Info servizio

Tipo: API REST  
Content-Type: application/json  
Url servizio: <https://api.contactlab.it/hub/v1/>

Parametri necessari

Workspace

Primo parametro da settare è quello del workspace ([https://api.contactlab.it/hub/v1/workspace/{id-workspace}/](https://api.contactlab.it/hub/v1/workspace/%7bid-workspace%7d/)). Procede ogni risorsa ed identifica uno specifico ambiente di lavoro.

Nodes

Altro Elemento fondamentale oltre il workspace in contacthub è il nodo. I nodi in contacthub sono dei “contenitori” in cui inserire i customer. I nodi possono essere di due tipologie

* Entry node: nodi in cui si possono scrivere e leggere i customer (lista dei nodi entry configurati <https://api.contactlab.it/hub/v1/workspaces/1/configuration/nodes/entries>). Nella definizione dei nodi di tipo entry vengono anche scelte le properties obbligatorie.
* View node: nodi in cui è possibile solo leggere i dati del customer. Questo perché i view node sono nodi che vengono creati attraverso (per adesso non li stiamo prendendo in considerazione)

Souces

La source è l’abilitazione di un’applicazione ad operare su uno o più nodi. La creazione di una source genera un token da utilizzare nell’autenticazione. Il token non scade mai. Può essere disabilitato o rigenerato.

Autenticazione

Nell’header va passato il parametro “Authorization: Bearer {token-della-source}”.

Risorse principali

Le due risorse principali sono:

/customers -> contenitori dei profili di customer:

* Id -> unique identifier assegnato da contachub
* externalId -> the external id of customer
* nodeId -> entry node `id`
* base -> properties predefinite da noi e di cui è possibile prendere il json schema attraverso la chiamata: <https://api.contactlab.it/hub/v1/workspaces/1/models/properties/base>
* extended -> parte “extended”, definita nel workspace dall’utilizzatore, di cui è possibile prendere il json schema attraverso la chiamata: <https://api.contactlab.it/hub/v1/workspaces/1/configuration/properties/extended>
* extra -> è una properties in cui è possibile salvare di tutto. Su quel campo non ci sono controlli
* tags -> I tag assegnati al customer. I customer possono essere di tipo manual (inseriti dall’utente) ed auto (autogenerati da elaborazioni).

/events -> contenitori degli eventi. Gli eventi sono normalmente associati ad un customer e con una data.

* id -> Unique identifier of event rilasciato da contacthub
* customerId -> customer id
* bringBackProperties -> oggetto con cui è possibile inserire eventi anonimi o eventi assegnati ad un externalId
* type -> the name of type event
* context -> contest dove è avvenuto l’evento. La lista dei contesti disponibili è qui: /workspaces/{id-workspace}/models/contexts
* properties -> the json schema relatico all’event type,
* date -> data di quando è avvenuto l’evento. In caso sia lasciato nullo viene inserito l’orario della chiamata API

Gli eventi sono “built in” in contacthub. Ogni “event type” ha le proprie properties definite da passare nell’inserimento. Per avere la lista degli eventi definiti: <https://api.contactlab.it/hub/v1/workspaces/1/configuration/events/>

E’ possibile poter inserire eventi anonimi o riferiti ad externalId. Questo è possibile attraverso la property bringBackProperties della risorsa events.

Funzionalità

SDK

* Prima fase
* Customers List -> GET /workspaces/{id-workspace}/customers?nodeId={node-id}
* Get Customer (id, externalId)
* Id: GET /workspaces/{id-workspace}/customers/{id}
* externalId: GET /workspaces/{id-workspace}/customers?externalId={externalId}&nodeId={nodeId}
* Add Customer -> POST /workspaces/{id-workspace}/customers
* Update Customer -> PUT /workspaces/{id-workspace}/customers/{id}
* Delete Customer -> DELETE /workspaces/{id-workspace}/customers/{id}
* Get customer like -> GET /workspaces/{id-workspace}/customers/{id}/likes/
* Add customer like -> POST /workspaces/{id-workspace}/customers/{id}/likes/
* Get customer subscription -> GET /workspaces/{id-workspace}/customers/{id}/subscriptions/
* Add customer subscription -> POST /workspaces/{id-workspace}/customers/{id}/subscriptions/
* Get customer education -> GET /workspaces/{id-workspace}/customers/{id}/educations/
* Add customer education -> POST /workspaces/{id-workspace}/customers/{id}/educations/
* Get customer job -> GET /workspaces/{id-workspace}/customers/{id}/jobs/
* Add customer job -> POST /workspaces/{id-workspace}/customers/{id}/jobs/
* Add customer session -> POST /workspaces/{id-workspace}/customers/{id}/sessions/
* Get customer session -> GET /workspaces/{id-workspace}/customers/{id}/sessions/
* Delete customer session -> DELETE /workspaces/{id-workspace}/customers/{id}/sessions/{sessionId}
* Get Customer Event (context, type)
* Senza filtro: GET /workspaces/{id-workspace}/events?customerId={customerId}
* Con filtro: GET /workspaces/{id-workspace}/events?customerId={customerId}&context={context}&type={type}
* Add Customer Event -> POST /workspaces/{id-workspace}/events
* Add Anonymus Event -> POST /workspaces/{id-workspace}/events
* Seconda fase:
* Customer query
* Get Segment
* Add Segment
* Get segment Customer

INTEGRAZIONE

* Gestione senza eventi anonimi:
* Customers List
* Get Customer (id, externalId)
* Add Customer
* Update Customer
* Delete Customer
* Get Customer Event (context, type)
* Add Customer Event
* Gestione con eventi anonimi:
* Customers List
* Get Customer (id, externalId)
* Add Customer
* Update Customer
* Delete Customer
* Add customer session
* Get customer session
* Delete customer session
* Get Customer Event (context, type)
* Add Customer Event
* Add Anonymus Event

Casi d’uso

Inserimento di un customer solo properties base

La chiamata per inserire un customer è la POST sul metodo /workspaces/{id-workspace}/customers. Per inserire le properties base (built-in in contactlhub) è necessario passare i valori da inserire nella property “base” dell’oggetto customer. Il json schema delle properties base fornite da contacthub è disponibile chiamando in GET il metodo /workspaces/{id-workspace}/models/properties/base.

Inserimento di un customer con properties base + properties extended

La chiamata per inserire un customer è la POST sul metodo /workspaces/{id-workspace}/customers. Per inserire le properties base (built-in in contactlhub) è necessario passare i valori da inserire nella property “base” dell’oggetto customer. Il json schema delle properties base fornite da contacthub è disponibile chiamando in GET il metodo /workspaces/{id-workspace}/models/properties/base.

Le properties extended sono le properties definite da chi sta usando contacthub per integrare quelle built in in contachub. Il json schema delle properties extended definite è possibile averlo chiamando in GET il metodo /workspaces/{id-workspace}/configuration/properties/extended. I valori da inserire devono essere passati property “extended” dell’oggetto customer.

Inserimento di un customer properties base + tag

La chiamata per inserire un customer è la POST sul metodo /workspaces/{id-workspace}/customers. Per inserire le properties base (built-in in contactlhub) è necessario passare i valori da inserire nella property “base” dell’oggetto customer. Il json schema delle properties base fornite da contacthub è disponibile chiamando in GET il metodo /workspaces/{id-workspace}/models/properties/base.

I tag vanno inseriti nella property “tags” dell’oggetto customer. Sono divisi in due categorie:

* Manual: tag inseriti manualmente
* Auto: tag inseriti automaticamente da applicazioni a seguito di elaborazioni

Inserimento di un customer properties base + extra

La chiamata per inserire un customer è la POST sul metodo /workspaces/{id-workspace}/customers. Per inserire le properties base (built-in in contactlhub) è necessario passare i valori da inserire nella property “base” dell’oggetto customer. Il json schema delle properties base fornite da contacthub è disponibile chiamando in GET il metodo /workspaces/{id-workspace}/models/properties/base.

La property “extra” dell’oggetto customer è una property in cui è possibile salvare qualsiasi tipo di dato (stringa/json).

Aggiornamento di un customer attraverso il customer id

La chiamata per aggiornare un customer conoscendo il customer id è la PUT sul metodo /workspaces/{id-workspace}/customers/{id}. Va ripassato tutto l’oggetto customer in quanto non è previsto l’aggiornamento parziale (PATCH). Quindi se non si conoscono tutti i valori va fatta prima la chiamata in GET sul metodo /workspaces/{id-workspace}/customers/{id} per riprendere i valori delle del customer che si andrà ad aggiornare.

Aggiunta di un tag attraverso il customer id

La chiamata per aggiungere un tag al customer conoscendo il customer id è la PUT sul metodo /workspaces/{id-workspace}/customers/{id}. Come per l’aggiornamento delle properties (base/extended/base) va ripassato tutto l’oggetto customer in quanto non è previsto l’aggiornamento parziale (PATCH). Quindi se non si conoscono tutti i valori va fatta prima la chiamata in GET sul metodo /workspaces/{id-workspace}/customers/{id} per riprendere i valori delle del customer che si andrà ad aggiornare.

Aggiornamento del customer attraverso l’external id

La chiamata per aggiornare un customer è la PUT sul metodo /workspaces/{idworkspace}/customers/{id}. Questo prevede la conoscenza del customer id. Quindi non conoscendo il customer id va prima fatta la ricerca attraverso l’external id del customer id. Questo è possibile farlo richiamando in GET il metodo /workspaces/{id-workspace}/customers passando in querystring il paramento externalId valorizzato con external id che si vuole ricercare. L’output della chiamata sarà l’oggetto customer, comprensivo di customer id. L’aggiornamento prevedere il passaggio dell’intero oggetto Customer in quanto non è abilitata l’aggiornamento parziale (PATCH).

Inserimento di un evento di un customer già presente in contacthub

La chiamata per inserire un evento è il POST sul metodo /workspaces/{id-workspace}/events. L’evento è caratterizzato da un tipo e da un contesto. Il contesto è “l’ambiente” in cui viene generato l’evento. Non esiste una relazione gerarchica tra i due elementi. Quindi un tipo potrebbe essere associato a più contesti. Esempio l’evento di tipo “acquisto” può essere inserito in un contesto “ecommerce” o “retail”. I tipi di eventi sono predefiniti da contacthub e la lista è disponibile attraverso la chiamata in GET del metodo /workspaces/{id-workspace}/configuration/events. Nella definizione del tipo di evento c’è il dettaglio delle properties, definite attraverso un json schema, da passare al momento dell’inserimento dell’evento. Per ogni tipo di evento oltre alla struttura predefinita c’è una property aggiuntiva “extraProperties” in cui poter inserire liberamente properties (semplici, array, oggetti) con cui estendere le informazioni dell’evento da inserire. In contacthub sono stati definiti anche due tipi di eventi generici con cui poter inserire tipi di eventi ancora non ancora censiti ed in cui solo la property del nome dell’evento è stato definita. Questi eventi generici sono: genericActiveEvent (eventi fatti attivamente dal

customer) e genericPassiveEvent (eventi in cui il customer è passivo. Esempio: ricezione di un’offerta telefonica, etc.).

Per gli eventi di cui si conosce il customer id la property “bringBackProperties” va ignorata.

Inserimento di un evento di un customer di cui si conosce l’external id

L’inserimento dell’evento di un customer di cui si conosce solo l’external id è molto simile a quella di cui si conosce il customer id in quanto la property “customerId” va lasciata nulla e va valorizzata la property “bringBackProperties”. L’oggetto BringBackProperty dovrà essere valorizzato cosi:

* type: EXTERNAL\_ID
* value: valore dell’external id
* nodeId: nodo a cui attribuire l’evento

In questo modo contacthub assegnerà l’evento al customer del nodo specificato con l’external id indicato. E’ possibile inserire anche eventi attraverso l’external id di un customer ancora non inserito. In questo caso contacthub inserirà automaticamente un customer sul nodo specificato valorizzando solo l’external id.

Inserimento di un evento di un customer anonimo

Contacthub permette il salvataggio di eventi anonimi di cui si prevede che prima o poi venga identificato il customer. Questo è possibile in quanto contacthub mette disposizione un buffer limitato di eventi “parcheggiati” ed assegnati ad un “SESSION\_ID”. Il session id è un identificativo univoco che va assegnato al customer anonimo. L’assegnazione del session id è a carico di chi usa contacthub.

L’inserimento dell’evento di un customer anonimo è molto simile a quella di cui si conosce il customer id in quanto la property “customerId” va lasciata nulla e va valorizzata la property “bringBackProperties”. L’oggetto BringBackProperty dovrà essere valorizzato cosi:

* type: SESSION\_ID
* value: valore del session id
* nodeId: nodo a cui attribuire l’evento

In questo modo contacthub parcheggerà l’evento in attesa che il customer si identifichi.

Identificazione di un customer anonimo Se un customer anonimo a cui è stato assegnato un session id si identifica per ricondurre a lui gli eventi “parcheggiati”, oltre l’inserimento del customer attraverso il POST sul metodo /workspaces/{idworkspace}/customers, va fatta la POST sul metodo /workspaces/{idworkspace}/customers/{id}/sessions per assegnare il session id al customer appena inserito.

Questo ricondurrà automaticamente tutti gli eventi parcheggiati con quel session id al customer appena inserito.