Hubster.io Documentation

Release 1.0.0

Hubster.io

Dec 29, 2020

Introduction

Hubster is an open-ended *Unified Message Platform as a Service* **(PaaS)** and like all good platforms, we try our best to adhere to industry standards and best practices.

So, what is meant by an **Open-ended** platform? An open-end platform allows a business to extend the platform by enabling the business to bring in their own integration or plugins.

At Hubster, we provide following ways on how you can extend our platform:

Integration (BYOI):
If you have an integration that Hubster does not currently support or it is unique to your business, you can easily add it to the mix using our direct API. There are no limits on how many customer, agent, bot, or CRM integrations you can add.

Pipeline plugins (BYOP)
Businesses can control the message pipeline by injecting their custom plugins. You can enrich messages, alter, or even control the follow how messages are redirected to any given participant involved in the conversation.

Webhooks
Businesses can add their own webhooks and will only be triggered based on *filter rules*. This is a great way to monitor certain activities that are important to the business. It should be noted that standard Webhooks are based on a one-way communication protocol know as *fire-and-forget*.

Dynamic Commands
Dynamic commands are a powerful concept allowing agents and/or bots through simple text, instruct Hubster to invoke your backend service to formulate the appropriate response. For example, if your business provides a command to a list a line of clothing specific to the user's profile, your system can check to see what are the best options and construct a targeted response that is meaningful to the end user.

Our *APIs* are designed using **REST** principles and most of our payloads are structured using **JSON**. Any exception to this rule will be noted where necessary.

Note: Hubster APIs incorporate cross-origin resource sharing (**CORS**) whereby facilitating web applications to freely use our API in an authenticated and secure manner.

Please help us make this experience even better
If you find any errors or a section is not as clear or lacking details, please don't hesitate to contact us at support@hubster.io

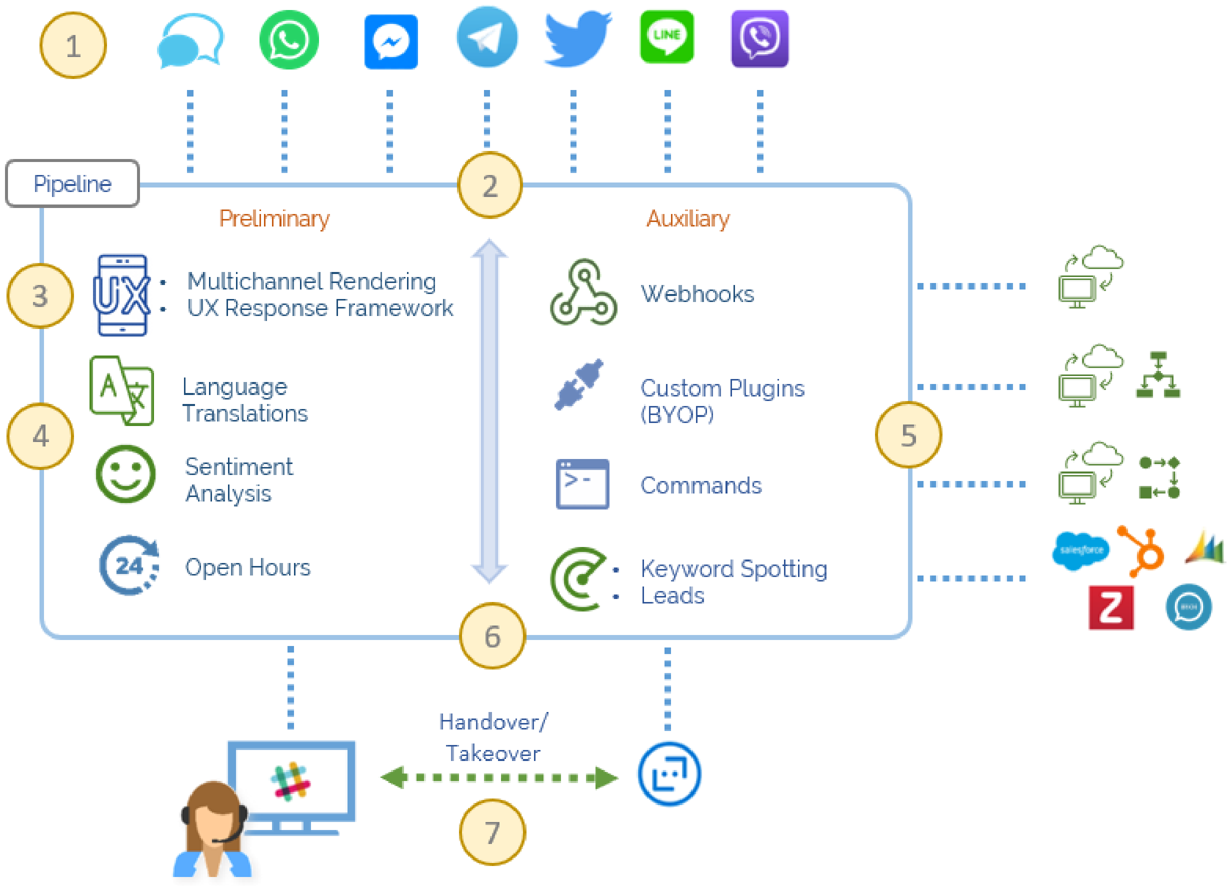
CHAPTER 1

The Big Picture

1.1 High Level Architecture

Hubster's **open-ended** platform was designed for simplicity, yet power enough to allow a business to extend the platform to meet their specific needs, on a per *hub* basis. Being **open-end** provides a business the flexibility to enrich the messaging pipeline by injecting their own custom *integrations and plugins.*

Engine

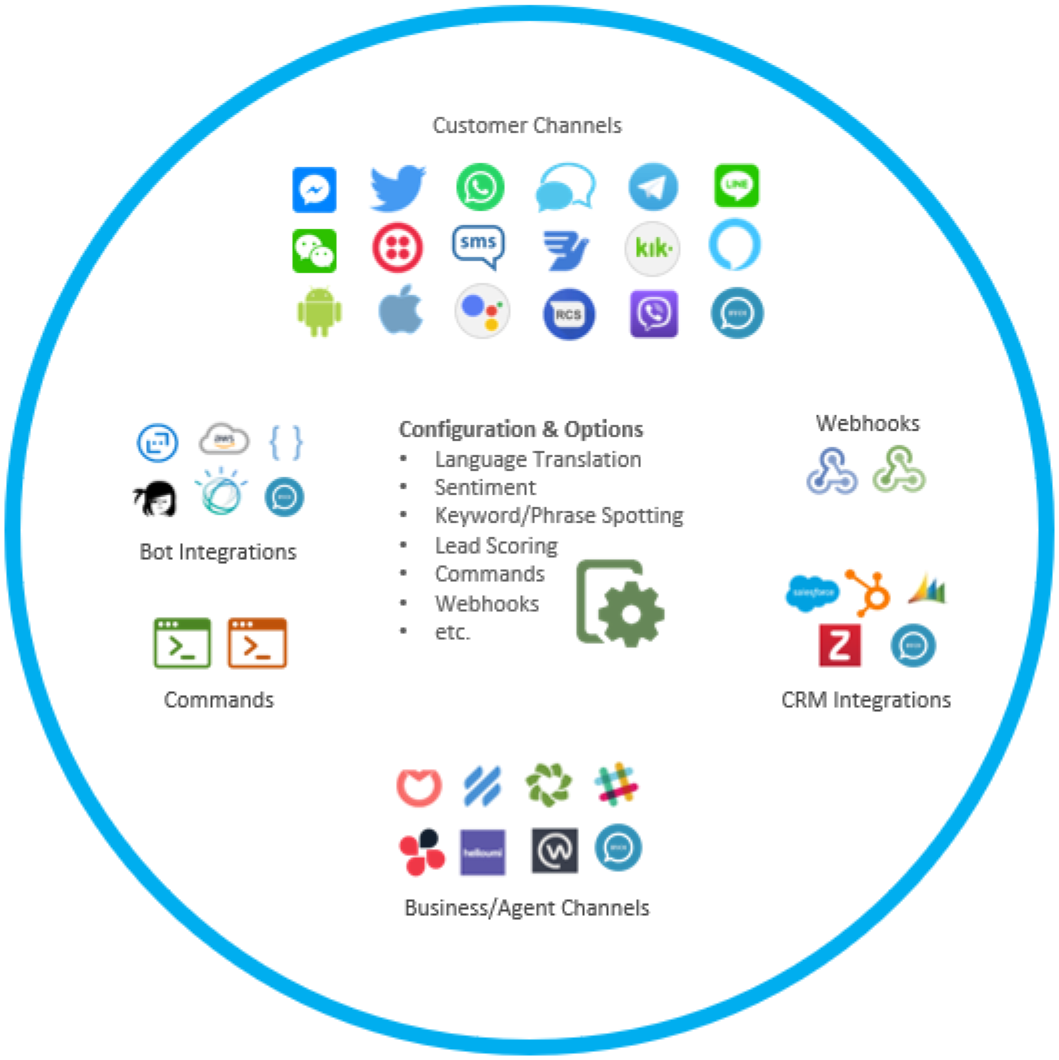


Hubster's Engine workflow and feature annotation:

1. A customer channel initiates a conversation with the engine
2. The engine reads the channel's *hub* configuration and starts the *pipeline* workflow
3. The pipeline *reverse engineers* the channel's proprietary format and constructs a common Hubster format known as an **activity**
4. Based on the hub's configuration and channel *source type*, the pipeline determines the appropriate **preliminary** flow actions required
5. The pipeline then determines the appropriate **auxiliary** flow actions required
6. Once both **preliminary** and **auxiliary** flows have been executed, the pipeline then determines the **active business destination** and reverse engineers the activity to the proprietary format specific to the destination source – agent or bot
7. The agent may initiate a **takeover** from a bot, handles the request, and eventually hands the conversation back to the bot. Conversely, if the bot has difficulty handling a request, the bot can initiate a **handover** and redirect the conversation to the agent.

1.2 The Hub Anatomy

At Hubster, a **hub** (hence our company name), is the center where all configurations are managed and stored. Hubs are used by engine's *pipeline* which drives the workflows and what actions are taken. A business can create as many hubs needed, with each having a specific configuration for a given business segment. For example, a business can create hubs for various lines-of-businesses, campaigns, events, and more.



TODO: Hub Integration with image

1.2.1 Customer Channels

TODO

1.2.2 Business Channels

TODO

- **Agents**
- **Bots**
- **Handover rules**

1.2.3 CRM Channels

TODO

1.2.4 Webhooks

TODO

1.2.5 Commands

TODO

1.2.6 Preliminaries

TODO

1.3 Pipeline

TODO

1.4 Bring your own Integration (BYOI)

TODO

1.5 UX Multi-rendering/Response Framework

TODO

CHAPTER 2

Terminology

2.1 Source

TODO

CHAPTER 3

Integrations

TODO

CHAPTER 4

Support

TODO

CHAPTER 5

Webchat Component

This section will describe how to embed and configure the Hubster Webchat component onto your website. Different configuration examples will be provided and a detail explanation describing each configuration property available.

5.1 Adding to Website

When working with Hubster's Webchat component, you must first embed the component onto every webpage where web-chatting is desired on your website.

Below is a snippet of the default configuration to enable web-chatting on your webpage. It should be noted that it's best to add the webchat component at the very tail end of the HTML webpage.

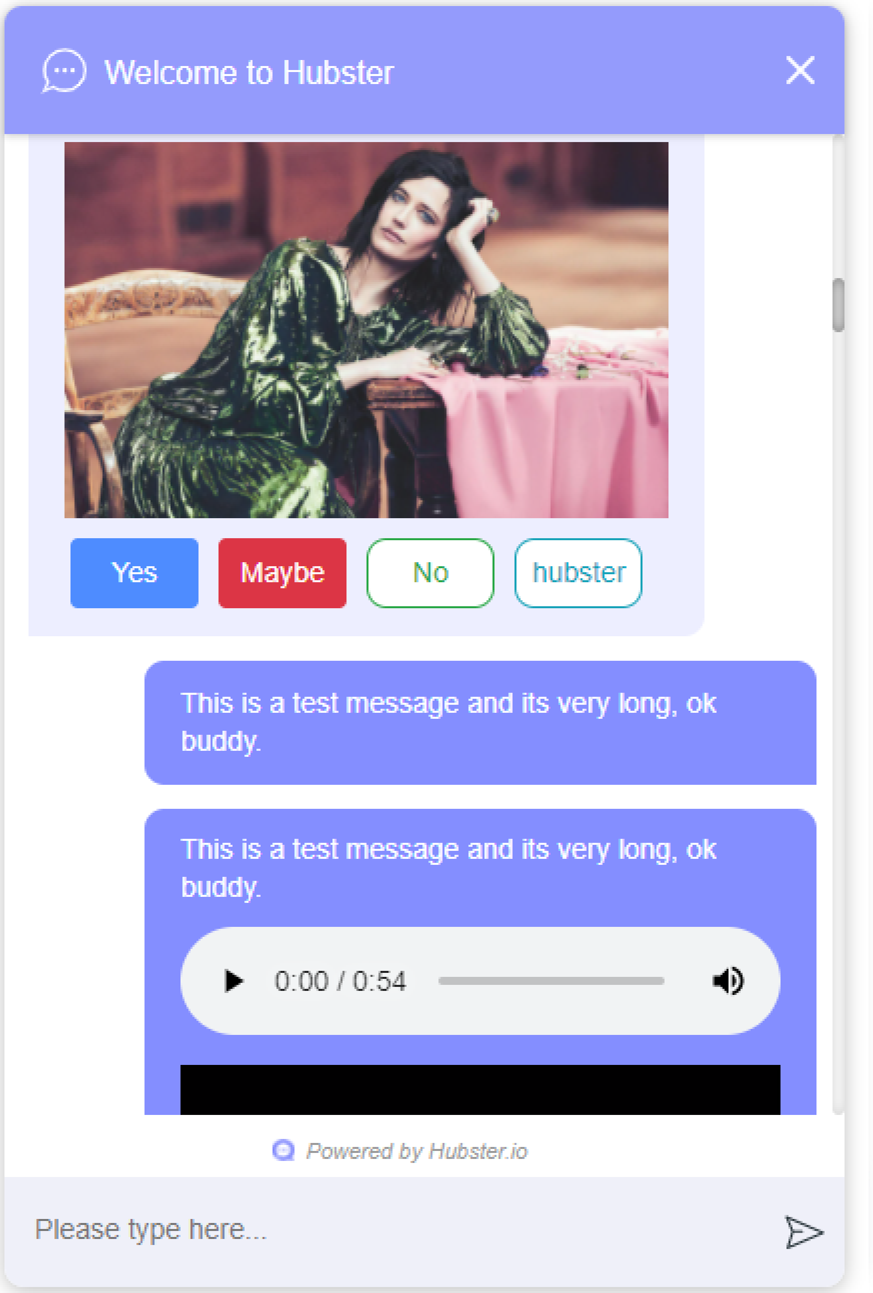
```
<!DOCTYPE html>
<html  lang="en">
<head>
    ...
</head>
<body>
    ...

    <!-- Webchat script -->
    <hubster-webchat></hubster-webchat>
    <script>
    window.HUBSTER_CONFIG = {
        // remove these lines when deploying to production
        engineEndpoint: 'https://demo-engine.hubster.io',
        eventsEndpoint: 'https://demo-events.hubster.io',
        integrationId: 'B205B3EC-A9D7-4243-B88C-017533957DBE',
        sessionTTL: 43200
    };
    </script>
    <script src="https://hubsterdevcdn.azureedge.net/pub/scripts/webchat/hubster-
↪webchat-1.0.min.js"></script>
```

(continues on next page)

(continued from previous page)

```
</body>
</html>
```

By default, the webchat component will look like this:



This is a test message and its very long, ok buddy.

This is a test message and its very long, ok buddy.

Configuring the look-and-feel

Although, Hubster loves the default look, we realize our customers need the ability to change how the webchat component looks-and-feels on their website. A lot of consideration and flexible has been provided, giving customers a wide range of style settings to configure their webchat component.

Below is an example configuration on how one would style the component using a blue theme.

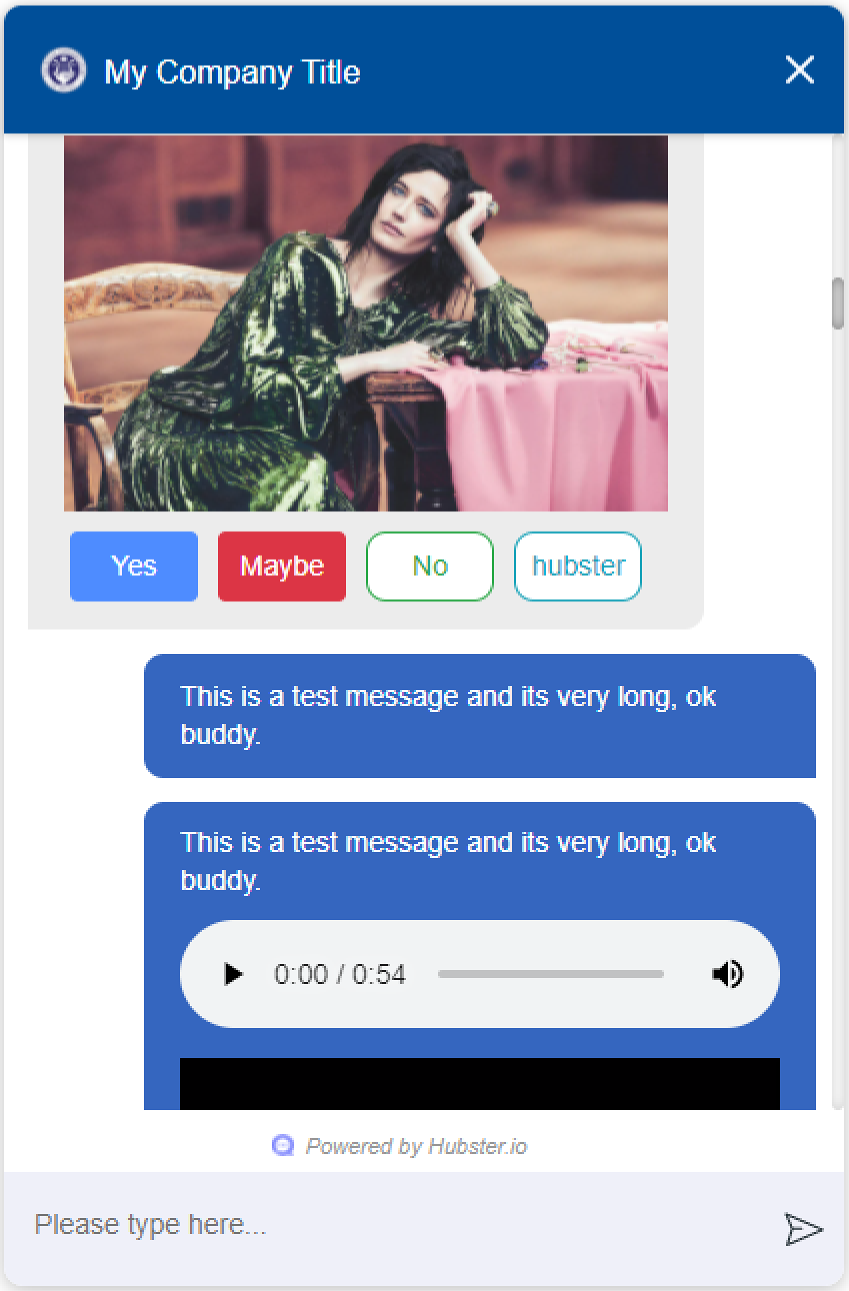
```
<!DOCTYPE html>
<html  lang="en">
<head>
    ...
</head>
<body>
    ...

    <!-- Webchat script -->
    <hubster-webchat></hubster-webchat>
    <script>
    window.HUBSTER_CONFIG = {
        // remove these lines when deploying to production
        engineEndpoint: 'https://demo-engine.hubster.io',
        eventsEndpoint: 'https://demo-events.hubster.io',
        integrationId: 'B205B3EC-A9D7-4243-B88C-017533957DBE',
        sessionTTL: 43200,
        styling: false,
        styles: {
            header: {
                title: 'My Company Title',
                iconUrl: 'https://hubsterdevcdn.azureedge.net/pub/demo/webchat/rcm/
↪chat_logo.png',
                style: {
                    'backgroundColor': '#004F99'
                }
            },
            mount: {
                style: {
                    'backgroundColor': '#004F99',
                    'bottom': '5rem',
                    'right': '.9rem',
                    'z-index': '100'
                },
            },
            userTextMessage: {
                'backgroundColor': '#3566BF'
            },
            agentTextMessage: {
                'color': '#22186e',
                'backgroundColor': '#ECECEC'
            },
            botTextMessage: {
                'color': '#22186e',
                'backgroundColor': '#ECECEC'
            },
            footer: {
                maxInputHeight: '40px'
            }
        },
        onMount(mounted) {
            console.log('Mounted:' + mounted);
        },
        mountOnLoad() {
            return -1;
        },
        onReceivedActivity(activity) {
```

(continues on next page)

(continued from previous page)

```
            console.log(activity);
        },
        onConversation() {
            return {
                "bindingKey": "some_unique_userId",
                "properties": {
                    "profile": {
                        "full name": "SomeUserName",
                        "gender": "female",
                        "custom1": "value1",
                        "custom2": "value2",
                        ...
                    },
                    "additional": {
                        "custom1": "value1",
                        "custom2": "value2",
                        ...
                    }
                }
            };
        }
    };
    </script>
    <script src="https://hubsterdevcdn.azureedge.net/pub/scripts/webchat/hubster-
↪webchat-1.0.min.js"></script>
</body>
</html>
```

The above configuration yields the following theme.



This is a test message and its very long, ok buddy.

This is a test message and its very long, ok buddy.

Note: Before going live, it's best to **style** the webchat component first to suite your site's look-and-feel. This however, can be a time consuming task, but luckily Hubster makes this easy.

Hubster provides configuration property called **styling** (see the example above). By default, this property is set to **false**. However, by setting this property to **true**, Hubster will provide a list of all the available webchat widgets for you to style. Most widgets share the same styling property, meaning that, if you change one style, it may affect one or more other widgets.

Hubster provides a wide range of styles that can be change, however, try to avoid changing styles that may affect sizes and position as we may not guarantee the visual behavior.

Furthermore, when **styling** is set to **true**, the webchat component disables all backend communications with Hubster services.

5.2 Configuration

Property	Mandatory	Comments
engineEndpoint	No	When developing in demo mode, set this value to https://demo-engine.hubster.io The default is: https://engine.hubster.io (production)
eventsEndpoint	No	When developing in demo mode, set this value to: https://demo-events.hubster.io The default is: https://events.hubster.io (production)
sessionTTL	No	How long the user's conversation should last in mins. The user's conversion is maintain even after they close their browser. If the user does not visit the site after the sessionTLL has expired, a new conversation will be established. Sessions are based on a rolling window, meaning that the session's start time will reset if the user revisits the site prior to the session expiring. The default is 1440 (1 day).
integrationId	Yes	The integration id for this Webchat integration.
openOnNewMessage	No	This tells the webchat component when closed, to open the chat window if a new message immediately arrives. The default is false.
hideCloseIcon	No	Don't show the header's close Icon. The default is false. **Note**: If the hideCloseIcon property is set to true, then there is no way to close the webchat window.
fullWindow	No	If set to to true, the webchat window will open in full window mode.
styling	No	Set this value to true when styling the webchat component. The webchat will provide a list of all the available webchat widgets for you to style. The default is false.
styles	No	A collection of styles. See the *Styles Configuration* section.
mountOnLoad	No	A **JavaScript** method that will be invoked when the webchat component is first loaded on the host webpage. This method when provided, allows the developer to determine how long of a **delay** (in milliseconds) the webchat component should wait before **mounting** (open). On page reload – a return value < 0 indicates no mounting should occur – a return value >= 0 indicates mount after number of milliseconds *mountOnLoad(): number;*
onMount	No	A **JavaScript** method that will be invoked when the webchat component has been mounted (open) or docked (closed). *onMount(mounted: boolean): void;*
onReceivedActivity	No	A **JavaScript** method that will be invoked when the webchat component receives an *activity* from the Hubster Engine service before it's displayed on the webchat list. *onReceivedActivity(activity: Activity): void;*
onBeforeActivitySend	No	A **JavaScript** method that will be invoked when the webchat component is about to send a user *activity* to Hubster's Engine service. The programer has the option to inspect the activity and perform any action as deemed necessary.

Styles Configuration

Hubster wanted to provide an easy, consistent and standard way styling your webchat component. Hubster decided to adhered to the **HTML** *style* property format as shown below.

```
<div styles='color: red; background-color: yellow'>...</div>
```

The only stipulation is that any style property that is normally **hyphenated**, will be replace with its **camelCase** equivalent:

- **text-decoration** will be represented as **'textDecoration'**
- **z-index** will be represented as **'zIndex'**
- **background-color** will be represented as **'backgroundColor'**
- **color** will be represented as **'color'** (in this example, the property name remains the same)
- and so on. . .

See a more formal example below.

```
window.HUBSTER_CONFIG = {
    styles: {
        mount: {
            iconUrl: 'https://cdn.com/logo.png',
            style: {
                'backgroundColor': '#004F99',
                'bottom': '5rem',
                'right': '.9rem',
                'zIndex': '100'
            },
        },
        userTextMessage: {
            'backgroundColor': '#3566BF'
        },
    },
    ...
};
```

Note: Values for each style property must be incased in single **'quotes'**. This is true even for properties that accept numeric values. If values are not provided, the webchat component will assume its defaults.

Style	Comments
chatBackgroundColor	This style controls the background color for the webchat list view. Examples: '#ABDCEF' or 'red' . . .
userTextMessage	This style controls the **user text message** bubble. See example below: `window.HUBSTER_CONFIG = {` `styles: {` `userTextMessage: {` `'color': 'green'` `'backgroundColor': '#CFCFCF'` `},` `},` `...` `};`
agentTextMessage	This style controls the **agent text message** bubble. See example below: `window.HUBSTER_CONFIG = {` `styles: {` `agentTextMessage: {` `'color': 'blue'` `'backgroundColor': 'white'` `},` `},` `...` `};`
botTextMessage	This style controls the **bot text message** bubble. See example below: `window.HUBSTER_CONFIG = {` `styles: {` `botTextMessage: {` `'color': 'yellow'` `'backgroundColor': '#00FF00'` `},` `},` `...` `};`
header	This style controls the **header** of the webchat component. See example below: `window.HUBSTER_CONFIG = {` `styles: {` `header: {` `title: 'My Company Title',` `iconUrl: 'https://cdn.com/logo.png',` `style: {` `'color': 'white'` `'backgroundColor': 'blue'` `}` `}` `},` `...` `};`
footer	

5.3 Webchat Script Versions

Version	Reference
1.0	https://hubsterdevcdn.azureedge.net/pub/scripts/webchat/hubster-webchat-1.0.min.js

CHAPTER 6

Webhooks

This section describes the necessary steps and considerations when developing a webhook integration. An example is provided, detailing how to implement your webhook endpoint to trust incoming payloads sent by Hubster.

Webhooks are only supported by the following integration types:

- System
- Direct
- BYOI (Bring your own Integration)

Warning: For **Direct** and **BYOI** integration types, they have the option to either receive activities via **webhooks** or **websockets**. If an integration has been configured using websockets and its endpoint is unreachable, Hubster will not enforce it's *retry policy* as websockets adhere to **fire-and-forget** paradigm.

Furthermore, if an integration is configured for **websockets**, then this section is **not applicable**. Websockets are secured through an authenticated connection thus HMAC verification is redundant and unnecessary.

6.1 HMAC Signature Validation

If your custom integration was configured to receive activities via webhooks, it's important to ensure that the request your integration receives comes from a trusted source, in this case Hubster.

Hubster uses the HMAC when signing and verify signatures. When you *create* and configured your custom integration for a given hub, the results of the request will yield two properties - **publicSigningKey** and **privateSigningKey** respectively.

To obtain your customer integration's public/private key pair, just call the following API:

GET */api/v1/integrations/{integrationId}*

```
{
    "integrationTypeId": "System",
    "channelId": "System",
    "name": "My cool integration name.",
    "configuration": {
        "events": [
            "message"
        ],
        "webhookUrl": "https://url_end_point.com",
        "publicSigningKey": "3EF951F619CD4F5E820C73622C0F1A3C",
        "privateSigningKey": "FA96D15568654A4482772E00BA941BCB"
    }
}
```

The **publicSigningKey** is not used by Hubster when signing the request. However, the business can use the **publicSigningKey** as a reference key to obtain their **privateSigningKey** from where they manage their secrets.

Warning: Please make sure all **private keys** are stored securely. If you suspect your private key was compromised, you can regenerate new **public/private key pair** by updating your custom integration. Click *here* for more info.

6.2 C# Sample

Below is an example code snippet using **C# ASP.NET Core**. This sample validates that the request is trusted using HMAC, and once trusted, consumes the activities. The code is fairly straight forward and should be easily transferable to other programing languages.

The main steps are as follows:

1. Extract the **Signature** from the request header
2. Extract the **Public Key** from the request header
3. Obtain the **Private Key** from your secure keystore and convert it to a **UTF8** byte array
4. Obtain the raw request body in byte array form
5. Produce the a **HMAC** hash (signature) by using the raw request body and applying the **Private Key**
6. Take the signature array produced from the step above and convert to 64 base encoding
7. Compare if signatures match
8. If signatures match then consume the *activities* as needed
9. If signatures don't match then return **Forbidden** (403)

This sample is based off APS.NET Core, using C#. To see the full example, head over to Hubster's public sample repo.

```
[ApiController]
[Route("[controller]")]
public class WebhooksController : ControllerBase
{
    [HttpPost("activities")]
    public async Task<IActionResult> ReceiveActivities()
    {
```

(continues on next page)

(continued from previous page)

```
        var publicKey = Request.Headers["x-hubster-public-key"].ToString();
        var headerSignature = Request.Headers["x-hubster-signature"].ToString();

        if (string.IsNullOrWhiteSpace(publicKey)
        || string.IsNullOrWhiteSpace(headerSignature))
        {
            return StatusCode((int)HttpStatusCode.Forbidden, "Forbidden");
        }

        var privateKey = await GetPrivateKeyAsync(publicKey);

        var rawBody = new byte[(int)Request.ContentLength];
        await Request.BodyReader.AsStream().ReadAsync(rawBody);

        // now preform HMAC signature check

        using (var hasher = new HMACSHA256(privateKey))
        {
            var byteSignature = hasher.ComputeHash(rawBody);
            var signature = Convert.ToBase64String(byteSignature);

            if (signature != headerSignature)
            {
                _logger.LogWarning("Invalid signature");
                return StatusCode((int)HttpStatusCode.Forbidden, "Forbidden");
            }
        }

        // at this point the request is now trusted
        // and it came from Hubster

        var json = Encoding.UTF8.GetString(rawBody);
        var activityConverter = new DirectMessageJsonConverter();
        var activities = JsonConvert.DeserializeObject<SystemOutboundDataModel>(json,␣
↪activityConverter);

        // you now have a list of activities you can process, etc.

        return Ok();
    }

    private Task<byte[]> GetPrivateKeyAsync(string publicKey)
    {
        // NOTE: for sake of sample, we are hard-coding the private key
        // however, you should use the public key as an indexer to get
        // the private key in some secure store like KeyVault, etc.

        var privateKey = "FA96D15568654A4482772E00BA941BCB";
        var bPrivateKey = Encoding.UTF8.GetBytes(privateKey);

        return Task.FromResult(bPrivateKey);
    }
}
```

Note:

If you're using **.NET Core**, the following nuget package contains all the activity model definitions.

Hubster.Abstractions

```
Install-Package Hubster.Abstractions -Version 1.0.2
```

To see a list of activity models, see our public github for direct reference.

6.3 Webhook - Header

Key	Value
x-source-system	The sending system source. This value will always be **engine.hubster.io**
x-hub-id	The hub that that triggered integration belongs too.
x-integration-id	The integration that was triggered.
x-conversation-id	The conversation that this activity was enacted on.
x-hubster-public-key	The public key for this integration. The endpoint can use this value to determine the **private key** used to sign the payload.
x-hubster-signature	The HMAC signature of the payload.

6.4 System - Payload

Webhook endpoints will receive a payload that looks similar to the JSON snippet shown below. The root node contains the **conversation** details and the **activities** node contains one or more *activities*.

```
{
  "hubId": "00000000-0000-0000-0000-000000000001",
  "tenantId": "00000000-0000-0000-0000-000000000002",
  "integrationId": "00000000-0000-0000-0000-000000000003",
  "conversationId": "00000000-0000-0000-0000-000000000004",
  "conversationProperties": {
    "profile": {
      "device": "Direct",
      "full name": "Some customer name",
      "prop1": "value1",
      "prop2": "value2"
    },
    "additional": {
      "prop1": "value1",
      "prop2": "value2"
    }
  },
  "activities": [
    {
      "type": "message",
      "eventTrigger": "message:customer",
      "eventId": 1603933721542,
      "externalId": "my-external-id",
      "isEcho": false,
      "interactionId": "00000000-0000-0000-0000-000000000005",
      "flowProcess": "Default",
      "sender": {
        "integrationId": "00000000-0000-0000-0000-000000000001",
        "integrationType": "Customer",
```

(continues on next page)

(continued from previous page)

```
        "channelType": "Direct",
        "tokenId": "t+8qymYD1jp7wDSHG+3eUA=="
      },
      "recipient": {
        "integrationId": "00000000-0000-0000-0000-000000000006",
        "integrationType": "Agent",
        "channelType": "Direct",
        "tokenId": "971480cb-938c-4dfd-be4e-01756c833490.00000000-0000-0000-0000-
↪000000000003"
      },
      "message": {
        "type": "text",
        "text": "Hi there!"
      }
    }
  ]
}
```

6.5 Direct - Payload

The Direct outbound payload is similar to the Webhook outbound payload accept that, rather than having a collection of activities, the activities will be replaced with an *activity* node.

```
{
  "hubId": "00000000-0000-0000-0000-000000000001",
  "tenantId": "00000000-0000-0000-0000-000000000002",
  "integrationId": "00000000-0000-0000-0000-000000000003",
  "conversationId": "00000000-0000-0000-0000-000000000004",
  "conversationProperties": {
    "profile": {
      "device": "Direct",
      "full name": "Some customer name",
      "prop1": "value1",
      "prop2": "value2"
    },
    "additional": {
      "prop1": "value1",
      "prop2": "value2"
    }
  },
  "activity": {
      "type": "message",
      "eventTrigger": "message:customer",
      "eventId": 1603933721542,
      "externalId": "my-external-id",
      "isEcho": false,
      "interactionId": "00000000-0000-0000-0000-000000000005",
      "flowProcess": "Default",
      "sender": {
        "integrationId": "00000000-0000-0000-0000-000000000001",
        "integrationType": "Customer",
        "channelType": "Direct",
        "tokenId": "t+8qymYD1jp7wDSHG+3eUA=="
      },
```

(continues on next page)

(continued from previous page)

```
      "recipient": {
        "integrationId": "00000000-0000-0000-0000-000000000006",
        "integrationType": "Agent",
        "channelType": "Direct",
        "tokenId": "971480cb-938c-4dfd-be4e-01756c833490.00000000-0000-0000-0000-
↪000000000003"
      },
      "message": {
        "type": "text",
        "text": "Hi there!"
      }
    }
}
```

6.6 Activity Event Filters

Below are list of of activity events that **system** integrations can register too. System integrations must register to at least one event but can register to more as deemed necessary. Hubster will only send events once, to one of the following events if triggered.

Event	Description
message	Hubster will notify the webhook on **all message** activities for the given hub.
message:customer	Hubster will only notify the webhook on **all customer message** activities for the given hub.
message:agent	Hubster will only notify the webhook on all **agent message activities** for the given hub.
message:bot	Hubster will only notify the webhook on all **bot message activities** for the given hub.

6.7 Webhook Retry Policy

Retry Attempt	Next Retry Period	Timeout Before Retry
0 x 2 minutes	0 minutes (immediate)	10 seconds
1 x 2 minutes	2 minutes	10 seconds
2 x 2 minutes	4 minutes	10 seconds
3 x 2 minutes	6 minutes	10 seconds
4 x 2 minutes	8 minutes	10 seconds
5 x 2 minutes	10 minutes	10 seconds

Warning: Once all retries attempts are exhausted, Hubster will send a **notification** to the tenant account holder with details to as to why the endpoint failed. It is up to the the account holder to rectify their integration issue.

CHAPTER 7

Direct Integrations

TODO

CHAPTER 8

Overview

This section will provide guidance and usage on how to interact with Hubster's APIs. The goal of Hubster is to provide a consistent and systematic approach across all our API resources, in terms of style, format and responses.

Hubster's APIs are designed using **REST** principles and most of our payloads are structured using **JSON**. Any exception to this rule will be noted where necessary.

Note: Hubster APIs incorporate cross-origin resource sharing (**CORS**) whereby facilitating web applications to freely use our API in an authenticated and secure manner.

8.1 Environments

Below are the list of Hubster environments:

Identity	API Resource
Demo	https://demo-identity.hubster.io
Production	https://identity.hubster.io

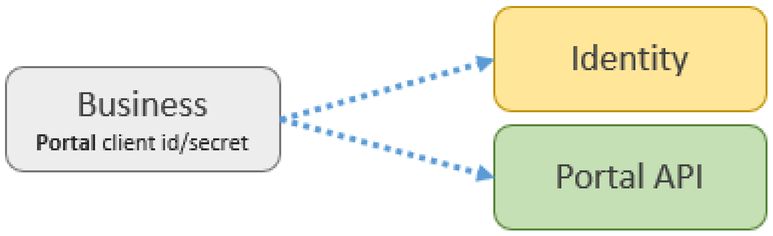
Portal	API Resource
Demo	https://demo-portal-api.hubster.io
Production	https://portal-api.hubster.io

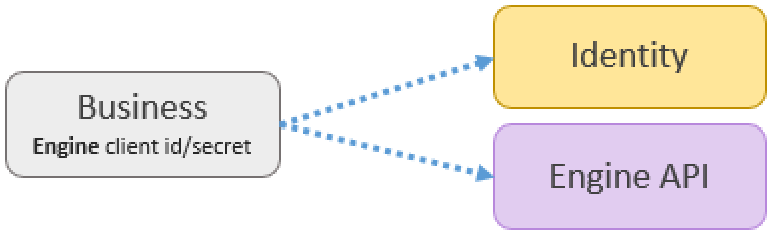
Engine	API Resource
Demo	https://demo-engine.hubster.io
Production	https://engine.hubster.io

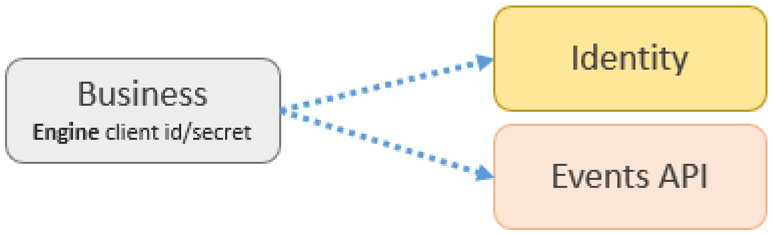
Events	API Resource
Demo	https://demo-events.hubster.io
Production	https://events.hubster.io

8.2 Identity to API Resource Interaction

Below is a depiction on how a business application first obtains an **access token**, using Hubster's identity service, which can later be used to make authenticated requests against API services, such as Hubster's Portal, Engine and Events resources.







Note: API **access tokens** use the **client_credential grant_type**, Description you can only obtain a access tokens using **client_ids** and **client_secrets** respectively. These access tokens are longed-lived and last no longer than 30 days. When an access token expires, accessing an API resource yields an HTTP Status **401** - *Unauthorized Access*.

Furthermore, access tokens are only meant for their indented API resource. For example, if a business obtains an access token for the Portal API resource, the access token cannot be used for the to gain access to the Engine API resource, and likewise. These design was intentional as scopes for each API resource are vastly different. *There's an exception to this rule. Accessing the Events API requires an Engine access token.*

To obtain an **access_token**, please refer to the *Authentication API* for more details.

8.3 Standard Error Response

Hubster's standard error response model is consistent across all API resources. The only difference being are the actual *error codes* returned by each API resource.

Below is an example of a **Bad Request** (400) returned by the Portal API.

Response

```
{
    "status": 400,
    "errors": [
        {
            "code": 301,
            "description": "Property 'name' is required. (code: PRT000301)"
        },
        {
            "code": 206,
            "description": "Value '50000' is not valid. (code: PRT000206)"
        }
    ]
}
```

Property	Description
status	The *HTTP Status code*. This will value always equals header's HTTP Status code.
errors	The list of *errors*.

8.4 Paginated Results

In some cases, Hubster may return a **paginated** response whereby, the business will need to re-query the next result, based on **page number** and **page size**. This is typically when certain **GET** requests may yield a large number of records.

Below is an example from Portal API resource returning Hubs as paginated response.

Response : 200 (OK)

```
{
    "pageNumber": 0,
    "pageSize": 50,
    "total": 2,
    "results": [
        {
            "hubId": "00000000-0000-0000-0000-0000000000a2",
            "tenantId": "00000000-0000-0000-0000-000000000001",
            "name": "Dev Hub 1",
            "description": "Dev Hub 1 (Websocket)",
            "statusId": 2000
        },
        {
            "hubId": "00000000-0000-0000-0000-0000000000a3",
            "tenantId": "00000000-0000-0000-0000-000000000001",
            "name": "Hubster Demo (blank)",
            "description": "Hubster Demo mainly used for Videos",
            "statusId": 2000
```

(continues on next page)

(continued from previous page)

```
            }
        ]
}
```

Property	Description
pageNumber	The requested page number.
pageSize	The requested page size.
total	The total number of results across all pages. **Note**: the total number of items does not necessary equal the number of result items.
results	A list of response models returned by the API resource. **Note**: the result models may differ on per call basis.

8.5 HTTP Status Codes

Hubster API HTTP Status codes.

HTTP Status	Description
200	OK response. The body of the response will include the data requested.
201	OK response. The response will content no data.
400	Bad request. The body of the response will have more info.
401	Unauthorized. Token is invalid.
403	Forbidden. Access to the requested resource is forbidden.
404	Not found. Resource not found.
408	Timed out. The request timed out.
409	Conflict. The request caused a conflict.
410	Not available. The request is not available.
417	Expectation Failed. The operation was aborted.
429	Too many requests. API usage limit has been reached.
500	Internal server error. There was an internal issue with the service.
501	Not implemented. The request is not implemented.
503	Service unavailable. The service is unavailable.

CHAPTER 9

Identity

The Identity Service is mainly used for Authenticating client credentials and upon success, returns a **Bearer access token**.

Below is the list of API collections for Hubster's Identity service.

9.1 Authentication

POST */connect/token*

Headers

Header	Description
Content-Type	`application/x-www-form-urlencoded`

Request Properties

Property	Mandatory	Description
grant_type	Yes	Has to be `client_credentials`.
client_id	Yes	The client id of the Hubster service you plan to authenticate against. Typical, it will be one of following: – **hubster.portal.api.000000000. . . .** – **hubster.engine.api.000000000. . . .** Please refer to *Identity to API Resource Interaction* for the resources accessible on a per **client_id** basis.
client_secret	Yes	The client secret for the **client_id** used.

Note: The request body is of **grant_type** see the following example format below:

grant_type=client_credentials&client_id=hubster.portal.api.0000000. . . &client_secret=SMWvD7W. . .

Response : 200 (OK)

```
{
    "access_token": "eyJhbGciOiJSUzI1NiIsImtpZCI...",
    "expires_in": 2592000,
    "token_type": "Bearer",
    "scope": "hubster-portal-api"
}
```

CHAPTER 10

Portal

The Portal Service is mainly use to manage and configure a tenant's hubs, integrations and access tokens.

Below is the list of API collections for Hubster's Portal service.

10.1 Tenants

10.1.1 Get Client

Gets a Client.

GET */api/v1/tenants/clients/{clientId}*

Headers

Header	Description
Authorization	Bearer `your portal access token`
Content-Type	`application/json`

Url Segments

Segment	Description
clientId	The client id to get.

Response : 200 (OK)

```
{
  "tenantId": "B205B3EC-A9D7-4243-B88C-017533957DEB",
  "clientId": "hubster.portal.api.B205B3ECA9D74243B88C017533957DEB",
  "name": "Hubster Portal API Client (Public)",
  "enabled": true,
```

(continues on next page)

(continued from previous page)

```
  "allowedScopes": [
    "hubster-portal-api"
  ],
  "claims": [
    {
      "type": "role",
      "value": "admin"
    },
    {
      "type": "tenant",
      "value": "B205B3EC-A9D7-4243-B88C-017533957DEB"
    }
  ],
  "secrets": [
    {
      "id": 9,
      "name": "my secret 1",
      "token": "SMWvD7WUAn8bkdl..."
    },
    {
      "id": 15,
      "name": "my secret 2",
      "token": "SMWvD7WUAn8bkdl..."
    },
    {
      "id": 16,
      "name": "my secret 3",
      "token": "SMWvD7WUAn8bkdl...",
      "expiration": "2030-01-31T00:00:00"
    }
  ]
}
```

HTTP Status	Description
200	OK response. The body of the response will include the data requested.
401	Unauthorized. Token is invalid.
403	Forbidden. Access to the requested resource is forbidden.
404	Not found. Resource not found.
408	Timed out. The request timed out.
429	Too many requests. API usage limit has been reached.
500	Internal server error. There was an internal issue with the service.
503	Service unavailable. The service is unavailable.

10.1.2 Add Client Token

Adds a new Client Token.

POST */api/v1/tenants/clients/{clientId}/tokens*

Headers

Header	Description
Authorization	Bearer `your portal access token`
Content-Type	`application/json`

Url Segments

Segment	Description
clientId	The client id affected.

Request Properties

Property	Mandatory	Description
name	Yes	Unique tenant name for Hub.
expiration	No	Expiration date when this token will no longer be valid. If no expiration date was provide, then this token lives forever.

Example Request Body

```
{
  "name": "my secret 3",
  "expiration": "2030-01-31T00:00:00"
}
```

Response : 200 (OK)

```
{
  "id": 16,
  "name": "my secret 3",
  "token": "7AQNCUKAXdCg1M...",
  "expiration": "2030-01-31T00:00:00"
}
```

HTTP Status	Description
200	OK response. The body of the response will include the data requested.
400	Bad request. The body of the response will have *more info*.
401	Unauthorized. Token is invalid.
403	Forbidden. Access to the requested resource is forbidden.
408	Timed out. The request timed out.
429	Too many requests. API usage limit has been reached.
500	Internal server error. There was an internal issue with the service.
503	Service unavailable. The service is unavailable.

10.1.3 Delete Client Token

Deletes (revokes) a Client Token.

DELETE */api/v1/tenants/clients/{clientId}/tokens/{tokenId}*

Headers

Header	Description
Authorization	Bearer `your portal access token`
Content-Type	`application/json`

Url Segments

Segment	Description
clientId	The client id affected.
tokenId	The the token id to delete.

Response : 200 (OK)

HTTP Status	Description
200	OK response. The body of the response will include the data requested.
401	Unauthorized. Token is invalid.
403	Forbidden. Access to the requested resource is forbidden.
408	Timed out. The request timed out.
429	Too many requests. API usage limit has been reached.
500	Internal server error. There was an internal issue with the service.
503	Service unavailable. The service is unavailable.

10.2 Hubs

10.2.1 Create

Creates a Hub.

POST */api/v1/hubs*

Headers

Header	Description
Authorization	Bearer `your portal access token`
Content-Type	`application/json`

Request Properties

Property	Mandatory	Description
name	Yes	Unique Hub name for tenant.
description	Yes	Hub description.
closeDormantConversation	No	The number of days to close this conversation if dormant. If no value was supplied or is equal to 0 (zero), the conversation will remain open. It should be noted that conversations can be closed by the business at any time.
statusId	No	Hub status. Valid options are: – *Active* = 2000 – *Paused* = 2002 Default is *Active* = 2000, if no value supplied.

Example Request Body

```
{
    "name": "Your New Cool Hub Name",
    "description": "This hub is cool",
    "closeDormantConversation": 30,
    "statusId": 2000
}
```

Response : 200 (OK)

```
{
    "hubId": "3bc1e69f-c520-446f-ab2c-01751fd66a31",
    "tenantId": "abc1e69f-c888-875f-ee2c-45789fd66a00",
    "name": "Your New Cool Hub Name",
    "description": "This hub is cool",
    "closeDormantConversation": 30,
    "statusId": 2000
}
```

HTTP Status	Description
200	OK response. The body of the response will include the data requested.
400	Bad request. The body of the response will have *more info*.
401	Unauthorized. Token is invalid.
403	Forbidden. Access to the requested resource is forbidden.
408	Timed out. The request timed out.
429	Too many requests. API usage limit has been reached.
500	Internal server error. There was an internal issue with the service.
503	Service unavailable. The service is unavailable.

10.2.2 Update

Updates a Hub.

PUT */api/v1/hubs/{hubId}*

Headers

Header	Description
Authorization	Bearer `your portal access token`
Content-Type	`application/json`

Url Segments

Segment	Description
hubId	The hub id affected.

Request Properties

Property	Mandatory	Description
name	No	Unique Hub name for tenant.
description	No	Hub description.
closeDormantConversation	No	The number of days to close this conversation if dormant. If no value was supplied or is equal to 0 (zero), the conversation will remain open. It should be noted that conversations can be closed by the business at any time.
statusId	No	Hub status. Valid options are: – *Active* = 2000 – *Paused* = 2002 Default is *Active* = 2000, if no value supplied.

Example Request Body

```
{
   "name": "Your New Cool Hub Name",
   "description": "This hub is cool",
   "closeDormantConversation": 30,
   "statusId": 2000
}
```

Response : 200 (OK)

```
{
    "hubId": "3bc1e69f-c520-446f-ab2c-01751fd66a31",
    "tenantId": "abc1e69f-c888-875f-ee2c-45789fd66a00",
    "name": "Your New Cool Hub Name",
    "description": "This hub is cool",
    "closeDormantConversation": 30,
    "statusId": 2000
}
```

HTTP Status	Description
200	OK response. The body of the response will include the data requested.
400	Bad request. The body of the response will have *more info*.
401	Unauthorized. Token is invalid.
403	Forbidden. Access to the requested resource is forbidden.
404	Not found. Resource not found.
408	Timed out. The request timed out.
429	Too many requests. API usage limit has been reached.
500	Internal server error. There was an internal issue with the service.
503	Service unavailable. The service is unavailable.

10.2.3 Delete

Deletes a Hub.

Warning: This will delete all integrations and their registrations to their service provider.

DELETE */api/v1/hubs/{hubId}*

Headers

Header	Description
Authorization	Bearer `your portal access token`
Content-Type	`application/json`

Url Segments

Segment	Description
hubId	The hub id affected.

Response : 200 (OK)

HTTP Status	Description
200	OK response. The body of the response will include the data requested.
401	Unauthorized. Token is invalid.
403	Forbidden. Access to the requested resource is forbidden.
404	Not found. Resource not found.
408	Timed out. The request timed out.
429	Too many requests. API usage limit has been reached.
500	Internal server error. There was an internal issue with the service.
503	Service unavailable. The service is unavailable.

10.2.4 Get

Gets a Hub.

GET */api/v1/hubs/{hubId}*

Headers

Header	Description
Authorization	Bearer `your portal access token`
Content-Type	`application/json`

Url Segments

Segment	Description
hubId	The hub id to get.

Response : 200 (OK)

```
{
    "hubId": "3bc1e69f-c520-446f-ab2c-01751fd66a31",
    "tenantId": "abc1e69f-c888-875f-ee2c-45789fd66a00",
    "name": "Your New Cool Hub Name",
```

(continues on next page)

(continued from previous page)

```
    "description": "This hub is cool",
    "closeDormantConversation": 30,
    "statusId": 2000
}
```

HTTP Status	Description
200	OK response. The body of the response will include the data requested.
401	Unauthorized. Token is invalid.
403	Forbidden. Access to the requested resource is forbidden.
404	Not found. Resource not found.
408	Timed out. The request timed out.
429	Too many requests. API usage limit has been reached.
500	Internal server error. There was an internal issue with the service.
503	Service unavailable. The service is unavailable.

10.2.5 Get Collection

Gets a list of Hubs.

GET */api/v1/hubs*

Headers

Header	Description
Authorization	Bearer `your portal access token`
Content-Type	`application/json`

Request Arguments

Argument	Mandatory	Description
pageNumber	No	The requested page number. *Must be >= 0*
pageSize	No	The requested page size. *Must be >= 1 and <= 100*

Response : 200 (OK)

paginated

```
{
      "pageNumber": 0,
      "pageSize": 50,
      "total": 2,
      "results": [
         {
          "hubId": "3bc1e69f-c520-446f-ab2c-01751fd66a31",
          "tenantId": "abc1e69f-c888-875f-ee2c-45789fd66a00",
          "name": "Your New Cool Hub Name",
          "description": "This hub is cool",
```

(continues on next page)

(continued from previous page)

```
            "closeDormantConversation": 30,
            "statusId": 2000
          },
          {
            "hubId": "3bc1e69f-c520-446f-ab2c-01751fd66a32",
            "tenantId": "abc1e69f-c888-875f-ee2c-45789fd66a01",
            "name": "Your New Cool Hub Name 2",
            "description": "This hub is cool 2",
            "closeDormantConversation": 30,
            "statusId": 2000
          }
        ]
    }
```

HTTP Status	Description
200	OK response. The body of the response will include the data requested.
400	Bad request. The body of the response will have *more info*.
401	Unauthorized. Token is invalid.
403	Forbidden. Access to the requested resource is forbidden.
408	Timed out. The request timed out.
429	Too many requests. API usage limit has been reached.
500	Internal server error. There was an internal issue with the service.
503	Service unavailable. The service is unavailable.

10.3 Integrations

10.3.1 Create

Creates an Integration.

POST */api/v1/integrations/hubs/{hubId}*

Headers

Header	Description
Authorization	Bearer `your portal access token`
Content-Type	`application/json`

Url Segments

Segment	Description
hubId	The hub id affected.

Request Properties

Property	Mandatory	Description
channelId	Yes	Has to be one of the following *Channel Types*.
name	Yes	Unique name for integration per Hub.
statusId	No	Integration status. Valid options are: – *Active* = 3000 – *Paused* = 3002 Default is *Active* = 3000, if no value supplied.
configuration	Yes	See *configuration* properties for each individual **channelId**.

Example Request Body

Note: The request body below uses **TwilioSMS** as an example. It should be noted that *configuration* properties for each channel type differs. Please use the correct configuration specific to the **channelId** value defined.

```
{
  "channelId": "TwilioSMS",
  "name": "My cool integration name.",
  "statusId": 3000,
  "configuration": {
    "authToken": "cb8c5367c3c4586ecb589e25570...",
    "accountSid": "AC1fc1c1722444b0c6313d3ae988b...",
    "numberSid": "PN667435536f4d1cefdf054abf99..."
  }
}
```

HTTP Status	Description
200	OK response. The body of the response will include the data requested.
400	Bad request. The body of the response will have *more info*.
401	Unauthorized. Token is invalid.
403	Forbidden. Access to the requested resource is forbidden.
408	Timed out. The request timed out.
429	Too many requests. API usage limit has been reached.
500	Internal server error. There was an internal issue with the service.
503	Service unavailable. The service is unavailable.

Configurations

TwilioSMS

```
{
  "authToken": "cb8c5367c3c4586ecb589e25570...",
  "accountSid": "AC1fc1c1722444b0c6313d3ae988b...",
  "numberSid": "PN667435536f4d1cefdf054abf99..."
}
```

Property	Mandatory	Description
authToken	Yes	Twilio authorization token.
accountSid	Yes	Twilio account SID.
numberSid	Yes	Twilio phone number SID.

Response 200 (OK)

```
{
  "integrationId": "00000000-0000-0000-0000-000000000000",
  "hubId": "00000000-0000-0000-0000-000000000000",
  "integrationTypeId": "Customer",
  "channelId": "TwilioSMS",
  "name": "My cool integration name.",
  "statusId": 3000,
  "configuration": {
    "accountSid": "AC1fc1c1722444b0c6313d3da98...",
    "authToken": "cb8c5367c3c4586ecb589e25570....",
    "numberSid": "PN667435536f4d1cefdf054ecf9....",
    "phoneNumber": "+16476960000",
    "capabilities": {
      "mms": true,
      "sms": true,
      "voice": true
    }
  }
}
```

Messenger

```
{
  "pageAccessToken":
↪"EAAFBmgAdBToBADCvmo5w10tmlh97uxhtorpi5Adrdo0wtwFfXfkNxxLAY29AxwBHJNfXH5rR..."
}
```

Property	Mandatory	Description
pageAccessToken	Yes	Facebook page access token.

Response 200 (OK)

```
{
  "integrationId": "00000000-0000-0000-0000-000000000000",
  "hubId": "00000000-0000-0000-0000-000000000000",
  "integrationTypeId": "Customer",
  "channelId": "Messenger",
  "name": "My cool integration name.",
  "statusId": 3000,
  "configuration": {
    "appId": "35360465938...",
    "pageId": "1013889883...",
    "pageAccessToken": "EAAFBm..."
  }
}
```

WebChat

```
{
  "allowedOrigins": [
    "localhost",
    "hubster.io"
  ],
  "start": [
    {
      "type": "text",
      "text": "Welcome to Hubster! How can we help you?"
    }
  ]
}
```

Property	Mandatory	Description
allowedOrigins	Yes	One or more domains hosting the WebChat component.
start	No	An array of Hubster *messages types*.

Response 200 (OK)

```
{
    "integrationId": "00000000-0000-0000-0000-000000000000",
    "hubId": "00000000-0000-0000-0000-000000000000",
    "integrationTypeId": "Customer",
    "channelId": "WebChat",
    "name": "Webchat",
    "statusId": 3000,
    "configuration": {
        "AllowedOrigins": [
            "localhost",
            "hubster.io"
        ],
        "Echo": true,
        "Start": [
            {
                "type": "text",
                "text": "Welcome to Hubster! How can we help you?"
            }
        ]
    }
}
```

Direct

```
{
  "integrationType": "Agent",
  "echo": true,
  "webhookUrl": "https://url_end_point.com"
}
```

Property	Mandatory	Description
integrationType	Yes	Must be a supported *integration* type.
echo	No	If yes, when an activity is received from this integration, it will echo it back.
webhookUrl	No	The endpoint to receive Hubster *Activities*. If not supplied, activities will be delivered via websockets.
start	No	An array of Hubster *messages types*.

Response 200 (OK)

```
{
  "integrationId": "00000000-0000-0000-0000-000000000000",
  "hubId": "00000000-0000-0000-0000-000000000000",
  "integrationTypeId": "Agent",
  "channelId": "Direct",
  "name": "My cool integration name.",
  "statusId": 3000,
  "configuration": {
  "integrationType": "Agent",
  "echo": true,
  "webhookUrl": "https://url_end_point.com",
  "publicSigningKey": "6DF60E ...",
  "privateSigningKey": "E0A42 ...",
  "start": [
    {
      "type": "text",
      "text": "Welcome to Hubster! How can we help you?"
    }
  ]
}
```

System

```
{
  "webhookUrl": "https://url_end_point.com",
  "events": [
    "message:customer",
    "message:agent",
    "message:bot"
  ]
}
```

Property	Mandatory	Description
webhookUrl	Yes	The endpoint to receive Hubster *Activities*
events	Yes	The *activity event filter(s)* to be event on.

Response 200 (OK)

```
{
    "integrationId": "00000000-0000-0000-0000-000000000000",
    "hubId": "00000000-0000-0000-0000-000000000000",
    "integrationTypeId": "System",
    "channelId": "System",
    "name": "My cool integration name.",
    "statusId": 3000,
    "configuration": {
        "events": [
            "message:customer",
            "message:agent",
            "message:bot"
        ],
        "webhookUrl": "https://url_end_point.com",
        "publicSigningKey": "6DF60E ...",
        "privateSigningKey": "E0A42 ...",
```

(continues on next page)

(continued from previous page)

```
    }
}
```

Slack

```
{
  "code": "EAAFBmgAdBToBADCvmo5w10tmlh97uxhtorpi5Adrdo0wtwFfXfkNxxLAY29AxwBHJNfXH5rR..
↪.",
  "nonce" : "mo5w10t.mlh97uxh"
}
```

Property	Mandatory	Description
code	Yes	Slack oauth2 code.
nonce	Yes	Verification signature.

Response 200 (OK)

```
{
  "integrationId": "00000000-0000-0000-0000-000000000000",
  "hubId": "00000000-0000-0000-0000-000000000000",
  "integrationTypeId": "Agent",
  "channelId": "Slack",
  "name": "My cool integration name.",
  "statusId": 3000,
  "configuration": {
    "botAccessToken": "xoxb-193043142226-...",
    "appAccessToken": "xoxp-193043142226-...",
    "defaultPublicChannel": "general",
    "teamId": "T5P19488N",
    "botName": "Hubster.io"
  }
}
```

10.3.2 Update

Updates an Integration.

POST */api/v1/integrations/hubs/{integrationId}*

Headers

Header	Description
Authorization	Bearer `your portal access token`
Content-Type	`application/json`

Url Segments

Segment	Description
integrationId	The integration id affected.

Request Properties

Property	Mandatory	Description
name	No	Unique name for integration per Hub.
statusId	No	Integration status. Valid options are: – *Active* = 3000 – *Paused* = 3002
configuration	No	See *configuration* properties for each individual **channelId**.

Example Request Body

```
{
  "name": "Direct",
  "statusId": 3002,
  "configuration": {
     "Echo": true,
     "webhookUrl": "http://hubster.io/v1/api/integration?customer=1"
  }
}
```

Configurations

Note: If you need to update any configuration value, you need to provide **all required** values specific to that channel type. In other words, the complete **configuration** object will replace the old one.

Warning: The following integration types cannot have their **configuration** values updated due to re-authenticating with their respective service providers. Any attempt will be ignored.

- **TwilioSMS**
- **Messenger**
- **Slack**

If you need to update their configuration, you must first **delete** the original integration and **recreate** a new one.

WebChat

```
{
  "allowedOrigins": [
      "localhost",
      "hubster.io"
  ],
  "start":
      [
        {
          "type": "text",
          "text": "Welcome to Hubster! How can we help you?"
        }
      ]
}
```

Property	Mandatory	Description
allowedOrigins	Yes	One or more domains hosting the WebChat component.
start	No	An array of Hubster *messages types*.

Response 200 (OK)

```
{
    "integrationId": "00000000-0000-0000-0000-000000000000",
    "hubId": "00000000-0000-0000-0000-000000000000",
    "integrationTypeId": "Customer",
    "channelId": "WebChat",
    "name": "Webchat",
    "statusId": 3000,
    "configuration": {
        "AllowedOrigins": [
            "localhost",
            "hubster.io"
        ],
        "Echo": true,
        "Start": [
            {
                "type": "text",
                "text": "Welcome to Hubster! How can we help you?"
            }
        ]
    }
}
```

Direct

```
{
  "integrationType": "Agent",
  "echo": true,
  "webhookUrl": "https://url_end_point.com",
  "regenerateKeys": true,
  "start": [
    {
      "type": "text",
      "text": "Welcome to Hubster! How can we help you?"
    }
  ]
}
```

Property	Mandatory	Description
integrationType	Yes	Must be a supported *integration* type.
echo	No	If yes, when an activity is received from this integration, it will echo it back.
webhookUrl	No	The endpoint to receive Hubster *Activities*. If not supplied, activities will be delivered via websockets.
regenerateKeys	No	This forces a new set of public/private keys to be generated.
start	No	An array of Hubster *messages types*.

Response 200 (OK)

```
{
  "integrationId": "00000000-0000-0000-0000-000000000000",
```

(continues on next page)

(continued from previous page)

```
  "hubId": "00000000-0000-0000-0000-000000000000",
  "integrationTypeId": "Agent",
  "channelId": "Direct",
  "name": "My cool integration name.",
  "statusId": 3000,
  "configuration": {
  "integrationType": "Agent",
  "echo": true,
  "webhookUrl": "https://url_end_point.com",
  "publicSigningKey": "6DF60E ...",
  "privateSigningKey": "E0A42 ...",
  "start": [
    {
      "type": "text",
      "text": "Welcome to Hubster! How can we help you?"
    }
  ]
}
```

System

```
{
  "webhookUrl": "https://url_end_point.com",
  "regenerateKeys": true,
  "events": [
    "message:customer",
    "message:agent",
    "message:bot"
  ]
}
```

Property	Mandatory	Description
webhookUrl	Yes	The endpoint to receive Hubster *Activities*
regenerateKeys	No	This forces a new set of public/private keys to be generated.
events	Yes	The *activity event filter(s)* to be event on.

Response 200 (OK)

```
{
    "integrationId": "00000000-0000-0000-0000-000000000000",
    "hubId": "00000000-0000-0000-0000-000000000000",
    "integrationTypeId": "System",
    "channelId": "System",
    "name": "My cool integration name.",
    "statusId": 3000,
    "configuration": {
        "events": [
            "message:customer",
            "message:agent",
            "message:bot"
        ],
        "webhookUrl": "https://url_end_point.com",
        "publicSigningKey": "6DF60E ...",
        "privateSigningKey": "E0A42 ...",
    }
```

(continues on next page)

(continued from previous page)

```
}
```

10.3.3 Get

Gets an Integration.

GET */api/v1/integrations/{integrationId}*

Headers

Header	Description
Authorization	Bearer `your portal access token`
Content-Type	`application/json`

Url Segments

Segment	Description
integrationId	The integration to get.

Response 200 (OK)

Note: The request body below uses **TwilioSMS** as an example. It should be noted that *configuration* properties for each channel type differs.

```
{
  "integrationId": "00000000 ...",
  "hubId": "00000000 ...",
  "inboundId": "AC1fc1c1722444b0...",
  "integrationTypeId": 2,
  "channelId": 102,
  "name": "Twilio Test Number: 1647...",
  "statusId": 3000,
  "configuration": {
    "AcccountSid": "AC1fc1c172244...",
    "AuthToken": "cb8c5367c3c458...",
    "NumberSid": "PN667435536f4d...",
    "PhoneNumber": "+1647...",
      "Capabilities": {
      "Mms": true,
      "Sms": true,
      "Voice": true
      }
    }
}
```

HTTP Status	Description
200	OK response. The body of the response will include the data requested.
401	Unauthorized. Token is invalid.
403	Forbidden. Access to the requested resource is forbidden.
404	Not found. Resource not found.
408	Timed out. The request timed out.
429	Too many requests. API usage limit has been reached.
500	Internal server error. There was an internal issue with the service.
503	Service unavailable. The service is unavailable.

Response Body Examples

TwilioSMS

```
 {
    "integrationId": "00000000-0000-0000-0000-000000000000",
    "hubId": "00000000-0000-0000-0000-000000000000",
    "inboundId": "AC1fc1c1722444b0c6313d3....",
    "integrationTypeId": "Customer",
    "channelId": "TwilioSMS",
    "name": "Twilio Test Number: 16476960489",
    "statusId": 3000,
    "configuration": {
        "AccountSid": "AC1fc1c1722444b0c6313d...",
        "AuthToken": "cb8c5367c3c4586ecb589e2...",
        "NumberSid": "PN667435536f4d1cefdf054...",
        "PhoneNumber": "+16476960489",
        "Capabilities": {
            "Mms": true,
            "Sms": true,
            "Voice": true
        }
    }
}
```

Messenger

```
{
    "integrationId": "00000000-0000-0000-0000-000000000000",
    "hubId": "00000000-0000-0000-0000-000000000000",
    "inboundId": "27623838....",
    "integrationTypeId": "Customer",
    "channelId": "Messenger",
    "name": "Messenger: Hubster Biz",
    "statusId": 3000,
    "configuration": {
        "AppId": "218851140...",
        "PageId": "27623838...",
        "PageAccessToken": "EAAfGcISnoh0BAEZBihIAC..."
    }
}
```

Web Chat

```
 {
    "integrationId": "00000000-0000-0000-0000-000000000000",
```

(continues on next page)

(continued from previous page)

```
    "hubId": "00000000-0000-0000-0000-000000000000",
    "integrationTypeId": "Customer",
    "channelId": "WebChat",
    "name": "Webchat for Hubster Demo (Blank) ",
    "statusId": 3000,
    "configuration": {
        "allowedOrigins": [
            "localhost",
            "hubster.io"
        ],
        "echo": true,
        "start": [
            {
                "type": "text",
                "text": "Welcome to Hubster! How can we help you?"
            }
        ]
    }
}
```

Direct

```
{
    "integrationId": "00000000-0000-0000-0000-000000000000",
    "hubId": "00000000-0000-0000-0000-000000000000",
    "integrationTypeId": "Customer",
    "channelId": "Direct",
    "name": "Direct Customer (Webhook)",
    "statusId": 3000,
    "configuration": {
        "WebhookUrl": "http://localhost:5100/v1/api/integration?customer=1",
        "publicSigningKey": "6DF60E ...",
        "privateSigningKey": "E0A42 ...",
        "Echo": false,
        "WelcomeMessage": "Welcome to Hubster! How can we help you?"
    }
}
```

System

```
{
    "integrationId": "00000000-0000-0000-0000-000000000000",
    "hubId": "00000000-0000-0000-0000-000000000000",
    "integrationTypeId": "System",
    "channelId": "System",
    "name": "My cool integration name.",
    "statusId": 3000,
    "configuration": {
        "events": [
            "message:customer",
            "message:agent",
            "message:bot"
        ],
        "webhookUrl": "https://url_end_point.com",
        "publicSigningKey": "6DF60E ...",
        "privateSigningKey": "E0A42 ...",
    }
```

(continues on next page)

(continued from previous page)

```
}
```

Slack

```
{
    "integrationId": "00000000-0000-0000-0000-000000000000",
    "hubId": "00000000-0000-0000-0000-000000000000",
    "integrationTypeId": "Agent",
    "channelId": "Slack",
    "name": "Slack for Hubster Demo",
    "statusId": 3000,
    "configuration": {
        "BotAccessToken": "xoxb-...",
        "AppAccessToken": "xoxp-...",
        "DefaultPublicChannel": "general",
        "TeamId": "T017QM...",
        "BotName": "Hubster.io"
    }
}
```

10.3.4 Get by Channel Type

Gets a list of integrations for a given *Channel Type*.

GET */api/v1/integrations/hubs/{hubId}/{channelType}*

Headers

Header	Description
Authorization	Bearer `your portal access token`
Content-Type	`application/json`

Url Segments

Segment	Description
hubId	The hub id to obtain all integrations.
channelType	The *Channel Type* to filter by.

Response : 200 (OK)

```
[
    {
        "integrationId": "00000000-0000-0000-0000-000000000001",
        "hubId": "00000000-0000-0000-0000-000000000000",
        "integrationTypeId": "Customer",
        "channelId": "Direct",
        "name": "Direct Customer 2 (Webhook)",
        "statusId": 3000,
        "configuration": {
            "WebhookUrl": "http://localhost:5100/v1/api/integration?customer=1",
            "publicSigningKey": "6DF60E ...",
            "privateSigningKey": "E0A42 ...",
            "Echo": false,
```

(continues on next page)

(continued from previous page)

```
            "WelcomeMessage": "Welcome to Hubster! How can we help you?"
        }
    },
    {
        "integrationId": "00000000-0000-0000-0000-000000000002",
        "hubId": "00000000-0000-0000-0000-000000000000",
        "integrationTypeId": "Agent",
        "channelId": "Direct",
        "name": "Direct Agent (Websocket)",
        "statusId": 3000,
        "configuration": {
            "Echo": true
        }
    }
]
```

HTTP Status	Description
200	OK response. The body of the response will include the data requested.
401	Unauthorized. Token is invalid.
403	Forbidden. Access to the requested resource is forbidden.
408	Timed out. The request timed out.
429	Too many requests. API usage limit has been reached.
500	Internal server error. There was an internal issue with the service.
503	Service unavailable. The service is unavailable.

10.3.5 Get Collection

Gets a list of integrations.

GET */api/v1/integrations*

Headers

Header	Description
Authorization	Bearer `your portal access token`
Content-Type	`application/json`

Url Arguments

Argument	Mandatory	Description
hubId	No	Filter by hub id.
pageNumber	No	The requested page number. *Must be >= 0.*
pageSize	No	The requested page size. *Must be >= 1 and <= 100.*

Response : 200 (OK)

paginated

```
{
      "pageNumber": 0,
      "pageSize": 50,
      "total": 2,
      "results": [
         {
          "hubId": "3bc1e69f-c520-446f-ab2c-01751fd66a31",
          "tenantId": "abc1e69f-c888-875f-ee2c-45789fd66a00",
          "name": "Your New Cool Hub Name",
          "description": "This hub is cool",
          "closeDormantConversation": 30,
          "statusId": 2000
        },
        {
          "hubId": "3bc1e69f-c520-446f-ab2c-01751fd66a32",
          "tenantId": "abc1e69f-c888-875f-ee2c-45789fd66a01",
          "name": "Your New Cool Hub Name 2",
          "description": "This hub is cool 2",
          "closeDormantConversation": 30,
          "statusId": 2000
        }
      ]
  }
```

HTTP Status	Description
200	OK response. The body of the response will include the data requested.
400	Bad request. The body of the response will have *more info*.
401	Unauthorized. Token is invalid.
403	Forbidden. Access to the requested resource is forbidden.
408	Timed out. The request timed out.
429	Too many requests. API usage limit has been reached.
500	Internal server error. There was an internal issue with the service.
503	Service unavailable. The service is unavailable.

10.4 Transfer Commands

Note: Transfer Commands are only respected by the **WebChat** device at this time. Sending transfers to other devices will be ignored.

10.4.1 Create

Creates a Transfer Command.

POST */api/v1/commands/transfers/{hubId}*

Headers

Header	Description
Authorization	Bearer `your portal access token`
Content-Type	`application/json`

Url Segments

Segment	Description
hubId	The Hub id to create commands for.

Request Properties

Property	Mandatory	Description
name	Yes	Unique command name per Hub.
description	No	Command description.
url	Yes	Command url.
linkDescription	Yes	The description used for the link anchor on the WebChat device.
mountDelay	No	Amount of in milliseconds before mounting occurs. Default is 0, if no value supplied. mountDelay < 0, will not mount the WebChat mountDelay = 0, will immediately mount the WebChat

Example Request Body

```
{
    "name": "new cool command",
    "description": "Description for command",
    "url": "https://mysite.io",
    "linkDescription": "Click here to be transferred",
    "mountDelay": 1000
}
```

Response : 200 (OK)

```
{
    "commandId": "5a0623b0-d51a-4b3c-ace1-0175e34fae2f",
    "hubId": "00000000-0000-0000-0000-0000000000a1",
    "name": "new cool command",
    "description": "Description for command",
    "url": "https://mysite.io",
    "linkDescription": "Click here to be transferred",
    "mountDelay": 1000
}
```

HTTP Status	Description
200	OK response. The body of the response will include the data requested.
400	Bad request. The body of the response will have *more info*.
401	Unauthorized. Token is invalid.
403	Forbidden. Access to the requested resource is forbidden.
408	Timed out. The request timed out.
429	Too many requests. API usage limit has been reached.
500	Internal server error. There was an internal issue with the service.
503	Service unavailable. The service is unavailable.

10.4.2 Update

Updates a Transfer Command.

PUT */api/v1/commands/transfers/{commandId}*

Headers

Header	Description
Authorization	Bearer `your portal access token`
Content-Type	`application/json`

Url Segments

Segment	Description
commandId	The command id affected.

Request Properties

Property	Mandatory	Description
name	No	Unique command name per Hub.
description	No	Command description.
url	No	Command url.
linkDescription	Yes	The description used for the link anchor on the WebChat device.
mountDelay	No	Amount of in milliseconds before mounting occurs. Default is 0, if no value supplied. mountDelay < 0, will not mount the WebChat mountDelay = 0, will immediately mount the WebChat

Example Request Body

```
{
    "name": "new cool command",
    "description": "Description for command",
    "url": "https://mysite.io",
    "linkDescription": "Click here to be transferred",
    "mountDelay": 1000
}
```

Response : 200 (OK)

```
{
    "commandId": "5a0623b0-d51a-4b3c-ace1-0175e34fae2f",
    "hubId": "00000000-0000-0000-0000-0000000000a1",
    "name": "new cool command",
    "description": "Description for command",
    "url": "https://mysite.io",
    "linkDescription": "Click here to be transferred",
    "mountDelay": 1000
}
```

HTTP Status	Description
200	OK response. The body of the response will include the data requested.
400	Bad request. The body of the response will have *more info*.
401	Unauthorized. Token is invalid.
403	Forbidden. Access to the requested resource is forbidden.
404	Not found. Resource not found.
408	Timed out. The request timed out.
429	Too many requests. API usage limit has been reached.
500	Internal server error. There was an internal issue with the service.
503	Service unavailable. The service is unavailable.

10.4.3 Delete

Deletes a Transfer Command.

DELETE */api/v1/commands/transfers/{commandId}*

Headers

Header	Description
Authorization	Bearer `your portal access token`
Content-Type	`application/json`

Url Segments

Segment	Description
commandId	The transfer command id.

Response : 200 (OK)

HTTP Status	Description
200	OK response. The body of the response will include the data requested.
401	Unauthorized. Token is invalid.
403	Forbidden. Access to the requested resource is forbidden.
404	Not found. Resource not found.
408	Timed out. The request timed out.
429	Too many requests. API usage limit has been reached.
500	Internal server error. There was an internal issue with the service.
503	Service unavailable. The service is unavailable.

10.4.4 Get

Gets a Transfer Command.

GET */api/v1/commands/transfers/{commandId}*

Headers

Header	Description
Authorization	Bearer `your portal access token`
Content-Type	`application/json`

Url Segments

Segment	Description
commandId	The transfer command id to get.

Response : 200 (OK)

```
{
    "commandId": "5a0623b0-d51a-4b3c-ace1-0175e34fae2f",
    "hubId": "00000000-0000-0000-0000-0000000000a1",
    "name": "new cool command",
    "description": "Description for command",
    "url": "https://mysite.io",
    "linkDescription": "Click here to be transferred",
    "mountDelay": 1000
}
```

HTTP Status	Description
200	OK response. The body of the response will include the data requested.
401	Unauthorized. Token is invalid.
403	Forbidden. Access to the requested resource is forbidden.
404	Not found. Resource not found.
408	Timed out. The request timed out.
429	Too many requests. API usage limit has been reached.
500	Internal server error. There was an internal issue with the service.
503	Service unavailable. The service is unavailable.

10.4.5 Get Collection

Gets a list of Transfer Commands.

GET */api/v1/commands/transfers/hub/{hubId}*

Headers

Header	Description
Authorization	Bearer `your portal access token`
Content-Type	`application/json`

Url Segments

Segment	Description
hubId	The Hub id to get transfer commands for.

Response : 200 (OK)

```
[
    {
        "commandId": "5a0623b0-d51a-4b3c-ace1-0175e34fae2f",
        "hubId": "00000000-0000-0000-0000-0000000000a1",
        "name": "new command 2",
        "description": "Description for command",
```

(continues on next page)

(continued from previous page)

```
        "url": "https://mysite.io",
        "linkDescription": "Click here to be transferred",
        "mountDelay": 1000
    }
]
```

HTTP Status	Description
200	OK response. The body of the response will include the data requested.
400	Bad request. The body of the response will have *more info*.
401	Unauthorized. Token is invalid.
403	Forbidden. Access to the requested resource is forbidden.
408	Timed out. The request timed out.
429	Too many requests. API usage limit has been reached.
500	Internal server error. There was an internal issue with the service.
503	Service unavailable. The service is unavailable.

10.5 Response Commands

10.5.1 Create

Creates a Response Command.

POST */api/v1/commands/responses/{hubId}*

Headers

Header	Description
Authorization	Bearer `your portal access token`
Content-Type	`application/json`

Url Segments

Segment	Description
hubId	The Hub id to create commands for.

Request Properties

Property	Mandatory	Description
name	Yes	Unique command name per Hub.
category	No	Command category.
description	No	Command description.
integrationTypeId	Yes	Type of integration command is used for. Valid options are: – *agent* – *customer*
type	Yes	Response command type. Valid options are: – *message* – *event*
responses	Yes	A list of *Message* or *Event* types.

Example Request Body

```
{
  "name": "contact-eva.green",
  "category": "contacts",
  "description": "Eva's Contact",
  "integrationTypeId": "customer",
  "type": "message",
  "responses": [
      {
          "type": "text",
          "text": "Below is my contact info"
      },
      {
          "type": "contact",
          "imageUrl": "https://image.png",
          "title": "Eva Green",
          "properties": [
              {
                  "key": "Title",
                  "value": "Mighty Health"
              },
              {
                  "key": "Address",
                  "value": "108 Kirkbride Crescent, Maple, ON",
                  "type": "address;work"
              },
              {
                  "key": "Cell",
                  "value": "+1 (714) 873-6202",
                  "type": "phone;cell"
              },
              {
                  "key": "Email",
                  "value": "eva@mightyhealth.com",
```

(continues on next page)

(continued from previous page)

```
                    "type": "email"
                }
            ],
            "channels": [
                {
                    "type": "Webchat",
                    "metadata": [
                        {
                            "key": "caption-show",
                            "value": "true"
                        },
                        {
                            "key": "caption-color",
                            "value": "white"
                        }
                    ]
                }
            ]
        }
    ]
}
```

Response : 200 (OK)

```
{
  "commandId": "00000000-0000-0000-0000-000000000005",
  "hubId": "00000000-0000-0000-0000-0000000000a1",
  "name": "contact-eva.green",
  "category": "contacts",
  "description": "Eva's Contact",
  "integrationTypeId": "customer",
  "type": "message",
  "responses": [
      {
          "type": "text",
          "text": "Below is my contact info"
      },
      {
          "type": "contact",
          "imageUrl": "https://image.png",
          "title": "Eva Green",
          "properties": [
              {
                  "key": "Title",
                  "value": "Mighty Health"
              },
              {
                  "key": "Address",
                  "value": "108 Kirkbride Crescent, Maple, ON",
                  "type": "address;work"
              },
              {
                  "key": "Cell",
                  "value": "+1 (714) 873-6202",
                  "type": "phone;cell"
              },
              {
```

(continues on next page)

(continued from previous page)

```
                    "key": "Email",
                    "value": "eva@mightyhealth.com",
                    "type": "email"
                }
            ],
            "channels": [
                {
                    "type": "Webchat",
                    "metadata": [
                        {
                            "key": "caption-show",
                            "value": "true"
                        },
                        {
                            "key": "caption-color",
                            "value": "white"
                        }
                    ]
                }
            ]
        }
    ]
}
```

HTTP Status	Description
200	OK response. The body of the response will include the data requested.
400	Bad request. The body of the response will have *more info*.
401	Unauthorized. Token is invalid.
403	Forbidden. Access to the requested resource is forbidden.
408	Timed out. The request timed out.
429	Too many requests. API usage limit has been reached.
500	Internal server error. There was an internal issue with the service.
503	Service unavailable. The service is unavailable.

10.5.2 Update

Updates a Response Command.

PUT */api/v1/commands/responses/{commandId}*

Headers

Header	Description
Authorization	Bearer `your portal access token`
Content-Type	`application/json`

Url Segments

Segment	Description
commandId	The command id affected.

Request Properties

Property	Mandatory	Description
name	Yes	Unique command name per Hub.
category	No	Command category.
description	No	Command description.
integrationTypeId	Yes	Type of integration command is used for. Valid options are: – *agent* – *customer*
type	Yes	Response command type. Valid options are: – *message* – *event*
responses	Yes	A list of *Message* or *Event* types.

Example Request Body

```
{
  "category": "Office contacts",
  "description": "Eva's Office Contact",
}
```

Response : 200 (OK)

```
{
  "commandId": "00000000-0000-0000-0000-000000000005",
  "hubId": "00000000-0000-0000-0000-0000000000a1",
  "name": "contact-eva.green",
  "category": "Office contacts",
  "description": "Eva's Office Contact",
  "integrationTypeId": "customer",
  "type": "message",
  "responses": [
      {
          "type": "text",
          "text": "Below is my contact info"
      },
      {
          "type": "contact",
          "imageUrl": "https://image.png",
          "title": "Eva Green",
          "properties": [
              {
                  "key": "Title",
                  "value": "Mighty Health"
              },
              {
                  "key": "Address",
                  "value": "108 Kirkbride Crescent, Maple, ON",
                  "type": "address;work"
```

(continues on next page)

(continued from previous page)

```
                },
                {
                    "key": "Cell",
                    "value": "+1 (714) 873-6202",
                    "type": "phone;cell"
                },
                {
                    "key": "Email",
                    "value": "eva@mightyhealth.com",
                    "type": "email"
                }
            ],
            "channels": [
                {
                    "type": "Webchat",
                    "metadata": [
                        {
                            "key": "caption-show",
                            "value": "true"
                        },
                        {
                            "key": "caption-color",
                            "value": "white"
                        }
                    ]
                }
            ]
        }
    ]
}
```

HTTP Status	Description
200	OK response. The body of the response will include the data requested.
400	Bad request. The body of the response will have *more info*.
401	Unauthorized. Token is invalid.
403	Forbidden. Access to the requested resource is forbidden.
404	Not found. Resource not found.
408	Timed out. The request timed out.
429	Too many requests. API usage limit has been reached.
500	Internal server error. There was an internal issue with the service.
503	Service unavailable. The service is unavailable.

10.5.3 Delete

Deletes a Response Command.

DELETE */api/v1/commands/responses/{commandId}*

Headers

Header	Description
Authorization	Bearer `your portal access token`
Content-Type	`application/json`

Url Segments

Segment	Description
commandId	The response command id.

Response : 200 (OK)

HTTP Status	Description
200	OK response. The body of the response will include the data requested.
401	Unauthorized. Token is invalid.
403	Forbidden. Access to the requested resource is forbidden.
404	Not found. Resource not found.
408	Timed out. The request timed out.
429	Too many requests. API usage limit has been reached.
500	Internal server error. There was an internal issue with the service.
503	Service unavailable. The service is unavailable.

10.5.4 Get

Gets a Response Command.

GET */api/v1/commands/responses/{commandId}*

Headers

Header	Description
Authorization	Bearer `your portal access token`
Content-Type	`application/json`

Url Segments

Segment	Description
commandId	The response command id to get.

Response : 200 (OK)

```
{
  "commandId": "00000000-0000-0000-0000-000000000005",
  "hubId": "00000000-0000-0000-0000-0000000000a1",
  "name": "contact-eva.green",
  "category": "Office contacts",
  "description": "Eva's Office Contact",
  "integrationTypeId": "customer",
  "type": "message",
  "responses": [
      {
          "type": "text",
          "text": "Below is my contact info"
      },
      {
          "type": "contact",
          "imageUrl": "https://image.png",
```

(continues on next page)

(continued from previous page)

```
          "title": "Eva Green",
          "properties": [
              {
                  "key": "Title",
                  "value": "Mighty Health"
              },
              {
                  "key": "Address",
                  "value": "108 Kirkbride Crescent, Maple, ON",
                  "type": "address;work"
              },
              {
                  "key": "Cell",
                  "value": "+1 (714) 873-6202",
                  "type": "phone;cell"
              },
              {
                  "key": "Email",
                  "value": "eva@mightyhealth.com",
                  "type": "email"
              }
          ],
          "channels": [
              {
                  "type": "Webchat",
                  "metadata": [
                      {
                          "key": "caption-show",
                          "value": "true"
                      },
                      {
                          "key": "caption-color",
                          "value": "white"
                      }
                  ]
              }
          ]
      }
  ]
}
```

HTTP Status	Description
200	OK response. The body of the response will include the data requested.
401	Unauthorized. Token is invalid.
403	Forbidden. Access to the requested resource is forbidden.
404	Not found. Resource not found.
408	Timed out. The request timed out.
429	Too many requests. API usage limit has been reached.
500	Internal server error. There was an internal issue with the service.
503	Service unavailable. The service is unavailable.

10.5.5 Get Collection

Gets a list of Response Commands.

GET */api/v1/commands/responses/hub/{hubId}*

Headers

Header	Description
Authorization	Bearer `your portal access token`
Content-Type	`application/json`

Url Segments

Segment	Description
hubId	The Hub id to get response commands for.

Response : 200 (OK)

```
[
    {
      "commandId": "00000000-0000-0000-0000-000000000005",
      "hubId": "00000000-0000-0000-0000-0000000000a1",
      "name": "contact-eva.green",
      "category": "Office contacts",
      "description": "Eva's Office Contact",
      "integrationTypeId": "customer",
      "type": "message",
      "responses": [
          {
              "type": "text",
              "text": "Below is my contact info"
          },
          {
              "type": "contact",
              "imageUrl": "https://image.png",
              "title": "Eva Green",
              "properties": [
                  {
                      "key": "Title",
                      "value": "Mighty Health"
                  },
                  {
                      "key": "Address",
                      "value": "108 Kirkbride Crescent, Maple, ON",
                      "type": "address;work"
                  },
                  {
                      "key": "Cell",
                      "value": "+1 (714) 873-6202",
                      "type": "phone;cell"
                  },
                  {
                      "key": "Email",
                      "value": "eva@mightyhealth.com",
                      "type": "email"
                  }
              ],
              "channels": [
                  {
```

(continues on next page)

(continued from previous page)

```
                        "type": "Webchat",
                        "metadata": [
                            {
                                "key": "caption-show",
                                "value": "true"
                            },
                            {
                                "key": "caption-color",
                                "value": "white"
                            }
                        ]
                    }
                ]
            }
        ]
    },
    {
        "commandId": "00000000-0000-0000-0000-000000000006",
        "hubId": "00000000-0000-0000-0000-0000000000a1",
        "name": "Command-5",
        "category": "Command-5",
        "description": "Command-5",
        "integrationTypeId": "agent",
        "type": "event",
        "responses": [
            {
                "type": "payload",
                "payloadType": "type1",
                "payload": {
                    "payload": "Hello"
                }
            }
        ]
    }
]
```

HTTP Status	Description
200	OK response. The body of the response will include the data requested.
400	Bad request. The body of the response will have *more info*.
401	Unauthorized. Token is invalid.
403	Forbidden. Access to the requested resource is forbidden.
408	Timed out. The request timed out.
429	Too many requests. API usage limit has been reached.
500	Internal server error. There was an internal issue with the service.
503	Service unavailable. The service is unavailable.

CHAPTER 11

Engine

The Engine Service is a realtime eventing service that coordinates conversation workflows between customer, business and external channels.

See the *The Big Picture* for more info.

Below is the list of API collections for Hubster's Engine service.

11.1 Conversations

11.1.1 Get Conversation

Get Conversation object using conversation identifier.

GET */api/v1/conversations/{conversationId}*

Headers

Header	Description
Authorization	Bearer `your engine access token`
Content-Type	`application/json`

Url Segments

Segment	Description
conversationId	The conversation id to get.

Response : 200 (OK)

```
{
    "conversationId": "a9cff9d8-0cb2-40f9-b787-01756ca7b92f",
    "integrationId": "...000000000023",
```

(continues on next page)

(continued from previous page)

```
    "tokenId": "9jP61d8EfmVgSnUtBZsMNw==",
    "integration": {
        "integrationId": "...000000000023",
        "hubId": "...0000000000a2",
        "hub": {
            "hubId": "...0000000000a2",
            "tenantId": "...000000000001",
            "name": "Dev Hub (Websocket)",
            "description": "Dev Hub 2 (Websocket)",
            "statusId": 2000,
            "created": "2019-01-01T00:00:00",
            "modified": "2019-01-01T00:00:00"
        },
        "integrationTypeId": "Customer",
        "channelId": "Direct",
        "name": "Direct Customer Webhook",
        "statusId": 3000,
        "created": "2018-01-01T00:00:00",
        "modified": "2018-01-01T00:00:00"
    },
    "properties": {
        "profile": {
            "device": "Direct",
            "full name": "Any name"
        },
         "additional": {}
    },
    "openedDateTime": "2019-10-28T00:42:12.6452901"
}
```

HTTP Status	Description
200	OK response. The body of the response will include the data requested.
401	Unauthorized. Token is invalid.
403	Forbidden. Access to the requested resource is forbidden.
404	Not found. Resource not found.
408	Timed out. The request timed out.
429	Too many requests. API usage limit has been reached.
500	Internal server error. There was an internal issue with the service.
503	Service unavailable. The service is unavailable.

11.1.2 Establish Conversation

Establish (create) new Conversation object.

POST */api/v1/conversations/establish*

Headers

Header	Description
Authorization	Bearer `your engine access token`
Content-Type	`application/json`

Request Properties

Property	Mandatory	Description
integrationId	Yes	Integration identifier.
binding	Yes	Binding key used to get conversation.
channelType	Yes	Type of channel to filter by. Valid option is: – *Direct* = 2000
properties	No	See *properties* object for more info.

Note: Properties can be though of as **metadata** bound to a conversation during establishment. When working for Direct interaction types, the developer can assigned any number properties as necessary. Properties can have one or more collections each containing key/value pairs. In the example below, two such collections have been defined.

By default, Hubster will generate a **profile** collection if not defined. This collection will contain at most, a **device** and **full name** key/value pair. The developer can override anyone of the following values but can also add their own custom key/value pair:

- **profile.device**
- **profile.full name**
- **profile.first name**
- **profile.last name**
- **profile.user name**
- **profile.gender**
- **profile.local**
- **profile.time zone**
- **profile.imageUrl**
- **profile.phone**
- **profile.address**
- **profile.email**

If **profile.device** is not provided, Hubster will default to **Direct**.
If **profile.full name** is not provided, Hubster will default to a random **fun-name**.

Properties

```
{
    "profile": {
        "device": "Direct",
        "full name":"User name here",
        "phone": "416-555-0001",
        "some_custom1": "value1",
        "some_custom2": "value2"
    },
    "additional": {
        "some_custom1": "value1",
```

(continues on next page)

(continued from previous page)

```
        "some_custom2": "value2"
    }
}
```

Response 200 (OK)

```
{
    "tenantId": "...000000000001",
    "hubId": "...0000000000a2",
    "integrationId": "...000000000023",
    "conversationId": "a9cff9d8-0cb2-40f9-b787-01756ca7b92f",
    "tokenId": "9jP61d8EfmVgSnUtBZsMNw==",
    "openedDateTime": "2020-10-28T00:42:12.6452901Z",
    "isNew": true,
    "properties": {
        "profile": {
            "device": "Direct",
            "full name": "User name here"
        },
    "additional": {}
    }
}
```

11.2 Interactions

11.2.1 Get Activities

Get *Activities* for a given conversation.

GET */api/v1/interactions/activities/{conversationId}*

Headers

Header	Description
Authorization	Bearer `your engine access token`
Content-Type	`application/json`

Url Segments

Segment	Description
conversationId	The conversationId id used to retrieve *activities*.

Request Properties

Property	Mandatory	Description
type	Yes	The **sender** or **recipient** integration type that you want to retrieve activities for. For example, if **type** = 'agent', than all activities whereby, the sender or recipient type = 'agent', will be retrieved. Only the following *integration types* are valid: • Customer • Agent • Bot
leid	No	The last starting point event timestamp that you want to retrieve activities from for the given conversation. The value must be in UNIX epoch time in milliseconds (UTC) Default is 0, if no value supplied.

Response : 200 (OK)

```
[
    {
        "type": "message",
        "eventTrigger": "message:customer",
        "eventId": 1604758399703,
        "externalId": "some_external_id_0001",
        "isEcho": false,
        "interactionId": "1368fd94-bd85-4dcf-84c7-0175a30dead7",
        "flowProcess": "Default",
        "sender": {
            "integrationId": "00000000-0000-0000-0000-000000000020",
            "integrationType": "Customer",
            "channelType": "WebChat",
            "tokenId": "LDDKEgUsKUmjQzwjNAhL1w=="
        },
        "recipient": {
            "integrationId": "00000000-0000-0000-0000-000000000024",
            "integrationType": "Agent",
            "channelType": "Slack",
            "tokenId": "f7cbbd3d-3071-45d7-abed-01744cad6a7e.00000000-0000-0000-0000-
↪000000000024"
        },
        "message": {
            "text": "Hi there!",
            "type": "text"
        }
    },
    {
        "type": "message",
        "eventTrigger": "message:agent",
        "eventId": 1604783877593,
        "externalId": "some_external_id_0002",
```

(continues on next page)

(continued from previous page)

```
        "isEcho": false,
        "interactionId": "13da6b0b-0d66-4277-9e9d-0175a492adda",
        "flowProcess": "Default",
        "sender": {
            "integrationId": "00000000-0000-0000-0000-000000000024",
            "integrationType": "Agent",
            "channelType": "Slack",
            "tokenId": "f7cbbd3d-3071-45d7-abed-01744cad6a7e.00000000-0000-0000-0000-
↪000000000024"
        },
        "recipient": {
            "integrationId": "00000000-0000-0000-0000-000000000020",
            "integrationType": "Customer",
            "channelType": "WebChat",
            "tokenId": "LDDKEgUsKUmjQzwjNAhL1w=="
        },
        "message": {
            "text": "Hello! How can I help you today?",
            "type": "text"
        }
    }
]
```

HTTP Status	Description
200	OK response. The body of the response will include the data requested.
401	Unauthorized. Token is invalid.
403	Forbidden. Access to the requested resource is forbidden.
404	Not found. Resource not found.
408	Timed out. The request timed out.
429	Too many requests. API usage limit has been reached.
500	Internal server error. There was an internal issue with the service.
503	Service unavailable. The service is unavailable.

11.3 Direct

The **Direct** API is used primarily by *Direct Channel Types* when submitting *Message* or *Event Activity* types.

POST */inbound/customer/v1/direct/activity/{conversationId}*

Headers

Header	Description
Authorization	Bearer `your portal access token`
Content-Type	`application/json`

Url Segments

Segment	Description
conversationId	The conversation id to send this activity.

Request Properties

When sending activities, there's a minimal amount of header properties that are required. See details below:

Property	Mandatory	Description
type	Yes	The type of activity to send. This can only be *message type* or *event type*
sender.integrationId	See **Description**	If the source of **integrationId** is bound to a **customer** integration, then this property is **not** required. However, if the **integrationId** is bound to either an **agent** or **bot** integration, then this property **is** required.
externalId	No	This can be any string value and will be attached to the lifetime of this activity if provided by the sender. Typically this is used by tenants to maintain a reference or metadata to a given tenant resource. For the most part this value will be null.
message	See **Description**	When **type** is set to *message* then this node is required.
event	See **Description**	When **type** is set to *event* then this node is required.

Example Request Body

```
{
    "externalId": "some-external-id",
    "type": "message | event",
    "sender": {
        "integrationId": "00000000-0000-0000-0000-000000000001"
    },
    "message": {
        "type": "text",
        "text": "Hi there!"
    },
    "event": {
        "type": "payload",
        "payloadType": "my.payload.01",
        "payload": {
            "data1": "value1",
            "data2": "value2",
            "data3": "value3"
        }
    }
}
```

Response : 200 (OK)

```
{
    "status": 200,
    "eventId": 1609281295385,
    "externalId": "some-external-id",
    "hubId": "00000000-0000-0000-0000-0000000000a1",
    "conversationId": "290e83ff-0ae1-4a62-ae8e-01759ad73ffd",
    "integrationId": "00000000-0000-0000-0000-000000000001",
    "interactionId": "d645f70e-30e2-4649-938c-0176b0a3d41b"
}
```

Note: If the activity type was **event**, then the **interactionId** will not be part of the response.

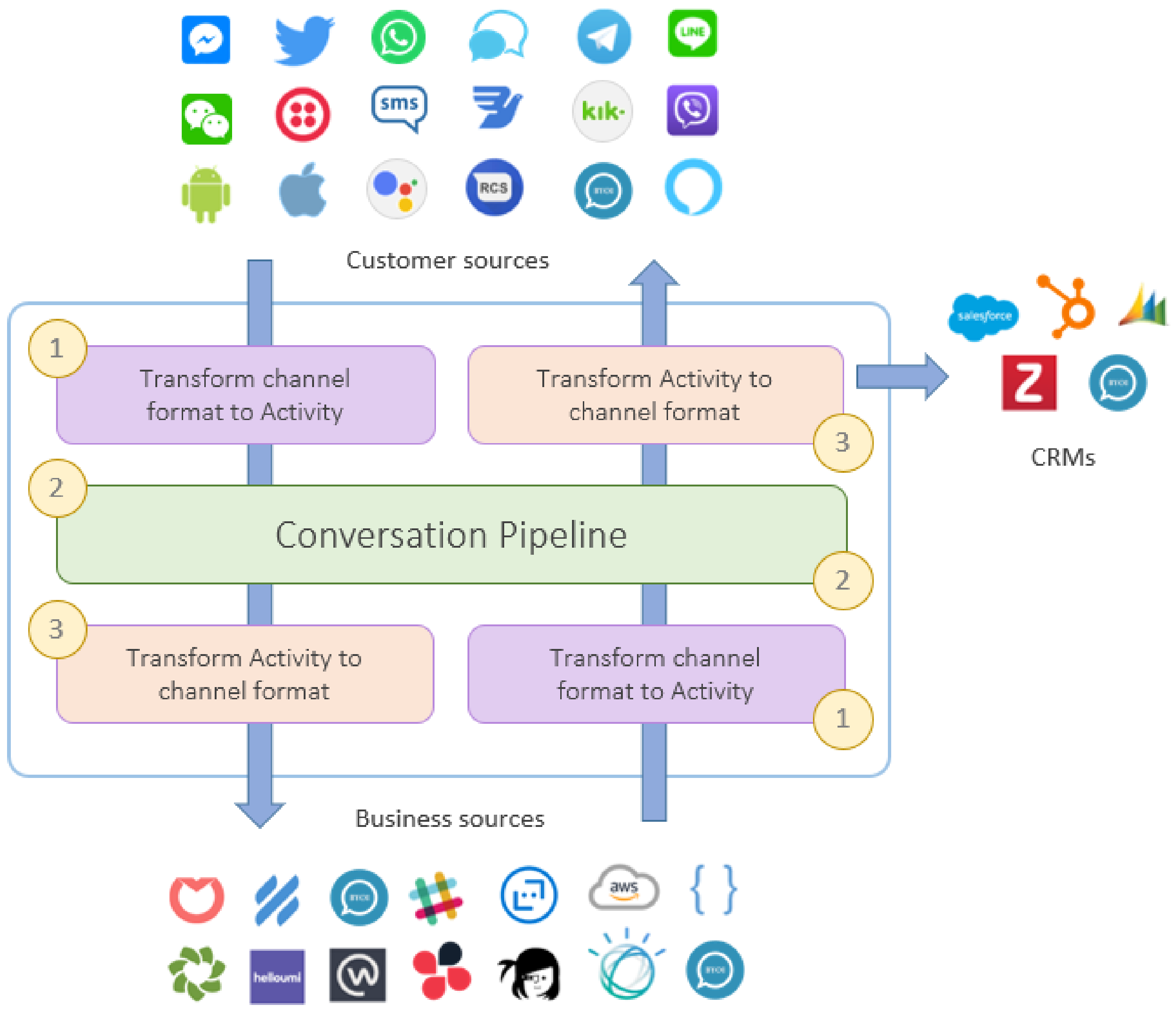
HTTP Status	Description
200	OK response. The body of the response will include the data requested.
400	Bad request. The body of the response will have *more info*.
401	Unauthorized. Token is invalid.
403	Forbidden. Access to the requested resource is forbidden.
408	Timed out. The request timed out.
429	Too many requests. API usage limit has been reached.
500	Internal server error. There was an internal issue with the service.
503	Service unavailable. The service is unavailable.

CHAPTER 12

Activities

When integrating with disparate systems, data coming in or data going out, most likely come in various shapes and formats. In Hubster's case, most integrations may have similar functionalities, however, each integration on their own, have proprietary formats that differ from each other. Because of these numerous differences, it's important to generalize their formats into a common format that is agnostic and most of all consistent. At Hubster, we call this common format an **Activity**. This in essence, is the hallmark behind the concept of **unified messaging**.

This section will explain what an activity is, it's constituents and how it's consumed by Hubster's conversation pipeline. See the depiction below for annotated description on how Hubster transforms both in and outbound data models.


Customer sources
1
Transform channel format to Activity
Transform Activity to channel format
3
2
Conversation Pipeline
2
3
Transform Activity to channel format
Transform channel format to Activity
1
Business sources
CRMs

How Hubster's UX Multi-rendering/Response Framework (RRF) works:

1. When data is sent by a specific integration type, the **inbound** RRF will transform this data into a Hubster activity for further consumption.
2. The conversation pipeline can now freely work and amend the activity without any knowledge of its original source format.
3. Once the conversation pipeline completes its workflow, it will send the response to the **outbound** RRF which transforms the activity to the appropriate integration type's proprietary format.

12.1 Activity

An activity is fairly simple structure that contains either a single *Message* or a *Event* type, but not both.

Note: For sake of sample, both **message** and **event** nodes are shown. However, these nodes are mutually exclusive and will never be presented together in actuality.

```
{
    "type": "message|action",
    "eventTrigger": "message:customer",
    "eventId": 1603933721542,
    "externalId": "my-external-id",
    "isEcho": false,
    "interactionId": "00000000-0000-0000-0000-000000000005",
    "flowProcess": "Default",
    "sender": {
        "integrationId": "00000000-0000-0000-0000-000000000001",
        "integrationType": "Customer",
        "channelType": "Direct",
        "tokenId": "t+8qymYD1jp7wDSHG+3eUA=="
    },
    "recipient": {
        "integrationId": "00000000-0000-0000-0000-000000000006",
        "integrationType": "Agent",
        "channelType": "Direct",
        "tokenId": "971480cb-938c-4dfd-be4e-01756c833490.00000000-0000-0000-0000-
↪000000000003"
    },
    "message": {
        "type": "text",
        "text": "Hi there!"
    },
    "event": {
        "type": "payload",
        "payloadType": "hubster.transfer",
        "payload": {
            "url": "http://localhost:4200",
            "label": "Click here to be transferred",
            "mount": 1000,
            "force": false
        }
    }
}
```

Property	Description
type	The is the type of activity being described. Can either be a **message** or **action**.
eventTrigger	The source of the trigger. Typically this is the **sender** of of the activity. See *Integration Types*
eventId	The epoch UNIX time in milliseconds when this event was initiated.
externalId	This can be any string value and will be attached to the lifetime of this activity if provided by the sender. Typically this is used by tenants to maintain a reference or metadata to a given tenant resource. For the most part this value will be null.
isEcho	A boolean state indicating wither the activity is an echo. Some custom integrations when sending an activity may wish to receive a feedback activity. This is because when sending an activity, the sender tends to send minimal data. Having echo enabled, the sender will receive a more enriched payload with additional data that can be important to the sender. For example, if the sender sends a youtube link in the message text, Hubster will convert the activity to youtube *message type* instead.
interactionId	The interaction id for this activity. This only applies to **message** types.
flowProcess	The pipeline flow that was taken. The current values are **Default** or **AutoReplay**.
sender	The sender *source* of this activity.
recipient	The recipient (receiver) *source* of this activity.
message	If the *activity.type* is **message** then this value is required. See *message type* for more details
event	If the *activity.type* is **event** then this value is required. See *event type* for more details

12.2 Activity Header

When sending activities, there's a minimal amount of header properties that are required. See details below:

Property	Mandatory	Description
type	Yes	The type of activity to send. This can only be *message* or *event*
sender.integrationId	**Description**	If the source of **integrationId** is bound to a **customer** integration, then this property is **not** required. However, if the **integrationId** is bound to either an **agent** or **bot** integration, then this property **is** required.
externalId	No	This can be any string value and will be attached to the lifetime of this activity if provided by the sender. Typically this is used by tenants to maintain a reference or metadata to a given tenant resource. For the most part this value will be null.

```
{
    "externalId": "some-external-id",
    "type": "message | event",
    "sender": {
        "integrationId": "00000000-0000-0000-0000-000000000001"
    },
    "message": {
        "type": "text",
        "text": "Hi there!"
    },
    "event": {
        "type": "payload",
        "payloadType": "my.payload.01",
        "payload": {
            "data1": "value1",
            "data2": "value2",
            "data3": "value3"
        }
    }
}
```

12.3 Activity Source

An activity will always contain a **sender** and **recipient** nodes. The the sender is the source of the activity and the recipient is the source that will receive the activity.

```
{
    "sender": {
        "integrationId": "00000000-0000-0000-0000-000000000001",
        "integrationType": "Customer",
        "channelType": "Direct",
        "tokenId": "t+8qymYD1jp7wDSHG+3eUA=="
    },
    "recipient": {
```

(continues on next page)

(continued from previous page)

```
        "integrationId": "00000000-0000-0000-0000-000000000006",
        "integrationType": "Agent",
        "channelType": "Direct",
        "tokenId": "971480cb-938c-4dfd-be4e-01756c833490.00000000-0000-0000-0000-
↪000000000003"
    }
}
```

Property	Description
integrationId	The integration id of the source.
integrationType	The *integration type* of the source.
channelType	The *channel type* of the source.
tokenId	Reserved for Hubster.

12.4 Message Types

An activity **message** supports the following **types**. Messages are an activity's **first-class-citizen** as they make up the majority of events being sent and received between integrations.

Note: Activities are send via the *Direct API* endpoint. Sending an activity is quite simple and requires minimal amount of header details. Once Hubster receives an activity to process, the engine will enrich the activity with more details, such as sources, etc. See *Activity* on this page.

12.4.1 Text

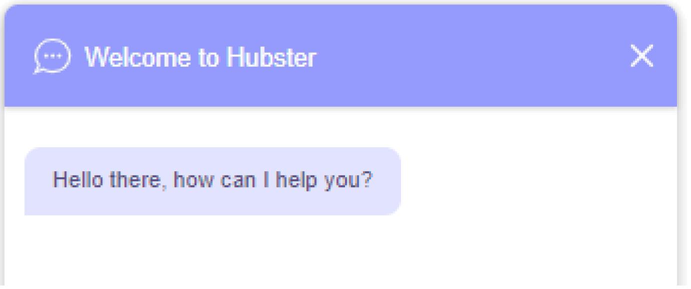
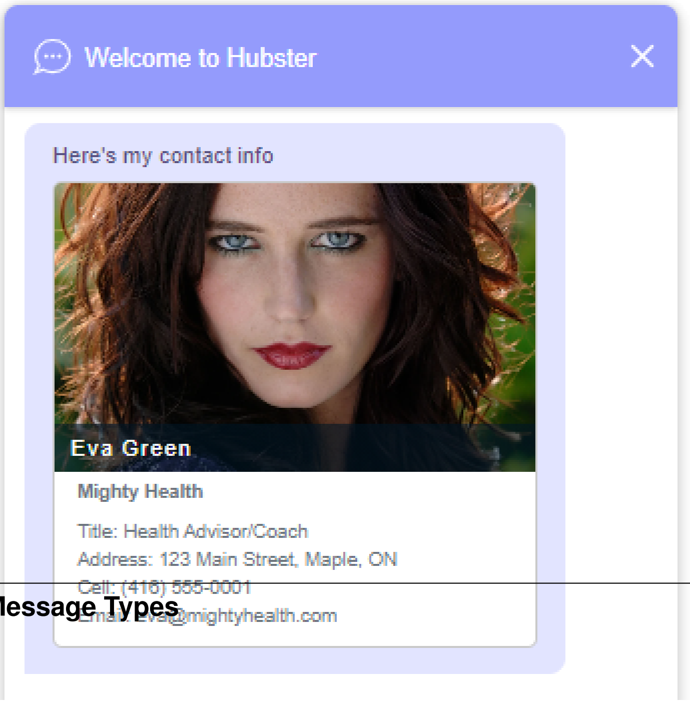
Sources allowed to send: **customer**, **agent** and **bot**.

Property	Mandatory	Description
type	Yes	Must be **text**.
text	See **note**	The text message to send. Links such as **Image**, **Youtube**, **Vimeo**, **Video**, **Audio** or **location**, may convert this message type to it's property message equivalent if no additional text was provided. If additional text was provided, then Hubster will add a message equivalent, such as **Youtube**, for example to the items array.
items	See **note**	A list of items containing zero or more of the following **messages types**: • youtube • vimeo • video • audio • image • attachment • location • contact • card
actions	See **note**	A list of actions containing zero or more of the following **action types**: • postback • reply • link

Note: The **text message** type must provide one or more of the following **mandatory** values:

- text
- items
- actions

Examples

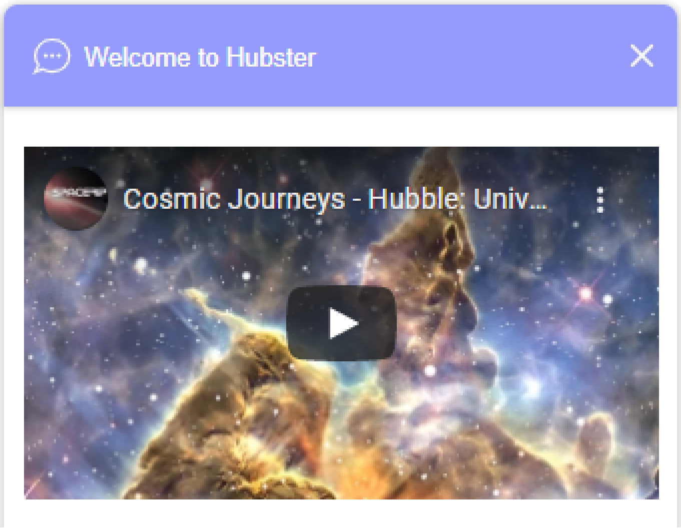
Request	View
`{ "type": "text", "text": "Hello there, how can I help you?" }`	 Welcome to Hubster Hello there, how can I help you?
{	 Welcome to Hubster Here's my contact info Eva Green Mighty Health Title: Health Advisor/Coach Address: 123 Main Street, Maple, ON Cell: (416) 555-0001 Email: eva@mightyhealth.com

12.4.2 Youtube

Sources allowed to send: **customer**, **agent** and **bot**.

Property	Mandatory	Description
type	Yes	Must be **youtube**.
url	Yes	The youtube url, which can be in anyone of the following formats: • *https://youtube.com/embed/x1245b* (preferred) • *https://youtube.com/watch?v=x1245b* • *https://m.youtube.com/watch?v=x1245b* • *https://youtu.be/watch?v=x1245b*

Example

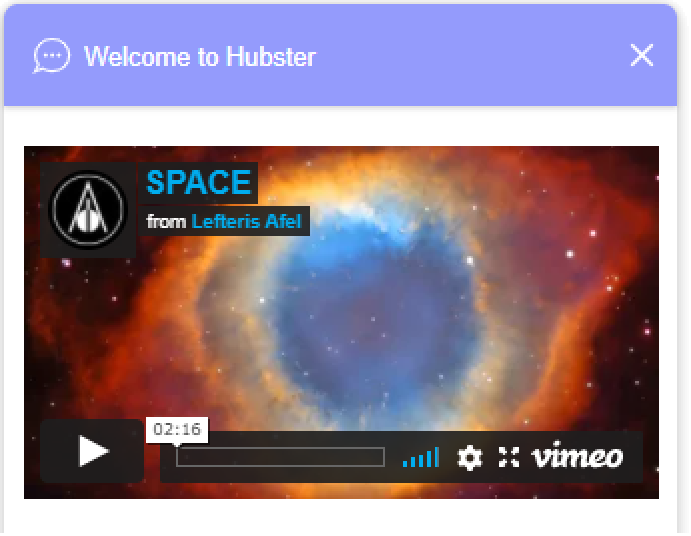
Request	View
`{ "type": "youtube", "url": "https://youtube.com/watch?v=x1245b" }`	

12.4.3 Vimeo

Sources allowed to send: **customer**, **agent** and **bot**.

Property	Mandatory	Description
type	Yes	Must be **vimeo**.
url	Yes	The Vimeo url, which can be in anyone of the following formats: • *https://player.vimeo.com/video/12345678* (preferred) • *https://vimeo.com/12345678*

Example

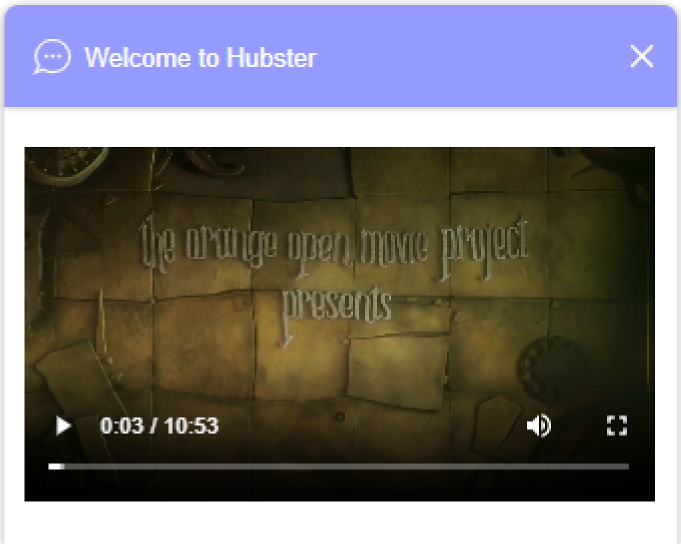
Request	View
`{ "type": "vimeo", "url": "player.vimeo.com/video/12345678" }`	

12.4.4 Video

Sources allowed to send: **customer**, **agent** and **bot**.

Property	Mandatory	Description
type	Yes	Must be **video**.
url	Yes	The video url, which can be in anyone of the following formats: • *.mp4* (preferred) • *.mov*
label	No	The label of this audio. Think of the label as a title to be displayed. **Note**: label is channel specific and may not render on certain channels.
mimeType	No	The mime type of the video. Hubster will try it's best to determine the mime type based on the **url**.

Example

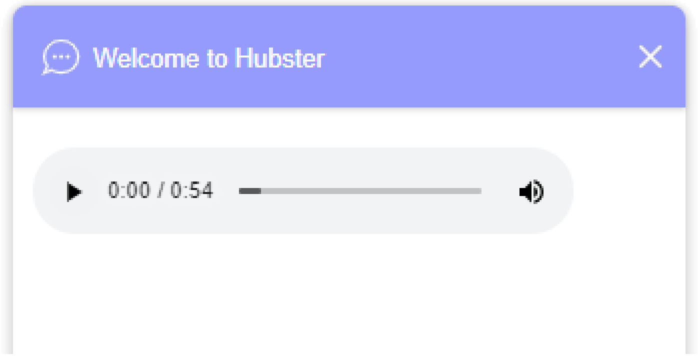
Request	View
`{ "type": "video", "url": "http://site.com/myvideo.mp4" }`	 Welcome to Hubster the orange open movie project presents 0:03 / 10:53

12.4.5 Audio

Sources allowed to send: **customer**, **agent** and **bot**.

Property	Mandatory	Description
type	Yes	Must be **audio**.
url	Yes	The audio url, which can be in anyone of the following formats: • *.mp3* (preferred) • *.mp4* • *.wav*
label	No	The label of this audio. Think of the label as a title to be displayed. **Note**: label is channel specific and may not render on certain channels.
mimeType	No	The mime type of this audio. Hubster will try it's best to determine the mime type based on the **url**.

Example

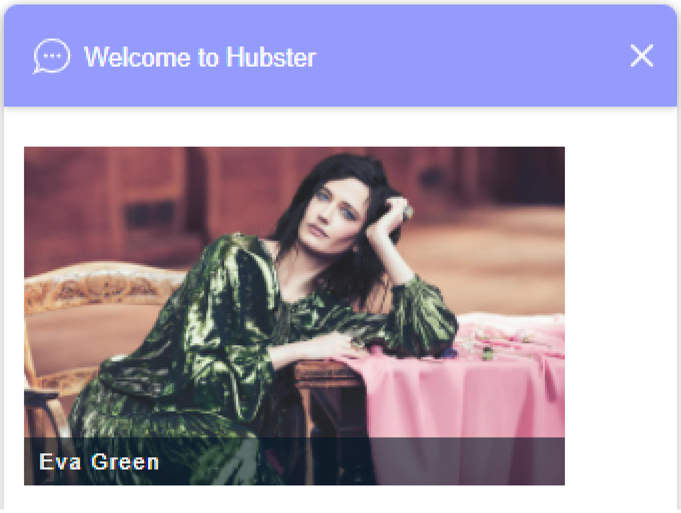
Request	View
`{ "type": "audio", "url": "http://site.com/myaudio.mp3" }`	

12.4.6 Image

Sources allowed to send: **customer**, **agent** and **bot**.

Property	Mandatory	Description
type	Yes	Must be **image**.
url	Yes	The image url.
urlAnchor	No	The url anchor. Used when user clicks on image.
alt	No	The alternate text for this image.
title	No	The text to show on the image. **Note**: title is channel specific and may not render on certain channels.
channels	No	Channel specific applied properties. The example below shows how to render the title on a **Webchat** channel.

Example

Request	View
	

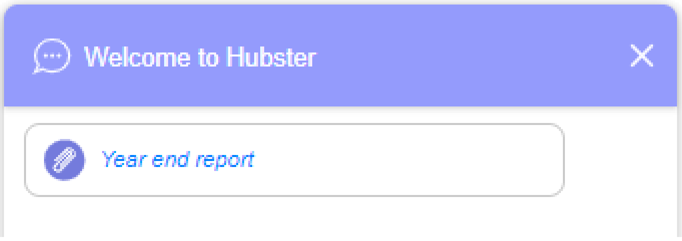
```
{
   "type": "image",
   "url": "http://site.com/myimage.png",
   "alt": "Some alternate text",
   "title": "Eva Green",
```

12.4.7 Attachment

Sources allowed to send: **customer**, **agent** and **bot**.

Property	Mandatory	Description
type	Yes	Must be **attachment**.
label	Yes	The label for this attachment.
mimeType	Yes	The mime type for this attachment i.e pdf, etc.
url	Yes	The attachment url.

Example

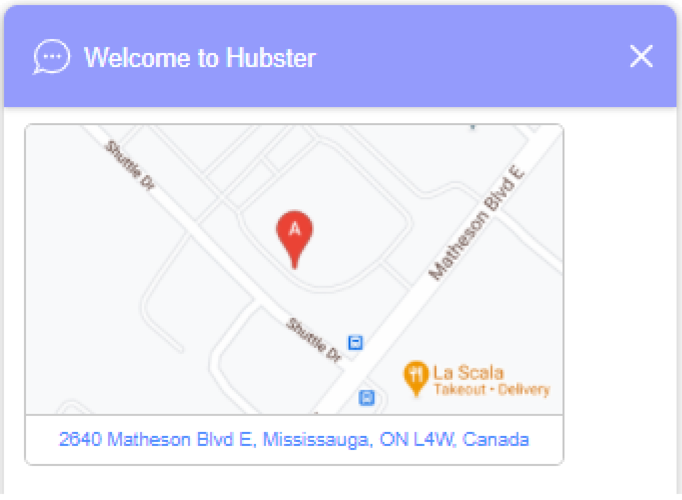
Request	View
`{ "type": "attachment", "label": "Year end report", "mimeType": "pdf", "url": "http://site.com/myfile.pdf" }`	

12.4.8 Location

Sources allowed to send: **customer**, **agent** and **bot**.

Property	Mandatory	Description
type	Yes	Must be **location**.
Address	See **note**	A fully qualified address.
latitude	See **note**	A latitude coordinate value. **Note**: The longitude coordinate value must be supplied.
longitude	See **note**	A longitude coordinate value. **Note**: The latitude coordinate value must be supplied.

Example

Request	View
`{ "type": "location", "address": "2640 Matheson, Mississauga, ON", "latitude": 43.8425254, "longitude": -79.5240196 }`	 Welcome to Hubster 2640 Matheson Blvd E, Mississauga, ON L4W, Canada

Note: Either a fully qualified **address** or a set of **latitude/longitude** coordinates must be supplied. If both **address** or **latitude/longitude** are supplied, Hubster will resort to using the **latitude/longitude** coordinates as the preferred option.

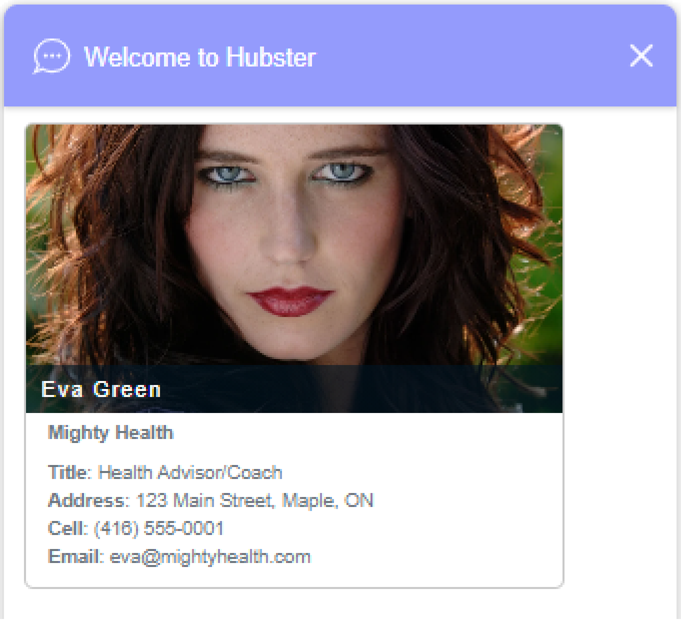
Please note, when using **latitude/longitude** coordinates, Hubster will try to yield the appropriate address. However, if the address yielded is not exact, then the **latitude/longitude** coordinates may be off. Alternatively, you can always use the **address** property without the need to provide **latitude/longitude** coordinates.

12.4.9 Contact

Sources allowed to send: **customer**, **agent** and **bot**.

Property	Mandatory	Description
type	Yes	Must be **contact**.
imageUrl	No	The image url to the contact.
title	Yes	At minimum, the contact message requires a title. i.e. Person's name, company name, job title, etc.
subtitle	No	A subtitle for the contact. i.e. company name, job title, etc.
properties	No	A tuplet made out of key/value/type set that can used to provide more metadata for the contact. See example. **Note:** The **type** portion of the tuplet is not required, however, if used, can provide additional metadata for certain property types. For example, if Hubster detects that a recipient device supports **vcards**, such as an SMS device, Hubster will create a contact element, allowing the recipient of the message to store the contact to their device's contact list. Hubster supports the following **vcard** types and their counterpart: • address; `work`, `home` • phone; `work`, `home`, `cell` • email
channels	No	Channel specific applied properties. The example below shows how to render the title on a **Webchat** channel.

Example

Request	View
	 Welcome to Hubster Eva Green Mighty Health **Title:** Health Advisor/Coach **Address:** 123 Main Street, Maple, ON **Cell:** (416) 555-0001 **Email:** eva@mightyhealth.com

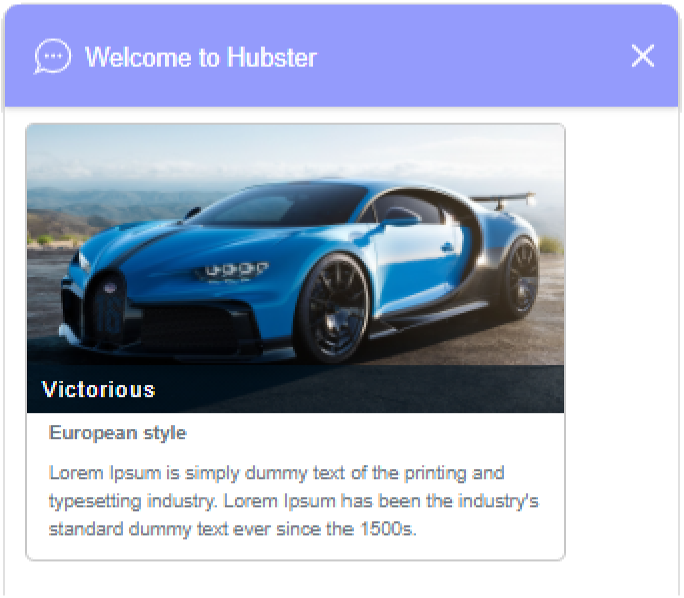
```
{
    "type": "contact",
    "imageUrl": "https://site.com/eva.png",
    "title": "Eva Green",
    "subtitle": "Mighty Health
```

12.4.10 Card

Sources allowed to send: **customer**, **agent** and **bot**.

Property	Mandatory	Description
type	Yes	Must be **card**.
urlType	No	If **url** is supplied, then this property is required. The possible types are as follows: • image • youtube • vimeo • video • audio
url	No	The link to the resource to display. The **urlType** property must be provided. The possible types are as follows: • image • youtube • vimeo • video • audio
fallbackImageUrl	No	When supplying a **url** that supports an image placeholder, such as youtube for example, and the link doesn't support an image, Hubster will use the **fallbackImageUrl** link as an alternate.
title	No	A title to display.
subtitle	No	A subtitle to display.
content	No	The content to display.
channels	No	Channel specific applied properties. The example below shows how to render the title on a **Webchat** channel. Note: only applicable if **urlType=image**

Example

Request	View
	

Welcome to Hubster

Victorious

European style

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s.

```
{
  "type": "card",
  "urlType": "image",
  "imageUrl": "https://site.com/car.png",
  "title": "Victorious",
```

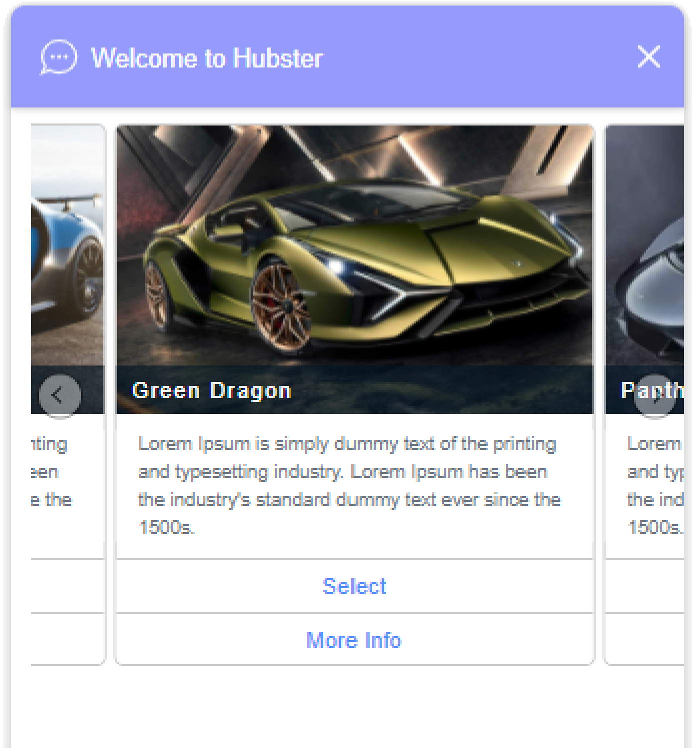
12.4.11 Carousel

Sources allowed to send: **agent** and **bot**.

Property	Mandatory	Description
type	Yes	Must be **carousel**.
items	Yes	Must contain one or more of the following message types: • image • youtube • vimeo • video • audio
channels	No	Channel specific applied properties. The example below shows how to render the title on a **Webchat** channel. Note: only applicable to image items.

Example

Request	View



Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s.

```
{
␣
↪␣
↪
↪"type
↪":␣
↪
↪"carousel
↪",
↪
␣
↪␣
↪
↪"items
↪":␣
↪[
␣
↪␣
↪␣
↪␣
↪
↪{
␣
↪␣
↪␣
↪␣
↪␣
↪␣
↪
↪"title
↪":␣
↪
↪"Victorious
↪",
↪
␣
↪␣
```

Note: Certain devices do not support carousels. If a device is unable to display a carousel, Hubster will render the carousel as a list.

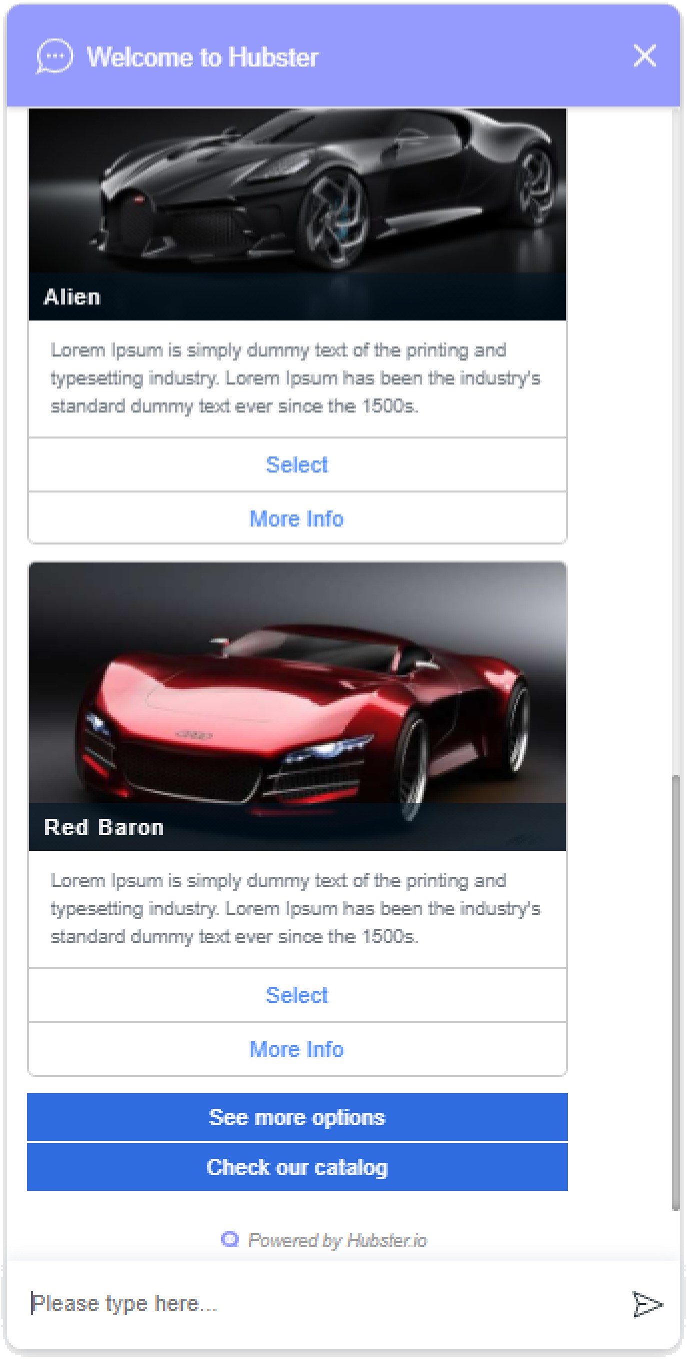
12.4.12 List

Sources allowed to send: **agent** and **bot**.

Note: Lists are similar to Carousels. The only differences are: how it's displayed and that a list provides the ability to offer a global set of **actions** for the message type.

Property	Mandatory	Description
type	Yes	Must be **list**.
items	Yes	Must contain one or more of the following message types: • image • youtube • vimeo • video • audio
actions	No	A list of actions containing zero or more of the following **action types**: • postback • reply • link
channels	No	Channel specific applied properties. The example below shows how to render the title on a **Webchat** channel. Note: only applicable to image items.

Example

Request	View
{ "type": "list", "items": [	

Welcome to Hubster

Alien

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s.

Red Baron

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s.

```
{
    "type": "list",
    "items": [
```

Note: Certain devices do not support lists. If a device is unable to display a list, Hubster will render the list as a carousel.

12.4.13 Commands

Sources allowed to send: **agent** and **bot**.

Commands are no different then sending a simple one line **text** message type. The main difference is when issuing a command it must start with a double (colon) **::** to be recognized. For example when sending this text, **::some_command** *-arg1 -arg2*, … Hubster will treat this as a command to be processed.

See examples below:

Request

```
{
  "type": "text",
  "text": "::resp -n contact.eva.green"
}
```

```
{
  "type": "text",
  "text": "::trans -force -n shopify"
}
```

12.5 Event Types

Event types are similar to message types and but are simpler in nature.

12.5.1 Basics

Note: Apart from the **payload** event type, the other types are currently not supported and will be on a future Hubster road-map.

```
{
  "type": "seen | typing_on | typing_off | payload"
}
```

Property	Mandatory	Description
type	Yes	Can be any one of the following types: • **seen** - *mark the message as been seen* • **typing_on** - *force the device to show the typing gif* • **typing_off** - *force the device to turn off the typing gif* • **payload** - *the event contains a payload - see payload description below*

12.5.2 Payload

Property	Mandatory	Description
type	Yes	Must be **payload**
payloadType	Yes	A unique identifier that describes the payload type.
payload	Yes	Can be any **json** object.

Example

```
{
  "type": "payload",
  "payloadType": "my.payload.01",
  "payload": {
      "data1": "value1",
      "data2": "value2",
      "data3": "value3"
  }
}
```

CHAPTER 13

Types

13.1 Activity Types

Type	Description
Message	The *activity* contains an *message* node.
Event	The *activity* contains an *event* node.

13.2 Channel Types

Type	Source	Value	Description
Direct	General	1	Direct channel.
Bot	Business	2	Bot channel.
System	Business	3	System type. This type is mainly used for Webhooks.
Messenger	Customer	101	Facebook Messenger channel.
TwilioSMS	Customer	102	Twilio SMS channel.
Line	Customer	103	Line channel.
Telegram	Customer	104	Telegram channel.
Kik	Customer	105	Kik channel.
WebChat	Customer	106	WebChat channel.
Slack	Business	1001	Slack channel.

13.3 Integration Types

Type	Description
Customer	Integration type for Customers.
Agent	Integration type for Agents.
Bot	Integration type for Bots.
System	Integration type for Webhooks.

CHAPTER 14

Error Codes

Below are a list of all possible REST API error codes for all Hubster API related services.

14.1 Identity

Below is a full list of all possible Hubster Identity REST API error codes.

Error	HTTP Status	Description
IDT000100	500	System Error.

14.2 Portal

Below is a full list of all possible Hubster Portal REST API error codes.

Error	HTTP Status	Description
PRT000100	500	System Error.
PRT000101	403	Forbidden.
PRT000102	401	Unauthorized access.
PRT000103	501	Requested operation is not implemented.
PRT000104	410	Requested resource or operation is not available.
PRT000105	404	Requested resource was not found.
PRT000106	409	Resource you are trying to create already exists.
PRT000107	417	Current request or operation is not valid.
PRT000108	408	Request took too long to execute and timed out.

Continued on next page

Table 1 – continued from previous page

Error	HTTP Status	Description
PRT000109	417	Requested action was aborted.
PRT000110	403	Requested action is not allowed.
PRT000200	400	Required query parameter was not supplied.
PRT000201	400	One or more parameters have invalid format.
PRT000202	400	Please correct supplied format for used parameter(s).
PRT000203	400	Provided `GUID` has bad format.
PRT000204	400	Provided date has bad format.
PRT000205	400	One or more parameters have invalid date format.
PRT000206	400	Provided parameter value is not supported (out of range).
PRT000207	400	Provided parameter is out of predefined range.
PRT000208	400	Provided parameter has to be greater than zero.
PRT000209	400	Parameter must be greater than or equal to zero.
PRT000210	400	Invalid Number format provided.
PRT000211	400	Invalid email format provided.
PRT000212	400	Criteria parameter is required when supplying a `searchBy` parameter.
PRT000300	400	Root body section is missing.
PRT000301	400	Required property is missing.
PRT000302	400	Property has is invalid type.
PRT000303	400	Validation failed. Property not supported.
PRT000304	400	Request parameter has bad format. Expected to be a valid `decimal` value.
PRT000305	400	Request parameter has bad format. Expected to be a valid `GUID` value.
PRT000306	400	Request collection must contain one or more elements.
PRT000307	400	Messaged was empty.
PRT000308	400	Request body must contain Location, either an address and/or latitude/longitude coordinates.
PRT000400	400	Tenant already exists.
PRT000401	400	User already exists.
PRT000599	404	User not found.
PRT000600	400	Name already exists.
PRT000699	404	Hub not found.
PRT000700	400	An integration with name already exists.
PRT000701	400	An integration with same name has already been assign to a hub. You can only add this channel once across all hubs.
PRT000799	404	Integration not found.

14.3 Engine

Below is a full list of all possible Hubster Engine REST API error codes.

Error	HTTP Status	Description
ENG000100	500	System Error.
ENG000101	403	Forbidden.
ENG000102	401	Unauthorized access.
ENG000103	501	Requested operation is not implemented.

Continued on next page

Table 2 – continued from previous page

Error	HTTP Status	Description
ENG000104	410	Requested resource or operation is not available.
ENG000105	404	Requested resource was not found.
ENG000106	409	Resource you are trying to create already exists.
ENG000107	417	Current request or operation is not valid.
ENG000108	408	Request took too long to execute and timed out.
ENG000109	417	Requested action was aborted.
ENG000110	403	Requested action is not allowed.
ENG000200	400	Required query parameter was not supplied.
ENG000201	400	One or more parameters have invalid format.
ENG000202	400	Please correct supplied format for used parameter(s).
ENG000203	400	Provided `GUID` has bad format.
ENG000204	400	Provided date has bad format.
ENG000205	400	One or more parameters have invalid date format.
ENG000206	400	Provided parameter value is not supported (out of range).
ENG000207	400	Provided parameter is out of predefined range.
ENG000208	400	Provided parameter has to be greater than zero.
ENG000209	400	Parameter must be greater than or equal to zero.
ENG000210	400	Invalid Number format provided.
ENG000211	400	Invalid email format provided.
ENG000212	400	Criteria parameter is required when supplying a `searchBy` parameter.
ENG002000	400	Provided tenant is invalid.
ENG002001	400	Your account is disabled.
ENG002002	400	Account evaluation period has expired.
ENG003000	400	Conversation request requires `body` to be present.
ENG003001	400	Conversation request is missing required property.
ENG003002	400	Conversation request parameter has bad format. Expected to be a valid `GUID` value.
ENG005000	400	Direct Inbound request requires `body` to be present.
ENG005001	400	Direct Inbound request is missing required property.
ENG005002	400	Direct Inbound request does not support provided property.
ENG005003	400	Direct Inbound request must contain one of the following sections: `root.message` or `root.event`.
ENG005004	400	Direct Inbound request can only contain one root with the following sections: `root.message` or `root.event`.
ENG005015	400	Direct Inbound request collection must contain one or more elements.
ENG005020	400	Direct Inbound request parameter has bad format. Expected to be a valid `GUID` value.
ENG005021	400	Direct Inbound request parameter has bad format. Expected to be a valid `decimal` value.
ENG005023	400	Direct Inbound request body was empty.
ENG005024	400	Direct Inbound request body must contain `Location`, either an `address` and/or `latitude/longitude` coordinates.
ENG005500	404	Hub does not exist.
ENG005501	400	Provided Hub does not have any Agent or Bot integration configured to receive or interact with customer messages.
ENG006000	404	Provided integration does not exist.
ENG006500	404	Provided conversation does not exist.
ENG006501	400	Customer is no longer responding to messages.
ENG006502	400	Your Hubster integration has been terminated and is no longer active. Please contact your Administrator.

Continued on next page

Table 2 – continued from previous page

Error	HTTP Status	Description
ENG006503	400	Conversation was paused.
ENG007500	400	Conversation encountered a web related issue.
ENG007501	400	Conversation encountered a web security related issue.
ENG007502	400	Conversation encountered a runtime related issue.
ENG007510	400	Customer failed to receive your message. This was due to an unauthorized issue on their end. Please check with your Administrator.
ENG007511	400	A web related issue was detected on Hub.
ENG007512	400	An unreachable web-endpoint was detected on Hub.
ENG008000	400	Message Spark encountered a web related issue.
ENG008001	400	Message Spark encountered a web security related issue.
ENG008002	400	Message Spark encountered a runtime related issue.
ENG008500	400	No upload files were provided.
ENG008501	400	Invalid `URL` was provided.
ENG008502	400	File you submitted was not received by the other party.
ENG008503	400	The other party tried to send you a file but failed.
ENG009000	400	Invalid command. You must have an actually command in front of the double colon e.g. `::mycommand [args]...`
ENG009001	400	Unknown command.
ENG009200	400	Command was not found. Please type `::{1} --list` to see the full list of available commands.
ENG009201	400	No commands have been configured for this hub.
ENG009202	400	No commands were found for the category.
ENG009299	400	There was an error while executing command. Please contact technical support.

14.4 Events

Below is a full list of all possible Hubster Events REST API error codes.

Error	HTTP Status	Description
EVT000100	500	System Error.