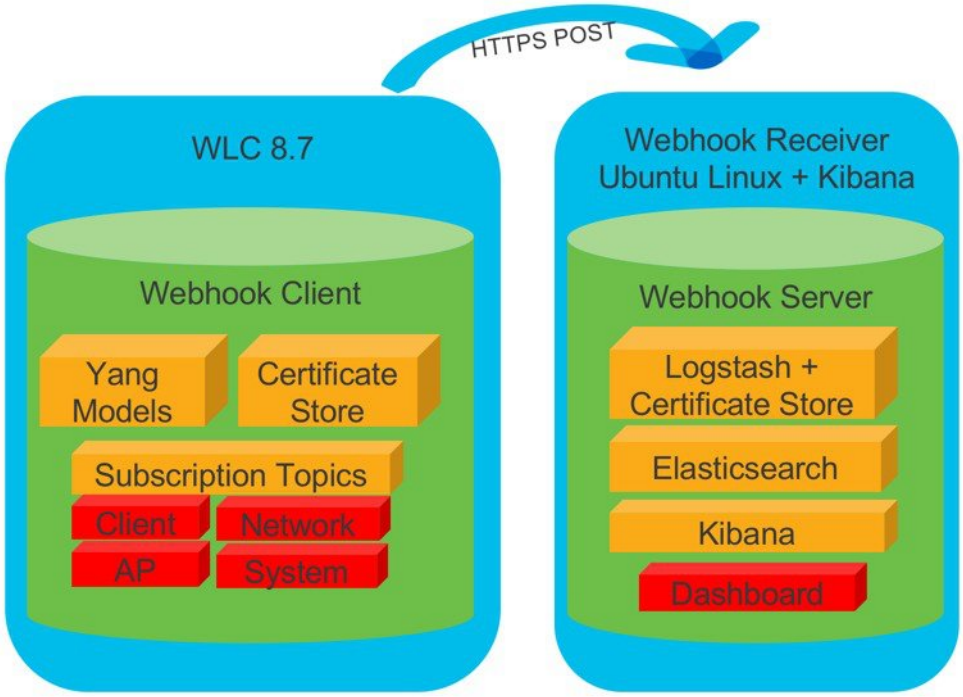
- ウェブフックの概要説明
- サポートされている主要機能のハイライト
- ウェブフック設定の導入および管理に関する詳細情報の提供
- ウェブフックがエクスポートする raw データの確認と表示に関するガイダンスの提供

表記法

ドキュメント表記の詳細は、『Cisco Technical Tips Conventions』を参照してください。

機能の概要

ウェブフック クライアントは WLC 内で実行され、HTTPS 接続を介して情報を外部のサードパーティ サーバに公開します。この情報は「ペイロード」と呼ばれ、各ペイロードには、YANG モデルで定義された固有の情報セットが含まれています。ペイロードには、AP、クライアント、およびシステム レベルの情報を含む、WLC のステータスに関する運用データが含まれています。



設定手順

GUI または CLI を使用してウェブフックを設定できます。まず CLI を使用してデータ公開（DX; Data Externalization）を有効にする必要があります。有効になっていない場合は再起動する必要があります。次のコマンドを使用して DX の状況を確認して有効にします。

```
(Cisco Controller) > show dx summary
(Cisco Controller) > config dx enable
```

ウェブフック サーバに、IP アドレスの代わりに DNS 名を使用することができます。そのため、次のコマンドを使用して、ネットワーク設定で定義されている DNS サーバの IP アドレスが WLC に設定されていることを確認してください。

```
(Cisco Controller) > config network dns serverip 208.67.222.222
(Cisco Controller) > show network summary
DNS Server IP............................. 208.67.222.222
```

CLI によるウェブフック設定

WLC でウェブフック サーバを正常に設定するには、いくつかの CLI コマンドが必要です。すべてのコマンドは「**config telemetry webhook**」セクションにあります。関連して「**show telemetry webhook summary**」コマンドも使用します。唯一の例外である、データ公開と DNS に関しては上記に記載しています。

```
(WLC-5520) >config telemetry webhook ?

auth-token     Configures Network webhook Auth-token
disable        Disables the webhook Service
enable         Enables the Webhook Service
on-change      Configures Webhook Data onchange publishing
subscribe      Add channels to Webhook Subscription
sync-interval  Configues Webhook Data Publising sync interval
unsubscribe    Unsubscribe from the channel
url            Configures Network Webhook Url
```

HTTPS の考慮事項

WLC とサードパーティ サーバ間の接続は HTTPS で保護され、証明書が WLC の証明書ストアにインストールされている必要があります。**transfer download** コマンドを使用して、TFTP または HTTP サーバから証明書を転送することで対応できます。

次のコマンドを使用して **webhook-ca-cert** 転送を設定します。

```
(Cisco Controller) > transfer download datatype webhook-ca-cert
(Cisco Controller) > transfer download filename <CA Cert File Name>
(Cisco Controller) > transfer download mode tftp
(Cisco Controller) > transfer download path /
(Cisco Controller) > transfer download serverip <TFTP server IP>
(Cisco Controller) > transfer download start
```

```
(WLC-5520) >transfer download start

Mode..............................................  TFTP
Data Type........................................  Webhook CA cert
TFTP Server IP...................................  10.10.10.49
TFTP Packet Timeout..............................  6
TFTP Max Retries.................................  10
TFTP Path........................................  /
TFTP Filename....................................  webhook.crt

This may take some time.
Are you sure you want to start? (y/N)
```

転送が成功すると、WLC上のウェブフック クライアントはウェブフック サーバとのセキュアな接続を確立できます。証明書を使用するために再起動する必要はありませんが、ウェブフック サービスがすでに有効になっている場合は、「**config telemetry webhook disable**」コマンド実行後に「**config telemetry webhook enable**」コマンドを実行してサービスを再起動する必要があります。

「Show**show certificate all**」コマンドを入力して「webhook-ca cert」を見つけ、証明書がインストールされていることを確認します。

```
(WLC-5520) >show certificate all

---------------  Verification Certificates ---------------
Certificate Name: Webhook CA cert
```

URL 設定

WLCがウェブフック ペイロードデータを送信する先のHTTPS URLを設定します。IPまたはDNS名で指定でき、ポートの指定は任意です。「SSL証明書の生成」の項を参照し、必要に応じて、必要なCA証明書を生成してインストールします。

```
(Cisco Controller) > config telemetry webhook url https://<IP or DNS name>:<Port>
```

Auth-Token

この英数字の値を使用して、ウェブフック サーバがWLCのウェブフック クライアントを検証する場合があります。ウェブフック サーバがこの設定を使用するとは限りませんが、サーバが使用するかどうかにかかわらず、WLCに設定する必要があります。この値をWLCのホスト名に設定することを推奨します。

```
(Cisco Controller) > config telemetry webhook auth-token <ALPHANUMERIC VALUE>
```

On-Change

CLIでのOn-Change設定では、一定の間隔ですべてのペイロードを送信するか、差分または更新されたペイロードだけを送信するかを定義します。ウェブフック サーバ上でのデータ操作をシンプルにするために、最初はOn-Changeを無

効にしてすべてのペイロードを受信し、差分データセットを効果的に使用するためのロジックがサードパーティのウェブフック サーバに組み込まれてから、有効にすることを推奨します。

```
(Cisco Controller) > config telemetry webhook on-change disable / enable
```

Sync-Interval

CLI での Sync-Interval 設定では、ウェブフック クライアントがウェブフック サーバにデータをパブリッシュする間隔を定義します。[fixed] に設定すると、結果は 5 分ごとにパブリッシュされます。[adaptive] に設定すると、結果は 30 秒以内にパブリッシュされます。更新の受信頻度を上げるには、この値を [adaptive] に設定することをお勧めします。

```
(Cisco Controller) > config telemetry webhook sync-interval adaptive / fixed
```

サブスクリプション

サブスクリプションのトピックおよび CLI で有効にした関連の YANG モデルの詳細については、後述する「サブスクリプションの設定」の項を参照してください。トピックを個別に登録することも、すべて選択して登録することも可能です。

```
(Cisco Controller) > config telemetry webhook subscribe <all / topic>
```

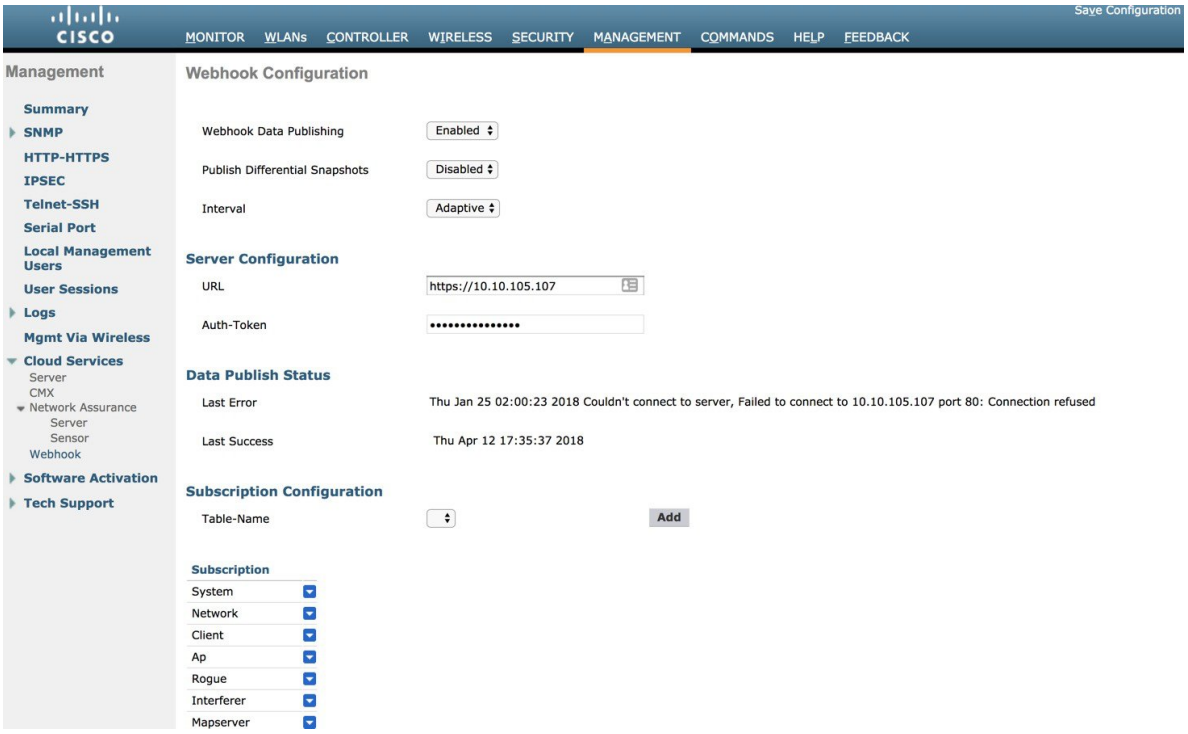
Enable または Disable

enable または disable コマンドを使用してウェブフックを有効または無効にします。

```
(Cisco Controller) > config telemetry webhook enable /disable
```

GUI によるウェブフック設定

GUI による設定画面には、[Management] タブを選択後、[Cloud Services]、[Webhook] の順に選択すればアクセスできます。URL、Auth-Token および HTTPS URL の設定の説明については、「CLI による設定」の項を参照してください。



ウェブフック データのパブリッシュ

ウェブフックを有効にするには、ドロップダウンを使用して [Enabled] を選択してから [Apply] を選択します。

ウェブフックを無効にするには、[Disabled] に変更します。

差分スナップショットのパブリッシュ

このオプションに対応する CLI コマンドは、「On-Change」です。[Enabled] に設定すると、差分スナップショットがパブリッシュされるため、変更または更新されたペイロードのみが送信されます。ウェブフック サーバ上でのデータ操作をシンプルにするために、On-Change を [Disabled] に設定して、一定間隔ですべてのペイロードを受信することを推奨します。

Interval

このオプションに対応する CLI コマンドは、「Sync-Interval」です。[fixed] に設定すると、結果は 5 分ごとにパブリッシュされます。[adaptive] に設定すると、結果は 30 秒ごとにパブリッシュされます。更新の受信頻度を上げるには、この値を [Adaptive] に設定することをお勧めします。

URL

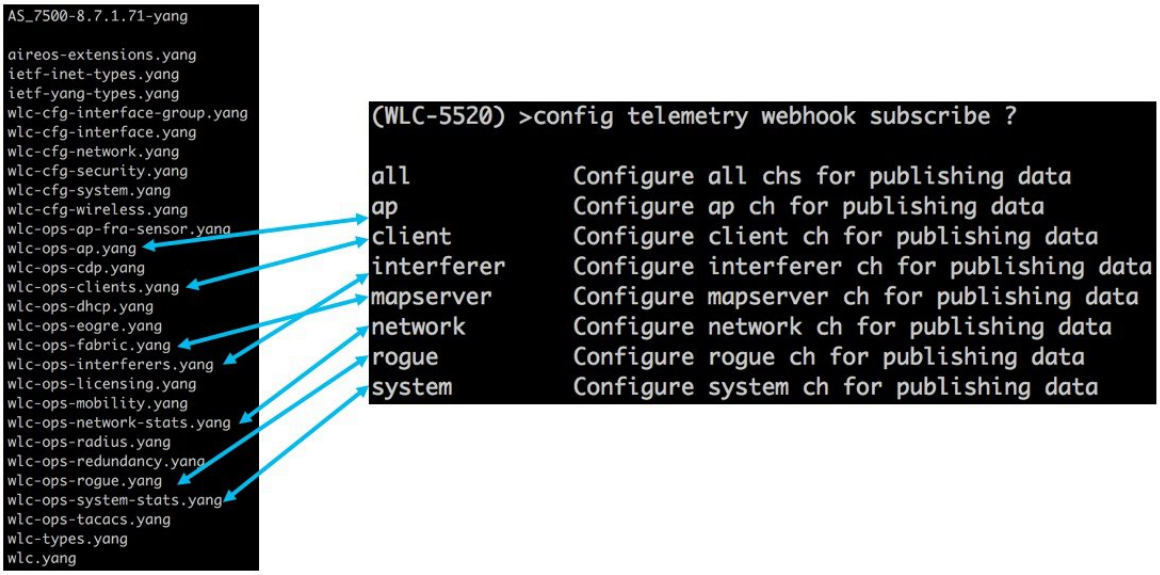
サードパーティのウェブフック サーバに対する HTTPS URL を設定して、結果をパブリッシュする方法の詳細については、上記を参照してください。

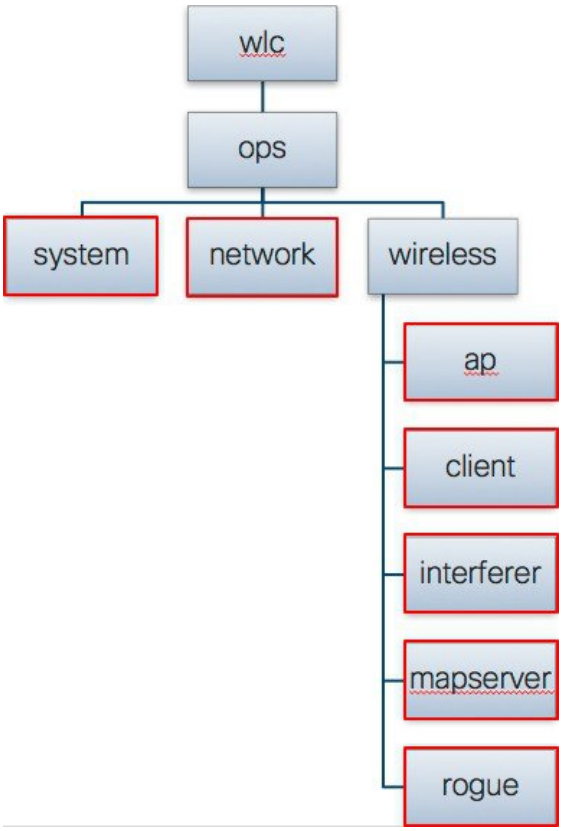
Auth-Token

Auth-Token の必須の設定については、上記を参照してください。

サブスクリプションの設定

公開されるデータは、CCO に 8.7 コードと合わせてリリースされる YANG モデルで定義されます。以下は、YANG モデルと CLI にリストされているサブスクリプション間のマッピングを示しています。これらは、GUI による設定も可能なサブスクリプションです。有効にすることができる7つのトピックまたはテーブルがあり、それぞれが異なるデータ セット、具体的には **AP**、**クライアント**、**干渉源**、**MapServer**、**ネットワーク**、**Rogue**、**システム**に関する運用データセットを定義しています。1 つまたは複数の YANG モデル サブスクリプションを選択すると、WLC のウェブフック クライアントが、関連するデータをウェブフック サーバにポストします。





システム サブスクリプションは wlc-ops-system-stats.yang モデルで定義され、メモリ、タイマー、CPU などの使用率とメトリックの詳細を示します。

ネットワーク サブスクリプションは wlc-ops-network-stats.yang モデルで定義され、各ネットワーク サービスの RADIUS、DHCP、TACACS の統計情報とカウンタの詳細を示します。

AP サブスクリプションは wlc-ops-ap.yang モデルで定義され、AP 名、MAC、接続しているクライアント数などの AP 情報を公開します。

クライアント サブスクリプションは wlc-ops-clients.yang モデルで定義され、クライアントの RSSI、MAC、接続している AP などのクライアントのメトリックと情報を公開します。

干渉源サブスクリプションは wlc-ops-interferers.yang モデルで定義され、さまざまな種類の RF 干渉源の詳細を示します。

MapServer サブスクリプションは wlc-ops-fabric.yang モデルで定義され、マップ サーバの詳細および、関連付けられているファブリック対応ワイヤレスカウンタを示します。このサブスクリプションには、CMX またはロケーションサービスに関する要素は含まれていません。

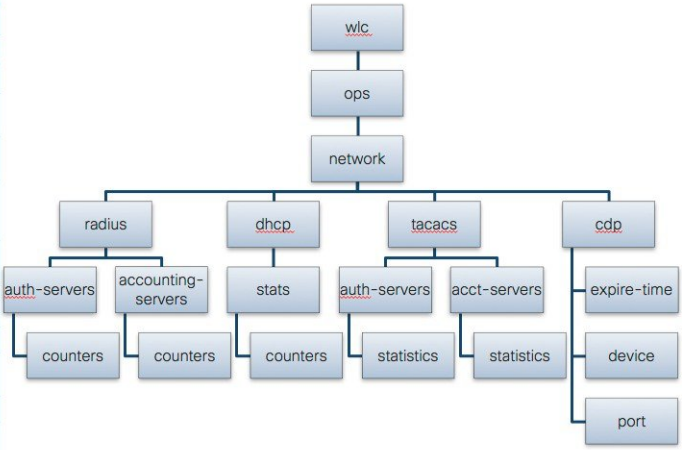
Rogue サブスクリプションは wlc-ops-rogue.yang モデルで定義され、関連付けられているレポート AP から送信されるさまざまな種類の Rogue と RF 情報の詳細を示します。

Yang モデル

YANG モデルを分析することで、モデルに含まれ、公開されるデータの詳細を把握できます。WLC-Ops-Network-stats.yang モデルの概要を次に示します。

8.7 Yang models – Network counters

Radius auth-servers counters	Radius account-servers counters	DHCP counters
Pending-requests	First-requests	tx-discovers
First-requests	retry-requests	rx-discovers
Retry-requests	Acct-responses	requests
Accept-responses	Malformed-messages	informs
Reject-responses	Bad-authenticator-messages	declines
Challenge-responses	Timeout-requests	releases
Malformed-messages	Unknown-messages	replies
Bad-authenticator-messages	Other-drops	offers
Timeout-requests	Pending-requests	acks
Unknown-messages		nacks
Other-drops		tx-failures
Consecutive-timeout-events		
Consecutive-response-failures		



このモデルでは、Radius、TACACS、DHCP、CDP 情報に関するカウンタと統計情報を含む、ネットワーク サブスクリプションの詳細が示されています。

ウェブフック ステータスの確認

ウェブフックのステータスは、CLI コマンド「**show telemetry webhook summary**」を使用して確認できます。GUI では、[Management] タブのウェブフック設定画面にある [Data Publishing Status] でも確認できます。[Last Success] の時刻が [Last Error] の時刻よりも新しいことを確認します。その場合は、ウェブフックがウェブフック サーバにデータを正常にポストできていることになります。

```
(WLC-5520) >config telemetry webhook enable
```

```
(WLC-5520) >show telemetry webhook summary

     Webhook url............................ https://10.10.105.107
     Webhook Mode........................... Enabled
     Webhook Onchange Mode.................. Enabled
     Webhook Sync Interval.................. Adaptive
     Webhook Subscription Topics............ all
                                                 Webhook Data Publish Status:
        Last Error......................... Thu Apr 12 18:43:45 2018 Couldn't connect to server
        Last Success.......................  Thu Apr 12 18:50:17 2018
```

Logstash および Python Flask 用の SSL 証明書の生成

WLC とウェブフック サーバ間の HTTPS 接続を保護するために、SSL 証明書を作成する必要があります。証明書は、認証局（CA）で署名することも、自己署名することもできます。

次の 2 つのコマンドを使用して、Logstash、Python Flask、および WLC のそれぞれで使用可能な自己署名 SSL 証明書を生成できます。プロンプトに従い、「What is your first and last name」というプロンプトが表示されたら、ウェブフックサーバの IP アドレスまたは DNS 名を入力します。

```
$ keytool -genkey -alias webhook -keyalg RSA -validity 365 -keystore /tmp/webhook.jks -keysize 2048
```

```
jcohoe@tme-virtual-machine:~$ keytool -genkey -alias webhook -keyalg RSA -validity 365 -keystore /tmp/webhook.jks -keysize 2048
Enter keystore password:
Re-enter new password:
What is your first and last name?
  [Unknown]:  10.10.105.107
What is the name of your organizational unit?
  [Unknown]:
What is the name of your organization?
  [Unknown]:
What is the name of your City or Locality?
  [Unknown]:
What is the name of your State or Province?
  [Unknown]:
What is the two-letter country code for this unit?
  [Unknown]:
Is CN=10.10.105.107, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown correct?
  [no]:  yes

Enter key password for <webhook>
        (RETURN if same as keystore password):
```

```
$ keytool -export -rfc -alias webhook -keystore /tmp/webhook.jks -file /tmp/webhook.crt -storepass password
```

```
jcohoe@tme-virtual-machine:~$ keytool -export -rfc -alias webhook -keystore /tmp/webhook.jks -file /tmp/webhook.crt -storepass password
Certificate stored in file </tmp/webhook.crt>

Warning:
The JKS keystore uses a proprietary format. It is recommended to migrate to PKCS12 which is an industry standard format using "keytool -imp
ortkeystore -srckeystore /tmp/webhook.jks -destkeystore /tmp/webhook.jks -deststoretype pkcs12".
```

Python Flask で使用する証明書を生成するには、次の追加コマンドを実行

```
$ keytool -importkeystore -srckeystore /tmp/webhook.jks -destkeystore /tmp/webhook.p12 -deststoretype PKCS12
-srcalias webhook -deststorepass password -destkeypass password
$ openssl pkcs12 -in /tmp/webhook.p12  -nokeys -out /tmp/webhook.crt
$ openssl pkcs12 -in /tmp/webhook.p12  -nodes -nocerts -out /tmp/webhook.key
```

```
jcohoe@tme-virtual-machine:~$ keytool -importkeystore -srckeystore /tmp/webhook.jks -destkeystore /tmp/webhook.p12 -deststoretype PKCS12 -s
rcalias webhook -deststorepass password -destkeypass password
Importing keystore /tmp/webhook.jks to /tmp/webhook.p12...
Enter source keystore password:
jcohoe@tme-virtual-machine:~$ openssl pkcs12 -in /tmp/webhook.p12  -nokeys -out /tmp/webhook.crt
Enter Import Password:
MAC verified OK
jcohoe@tme-virtual-machine:~$ openssl pkcs12 -in /tmp/webhook.p12  -nodes -nocerts -out /tmp/webhook.key
Enter Import Password:
MAC verified OK
```

最終的には、次の 4 つのファイルが出力されます。

```
jcohoe@tme-virtual-machine:~$ file /tmp/webhook.*
/tmp/webhook.crt: ASCII text
/tmp/webhook.jks: Java KeyStore
/tmp/webhook.key: ASCII text
/tmp/webhook.p12: data
```

Webhook.crt は WLC に転送されます。

Webhook.jks は Logstash で使用できます。

Webhook.crt と webhook.key は Python Flask で使用できます。

Webhook.p12 は使用されていないため、削除してかまいません。

.crt および .key ファイルは次のような内容になります。

```
jcohoe@tme-virtual-machine:~$ cat /tmp/webhook.crt
Bag Attributes
    friendlyName: webhook
    localKeyID: 54 69 6D 65 20 31 35 32 33 35 39 34 34 36 30 34 37 38
subject=/C=Unknown/ST=Unknown/L=Unknown/O=Unknown/OU=Unknown/CN=10.10.105.107
issuer=/C=Unknown/ST=Unknown/L=Unknown/O=Unknown/OU=Unknown/CN=10.10.105.107
-----BEGIN CERTIFICATE-----
MIIDgzCCAmugAwIBAgIEDgy1EDANBgkqhkiG9w0BAQsFADByMRAwDgYDVQQGEwdV
bmtub3duMRAwDgYDVQQIEwdVbmtub3duMRAwDgYDVQQHEwdVbmtub3duMRAwDgYD
VQQKEwdVbmtub3duMRAwDgYDVQQLEwdVbmtub3duMRYwFAYDVQQDEw0xMC4xMC4x
MDUuMTA3MB4XDTE4MDQxMzA0MzkwNFoXDTE5MDQxMzA0MzkwNFowcjEQMA4GA1UE
BhMHVW5rbm93bjEQMA4GA1UECBMHVW5rbm93bjEQMA4GA1UEBxMHVW5rbm93bjEQ
MA4GA1UEChMHVW5rbm93bjEQMA4GA1UECxMHVW5rbm93bjEWMBQGA1UEAxMNMTAu
MTAuMTA1LjEwNzCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAIUU2YRH
2weFM8VRyN8tW/2YF0AM6MWyBPqBDs0By9avORqBsL12yNNRuxucEolimawokS8G
dxNxkwV6JnNS3/n2hAHtg+h8yth1K1zhg/MUtXLTeJ6cX/59FskupHcqQtfTfkV1
3oovk5qam0uJ2AT+DzbmCZjIXpIZo8KNNArYUQJZ7dYpo41/20d4qHlkk8Ph+bD7
[illegible]
BMne0cdBBLisrpJmxbgZBxF548RaMQZPbvXaE1qdv5GzVhuG54gZ
-----END CERTIFICATE-----
jcohoe@tme-virtual-machine:~$
```

```
jcohoe@tme-virtual-machine:~$ cat /tmp/webhook.key
Bag Attributes
    friendlyName: webhook
    localKeyID: 54 69 6D 65 20 31 35 32 33 35 39 34 34 36 30 34
Key Attributes: <No Attributes>
-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQCFFNmER9sHhTPF
UcjfLVv9mBdADOjFsgT6gQ7NAcvWrzkagbC9dsjTUbsbnBKJYpmsKJEvBncTcZMF
eiZzUt/59oQB7YPofMrYdStc4YPzFLVy03ienF/+fRbJLqR3KkLX035Fdd6KL5Oa
mpjridgE/g825gmYyF6SGaPCjTQK2FECWe3WKaONf9jneIB5ZJPD4fmw+0lQcuZo
J6Cd+9e6mmunO4yko5RQzLeKmD/ou1nQ2n4cxJ98mmsLnHKmfjJ22VDBDoIIEfqk
7SbemFyuHHqQDOndnupt/mqCYJkWIAzh0lgNXQvmywYk2YCSwfaFimm6bRuXVz/F
2OSdkSejAgMBAAECggEAcBhJx3RG96SpINFsox+v1oFfRCIbhzoVm3NNpjyXF7Ps
I3mOB2iga+7x7aWer34P9sUM1VFax1PpBLoI3Sa3ZfLQ6lbHvvIuYfj/l3KVFBEj
iJe2RDpJ+eOZ/ITw0FqYsqXv3PlJ5iL/CoeiTCCE7OACNaVApL6L9tTmFK8fQdPW
RdjU03qdCnnYZDXQF6Q43xRg
-----END PRIVATE KEY-----
jcohoe@tme-virtual-machine:~$
```

Python を使用した Linux ウェブフック サーバ

ここでは、Flask ウェブフック フレームワークを使用してウェブフック サーバとして機能し、WLC からデータを受信する Python コードの例をいくつか示します。Linux サーバ IP でウェブフック URL を指定した後、Linux で次のコードを実行します。WLC から「**config telemetry webhook url http://<IP_OF_LINUX_SERVER>**」コマンドを入力し、「**config telemetry webhook enable**」コマンドを使用してサービスが有効になっていることを確認します。

Flash ウェブフックの詳細は、Github（https://ogma-dev.github.io/posts/simple-flask-webhook/）で確認できます。以下は最も基本的なウェブフック サーバの例です。必要に応じて SSL 証明書へのパスを更新してください。

```
from flask import Flask, request, abort
```

```
app = Flask(__name__)

@app.route('/', methods=['POST'])
def webhook():
    if request.method == 'POST':
        print(request.json)
        return '', 200
    else:
        abort(400)

if __name__ == '__main__':
    app.run(ssl_context=('/tmp/webhook.crt', '/tmp/webhook.key'),host='0.0.0.0', port=443)
```

Linux で Python Flask ウェブフックを開始する例は次のとおりです。

```
from flask import Flask, request, abort

app = Flask(__name__)

@app.route('/', methods=['POST'])
def webhook():
    if request.method == 'POST':
        print(request.json)
        return '', 200
    else:
        abort(400)

if __name__ == '__main__':
    app.run(ssl_context=('/tmp/webhook.crt', '/tmp/webhook.key'),host='0.0.0.0', port=443)
```

最初のペイロードが受信され、ウェブフック クライアントとサーバの設定が正しいことが確認されます。ウェブフック サーバが、ウェブフック クライアントに HTTP POST コード 200 を返します。

```
user@canyon-server:~$ sudo python webhook.py
 * Running on https://0.0.0.0:443/ (Press CTRL+C to quit)
{u'header': {u'sysname': u'AP3800-ME87', u'mgmtIp': u'10.10.10.50', u'sequen
ce': 0, u'timeStamp': u'1523553469', u'version': u'8.7.1.161', u'serialNum':
 u'FJC2025F030'}, u'payload': {u'wlc': {u'config': {u'wireless': {u'rogue-ig
nore-list': [], u'wlan': [{u'interface-group': None, u'ssid': u'messid87', u
'wlan-id': 1, u'atf-policy': 0, u'auth-type': u'WPA/WPA2+PSK', u'profile-nam
e': u'messid87', u'interface': u'management', u'opendns-mode': 1, u'opendns-
profile-name': None}], u'general': {u'fra-enabled': 1}, u'ap': [{u'mac': u'c
c:16:7e:ac:8d:60', u'mode': 2}], u'radio': [{u'radioType': 1, u'beam-forming
': 0}, {u'radioType': 2, u'beam-forming': 0}], u'apgroups': [], u'apModel':
[{u'model': u'AP4800', u'radio': [{u'radioType': 1, u'spatial-stream': 3, u'
protocol': 2, u'max-data-rate': 217}, {u'radioType': 2, u'spatial-stream': 3
, u'protocol': 3, u'max-data-rate': 2340}]}, {u'model': u'AP3802', u'radio':
```

受信した別のペイロードも参考までに以下に示します。

```
10.10.10.50 - - [12/Apr/2018 10:48:46] "POST / HTTP/1.1" 200 -
{u'header': {u'sysname': u'AP3800-ME87', u'mgmtIp': u'10.10.10.50', u
, u'version': u'8.7.1.161', u'serialNum': u'FJC2025F030'}, u'payload'
 [{u'ethernet-mac': u'00:42:68:C5:D2:1A', u'network': {u'ip': u'10.10
: 5, u'statistics': {u'output-no-buffer': 0, u'rx-unicast-pkts': 8445
 u'input-drops': 1, u'tx_nunicast-pkts': 0, u'input-errors': 0, u'tx-
s': 0, u'unknown-protocol': 0, u'output-errors': 0, u'rx-bytes': 7992
lision': 0, u'input-crc': 0, u'tx-bytes': 106262412, u'output-underru
u'name': u'GigabitEthernet0'}, {u'duplex-status': 5, u'statistics': {
: 0, u'input-aborts': 0, u'reset-count': 0, u'input-drops': 0, u'tx_n
-unicast-pkts': 12453, u'output-total-drops': 0, u'unknown-protocol':
```

ウェブフック サーバがセットアップされ、正常に動作していることが確認されました。

Elastic（ELK）スタックの Kibana を使用したウェブフック データの表示

Kibana は、データ表示に特化した、有名な無償のオープンソース ソフトウェア パッケージで、多くの場合 Logstash および Elasticsearch と一緒に使用して、「ELK」または「Elastic スタック」を構成します。ELK スタックをインストールする手順を説明しているガイドがオンライン上に多数ありますので、このドキュメントでは扱っていません。ELK スタックの保守担当者向けのドキュメントは、詳細なインストールプロセスが記載されているサイト（https://www.elastic.co/guide/en/elastic-stack/current/installing-elastic-stack.html）に用意されています。Ubuntu Linux 16.04 を使用している場合は、サマリー ガイドも https://www.rosehosting.com/blog/install-and-configure-the-elk-stack-on-ubuntu-16-04/ で入手できます。手順 6「Nginx をリバース プロキシとしてインストールする」をスキップし、代わりにデフォルト ポート の 5601 で Kibana に直接アクセスすることを推奨します。

Java セキュリティ設定

次の設定では Logstash プロセスが特権ポート 443 を使用するため、Java プロセスがポート 443 を使用できるようにするには手順を追加する必要があります。https://serverfault.com/questions/133415/how-to-allow-non-root-user-to-listen-on-privileged-port にあるガイドに従って次のコマンドを実行し、Java がポート 443 を使用できるようにします。

```
sudo setcap 'cap_net_bind_service=+ep' <path to Java executable>
Ensure the <path to Java> is to the binary executable, not a symbolic link.
For Ubuntu 16 with Java 8, the complete command is:
$ sudo setcap 'cap_net_bind_service=+ep' /usr/lib/jvm/java-8-oracle/jre/bin/java
```

Logstash 設定

ELK スタックの運用準備が完了したら、ウェブフック データを受信するために必要な手順がいくつかあります。前述のように、ウェブフックの受信には、Python Flask ウェブフックではなく Logstash を使用します。

/etc/logstash/logstash.conf ファイルを設定して、入力と出力の設定用に次の 2 行だけが含まれるようにします。この設定では、Logstash はポート 443 で HTTPS ウェブフック サーバとして機能し、解析結果をローカルで実行されている Elasticsearch に出力します。

```
input { http { ssl => "true" keystore => "/tmp/webhook.jks" keystore_password => "password"  port => 443 } }
output { elasticsearch {} }
```

設定が完了したら、logstash.conf ファイルは次のようになります。

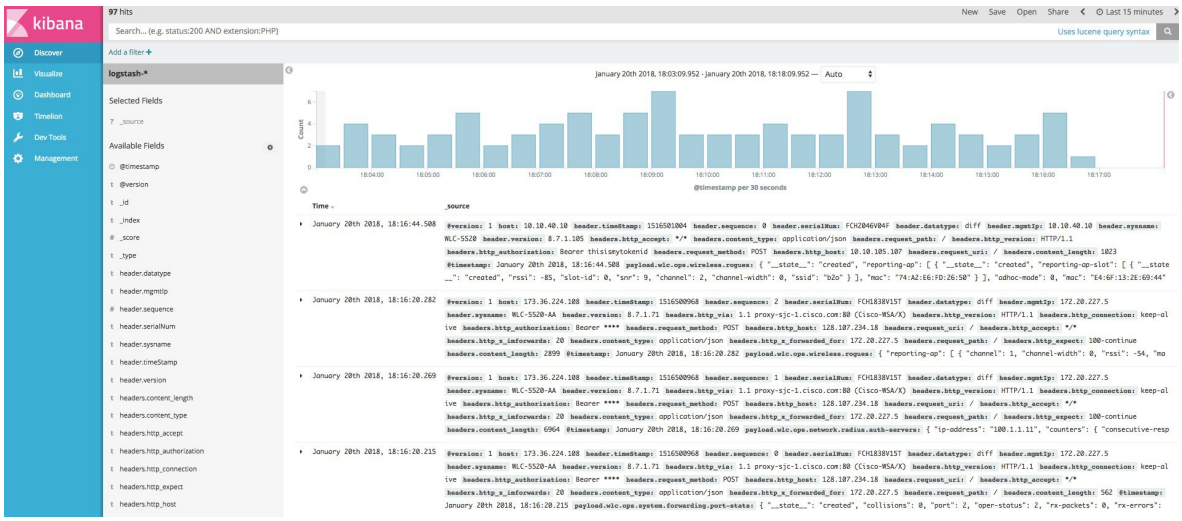
```
input { http { ssl => "true" keystore => "/tmp/webhook.jks" keystore_password => "password"  port => 443 } }
output { elasticsearch {} }
```

必ず、JKS 証明書キーストア ファイルへの正しいパスとキーストア パスワードを使用してください。

Logstash を再起動し、結果が http://localhost:5601 の Kibana に表示されることを確認します。

Kibana

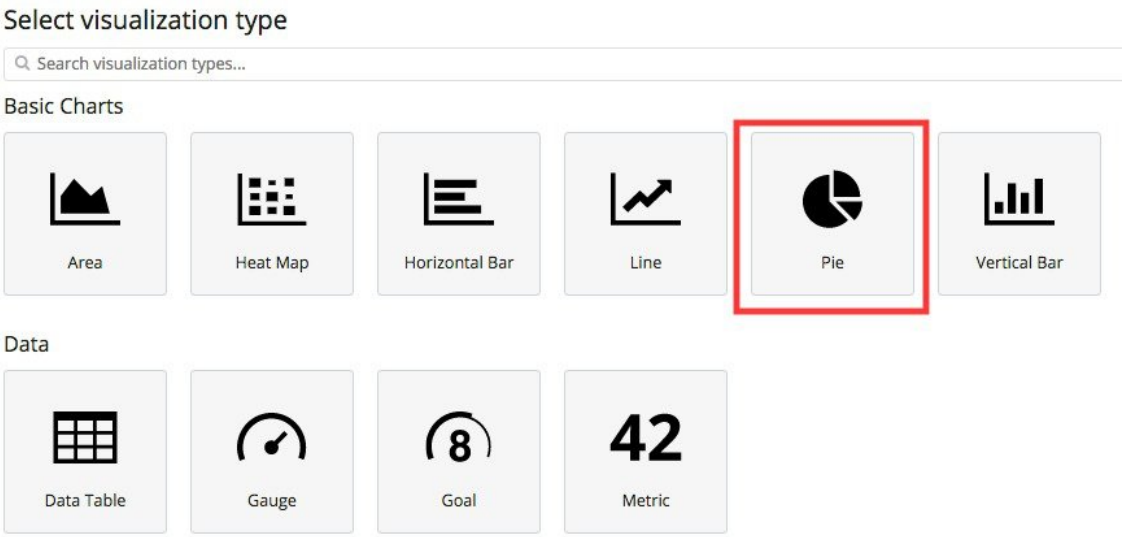
Kibana インターフェイスはデフォルトでは HTTP ポート 5601 上で実行され、最初の接続時には [Discover] ビューが表示されます。[Discover] ビューには、Logstash によってウェブフック クライアントから収集/処理され、Elasticsearch データストアにインデックス付きで保存されたデータが表示されます。WLC のウェブフック クライアントが送信したさまざまなペイロードとデータを含む



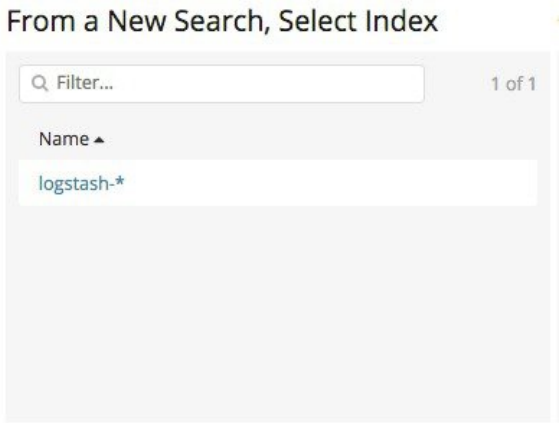
フィールドが表示されます。

表示設定の作成

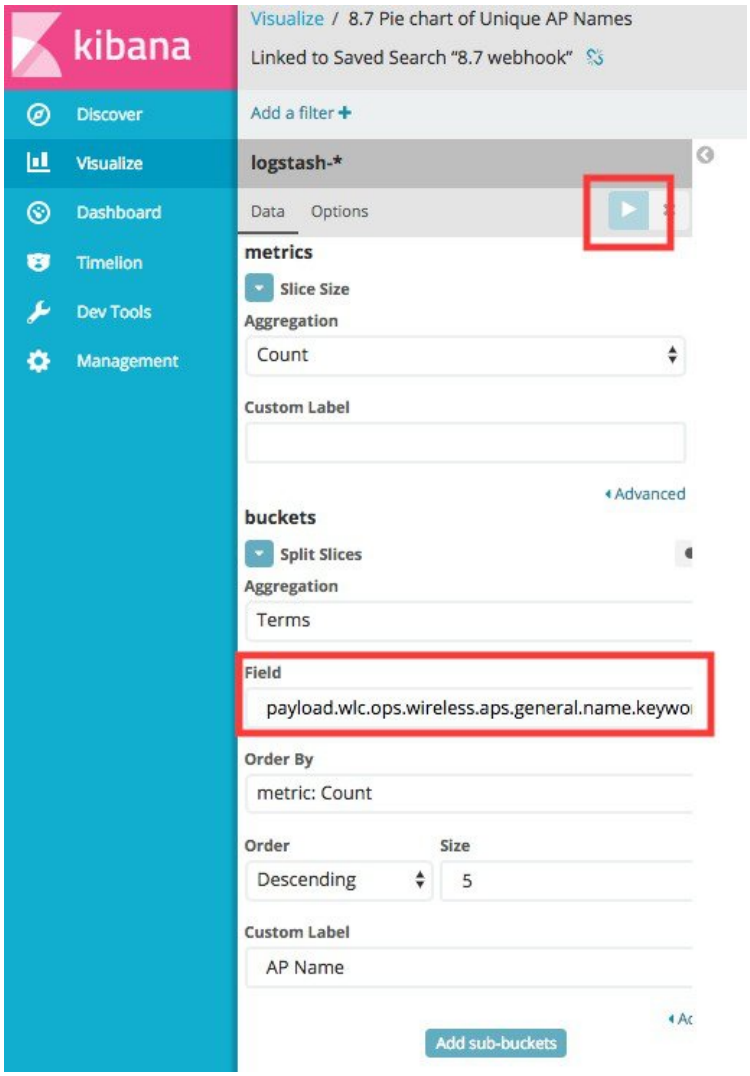
左側の [Visualize] タブをクリック後、[+] 記号をクリックして新しい表示設定を作成します。[Pie] グラフを選択します。



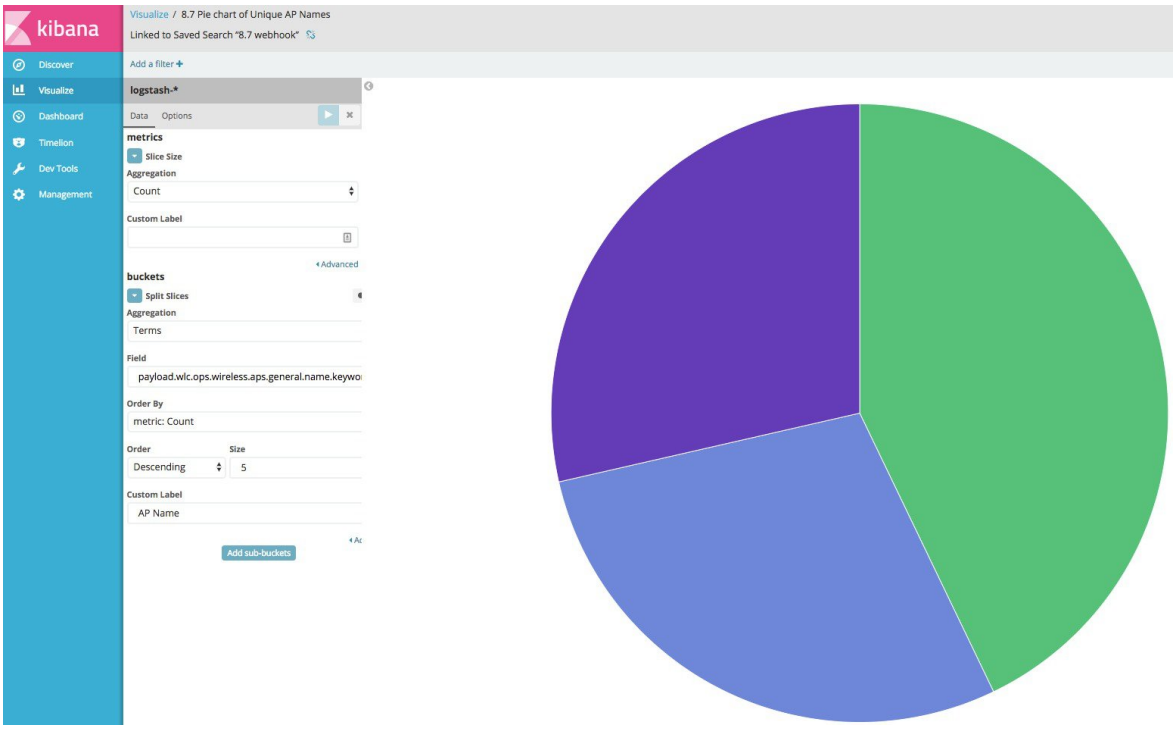
「logstash-*」インデックスを選択するか、すでに [Discover] タブに検索条件を作成して保存している場合は、保存済みの検索条件を選択します。



例として次のスクリーンショットを使用し、WLC に参加している AP 名を示す円グラフを作成します。この設定の場合、[Field] には「**payload.wlc.ops.wireless.aps.general.name.keyword**」を設定します。



右上の右矢印アイコンをクリックすると、以下のように結果が円グラフで表示されます。右上にある [Save] リンクを選択し、表示設定の名前を付けます。

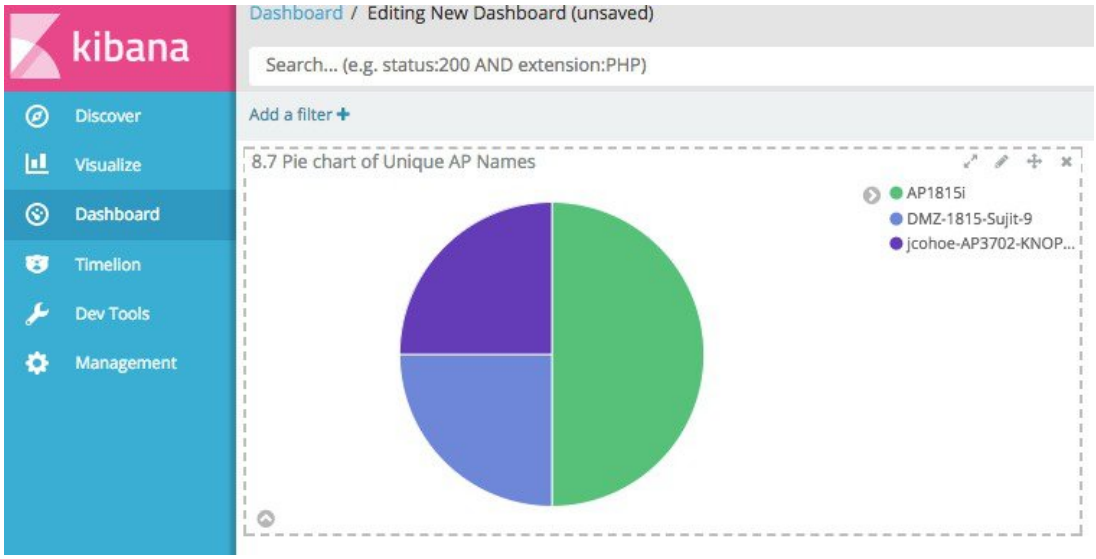


表示設定を追加する場合は、この手順を繰り返します。[Discover] タブを使用して、さまざまなペイロードに使用するフィールド名を決定します。AP およびクライアントのサブスクリプションの例を次に示します。

```
payload.wlc.ops.wireless.aps.general.name.keyword
payload.wlc.ops.wireless.aps.mac.keyword
payload.wlc.ops.wireless.clients.mac.keyword
payload.wlc.ops.wireless.clients.measurements.rf.rssi
payload.wlc.ops.wireless.clients.measurements.traffic.tx-bytes
payload.wlc.ops.wireless.clients.measurements.traffic.rx-bytes
```

ダッシュボードの作成

[Dashboard] タブを使用すると、複数の表示設定をダッシュボードに追加して簡単にデータを表示できます。左側のパネルから [Dashboard] リンクを選択後、[+] アイコンを選択して新しいダッシュボードを作成し、[Add] を選択して前に作成したデータ表示設定を追加します。



ダッシュボードにデータ表示設定を追加して、右上にある [Save] を選択すると、後で簡単に再表示できます。

まとめ

このベータ版ガイドの情報を活用することで、WLC でのウェブフック クライアントの設定、Python Flask ウェブフック コードによる設定の検証、Kibana でのウェブフック データの表示が可能になります。

【注意】シスコ製品をご使用になる前に、安全上の注意（**www.cisco.com/jp/go/safety_warning/**）をご確認ください。本書は、米国シスコ発行ドキュメントの参考和訳です。リンク情報につきましては、日本語版掲載時点で、英語版にアップデートがあり、リンク先のページが移動/変更されている場合がありますことをご了承ください。あくまでも参考和訳となりますので、正式な内容については米国サイトのドキュメントを参照ください。また、契約等の記述については、弊社販売パートナー、または、弊社担当者にご確認ください。

©2008 Cisco Systems, Inc. All rights reserved.
Cisco、Cisco Systems、およびCisco Systemsロゴは、Cisco Systems, Inc.またはその関連会社の米国およびその他の一定の国における登録商標または商標です。
本書類またはウェブサイトに掲載されているその他の商標はそれぞれの権利者の財産です。
「パートナー」または「partner」という用語の使用はCiscoと他社との間のパートナーシップ関係を意味するものではありません。(0809R)
この資料の記載内容は2008年10月現在のものです。
この資料に記載された仕様は予告なく変更する場合があります。

シスコシステムズ合同会社
〒107-6227 東京都港区赤坂9-7-1 ミッドタウン・タワー
http://www.cisco.com/jp
お問い合わせ先：シスコ コンタクトセンター
0120-092-255（フリーコール、携帯・PHS含む）
電話受付時間：平日 10:00～12:00、13:00～17:00
http://www.cisco.com/jp/go/contactcenter/