



HubSpot API 利用ガイド

最終更新日: 2019年5月

目次

1. 本ガイドについて	3
1-1. はじめに	3
1-2. 想定する読者	3
1-3. サンプルコードの動作環境	3
1-4. 用語の定義	4
1-5. HubSpot API の利用準備	4
1-6. HubSpot API の利用制限	5
2. Contacts API	5
2-1. コンタクトの作成	5
2-2. コンタクトの取得	7
2-3. コンタクトの更新	8
2-4. コンタクトの削除	10
3. Companies API	10
3-1. 会社の作成	11
3-2. 会社の取得	12
3-3. 会社の更新	13
3-4. 会社の削除	15
4. Deals API	15
4-1. 取引の作成	15
4-2. 取引の取得	17
4-3. 取引の更新	18
4-4. 取引の削除	19
5. Tickets API	19
5-1. チケットの作成	19
5-2. チケットの取得	21
5-3. チケットの更新	22
5-4. チケットの削除	23

6. Engagements API	23
6-1. アクティビティの作成	24
6-2. アクティビティの取得	25
6-3. アクティビティの更新	26
6-4. アクティビティの削除	26
7. CRM Associations API	27
7-1. オブジェクトの関連付け	27
7-2. オブジェクトの関連付けの削除	28
8. Forms API	29
8-1. 事前準備	29
8-2. フォームの送信 (Ajax)	30
8-3. フォームの送信 (サーバーサイドプログラム)	32
9. Tracking Code API	33
9-1. ページ閲覧履歴の記録	33
9-2. 訪問者の特定	35
10. HubSpotアプリ	36
10-1. 開発者アカウントの作成	36
10-2. HubSpotアプリの作成	37
11. OAuth認証	39
11-1. インストールリンクの作成	39
11-2. アクセストークンの取得	41
11-3. アクセストークンの更新	43
12. CRM拡張API	45
12-1. アプリの設定	45
12-2. サーバープログラムの準備	47
13. タイムラインAPI	50
13-1. アプリの設定	50
13-2. サーバープログラムの準備	51
14. ウェブフックAPI	52
14-1. アプリの設定	53
14-2. サーバープログラムの準備	55
15. おわりに	55
15-1. 開発で困った時は	55
15-2. 作者について	56

1. 本ガイドについて

1-1. はじめに

2019年5月現在、HubSpotはAPIドキュメントの日本語版を公開していません。そのため英語が母語ではないエンジニアの皆様にとって、HubSpotのAPIは分かりづらいものになっているかと存じます。本ガイドはそのギャップを埋めるために作成いたしました。

本ガイドでは、HubSpot APIの利用手順から実際に動作するサンプルコード、また「HubSpotアプリ（連携プログラム）」を作成して配布する方法について解説しています。英語のAPIドキュメントを参照する際の補足資料として本ガイドが皆様のお役に立てば嬉しく存じます。

1-2. 想定する読者

本ガイドは下記の読者を想定しています。

- 自社システムとHubSpotとのデータ連携を行うシステム担当者
- HubSpotのユーザーに自社サービスとの連携プログラムを配布するソフトウェアベンダー

本ガイドのサンプルコードはJavaScriptで記述していますが、言語仕様については説明していません。JavaScriptについて馴染みがない場合、[MDN Web Docs](https://developer.mozilla.org/ja/docs/)などのリソースでの事前の学習をお奨めします。

1-3. サンプルコードの動作環境

本ガイドのサンプルコードは下記のGitHubリポジトリで公開しています。

<https://github.com/shinobukawano/hubspot-api-examples>

サンプルコードはNode.jsというJavaScriptの実行環境で動作させることを想定しています。ご利用の端末にNode.jsがインストールされていない場合、下記のページよりインストーラをダウンロードして下さい。

Download | Node.js:

<https://nodejs.org/en/download/>

Node.jsを利用したサーバーサイドプログラミングについて馴染みがない場合は下記のリソースが参考になるかと存じます。

Express/Node のイントロダクション:

https://developer.mozilla.org/ja/docs/Learn/Server-side/Express_Nodejs/Introduction

また後半の「HubSpotアプリ」以降の章ではサンプルコードをHeroku（PaaS）環境にデプロイして動作させることを想定しています。アカウントをお持ちでない場合は、下記のページより作成して下さい。

クラウド・アプリケーション・プラットフォーム | Heroku:
<https://jp.heroku.com/>

HerokuでNode.jsアプリケーションをデプロイする手順については、下記のドキュメントをご参考頂ければと存じます。

Getting Started on Heroku with Node.js:
<https://devcenter.heroku.com/articles/getting-started-with-nodejs>

サンプルアプリのHerokuへのデプロイについてはGitHubリポジトリのREADMEをご参照下さい。

<https://github.com/shinobukawano/hubspot-api-examples/blob/master/README.md>

1-4. 用語の定義

- REST API
Webシステムを外部から利用するための規約です。HubSpotのAPIはREST APIとして設計されており、所定のエンドポイントURLに対してHTTPリクエストを発行することでデータを作成・取得・更新・削除することが可能です。
- JSON
JavaScriptのオブジェクトリテラルをベースにしたデータ交換のフォーマットです。HubSpotのAPIはJSONフォーマットでデータを受け渡しします。
- HubSpotアプリ
HubSpotのポータルにインストールできる連携プログラムです。HubSpotアプリをインストールすることで、サードパーティのプログラムに対して該当ポータルのデータにアクセスする許可を与えることができます。
認定のアプリについてはHubSpotのマーケットプレイスに掲載され、ユーザーが簡単にアプリを見つけることができるようになります。

1-5. HubSpot API の利用準備

HubSpotのAPIを利用するためには、主に二つの方法があります。

- APIキー
- OAuth認証

各HubSpotポータル毎にAPIキーを発行することができます。このキーの値をリクエストパラメータに含めることで対象ポータルのデータにアクセスすることが可能です。APIキーはHubSpotポータルの「設定」画面にある「統合」「APIキー」メニューから発行下さい。

設定

アカウントの既定値

HubSpot APIキー

インポートとエクスポート

HubSpot APIキーを使用して、すべてのHubSpotツールで利用できる同じAPIにアクセスします。これにより、開発者はHubSpotのAPIを使用してこのアカウントにフルアクセスして変更を行うことができるため、安全に保管し、公開はしないでください。

ウェブサイト

コミュニケーション

各キーは、各ユーザーではなくHubSpotアカウントに固有であり、同時に使用できるキーは1つのみです。いつでもAPIキーを無効にして新しいキーを生成できます。

コンタクトおよび会社

[HubSpot APIキーについてもっと詳しく](#)

サービス

APIキーを生成

セールス

ドメインとURL

プロパティ

ログを監査

マーケティング

ユーザーとチーム

レポート

統合

APIキー

Eコマース

Eメールの統合

日付	ユーザー	アクション	キー
APIキーを生成したユーザーはまだいません。			

ヘルプ

OAuth認証はHubSpotアプリを配布する際に必要となる認証方式です。フローを実装頂く必要がありますが、APIキーよりもセキュアな認証を実現できます。

1-6. HubSpot API の利用制限

HubSpotのAPIには下記の利用制限があります。

- 実行できるHTTPリクエストは1日40,000回まで
- 実行できるHTTPリクエストは1秒間に10回まで

APIのアドオン（月額60,000円 / 2019年4月時点）を購入頂くことで、APIの1日のHTTPリクエスト数を160,000回まで引き上げることが可能です。ご希望の際はHubSpotの営業担当者までご連絡下さい。

2. Contacts API

HubSpotの「コンタクト」オブジェクトを操作するためのAPIです。

2-1. コンタクトの作成

エンドポイントURL「<https://api.hubapi.com/contacts/v1/contact/>」にJSONデータを送信することでコンタクトを作成できます。利用するHTTPメソッドはPOSTです。コンタクトが作成されるとレコードを特定するための一意のID (vid) が採番され、値がレスポンスで返されます。

下記はAPI利用のサンプルコードです。Node.jsランタイムでの実行を想定しています。

```
// $ node scripts/contacts.js create

request({
  url: `https://api.hubapi.com/contacts/v1/contact/?hapikey=${API_KEY}`,
  method: "POST",
  json: true,
  body: {
    "properties": [{
      "property": "email",
      "value": "skawano+test@hubspot.com"
    }, {
      "property": "lastname",
      "value": "テスト"
    }, {
      "property": "firstname",
      "value": "太郎"
    }
  ]
})
  .then((r) => {
    console.log(r.vid)
  })
  .catch((e) => {
    console.log(e.message)
  })

=> 309801
```

またエンドポイントURL「[https://api.hubapi.com/contacts/v1/contact/createOrUpdate/email/{Eメール
アドレス}](https://api.hubapi.com/contacts/v1/contact/createOrUpdate/email/{Eメールアドレス})」では、コンタクトが存在しない場合は新規で作成し、既に存在する場合はコンタクトの値を更新することが可能です。

```
// $ node scripts/contacts.js createOrUpdate

request({
  url:
`https://api.hubapi.com/contacts/v1/contact/createOrUpdate/email/skawano+test@hu  
bspot.com/?hapikey=${API_KEY}`,
  method: "POST",
  json: true,
  body: {
    "properties": [{
```

```
    "property": "lastname",
    "value": "テスト"
  }, {
    "property": "firstname",
    "value": "太郎"
  }
]
}
}).then((r) => {
  console.log(r.vid)
}).catch((e) => {
  console.log(e.message)
})
```

=> 309801

参考) Create a new contact:

https://developers.hubspot.com/docs/methods/contacts/create_contact

参考) Create or update a contact:

https://developers.hubspot.com/docs/methods/contacts/create_or_update

2-2. コンタクトの取得

エンドポイントURL 「<https://api.hubapi.com/contacts/v1/contact/vid/{vid}/profile/>」 にリクエストすることでURLで指定したvidのコンタクトを取得できます。利用するHTTPメソッドはGETです。データはJSON形式で返却されます。

```
// $ node scripts/contacts.js getById 309801

request({
  url:
`https://api.hubapi.com/contacts/v1/contact/vid/${vid}/profile/?hapikey=${API_KEY}`,
  json: true
}).then((r) => {
  console.log(r)
}).catch((e) => {
  console.log(e.message)
})

=> { vid: 309801,
  properties:
  { ...
```

```
  lastname: { value: 'テスト', versions: [Array] },
  firstname: { value: '太郎', versions: [Array] },
  email: { value: 'skawano+test@hubspot.com', versions: [Array] },
  ...
```

またエンドポイントURL「<https://api.hubapi.com/contacts/v1/contact/email/{Eメールアドレス}/profile>」では、Eメールアドレスを利用してコンタクトを取得することが可能です。

```
// $ node scripts/contacts.js getByEmail

request({
  url:
`https://api.hubapi.com/contacts/v1/contact/email/skawano+test@hubspot.com/profi
le?hapikey=${API_KEY}`,
  json: true
}).then((r) => {
  console.log(r)
}).catch((e) => {
  console.log(e.message)
})

=> { vid: 309801,
  properties:
  { ...
    lastname: { value: 'テスト', versions: [Array] },
    firstname: { value: '太郎', versions: [Array] },
    email: { value: 'skawano+test@hubspot.com', versions: [Array] },
    ...
```

参考) Get a contact record by its vid:

https://developers.hubspot.com/docs/methods/contacts/get_contact

参考) Search for a contact by email address:

https://developers.hubspot.com/docs/methods/contacts/get_contact_by_email

2-3. コンタクトの更新

エンドポイントURL「<https://api.hubapi.com/contacts/v1/contact/vid/{vid}/profile>」にJSONデータを送信することでURLで指定したvidのコンタクトを更新できます。利用するHTTPメソッドはPOSTです。


```
// $ node scripts/contacts.js updateById 309801

request({
  url:
`https://api.hubapi.com/contacts/v1/contact/vid/${vid}/profile/?hapikey=${API_KEY}`,
  method: "POST",
  json: true,
  body: {
    "properties": [{
      "property": "lastname",
      "value": "更新: テスト"
    }, {
      "property": "firstname",
      "value": "更新: 太郎"
    }
  ]
}).then((r) => {
  console.log(r)
}).catch((e) => {
  console.log(e.message)
})
```

またエンドポイントURL「<https://api.hubapi.com/contacts/v1/contact/email/{Eメールアドレス}/profile>」では、Eメールアドレスを利用してコンタクトを更新することが可能です。

```
// $ node scripts/contacts.js updateByEmail

request({
  url:
`https://api.hubapi.com/contacts/v1/contact/email/skawano+test@hubspot.com/profile?hapikey=${API_KEY}`,
  method: "POST",
  json: true,
  body: {
    "properties": [{
      "property": "lastname",
      "value": "更新: テスト"
    }, {
      "property": "firstname",
      "value": "更新: 太郎"
    }
  ]
})
```

```
    }  
  }).then((r) => {  
    console.log(r)  
  }).catch((e) => {  
    console.log(e.message)  
  })  
})
```

参考) Update an existing contact:

https://developers.hubspot.com/docs/methods/contacts/update_contact

参考) Update an existing contact by email:

https://developers.hubspot.com/docs/methods/contacts/update_contact-by-email

2-4. コンタクトの削除

エンドポイントURL 「<https://api.hubapi.com/contacts/v1/contact/vid/{vid}>」 にリクエストすることでURLで指定したvidのコンタクトを削除できます。利用するHTTPメソッドはDELETEです。

```
// $ node scripts/contacts.js deleteById 309801  
  
request({  
  url:  
  `https://api.hubapi.com/contacts/v1/contact/vid/${vid}?hapikey=${API_KEY}`,  
  method: "DELETE",  
}).then((r) => {  
  console.log(r)  
}).catch((e) => {  
  console.log(e.message)  
})  
  
=> {"vid":309801,"deleted":true,"reason":"OK"}
```

尚、現時点ではEメールの値を利用してコンタクトを削除するAPIは公開していません。

参考) Delete a contact:

https://developers.hubspot.com/docs/methods/contacts/delete_contact

3. Companies API

HubSpotの「会社」オブジェクトを操作するためのAPIです。

3-1. 会社の作成

エンドポイントURL「<https://api.hubapi.com/companies/v2/companies>」にJSONデータを送信することで会社を作成できます。利用するHTTPメソッドはPOSTです。会社を作成されるとレコードを特定するための一意のID (companyId) が採番され、値がレスポンスで返されます。

```
// $ node scripts/companies.js create

request({
  url: `https://api.hubapi.com/companies/v2/companies?hapikey=${API_KEY}`,
  method: "POST",
  json: true,
  body: {
    "properties": [{
      "name": "domain",
      "value": "hubspot.jp"
    }, {
      "name": "name",
      "value": "HubSpot Japan"
    }, {
      "name": "description",
      "value": "インバウンドマーケティング&セールスソフトウェア"
    }
  ]
})
}.then((r) => {
  console.log(r.companyId)
}).catch((e) => {
  console.log(e.message)
})

=> 1902989304
```

Contacts APIとはpropertiesキーで渡すオブジェクトのパラメータが異なりますのでご注意ください。

参考) Create a Company:

https://developers.hubspot.com/docs/methods/companies/create_company

3-2. 会社の取得

エンドポイントURL「<https://api.hubapi.com/companies/v2/companies/{companyId}>」にリクエストすることでURLで指定したcompanyIdの企業を取得できます。利用するHTTPメソッドはGETです。データはJSON形式で返却されます。

```
// $ node scripts/companies.js getById 1902989304

request({
  url:
`https://api.hubapi.com/companies/v2/companies/${id}?hapikey=${API_KEY}`,
  json: true
}).then((r) => {
  console.log(r)
}).catch((e) => {
  console.log(e.message)
})

=> { ...
  companyId: 1902989304,
  properties:
  { ...
    description:
    { value: 'インバウンドマーケティング&セールスソフトウェア',
      ...
    },
    domain:
    { value: 'hubspot.jp',
      ...
    },
    name:
    { value: 'HubSpot Japan',
      ...
    },
    ...
  }
}
```

またエンドポイントURL「<https://api.hubapi.com/companies/v2/domains/{ドメイン名}/companies>」では、ドメイン名を利用して企業を取得することが可能です。

```
// $ node scripts/companies.js getByDomain
```

```

request({
  url:
`https://api.hubapi.com/companies/v2/domains/hubspot.jp/companies?hapikey=${API_
KEY}`,
  method: "POST",
  body: {
    "requestOptions": {
      "properties": [
        "domain",
        "name",
        "description"
      ]
    }
  },
  json: true
}).then((r) => {
  console.log(r)
}).catch((e) => {
  console.log(e.message)
})

=> { results:
  [ { ...
    companyId: 1902989304,
    properties: [ ... ],
    ...

```

上記のAPIでは利用するHTTPメソッドはPOSTで送信データにrequestOptionsの値を含める必要がある点にご注意下さい。requestOptionsでは取得したいプロパティを指定します。

参考) Get a Company:

https://developers.hubspot.com/docs/methods/companies/get_company

参考) Search for companies by domain:

https://developers.hubspot.com/docs/methods/companies/search_companies_by_domain

3-3. 会社の更新

エンドポイントURL「<https://api.hubapi.com/companies/v2/companies/{companyId}>」にJSONデータを送信することでURLで指定したcompanyIdの会社を更新できます。利用するHTTPメソッドはPUTです。

```
// $ node scripts/companies.js updateById 1902989304

request({
  url:
`https://api.hubapi.com/companies/v2/companies/${id}/?hapikey=${API_KEY}`,
  method: "PUT",
  json: true,
  body: {
    "properties": [{
      "name": "name",
      "value": "更新: HubSpot Japan"
    }, {
      "name": "description",
      "value": "更新: インバウンドマーケティング&セールスソフトウェア"
    }
  ]
})
}.then((r) => {
  console.log(r)
}).catch((e) => {
  console.log(e.message)
})

=> { ...
  companyId: 1902989304,
  properties:
  { ...
    name:
    { value: '更新: HubSpot Japan',
      ...
    },
    description:
    { value: '更新: インバウンドマーケティング&セールスソフトウェア',
      timestamp: 1555128724081,
      ...
    }
  }, ...
}
```

参考) Update a Company:

https://developers.hubspot.com/docs/methods/companies/update_company

3-4. 会社の削除

エンドポイントURL「`https://api.hubapi.com/companies/v2/companies/{companyId}`」にリクエストすることでURLで指定したcompanyIdの会社を削除できます。利用するHTTPメソッドはDELETEです。

```
// $ node scripts/companies.js deleteById 1902989304

request({
  url:
`https://api.hubapi.com/companies/v2/companies/${id}?hapikey=${API_KEY}`,
  method: "DELETE",
}).then((r) => {
  console.log(r)
}).catch((e) => {
  console.log(e.message)
})

=> {"companyId":1807324680,"deleted":true}
```

参考) Delete a Company:

https://developers.hubspot.com/docs/methods/companies/delete_company

4. Deals API

HubSpotの「取引」オブジェクトを操作するためのAPIです。

4-1. 取引の作成

取引を作成するためには、取引ステージのID (stagelid) を指定する必要があります。CRM Pipelines APIを利用することで、取引パイプラインの情報を取得できます。エンドポイントのURLは「`https://api.hubapi.com/crm-pipelines/v1/pipelines/deals`」で、HTTPメソッドGETでデータを取得します。

```
// $ node scripts/deals.js getPipelines

request({
  url:
`https://api.hubapi.com/crm-pipelines/v1/pipelines/deals?hapikey=${API_KEY}`,
  json: true
}).then((r) => {
```

```

    console.dir(r, { depth: 4 })
  }).catch((e) => {
    console.log(e.message)
  })
=> { results:
  [ { pipelineId: 'default',
    objectType: 'DEAL',
    ...
    stages:
      [ { stageId: '219ee00f-dc48-4734-86dc-924fb35e6da5',
        ... },
        ...

```

上記のAPIで取得した取引ステージのID (stageId) を利用して取引を作成します。

エンドポイントURLは「<https://api.hubapi.com/deals/v1/deal>」です。「dealstage」プロパティにCRM Pipelines APIで取得したstageIdを指定して下さい。また利用するHTTPメソッドはPOSTです。取引が作成されるとレコードを特定するための一意のID (dealId) が採番され、値がレスポンスで返されます。

```

// $ node scripts/deals.js create 219ee00f-dc48-4734-86dc-924fb35e6da5

request({
  url: `https://api.hubapi.com/deals/v1/deal?hapikey=${API_KEY}`,
  method: "POST",
  json: true,
  body: {
    "properties": [{
      "name": "dealstage",
      "value": stageId
    }, {
      "name": "dealname",
      "value": "新しい取引"
    }, {
      "name": "amount",
      "value": 30000
    }
  ]
}
}).then((r) => {
  console.log(r.dealId)
}).catch((e) => {
  console.log(e.message)

```



```
})
```

```
=> 736610655
```

参考) Get all pipelines for a specified object type:

https://developers.hubspot.com/docs/methods/pipelines/get_pipelines_for_object_type

参考) Create a Deal:

https://developers.hubspot.com/docs/methods/deals/create_deal

4-2. 取引の取得

エンドポイントURL「<https://api.hubapi.com/deals/v1/deal/{dealId}>」にリクエストすることでURLで指定したdealIdの取引を取得できます。利用するHTTPメソッドはGETです。データはJSON形式で返却されます。

```
// $ node scripts/deals.js getById 736610655

request({
  url: `https://api.hubapi.com/deals/v1/deal/${id}?hapikey=${API_KEY}`,
  json: true
}).then((r) => {
  console.log(r)
}).catch((e) => {
  console.log(e.message)
})

=> { ...
  dealId: 736610655,
  associations:
    { associatedVids: [],
      associatedCompanyIds: [],
      associatedDealIds: [],
      associatedTicketIds: [] },
  properties:
    { pipeline:
      { value: 'default',
        timestamp: 1555135601117,
        source: 'API',
        sourceId: null,
        versions: [Array] },
      dealname:
      { value: '新しい取引',
```

```
    timestamp: 1555135601117,
    source: 'API',
    sourceId: null,
    versions: [Array] },
  amount:
  { value: '30000',
    timestamp: 1555135601117,
    source: 'API',
    sourceId: null,
    versions: [Array] },
  dealstage:
  { value: '219ee00f-dc48-4734-86dc-924fb35e6da5',
    timestamp: 1555135601117,
    source: 'API',
    sourceId: null,
    versions: [Array] },
```

参考) Get a Deal:

https://developers.hubspot.com/docs/methods/deals/get_deal

4-3. 取引の更新

エンドポイントURL「<https://api.hubapi.com/deals/v1/deal/{dealId}>」にJSONデータを送信することでURLで指定したdealIdの取引を更新できます。利用するHTTPメソッドはPUTです。

```
// node scripts/deals.js updateById 736610655

request({
  url: `https://api.hubapi.com/deals/v1/deal/${id}?hapikey=${API_KEY}`,
  method: "PUT",
  json: true,
  body: {
    "properties": [{
      "name": "dealname",
      "value": "更新: 新しい取引"
    }, {
      "name": "amount",
      "value": 99999
    }
  ]
})
}.then((r) => {
  console.log(r)
```

```
}).catch((e) => {
  console.log(e.message)
})
```

参考) Update a Deal:

https://developers.hubspot.com/docs/methods/deals/update_deal

4-4. 取引の削除

エンドポイントURL「<https://api.hubapi.com/deals/v1/deal/{dealId}>」にリクエストすることでURLで指定したdealIdの取引を削除できます。利用するHTTPメソッドはDELETEです。

```
// $ node scripts/deals.js deleteById 736610655

request({
  url: `https://api.hubapi.com/deals/v1/deal/${id}?hapikey=${API_KEY}`,
  method: "DELETE",
}).then((r) => {
  console.log(r)
}).catch((e) => {
  console.log(e.message)
})
```

参考) Delete a Deal:

https://developers.hubspot.com/docs/methods/deals/delete_deal

5. Tickets API

HubSpotの「チケット」オブジェクトを操作するためのAPIです。

5-1. チケットの作成

チケットを作成するためには、パイプラインのID (pipelineId) とステージのID (stageId) を指定する必要があります。CRM Pipelines APIを利用することで、チケットパイプラインの情報を取得できます。エンドポイントのURLは「<https://api.hubapi.com/crm-pipelines/v1/pipelines/ticket>」で、HTTPメソッドGETでデータを取得します。

```
// $ node scripts/tickets.js getPipelines

request({
  url:
```

```

`https://api.hubapi.com/crm-pipelines/v1/pipelines/tickets?hapikey=${API_KEY}` ,
  json: true
}).then((r) => {
  console.dir(r, { depth: 4 })
}).catch((e) => {
  console.log(e.message)
})

=> { results:
  [ { pipelineId: '0',
    ...
    stages:
      [ { stageId: '1',
          label: 'New',
          ... },

```

上記のAPIで取得したパイプラインのID (pipelineId) とステージのID (stageId) を利用してチケットを作成します。

エンドポイントURLは「<https://api.hubapi.com/crm-objects/v1/objects/tickets>」です。「hs_pipeline」プロパティにCRM Pipelines APIで取得した「pipelineId」を、「hs_pipeline_stage」プロパティに「stageId」を指定して下さい。利用するHTTPメソッドはPOSTです。チケットが作成されるとレコードを特定するための一意のID (objectId) が採番され、値がレスポンスで返されます。

```

// $ node scripts/tickets.js create

request({
  url:
`https://api.hubapi.com/crm-objects/v1/objects/tickets?hapikey=${API_KEY}` ,
  method: "POST",
  json: true,
  body: [
    {
      "name": "subject",
      "value": "新規チケット"
    },
    {
      "name": "hs_pipeline",
      "value": "0"
    },
    {
      "name": "hs_pipeline_stage",

```

```
    "value": "1"
  }
]
}).then((r) => {
  console.log(r.objectId)
}).catch((e) => {
  console.log(e.message)
})

=> 26829590
```

参考) Get all pipelines for a specified object type:

https://developers.hubspot.com/docs/methods/pipelines/get_pipelines_for_object_type

参考) Create a ticket:

<https://developers.hubspot.com/docs/methods/tickets/create-ticket>

5-2. チケットの取得

エンドポイントURL「<https://api.hubapi.com/crm-objects/v1/objects/tickets/{objectId}>」にリクエストすることでURLで指定したobjectIdのチケットを取得できます。また取得したいプロパティの値をリクエストパラメータで指定する必要があります（例: ?properties=subject）。

利用するHTTPメソッドはGETです。データはJSON形式で返却されます。

```
// $ node scripts/tickets.js getById 26829590

request({
  url:
`https://api.hubapi.com/crm-objects/v1/objects/tickets/${id}?hapikey=${API_KEY}`
,
  qs: {
    properties: "subject"
  },
  json: true,
}).then((r) => {
  console.log(r)
}).catch((e) => {
  console.log(e.message)
})
```

```
=> { objectType: 'TICKET',
  portalId: 4148814,
  objectId: 26829590,
  properties:
    { subject:
      { versions: [Array],
        value: '新規チケット',
        timestamp: 1555138383923,
        source: 'API',
        sourceId: null } },
  isDeleted: false }
```

参考) Get a ticket by ID:

https://developers.hubspot.com/docs/methods/tickets/get_ticket_by_id

5-3. チケットの更新

エンドポイントURL 「<https://api.hubapi.com/crm-objects/v1/objects/tickets/{objectId}>」 にJSONデータを送信することでURLで指定したobjectIdのチケットを更新できます。利用するHTTPメソッドはPUTです。

```
// $ node scripts/tickets.js updateById 26829590

request({
  url:
`https://api.hubapi.com/crm-objects/v1/objects/tickets/${id}?hapikey=${API_KEY}`
,
  method: "PUT",
  json: true,
  body: [
    {
      "name": "subject",
      "value": "更新: 新しいチケット"
    }
  ]
}).then((r) => {
  console.log(r)
}).catch((e) => {
  console.log(e.message)
})

=> { objectType: 'TICKET',
  objectId: 26829590,
```

```
...
properties:
  { subject:
    { versions: [Array],
      value: '更新: 新しいチケット',
      timestamp: 1555139265251,
      source: 'API',
      sourceId: null } },
  ...
```

参考) Update a ticket:

<https://developers.hubspot.com/docs/methods/tickets/update-ticket>

5-4. チケットの削除

エンドポイントURL 「<https://api.hubapi.com/crm-objects/v1/objects/tickets/{objectId}>」 にリクエストすることでURLで指定したobjectIdのチケットを削除できます。利用するHTTPメソッドはDELETEです。

```
// node scripts/tickets.js deleteById 26829590

request({
  url:
  `https://api.hubapi.com/crm-objects/v1/objects/tickets/${id}?hapikey=${API_KEY}`
,
  method: "DELETE",
}).then((r) => {
  console.log(r)
}).catch((e) => {
  console.log(e.message)
})
```

参考) Delete a ticket:

<https://developers.hubspot.com/docs/methods/tickets/delete-ticket>

6. Engagements API

HubSpotの「アクティビティ」や「タスク」を操作するためのAPIです。

6-1. アクティビティの作成

エンドポイントURL「<https://api.hubapi.com/engagements/v1/engagements>」にJSONデータを送信することでアクティビティを作成できます。利用するHTTPメソッドはPOSTです。会社が作成されるとレコードを特定するための一意のID（engagementId）が採番され、値がレスポンスで返されます。

```
// $ node scripts/engagements.js create

request({
  url: `https://api.hubapi.com/engagements/v1/engagements?hapikey=${API_KEY}`,
  method: "POST",
  json: true,
  body: {
    "engagement": {
      "type": "CALL"
    },
    "metadata" : {
      "toNumber" : "03-1234-5678",
      "fromNumber" : "090-1234-5678",
      "status" : "COMPLETED"
    }
  }
}).then((r) => {
  console.log(r)
}).catch((e) => {
  console.log(e.message)
})

=> {
  engagement:
  { id: 2277261816,
    type: 'CALL',
    ... },
  associations:
  { contactIds: [],
    companyIds: [],
    dealIds: [],
    ownerIds: [],
    ... },
  attachments: [],
  metadata:
  { toNumber: '03-1234-5678',
```



```
fromNumber: '090-1234-5678',
status: 'COMPLETED',
unknownVisitorConversation: false } }
```

参考) Create an Engagement:

https://developers.hubspot.com/docs/methods/engagements/create_engagement

6-2. アクティビティの取得

エンドポイントURL 「<https://api.hubapi.com/engagements/v1/engagements/{engagementId}/>」 にリクエストすることでURLで指定したengagementIdのアクティビティを取得できます。利用するHTTPメソッドはGETです。データはJSON形式で返却されます。

```
// $ node scripts/engagements.js getById 2277261816

request({
  url:
`https://api.hubapi.com/engagements/v1/engagements/${id}?hapikey=${API_KEY}`,
  json: true
}).then((r) => {
  console.log(r)
}).catch((e) => {
  console.log(e.message)
})

=> {
  engagement:
  { id: 2277261816,
    type: 'CALL',
    ... },
  associations:
  { contactIds: [],
    companyIds: [],
    dealIds: [],
    ownerIds: [],
    ... },
  attachments: [],
  metadata:
  { toNumber: '03-1234-5678',
    fromNumber: '090-1234-5678',
    status: 'COMPLETED',
    unknownVisitorConversation: false } }
```

参考) GET an Engagement:

https://developers.hubspot.com/docs/methods/engagements/get_engagement

6-3. アクティビティの更新

エンドポイントURL 「<https://api.hubapi.com/engagements/v1/engagements/{engagementId}>」にJSONデータを送信することでURLで指定したcompanyIdの会社を更新できます。利用するHTTPメソッドはPATCHです。

```
// $ node scripts/engagements.js updateById 2277261816

request({
  url:
`https://api.hubapi.com/engagements/v1/engagements/${id}/?hapikey=${API_KEY}`,
  method: "PATCH",
  json: true,
  body: {
    metadata: {
      "toNumber" : "06-1234-5678",
    }
  }
}).then((r) => {
  console.log(r)
}).catch((e) => {
  console.log(e.message)
})
```

参考) Update an Engagement:

https://developers.hubspot.com/docs/methods/engagements/update_engagement-patch

6-4. アクティビティの削除

エンドポイントURL 「<https://api.hubapi.com/engagements/v1/engagements/{engagementId}>」にリクエストすることでURLで指定したcompanyIdの会社を削除できます。利用するHTTPメソッドはDELETEです。

```
// $ node scripts/engagements.js deleteById 2277261816

request({
```

```
url:
`https://api.hubapi.com/engagements/v1/engagements/${id}?hapikey=${API_KEY}`,
method: "DELETE" ,
}).then((r) => {
  console.log(r)
}).catch((e) => {
  console.log(e.message)
})
```

参考) Delete an Engagement:

<https://developers.hubspot.com/docs/methods/engagements/delete-engagement>

7. CRM Associations API

HubSpotのオブジェクトとオブジェクトを関連づけるためのAPIです。

7-1. オブジェクトの関連付け

エンドポイントURL「<https://api.hubapi.com/crm-associations/v1/associations>」にJSONデータを送信することでオブジェクトを関連づけることができます。利用するHTTPメソッドはPUTです。

下記はコンタクトと会社を関連付けるサンプルです。「fromObjectId」でコンタクトのvidを、「toObjectId」で会社のcompanyIdを指定しています。「category」には「HUBSPOT_DEFINED」を指定して下さい。

```
// $ node scripts/associations.js createAssociation 309851 1903970621

request({
  url:
`https://api.hubapi.com/crm-associations/v1/associations?hapikey=${API_KEY}`,
  method: "PUT",
  json: true,
  body: {
    "fromObjectId": fromId,
    "toObjectId": toId,
    "category": "HUBSPOT_DEFINED",
    "definitionId": 1
  }
}).then((r) => {
  console.log(r)
}).catch((e) => {
  console.log(e.message)
})
```

```
})
```

上記はコンタクトと会社を関連付けるサンプルですが、同じ方式でコンタクトと取引、会社とチケットなど様々なオブジェクト同士を関連付けることが可能です。その際は「definitionId」で対象の組み合わせを指定して下さい。例えば、コンタクトと会社の関連付けであれば「definitionId」は「1」に、コンタクトと取引の関連付けであれば「definitionId」は「4」になります。詳細については下記ドキュメントのテーブルを参照して下さい。

参考) CRM associations overview:

<https://developers.hubspot.com/docs/methods/crm-associations/crm-associations-overview>

参考) Associate CRM objects:

<https://developers.hubspot.com/docs/methods/crm-associations/associate-objects>

7-2. オブジェクトの関連付けの削除

エンドポイントURL「<https://api.hubapi.com/crm-associations/v1/associations/delete>」にJSONデータを送信することでオブジェクトの関連付けを削除することができます。利用するHTTPメソッドはPUTです。

下記はコンタクトと会社の関連付けを削除するサンプルです。送信データの仕様は関連付けを行う時と同様です。

```
// $ node scripts/associations.js deleteAssociation 309851 1903970621

request({
  url:
`https://api.hubapi.com/crm-associations/v1/associations/delete?hapikkey=${API_KEY}`,
  method: "PUT",
  json: true,
  body: {
    "fromObjectId": fromId,
    "toObjectId": toId,
    "category": "HUBSPOT_DEFINED",
    "definitionId": 1
  }
}).then((r) => {
  console.log(r)
}).catch((e) => {
  console.log(e.message)
})
```

参考) Delete an association:

<https://developers.hubspot.com/docs/methods/crm-associations/delete-association>

8. Forms API

HubSpotのフォーム送信をプログラムで行うためのAPIです。Webページ内から実行できるAjax版とサーバーサイドプログラム版の二種類のForms APIがあります。

8-1. 事前準備

Forms APIを利用するためには、事前に該当のHubSpotのポータルでフォームを作成しIDを発行する必要があります。Forms APIで利用したいフィールドを持つフォームを作成して下さい。

The screenshot shows the HubSpot Forms Editor interface. The browser address bar displays the URL: `https://app.hubspot.com/forms/4148814/editor/ad9898b1-0ac4-4a99-bdc9-f879c0e05f48/edit/form`. The page title is "新規フォーム (2019年4月13日 13:39:01) (1)". The interface includes a search bar for form fields, a list of contact properties, and a preview of the form with fields for "姓" (Last Name), "名" (First Name), "Eメール*" (Email), and "電話番号" (Phone Number). A red arrow points to the timestamp "13:39:01" in the page title, indicating the form ID.

作成したフォームのアドレスバーの「editor/」から「/edit/form」の間にある値がフォームのIDです。この値をForms APIで利用します。

参考) フォームの GUID を見つける:

https://knowledge.hubspot.com/jp/articles/kcs_article/forms/find-your-form-guid

尚、公開されていないフォームはForms APIで利用できませんので、ご注意ください。

8-2. フォームの送信 (Ajax)

エンドポイントURL「<https://api.hsforms.com/submissions/v3/integration/submit/{ポータルID}/{フォームID}>」にJSONデータを送信することでフォームを送信できます。利用するHTTPメソッドはPOSTです。APIの実行が完了するとコンタクトが（存在しない場合は）作成され、コンタクトのタイムラインにフォーム送信のログが追加されます。またブラウザのCookieとHubSpotコンタクトとの紐付けが行われます。

下記はサンプルコードです。（これまでの例と異なりWebブラウザ上での実行を想定しています）

```
// $ node browser/forms.js
// The app listening on http://localhost:3000

$.ajax({
  url:
    "https://api.hsforms.com/submissions/v3/integration/submit/4148814/ad9898b1-0ac4-4a99-bdc9-f879c0e05f48",
  type: "POST",
  dataType: "json",
  contentType: "application/json; charset=UTF-8",
  data : JSON.stringify({
    fields: [
      {
        name: "lastname",
        value: $("[name=lastname]").val()
      },
      {
        name: "firstname",
        value: $("[name=firstname]").val()
      },
      {
        name: "email",
        value: $("[name=email]").val()
      },
      {
        name: "phone",
        value: $("[name=phone]").val()
      }
    ],
    context: {
      hutk: getCookie("hubspotutk"),
      pageUri: window.location.href,
      pageName: document.title
    }
  })
});
```

```
    }  
  }},  
}).done(function(res) {  
  console.log(res)  
})
```

context/パラメータで後のアナリティクスで有用な情報を指定しています。これらはオプションですが、ぜひ指定されることをお奨めします。「hutk」パラメータではHubSpotのトラッキングコードが設定する「hubspotutk」の値を設定しています。Cookieの値は下記のような関数で取得して下さい。

```
window.getCookie = function (name) {  
  var nameEQ = name + "="  
  var ca = document.cookie.split(";")  
  for (var i = 0; i < ca.length; i++) {  
    var c = ca[i]  
    while (c.charAt(0) == " ") c = c.substring(1, c.length)  
    if (c.indexOf(nameEQ) == 0) return c.substring(nameEQ.length, c.length)  
  }  
  return null  
}
```

The screenshot shows a web browser window with the title "Forms API サンプル". The address bar shows "localhost:3000". The form contains the following elements:

- Input field: 川野
- Input field: 忍
- Input field: skawano+forms123@hu
- Input field: 1234-5678
- Radio buttons: Ajax版, サーバーサイド版
- Submit button: 送信

参考) Submit data to a form:

https://developers.hubspot.com/docs/methods/forms/submit_form_v3

8-3. フォームの送信（サーバーサイドプログラム）

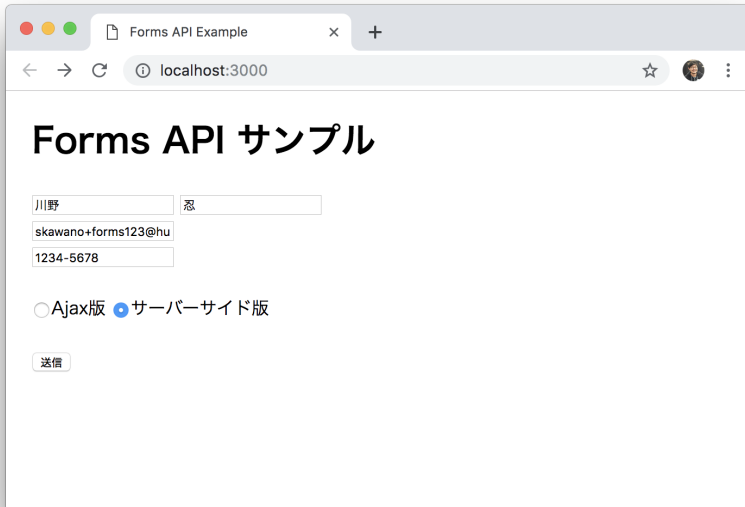
エンドポイントURL「<https://forms.hubspot.com/uploads/form/v2/{ポータルID}/{フォームID}>」にJSONデータを送信することでフォームを送信できます。利用するHTTPメソッドはPOSTです。本APIはサーバーサイドプログラムから実行する必要があります。

下記はサンプルコードです。（Node.jsランタイム上での実行を想定しています）

```
// $ node browser/forms.js
// The app listening on http://localhost:3000

request({
  url:
`https://forms.hubspot.com/uploads/form/v2/${PORTAL_ID}/${FORM_GUID}?hapikey=${API_KEY}`,
  method: "POST",
  headers: {
    "Content-Type": "application/x-www-form-urlencoded",
  },
  body: querystring.stringify({
    "firstname": req.body.firstname,
    "lastname": req.body.lastname,
    "email": req.body.email,
    "phone": req.body.phone,
    "hs_context": {
      "hutk": req.cookies.hubspotutk,
      "pageUrl": "http://www.example.com/form-page",
      "pageName": "Forms API Example"
    }
  })
}).then(() => {
  res.send("Thank you!")
}).catch(() => {
  res.send("Error...")
})
```

サーバーサイドからAPIを実行する際は必ず送信データにCookie「hubspotutk」の値を含めて下さい。これにより、ブラウザのCookieとHubSpotのコンタクトの紐付けが行われます。



参考) Submit data to a form:

https://developers.hubspot.com/docs/methods/forms/submit_form

9. Tracking Code API

訪問者のWebサイト閲覧履歴を記録するためのAPIです。HubSpotのトラッキングコードをページにインストールすることで、閲覧の履歴は自動的にHubSpotのコンタクトタイムラインに蓄積されます。しかし、シングルページWebアプリケーションなどURLが変わらないページの場合は画面遷移の情報は蓄積されません。本APIを利用することで、そのようなケースでも情報を蓄積できます。

9-1. ページ閲覧履歴の記録

下記はシングルページWebアプリでの画面遷移をトラッキングするサンプルです。ハッシュフラグメント変更イベント（onhashchange）のリスナーでHubSpotにページビュー情報を送信しています。

HTML

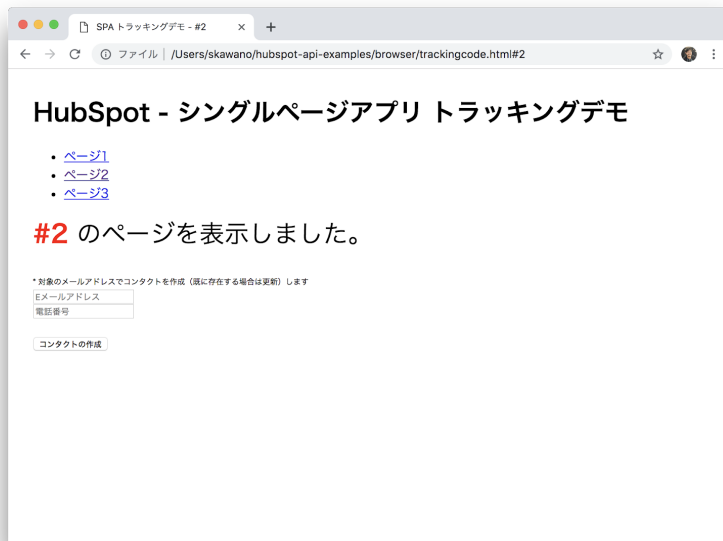
```
<body style="margin: 2em" onhashchange="render()">
  <h1>HubSpot - シングルページアプリ トラッキングデモ</h1>
  <ul>
    <li><a href="#1">ページ1</a></li>
    <li><a href="#2">ページ2</a></li>
    <li><a href="#3">ページ3</a></li>
  </ul>
  ...
```

JavaScript

```
function render() {
  var elem = document.getElementById('content');
  switch (location.hash) {
    case '#1':
      ...
    case '#2':
      ...
    case '#3':
      ...
  }
  _hsq.push(['setPath', location.hash]);
  _hsq.push(['trackPageView']);
}
```

ブラウザで「trackingcode.html」ファイルを開いてサンプルページを表示下さい。

```
$ open browser/trackingcode.html
```



4月 2019



ページビュー

2019年4月13日の23:07 GMT+9

HubSpot - シングルページアプリ トラッキングデモとその他8ページ

▼ セッションの詳細

今日 23:07 GMT+9	○	閲覧されたSPA トラッキングデモ - #1
今日 23:07 GMT+9	○	閲覧されたSPA トラッキングデモ - #2
今日 23:07 GMT+9	○	閲覧されたSPA トラッキングデモ - #3
今日 23:07 GMT+9	○	閲覧されたSPA トラッキングデモ - #2
今日 23:07 GMT+9	○	閲覧されたSPA トラッキングデモ - #1

さらに表示



参考) Track Page View:

https://developers.hubspot.com/docs/methods/tracking_code_api/track_page_view

9-2. 訪問者の特定

HubSpotのトラッキングコードがインストールされたページにおいて、下記のコードを実行することでブラウザのCookieとHubSpotのコンタクトを紐づけることができます。「email」プロパティで指定したメールアドレスのコンタクトが存在しない場合は新規で作成します。

```
_hsq.push(["identify", {  
  email: document.getElementById('email').value  
}]);  
_hsq.push(['trackPageView']);
```

また本APIを利用してコンタクトのプロパティの値を更新することも可能です。

```
_hsq.push(["identify", {  
  email: document.getElementById('email').value,  
  phone: document.getElementById('phone').value // 電話番号プロパティの値を更新  
}]);  
_hsq.push(['trackPageView']);
```

* 対象のメールアドレスでコンタクトを作成（既に存在する場合は更新）します

skawano+forms123@hu
03-1234-5678

コンタクトの作成



参考) Identify a visitor:

https://developers.hubspot.com/docs/methods/tracking_code_api/identify_visitor

10. HubSpotアプリ

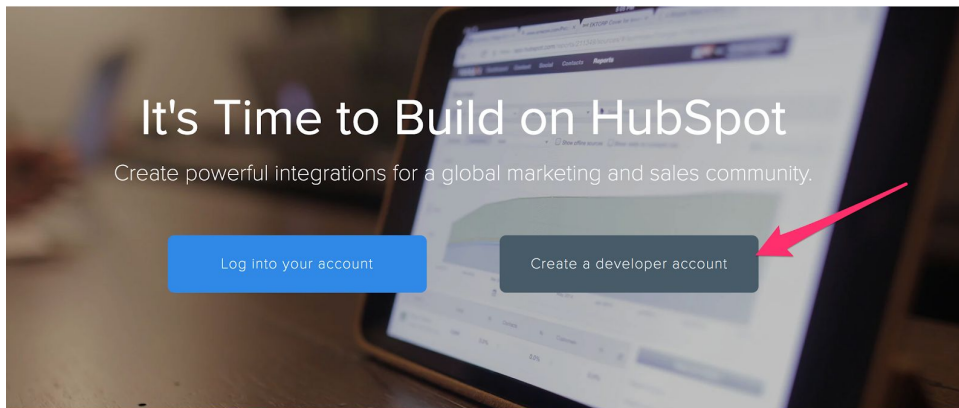
HubSpotアプリを作成することで、自社開発のソフトウェアサービスとHubSpotの連携プログラムを広く配布することができます。HubSpotアプリのマーケットプレイス「[HubSpot Connect](#)」では200以上のHubSpotアプリが公開されており、ユーザーが必要な機能を探して自身のポータルにアプリをインストールできるようになっています。

10-1. 開発者アカウントの作成

HubSpotアプリを作成するには、まずは開発者アカウントを作成する必要があります。開発者アカウントは下記サイトの「Create a Developer account」ボタンから作成して下さい。

HubSpot Developer Site:

<https://developers.hubspot.com/>



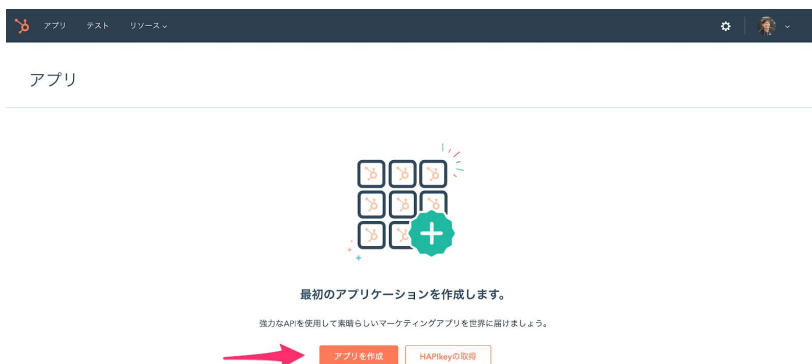
Why Build on HubSpot?

HubSpot is the central nervous system of tens of thousands of sales and marketing teams across the world. When you build on HubSpot, you plug directly into how they work.

開発者アカウントの作成後、「開発者ポータル」にログインできるようになります。開発者ポータルではHubSpotアプリの設定やテストポータルの作成、API実行ログの確認など、HubSpotアプリの運用に必要な作業を行うことができます。

10-2. HubSpotアプリの作成

開発者ポータルの「アプリ」メニューにある「アプリを作成」ボタンからHubSpotアプリを作成します。



アプリを作成 ×

アプリ名

アプリケーションの共有範囲

公開
すべてのポータルでこのアプリケーションを活用できる可能性があります。

非公開
このアプリケーションを活用できるのはあなたのポータルのみです。

アプリの配布を予定している場合は、共有範囲で「公開」をご選択下さい。

作成したアプリが一覧に追加され、詳細設定を行うことができますようになります。

アプリ テスト リソース

Shinobu Developer

< すべてのアプリに戻る

HubSpot API Sample App

- 詳細
- モニタリング
- 機能
 - ウェブフックサブスクリプション
 - タイムラインAPIイベント
 - CRM拡張API

アプリケーションの詳細

アプリID

アプリ名 *

作成者名

顧客ID

顧客の秘密

サポートの詳細

サポートの電話番号

設定画面では、アプリのロゴ画像やOAuth認証で許可を得る権限スコープ、公開時のアプリケーション情報（サポートメールアドレス、ドキュメントページのURL）などを設定します。

参考) How do I create an app in HubSpot?:

<https://developers.hubspot.com/docs/faq/how-do-i-create-an-app-in-hubspot>

参考) How do I create a test account?:

<https://developers.hubspot.com/docs/faq/how-do-i-create-a-test-account>

参考) Using app request monitoring:

<https://developers.hubspot.com/docs/faq/api-request-monitoring>

11. OAuth認証

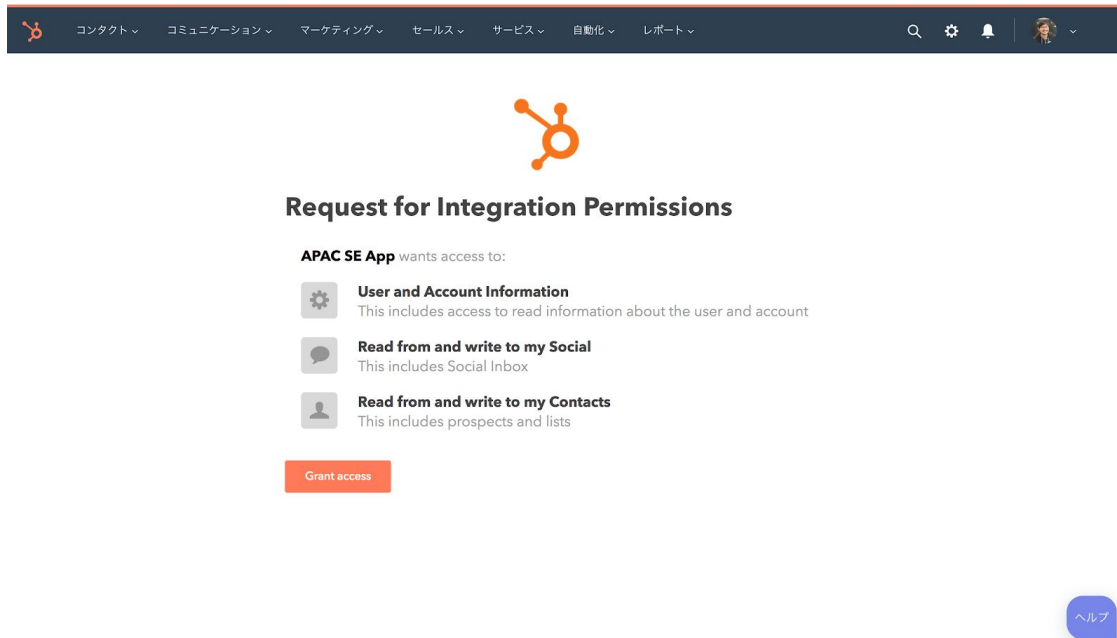
HubSpotアプリを配布するためにはOAuth認証のフローを作成する必要があります。アプリの設定画面で表示されている「顧客ID」や「顧客の秘密」のキーを利用してOAuthのフローを作成します。

11-1. インストールリンクの作成

まずはじめに、インストールリンクを作成します。下記はリンクのサンプルです。

```
https://app.hubspot.com/oauth/authorize?scope=contacts%20social&redirect_uri=https://www.example.com/auth-callback&client_id=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

リンクをクリックすることで、ユーザーはアプリのインストール画面にアクセスできます。



作成したHubSpotアプリを提供したい場合、該当アプリのインストールリンクをお客様に配布頂ければと存じます。またHubSpot Connectへの掲載にご興味がありましたら、[Connectプログラムのフォーム](#)よりお問い合わせ下さい。

インストールリンクでは、所定のURL「https://app.hubspot.com/oauth/authorize」のパラメータに認証に必要な下記3つの情報を付与します。

- 顧客ID
- スコープ
- リダイレクトURL

顧客IDは「client_id」パラメータで指定します。アプリ画面で発行されている「顧客ID」の値を指定して下さい。

スコープは「scope」パラメータで指定します。上記のサンプルでは「コンタクト」と「ソーシャル」へのアクセス権限を得るために「scope=contacts%20social」と指定しています。

設定するパラメータの値はアプリ設定画面で確認できます。

スコープ

ここでスコープを選択するには、ユーザーが統合を接続したときに認証URLに存在している必要があります。統合で任意のスコープを要求できることにご注意ください。ここで選択するのは、任意のセキュリティの強化対策です。

スコープエリパラメータ

&scope=

認証URLでこのスニペットを使用し、下で選択したスコープを要求します。認証URLの構築についてもっと詳しく [☞](#)

- アクションフォームを追加 ?
- タイムラインイベントを作成 ?
- ビジネスインテリジェンスAPIからの読み取り ?
- OAuthの基本機能 ?
- トランザクションEメールへのアクセス ?
- 連携同期機能へのアクセス ?
- Eコマースへのアクセス ?

読み取り/書き込み

- | | | |
|--|--|---|
| <input checked="" type="checkbox"/> コンタクト <small>?</small> | <input type="checkbox"/> コンテンツ <small>?</small> | <input type="checkbox"/> ファイル <small>?</small> |
| <input type="checkbox"/> フォーム <small>?</small> | <input type="checkbox"/> レポート <small>?</small> | <input type="checkbox"/> HubDB <small>?</small> |
| <input checked="" type="checkbox"/> ソーシャル <small>?</small> | <input type="checkbox"/> ワークフロー <small>?</small> | <input type="checkbox"/> チケット <small>?</small> |

リダイレクトURLは「redirect_uri」パラメータで指定します。ユーザーによる権限の付与が完了した後、リダイレクトさせたいURLを指定して下さい。

11-2. アクセストークンの取得

ユーザーによる権限の付与が行われると、インストールリンクの「redirect_uri」パラメータで指定したURLへのリダイレクトが行われます。リダイレクトの際、「code」というパラメータが付与され、その値を利用して該当ポータルへのデータにアクセスするためのトークンを取得できます。

```
https://www.example.com/auth-callback?code=XXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXX
```

エンドポイントのURL「https://api.hubapi.com/oauth/v1/token」にフォーム形式で所定の情報を送信することでアクセストークンを取得します。利用するHTTPメソッドはPOSTです。

下記はアクセストークンを取得するサンプルです。「grant_type」に「authorization_code」というキーワードを指定し、「client_id」には「顧客ID」の値を、「client_secret」には「顧客の秘密」の値を、「redirect_uri」にはインストールリンクで指定したリダイレクトURLを、「code」にはリダイレクトされた際にパラメータに追加された「code」の値を指定して下さい。

```
const tokens = await exchangeForTokens({
  grant_type: "authorization_code",
  client_id: CLIENT_ID,
  client_secret: CLIENT_SECRET,
  redirect_uri: REDIRECT_URI,
  code: req.query.code
})
```

その後、指定のエンドポイントにリクエストを送信するとアクセストークンが返されます。

```
exports.exchangeForTokens = async (form) => {
  try {
    const responseBody = await
request.post("https://api.hubapi.com/oauth/v1/token", {
  form
})
    return JSON.parse(responseBody)
  } catch (e) {
    log.error(e)
    throw new Error()
  }
}

=> {
  "access_token": "xxxx",
```

```
"refresh_token": "yyyyyyyyy-yyyy-yyyy-yyyyyyyyyyyy",  
"expires_in": 21600  
}
```

このトークンを利用して対象ポータルからデータを取得できます。下記はアクセストークンを利用して
コンタクト情報を更新するサンプルです。

```
// app/services/hs.js  
  
exports.getContactByEmail = async (email, accessToken) => {  
  try {  
    const responseBody = await  
    request.get(`https://api.hubapi.com/contacts/v1/contact/email/${email}/profile`,  
    {  
      headers: {  
        "Authorization": "Bearer " + accessToken  
      },  
      json: true  
    })  
    return responseBody  
  } catch (e) {  
    log.error(e)  
    throw new Error()  
  }  
}  
  
=> {  
  email: "bh@hubspot.com",  
  contact: {  
    vid: 51,  
    ...  
    properties: {  
      firstname: {  
        value: "Brian",  
        ...  
      }  
    }  
  }  
}
```

参考) OAuth 2.0 Quickstart Guide:

<https://developers.hubspot.com/docs/methods/oauth2/oauth2-quickstart>

参考) Initiate an Integration with OAuth 2.0:

<https://developers.hubspot.com/docs/methods/oauth2/initiate-oauth-integration>

参考) Using OAuth 2.0 Access Tokens:

<https://developers.hubspot.com/docs/methods/oauth2/using-access-tokens>

11-3. アクセストークンの更新

アクセストークンの有効期限は6時間のため、リフレッシュトークンを利用して定期的のアクセストークンを再取得する必要があります。

下記はアクセストークンを再取得するサンプルです。バッチプログラムで定期的に行うことを想定しています。

```
// app/jobs/app.js
const log = require("log4js").getLogger("jobs/app")
const db = require("../services/db")

const path = require("path")
require("dotenv").config({
  path: path.join(__dirname, "../.env")
})

const APP_ID = process.env.APP_ID
const CLIENT_ID = process.env.APP_CLIENT_ID
const CLIENT_SECRET = process.env.APP_CLIENT_SECRET
const SERVER_URL = process.env.SERVER_URL
const REDIRECT_URI = `${SERVER_URL}/oauth-callback`

const request = require("request-promise-native")

exports.refreshAccessToken = async () => {
  const accounts = await db.findAccounts({
    appId: APP_ID
  })

  accounts.forEach(async (account) => {
    try {
      const responseBody = await request({
        method: "POST",
        url: "https://api.hubapi.com/oauth/v1/token",
        headers: {
          "Content-Type": "application/x-www-form-urlencoded;charset=utf-8"
        },
        form: {
          "grant_type": "refresh_token",
          "client_id": CLIENT_ID,
```

```

        "client_secret": CLIENT_SECRET,
        "redirect_uri": REDIRECT_URI,
        "refresh_token": account.refreshToken
    }
  })

  const info = JSON.parse(responseBody)
  await db.updateAccount({
    email: account.email,
    hubId: account.hubId,
    appId: account.appId
  }, {
    accessToken: info.access_token,
    refreshToken: info.refresh_token
  })
} catch (e) {
  log.error(e)
}
})
}

```

```

// app/index.js
...

/* Job Config */
const schedule = require("node-schedule")
const appJob = require("./jobs/app")
schedule.scheduleJob({ hour: 23, minute: 0 }, appJob.refreshAccessToken)
schedule.scheduleJob({ hour: 3, minute: 0 }, appJob.refreshAccessToken)
schedule.scheduleJob({ hour: 7, minute: 0 }, appJob.refreshAccessToken)
schedule.scheduleJob({ hour: 11, minute: 0 }, appJob.refreshAccessToken)

```

参考) Get OAuth 2.0 Access Token and Refresh Tokens:

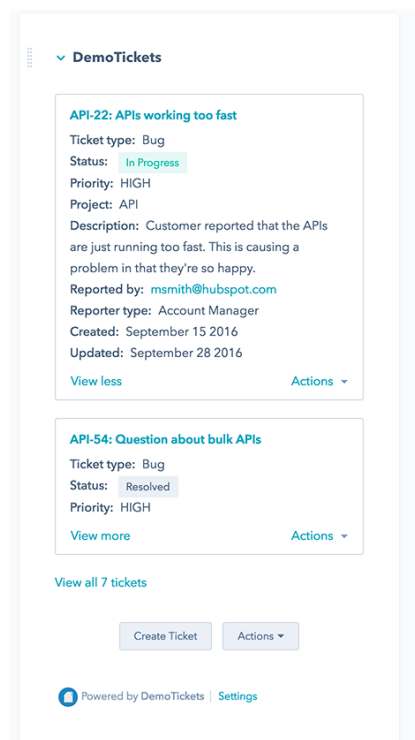
<https://developers.hubspot.com/docs/methods/oauth2/get-access-and-refresh-tokens>

参考) Refresh OAuth 2.0 Access Token:

<https://developers.hubspot.com/docs/methods/oauth2/refresh-access-token>

12. CRM拡張API

HubSpotアプリを作成することで、CRM拡張APIを利用できるようになります。本APIではHubSpotのコンタクト詳細画面、会社詳細画面、取引詳細画面に独自の「カード」を追加することができます。



カードでは、任意のデータを一覧で表示させたり、ボタンを追加して特定の処理を実行させることができます。

12-1. アプリの設定

CRM拡張APIを利用するためには、まずはアプリの設定を行います。アプリの「CRM拡張API」メニューを選択し設定画面を表示して下さい。最初にオブジェクトタイプを作成します。

[すべてのアプリに戻る](#)

HubSpot API Sample App

詳細

モニタリング

機能

ウェブフックサブスクリプション

タイムラインAPIイベント

CRM拡張API



製品をCRMに活用しましょう。

関連オブジェクトをHubSpot内コンタクト、会社、取引に関するレコードにリンクします。

オブジェクトタイプを作成



ヘルプ

オブジェクトタイプの名前とその定義情報を取得するためのURL（データ取得URI）を指定します。

オブジェクトタイプを作成

タイトル *

データ取得URI * i

作成 キャンセル

その後、詳細画面でカードを追加したいHubSpotオブジェクトを指定して下さい。

アプリ テスト リソース

Shinobu Developer

< すべてのアプリに戻る

HubSpot API Sample App

詳細

モニタリング

機能

ウェブフックサブスクリプション

タイムラインAPIイベント

CRM拡張API

< すべてのオブジェクトタイプに戻る

HubSpot API Sample App

データ取得URI

https://hubspot-api-sample-app.herokuapp.com/crm-extension

基本URI

+ 基本URIを追加

関連付けられたHubSpotオブジェクト

関連付けられたHubSpotオブジェクト

コンタクト

プロパティ

会社

取引

ヘルプ

参考) CRM Extensions API:

<https://developers.hubspot.com/docs/methods/crm-extensions/crm-extensions-overview>

12-2. サーバープログラムの準備

データ取得URIにアクセスした際に、所定のJSONデータを返却するよう準備します。下記はiFrameウィンドウを開くボタンを持つカードを指定するサンプルです。

```
// app/controllers/index.js

router.get("/crm-extension", (req, res) => {
  res.json({
    "results": [],
    "primaryAction": {
      "type": "IFRAME",
      "width": 890,
      "height": 748,
      "uri":
`_${SERVER_URL}/hubid/${req.query.portalId}/page?userEmail=${req.query.userEmail}
`,
      "label": "CRM Extension Page",
      "associatedObjectProperties": [
        "email"
      ]
    }
  })
})
```

```

    }
  })
})

=> {
  "results": [],
  "primaryAction": {
    "type": "IFRAME",
    "width": 890,
    "height": 748,
    "uri":
    "https://yourapp.herokuapp.com/hubid/undefined/page?userEmail=undefined",
    "label": "CRM Extension Page",
    "associatedObjectProperties": [
      "email"
    ]
  }
}
}

```

下記はボタンをクリックした際に発行されるリクエストを処理するサーバープログラムです。コンタクト画面から引き渡された情報をウィンドウに表示します。

```

// app/controllers/index.js

router.get("/hubid/:hubId/page", async (req, res) => {
  try {
    res.json({
      success: true,
      email: req.query.email,
      userEmail: req.query.userEmail
    })
  } catch (e) {
    res.sendStatus(500)
  }
})

```

オブジェクトの詳細画面を表示する際にデータ取得URIから拡張の定義データを取得します。定義が正しく行われている場合、カードに情報が表示されます。

The screenshot shows the HubSpot CRM interface. At the top, there are navigation tabs for 'サービス' (Services), '自動化' (Automation), and 'レポート' (Reports). A search bar, settings gear, and notification bell are also visible. The user's name 'ShinobuSpot' is in the top right. Below the navigation, there are tabs for 'Eメール' (Email), 'コール' (Calls), and 'タスク' (Tasks). The main content area shows a list of items, with the first item being 'HubSpot API Sample App'. A red arrow points to the 'CRM Extension Page' button next to this item. Below this, there is a section for '会社 (1)' (Company (1)) and '関連する会社数 (0)' (Number of related companies (0)), with a '会社を追加' (Add company) button.

ボタンをクリックするとウィンドウが開き情報が表示されます。

The screenshot shows a modal window titled 'HubSpot API Sample App - CRM Extension Page'. The window has a teal header with a close button (X). The main content area displays a JSON response: `{"success":true,"email":"skawano+forms123@hubspot.com","userEmail":"skawano@hubspot.com"}`. A red arrow points to the JSON data. In the bottom right corner, there is a 'ヘルプ' (Help) button.

13. タイムラインAPI

HubSpotアプリを作成することで、タイムラインAPIを利用できるようになります。本APIでは、コンタクトのタイムラインに任意の情報を表示するカードを追加することができます。

13-1. アプリの設定

タイムラインAPIを利用するためには、まずはアプリの設定を行います。アプリの「タイムラインAPIイベント」メニューを選択し設定画面を表示して下さい。ここではイベントタイプを作成し、タイムラインに追加する対象のHubSpotオブジェクトを指定します。

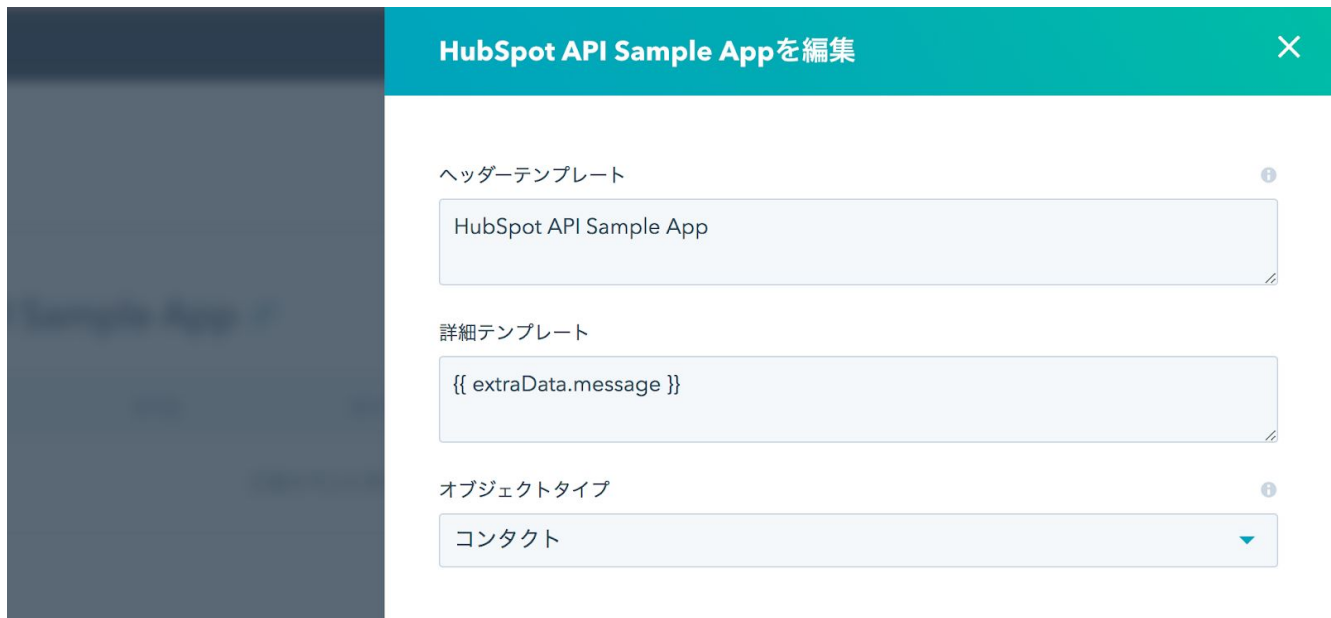


イベントタイプを作成するとIDが採番されます。このIDをAPIで利用します。

詳細	イベントタイプ	イベントタイプを作成
モニタリング		
機能		
ウェブフックサブスクリプション		
タイムラインAPIイベント		
CRM拡張API		

名前	ヘッダー	ID	オブジェクトタイプ
Approval	✓	389949	コンタクト

続いて、タイムラインのカードでデータを表示するためのテンプレートを作成します。



参考) Timeline API Overview:

<https://developers.hubspot.com/docs/methods/timeline/timeline-overview>

13-2. サーバープログラムの準備

エンドポイントURL 「`https://api.hubapi.com/integrations/v1/${appId}/timeline/event`」 にJSONデータを送信することで「`eventTypeid`」で指定したイベントタイプのカードをタイムラインに追加できます。利用するHTTPメソッドはPUTです。

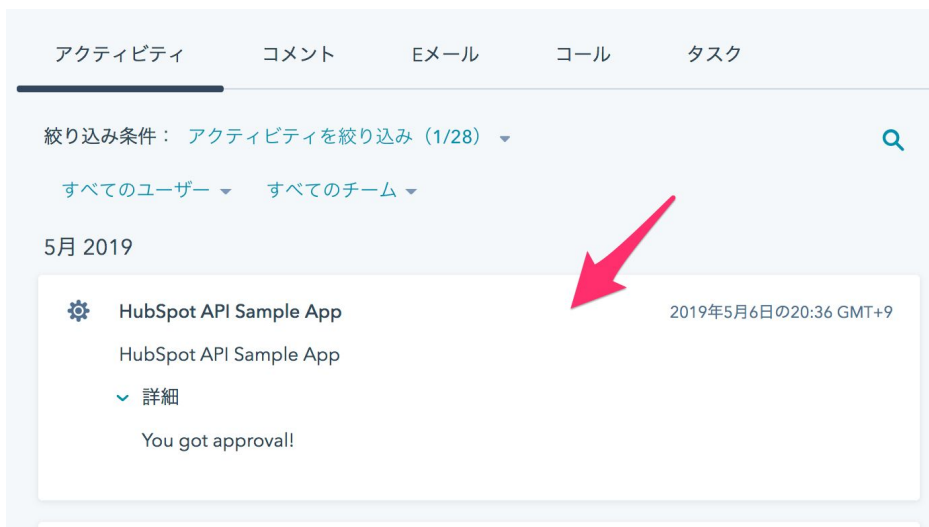
```
// app/services/hs.js

exports.createTimelineEvent = async (email, appId, eventTypeid, accessToken) =>
{
  try {
    const responseBody = await
request.put(`https://api.hubapi.com/integrations/v1/${appId}/timeline/event`, {
  headers: {
    "Authorization": "Bearer " + accessToken
  },
  json: true,
  body: {
    "id": Date.now(),
    eventTypeid,
    email,
    "extraData": {
      "message": "You got approval!"
    }
  }
}
```

```
    }
  })
  return responseBody
} catch (e) {
  log.error(e)
  throw new Error()
}
}
```

上記のサンプルでは対象コンタクトの特定にメールアドレスを利用しています。

```
fish /Users/skawano/git/hubspot-api-examples/app
~/g/h/app on master curl https://hubspot-api-sample-app.herokuapp.com/hubid/4148814/timeline/skawano+forms123@hubspot.com -X PUT
{"hubId":"4148814","email":"skawano+forms123@hubspot.com","success":true}
~/g/h/app on master 20:3
```



参考) Create or update a timeline event:

<https://developers.hubspot.com/docs/methods/timeline/create-or-update-event>

14. ウェブフックAPI

HubSpotアプリを作成することで、ウェブフックAPIを利用できるようになります。本APIではHubSpotオブジェクトの状態をトリガーに、外部プログラムに該当オブジェクトのデータを送信します。

14-1. アプリの設定

ウェブフックAPIを利用するためには、まずはアプリの設定を行います。アプリの「ウェブフックサブスクリプション」メニューを選択し設定画面を表示して下さい。続いて「サブスクリプション作成」ボタンでトリガー対象のHubSpotオブジェクトを指定します。

サブスクリプションでは、コンタクト、会社、または取引オブジェクトの「作成」「削除」「更新」のタイミングを指定することができます。「更新」イベントの場合は、対象のプロパティを指定して下さい。標準プロパティの他、カスタムプロパティも指定可能です。

ウェブフックサブスクリプションを作成 ×

興味のあるオブジェクトと詳しく知りたいイベントを選択して、ウェブフックサブスクリプションを作成してください。

どのオブジェクトタイプを選びますか？

コンタクト 会社 取引

どのイベントについて知りたいですか？

作成済み 削除 更新完了

どのプロパティを選びますか？

プロパティを選択するか、入力してください ▼

HubSpotの標準プロパティから選択するか、カスタムプロパティを入力します。

キャンセル 次へ

サブスクリプションの作成が完了したら、該当イベントが発生した際のデータを受け取るURLを指定します。

< すべてのアプリに戻る

HubSpot API Sample App

詳細

モニタリング

機能

ウェブフックサブスクリプション

タイムラインAPIイベント

CRM拡張API

ウェブフック

ターゲットURL

HubSpotはこのURL宛にJSONプレイロードを送信し、イベントがトリガーされたときにその詳細をお知らせします。
ウェブフックAPIについての詳細

イベントの抑制

10 1秒あたり

HubSpotがいつでも設定済みのURLに送信を試みるイベントの最大数。

サブスクリプションを作成

イベントのサブスクリプション

<input type="checkbox"/>	オブジェクト	イベント	イベント	エラー	ステータス
<input type="checkbox"/>	👤 コンタクト	作成	0	0	● 停止中

URL指定後、サブスクリプションを有効にして下さい。これによりイベント発生時に所定のデータが指定したURLに送信されます。

ウェブフックサブスクリプションの詳細

このサブスクリプションは有効です。 オン

このサブスクリプションは、次のイベントが発生したときにトリガーされます。

オブジェクト	イベント
👤 コンタクト	作成

このサブスクリプションをテスト

URL

URLを入力してこのサブスクリプションをテスト

JSONペイロードの例

```
[
  {
    "eventId": 1,
    "subscriptionId": 118651,
    "portalId": 5808466,
    "occurredAt": 1557132549196,
    "subscriptionType": "contact.creation",
    "attemptNumber": 0,
    "objectId": 123,
    "changeSource": "CRM",
    "changeFlag": "NEW",
    "appId": 193354
  }
]
```

14-2. サーバープログラムの準備

下記はウェブフックのリクエストを受け取るサンプルプログラムです。リクエストはPOSTメソッドで発行される点にご注意下さい。

```
// app/controllers/index.js

router.post("/webhook", (req, res) => {
  log.info(req.body)
  res.sendStatus(200)
})
```

発行されたリクエストについてはアプリの「モニタリング」メニューで確認できます。



The screenshot shows the HubSpot Developer Console interface. At the top, there are navigation links for 'アプリ', 'テスト', and 'リソース'. The user is identified as 'Shinobu Developer'. The main heading is 'HubSpot API Sample App'. Below this, there are three tabs: 'APIコール', 'ウェブフック', and 'CRM拡張機能'. A red arrow points to the 'ウェブフック' tab. Underneath, there are filters for '絞り込み条件' (Response, Status, Subscription, Today) and a search bar for '試行ID' and 'ログを検索'. A table displays the following data:

機能	HTTP	サブスクリプションタイプ	バッチID	試行開始	
ウェブフックサブスクリプション	●	200	contact.creation	700be729	2019年5月6日 20:41

At the bottom, there are pagination controls: '< 前 1 次へ >' and 'ページあたり 25'.

参考) Webhooks Overview:

<https://developers.hubspot.com/docs/methods/webhooks/webhooks-overview>

15. おわりに

15-1. 開発で困った時は

HubSpotコミュニティで開発者の皆様をサポートしています。英語でのコミュニケーションが必要になりますが、ぜひご活用下さい。

HubSpot Developers Forum:

<https://community.hubspot.com/t5/APIs-Integrations/bd-p/integrations>

またHubSpot APIのアップデートについては下記のブログで公開しています。

HubSpot Developers Changelog:

<https://developers.hubspot.com/changelog>

仕様変更など重要なアナウンスもありますので、ぜひご購読頂ければと存じます。

15-2. 作者について

本ガイドはHubSpotセールスエンジニアの川野 忍が作成しました。

私は過去に大企業向けCRMパッケージソフトウェアの研究開発に携わり、前職の米Sencha, Inc.では商用JavaScriptフレームワーク「Ext JS」の導入を行うWebエンジニアへの技術支援を行なっていました。Ext JSはSalesforceやMarketo、Xeroなど多くの成功したSaaSベンダーの製品で利用されています。

現在はHubSpot製品の導入を検討されているお客様に対して技術面での支援を行なっています。また週末は個人で運営しているソフトウェアサービスの開発とメンテナンスをしています。