

Monetico Online

SDK Hosted Fields « hostedfields.js »

Documentation



SOMMAIRE

1. Introduction	3
2. Références	4
3. Prérequis	5
3.1. Configuration du TPEv (Terminal de Paiement Electronique virtuel)	5
3.2. Génération du jeton de saisie de carte temporaire	5
3.3. Utilisation du champ <i>hosted field</i> de type <i>cardPreview</i>	5
4. Intégration	6
4.1. Sécurité des échanges	6
4.2. Inclure <i>hostedfields.js</i>	6
4.3. Environnements	6
4.3.1. Environnement de test dit « sandbox »	6
4.3.2. Environnement d'exploitation dit « live »	7
5. Références	8
MoneticoPaiement.HostedFields()	8
initialize(pointOfSale, token)	9
createField(fieldType, configuration)	10
field.generate(domElementId)	27
addEventListener(eventType, handler(event))	28
removeField(fieldType)	37
removeFields()	38
6. Informations et remarques pour l'intégration	39

1. Introduction

Le SDK en JavaScript « `hostedfields.js` » de Monetico online permet de générer au sein d'une page web HTML des éléments graphiques indépendants et sécurisés pour la saisie d'information de carte de paiement.

A chaque élément graphique généré correspondra un élément de la carte de paiement à saisir :

- le numéro de la carte de paiement ou *Personal Account Number (PAN)*
- la date d'expiration de la carte de paiement
- le cryptogramme visuel de la carte de paiement
- le nom du porteur de la carte de paiement

Il existe également des champs spécifiques aux parcours de paiement avec enregistrement d'une carte ou réutilisation d'une carte enregistrée, et correspondant aux fonctionnalités suivantes :

- le recueil du consentement pour l'enregistrement d'une carte de paiement
- la restitution des informations de carte au format PCI d'une carte enregistrée lors d'un parcours précédent, ainsi que la suppression de l'enregistrement de la carte
- la saisie d'une nouvelle carte lors d'un parcours avec réutilisation d'une carte enregistrée

Les informations saisies par le porteur lors d'un parcours de paiement intégrant les *hosted fields* seront intégralement gérées et stockées par Monetico online. Il suffira simplement de fournir au SDK un identifiant dit « jeton » de carte qui aura été généré au préalable via l'API de génération du jeton « `PaymentMeanTokenGenerator` » de Monetico online.

Une fois la saisie complète des différents champs, le jeton de carte sera utilisable pour effectuer une opération de paiement ainsi que des opérations de vérification des données de carte via l'API « `PaymentService` » de Monetico online.

Enfin, afin que les éléments graphiques générés par le SDK correspondent au mieux au style de la page web dans laquelle ils s'incrusteront, il sera possible de renseigner lors de l'intégration un certain nombre de paramètres qui définiront l'aspect visuel des différents champs.

Il sera notamment possible de définir plusieurs visuels pour un même champ en fonction de l'état de la saisie de celui-ci. Ces visuels seront alors affichés dynamiquement par le SDK lorsque le porteur renseignera les informations de carte de paiement (ex : on pourra définir une couleur de bordure différente si le champ est vide, rempli de manière correcte ou incorrecte).

2. Références

[1] [Documentation technique « Monetico online – Demande de jeton de saisie temporaire de moyen de paiement »](#)

[2] [Documentation technique « Monetico online – Hosted Fields – Intégration »](#)

3. Prérequis

3.1. Configuration du TPEv (Terminal de Paiement Electronique virtuel)

L'intégration du système *Hosted Fields* de Monetico online nécessite d'avoir souscrit au préalable à un pack « expert » Monetico.

Une fois la souscription finalisée, l'intégrateur doit être en possession des informations techniques suivantes :

- le numéro du TPEv
- la clé de sécurité du TPEv
- le nom du « code site » ou « code société » associé à la **configuration pour les appels aux API** de Monetico online (par opposition à la configuration dédiée à l'utilisation de la page de paiement déportée de Monetico)
- le code langue associé au « code site » décrit ci-dessus

Une demande doit également être faite auprès du support technique pour activer l'option « **tokenisation** » sur le TPEv et le « code site » utilisés.

3.2. Génération du jeton de saisie de carte temporaire

Les méthodes du SDK ne sont utilisables que si celui-ci est initialisé avec succès. Pour cela, un jeton de saisie de carte temporaire doit être généré au préalable via l'API « `PaymentMeanTokenGenerator` ».

Pour plus de détails sur l'intégration du service « `PaymentMeanTokenGenerator` » et du système *Hosted fields* dans sa globalité, se référer aux documents [\[1\]](#) et [\[2\]](#).

3.3. Utilisation du champ *hosted field* de type *cardPreview*

Le champ *hosted field* de type *cardPreview* n'est utilisable que dans le cadre d'un parcours de paiement avec réutilisation d'une carte déjà enregistrée lors d'un parcours précédent.

L'implémentation de tels parcours nécessite l'activation de l'option « **paiement express** » sur le contrat TEPv et doit faire l'objet d'une demande au support technique.

Pour plus de détails sur les différents parcours de paiement ou d'enregistrement de carte, se référer au document [\[2\]](#), section 5.

4. Intégration

4.1. Sécurité des échanges

L'utilisation du SDK « hostedfields.js » n'est possible que sur les sites web en HTTPS.

4.2. Inclure hostedfields.js

Afin de pouvoir profiter des fonctionnalités du SDK « hostedfields.js », la page web doit tout d'abord inclure la librairie JavaScript JQuery (cf. [ici](#) pour plus d'informations).

Le SDK « hostedfields.js » doit ensuite être inclus sur la page html où il sera utilisé de la manière suivante :

```
1 <script src="https://p.monetico-
2 services/hostedfields/hostedfields.js"></script>
```

4.3. Environnements

Le SDK « hostedfields.js » de Monetico online est disponible sur différents environnements :

- un environnement de test dit « sandbox »
- un environnement d'exploitation

L'environnement devra rester le même tout au long du parcours de paiement. Par exemple, un jeton de carte généré sur l'environnement d'exploitation ne pourra pas être utilisé avec le SDK disponible sur l'environnement de sandbox (et vice-versa). De même, un jeton correspondant à une carte saisie sur l'environnement de sandbox ne pourra pas être utilisé pour effectuer une opération de paiement sur l'environnement d'exploitation.

4.3.1. Environnement de test dit « sandbox »

Le rôle de notre serveur de test est de vous permettre de valider vos développements en vérifiant votre intégration à notre solution.

Le SDK en environnement de *sandbox* est disponible à l'URL suivante :

<https://p.monetico-services.com/test/hostedfields/hostedfields.js>

Pour tester l'intégration des *hosted fields* dans cet environnement, nous mettons à votre disposition les cartes de paiement suivantes :

Scénario	Réseau carte	3DSecure V1		3DSecure V2	
		Paiement accepté	Paiement refusé	Paiement accepté	Paiement refusé
La carte n'est pas enrôlée au 3DSecure	Visa	0000010000000002	0000010000000001	0000010000000021	0000010000000022
	Mastercard	0000030000000002	0000030000000001	0000030000000021	0000030000000022
L'authentification est effectuée avec succès sans challenge	Visa			0000010000000023	0000010000000024
	Mastercard			0000030000000023	0000030000000024
L'authentification est effectuée avec	Visa	0000010000000005	0000010000000006	0000010000000025	0000010000000026
	Mastercard	0000030000000005	0000030000000006	0000030000000025	0000030000000026

succès avec challenge					
L'authentification n'a pas pu être complétée (problème technique ou autre)	Visa		0000010000000008		0000010000000027
	Mastercard		0000030000000008		0000030000000027
Une tentative d'authentification a bien été effectuée. L'authentification n'a pas pu se faire mais une preuve a été générée (CAVV)	Visa	0000010000000007		0000010000000028	
	Mastercard	0000030000000007		0000030000000028	
L'authentification a échoué sans challenge	Visa				0000010000000029
	Mastercard				0000030000000029
L'authentification a échoué avec challenge	Visa		0000010000000004		0000010000000030
	Mastercard		0000030000000004		0000030000000030
L'authentification a été refusée par l'émetteur	Visa				0000010000000031
	Mastercard				0000030000000031

4.3.2. Environnement d'exploitation dit « live »

Après avoir validé vos développements, vous pourrez utiliser le SDK et l'API « PaymentService » en environnement d'exploitation, disponibles à l'URL suivante :

<https://p.monetico-services.com/hostedfields/hostedfields.js>

Attention : les cartes utilisées pour la saisie devront respecter l'algorithme de Luhn et les opérations de paiement effectuées suite à la saisie donneront lieu à de réels mouvements bancaires.

5. Références

MoneticoPaiement.HostedFields()

L'objet MoneticoPaiement.HostedFields est le point d'entrée du SDK des *hosted fields*. Une instance de cet objet doit être créée via la méthode MoneticoPaiement.HostedFields().

Exemple :

```
1 var hostedFields = MoneticoPaiement.HostedFields();
```

initialize(pointOfSale, token)

Cette méthode permet d'initialiser le SDK des *hosted fields*.

Renvoie `true` si le SDK est initialisé avec succès, `false` sinon.

Paramètres :

Champ	pointOfSale
Présence	Obligatoire
Description	Numéro de votre TPE virtuel
Format	7 caractères alphanumériques
Valeur(s) possible(s)	[A-Za-z0-9]{7}

Champ	token
Présence	Obligatoire
Description	Identifiant correspondant au jeton de saisie temporaire de carte et permettant d'authentifier l'utilisateur du SDK. Le <i>token</i> s'obtient en effectuant au préalable une demande de génération de jeton de carte à l'API de génération du jeton de Monetico online. Les autres méthodes du SDK ne seront accessibles que si cette méthode est appelée avec un <i>token</i> valide.
Format	Chaîne de caractères : UUID (RFC 4122)
Valeur(s) possible(s)	

Exemple :

```
1 hostedFields.initialize('9000001', '90d3fe3b-e82e-4633-a67c-70ec967c0852');
```

createField(fieldType, configuration)

Cette méthode permet de créer une instance d'un champ *hosted field*. Chaque champ ne peut être instancié qu'une seule fois par session, une session correspondant à un *token* donné.

Paramètres :

Champ	fieldType
Présence	Obligatoire
Description	Type du champ à créer
Format	Chaîne de caractères
Valeur(s) possible(s)	cardNumber, cardExpDate, cardCvx, cardHolderName, cardPreview, cardSwitch

Champ	configuration
Présence	Optionnelle
Description	Options de configuration du champ
Format	Objet
Valeur(s) possible(s)	

Liste des types de champ définissables :

Type	Parcours et présence	Description
cardNumber	Parcours de paiement sans enregistrement de carte : obligatoire Parcours avec enregistrement de carte : obligatoire Parcours de paiement avec réutilisation d'une carte enregistrée : interdit	Champ contenant le numéro de la carte de paiement
cardExpDate	Parcours de paiement sans enregistrement de carte : obligatoire Parcours avec enregistrement de carte : obligatoire Parcours de paiement avec réutilisation d'une carte enregistrée : interdit	Champ contenant la date d'expiration de la carte de paiement
cardCvx	Parcours de paiement sans enregistrement de carte : obligatoire Parcours avec enregistrement de carte : obligatoire	Champ contenant le cryptogramme visuel de la carte de paiement

	Parcours de paiement avec réutilisation d'une carte enregistrée : conditionnelle	
cardHolderName	Parcours de paiement sans enregistrement de carte : obligatoire Parcours avec enregistrement de carte : obligatoire Parcours de paiement avec réutilisation d'une carte enregistrée : interdit	Champ contenant le nom du porteur de la carte de paiement
cardPreview	Parcours de paiement sans enregistrement de carte : interdit Parcours avec enregistrement de carte : interdit Parcours de paiement avec réutilisation d'une carte enregistrée : facultatif	Champ restituant les informations d'une carte de paiement enregistrée (PAN PCI, date d'expiration et nom du porteur)
cardSwitch	Parcours de paiement sans enregistrement de carte : interdit Parcours avec enregistrement de carte : interdit Parcours de paiement avec réutilisation d'une carte enregistrée : facultatif	Champ permettant de saisir une autre carte que la carte de paiement enregistrée et proposée pour le paiement
cardRegistrationConsent	Parcours de paiement sans enregistrement de carte : interdit Parcours avec enregistrement de carte : obligatoire Parcours de paiement avec réutilisation d'une carte enregistrée : interdit	Champ permettant de recueillir le consentement du porteur pour l'enregistrement de sa carte

Note : la présence de chaque champ est spécifiée en retour du service de génération du jeton de saisie temporaire de carte « PaymentMeanTokenGenerator » lorsque le contexte de la génération du jeton est précisé en entrée du service. Se référer au document [\[1\]](#) pour plus de détails.

Note2 : pour plus de détails sur les différents parcours de paiement ou d'enregistrement de carte, se référer au document [\[2\]](#), section 5.

Objet « configuration » :

Propriété	Type	Présence
placeholder	Liste d'éléments de type Chaîne de caractères	Optionnelle
style	Objet	Optionnelle
texts	Objet	Optionnelle

displayType	Liste d'éléments de type Chaîne de caractères	Optionnelle
--------------------	--	-------------

Propriété	placeholder
Présence	Optionnelle, n'est pas utilisable pour un champ de type <i>cardPreview</i>
Description	Le <i>placeholder</i> correspond au libellé à afficher à l'intérieur du champ si celui-ci est vide. Pour un <i>hosted field</i> de type <i>cardExpDate</i>, il est conseillé de fournir une liste contenant 2 libellés. Le 1^{er} élément de la liste correspond au <i>placeholder</i> pour le champ du mois et le 2^{ème} élément à celui de l'année de la date d'expiration. Si la liste ne contient qu'un seul élément, le même <i>placeholder</i> est affiché pour le champs de saisie du mois et pour celui de l'année. Si la liste contient plus de 2 éléments, seuls les 2 premiers sont pris en compte. Pour les autres types de champ, seul le 1^{er} élément de la liste est pris en compte.
Format Valeur(s) possible(s)	Liste d'éléments de type chaîne de caractères, de longueur maximum de 255 caractères

Note : la saisie du mois et de l'année de la date d'expiration d'une carte s'effectue sur 2 chiffres, il est donc recommandé d'utiliser comme placeholders ['MM', 'AA'] pour ce champ.

Propriété	texts
Présence	Optionnelle, n'est utilisable que pour un champ de type <i>cardPreview</i> et <i>cardSwitch</i>
Description	Libellés des textes à afficher dans le champ <i>cardPreview</i> et le champ <i>cardSwitch</i>
Format Valeur(s) possible(s)	Objet Détails ci-dessous

Propriété	displayType
Présence	Optionnelle, n'est utilisable que pour un champ de type <i>cardRegistrationConsent</i>
Description	Apparence du champ de type <i>cardRegistrationConsent</i> Peut avoir la forme d'une case à cocher (apparence par défaut) ou d'un interrupteur
Format Valeur(s) possible(s)	Liste d'éléments de type chaîne de caractères Valeurs possibles : <i>checkBox</i> , <i>toggleSwitch</i> Détails ci-dessous

Propriété	style
Présence	Optionnelle

Description	Style CSS à appliquer au champ. Une variante de style peut être définie en fonction de l'état dans lequel se trouve le champ. Pour chaque variante de style, une liste de propriétés CSS peuvent être définies.
Format	Objet
Valeur(s) possible(s)	Détails ci-dessous

Objet « texts » :

Propriété	Description	Présence
cardSwitchButtonText	Chaîne de caractères Libellé indiquant la possibilité d'utiliser une autre carte de paiement que celle déjà enregistrée et proposée par le commerçant au porteur pour effectuer son paiement 255 caractères max	Obligatoire pour le champ <i>cardSwitch</i>
cardRemovalTitleText	Chaîne de caractères Libellé présent sur l'encart de confirmation de suppression de l'enregistrement d'une carte Titre du message de confirmation de suppression de l'enregistrement de la carte 255 caractères max	Obligatoire pour le champ <i>cardPreview</i>
cardRemovalMessageText	Chaîne de caractères Libellé présent sur l'encart de confirmation de suppression de l'enregistrement d'une carte Contenu du message de confirmation de suppression de l'enregistrement de la carte 255 caractères max	Obligatoire pour le champ <i>cardPreview</i>
cardRemovalConfirmButtonText	Chaîne de caractères Libellé présent sur l'encart de confirmation de suppression de l'enregistrement d'une carte Libellé du bouton de confirmation de suppression de l'enregistrement de la carte 255 caractères max	Obligatoire pour le champ <i>cardPreview</i>
cardRemovalCancelButtonText	Chaîne de caractères	Obligatoire pour le champ <i>cardPreview</i>

	Libellé du bouton d'annulation de suppression de l'enregistrement de la carte 255 caractères max	
--	---	--

Liste des valeurs possibles pour « displayType » :

displayType	Description
checkBox	Le champ « cardRegistrationConsent » aura l'apparence d'une case à cocher
toggleSwitch	Le champ « cardRegistrationConsent » aura l'apparence d'un interrupteur

Objet « style » :

Propriété	Type	Présence
baseClass	Objet	Optionnelle
emptyClass	Objet	Optionnelle
focusedClass	Objet	Optionnelle
completedClass	Objet	Optionnelle
invalidClass	Objet	Optionnelle
activeClass	Objet	Optionnelle
hoveredClass	Objet	Optionnelle

Propriété	baseClass
Présence	Optionnelle
Description	Style CSS par défaut à appliquer au champ
Format	Objet
Valeur(s) possible(s)	

Propriété	emptyClass
Présence	Optionnelle N'est pas définissable pour un champ de type <i>cardPreview</i> , <i>cardSwitch</i> ou <i>cardRegistrationConsent</i>
Description	Style CSS à appliquer au champ lorsque celui-ci est vide
Format	Objet
Valeur(s) possible(s)	

Propriété	focusedClass
Présence	Optionnelle N'est pas définissable pour le champs de type <i>cardSwitch</i> ou <i>cardRegistrationConsent</i>

Description	Style CSS à appliquer au champ lorsque celui-ci est en cours de saisie
Format	Objet
Valeur(s) possible(s)	

Propriété	completedClass
Présence	Optionnelle N'est pas définissable pour le champs de type <i>cardSwitch</i> ou <i>cardRegistrationConsent</i>
Description	Style CSS à appliquer au champ lorsque celui-ci contient des données valides pouvant correspondre à la fin de la saisie
Format	Objet
Valeur(s) possible(s)	

Propriété	invalidClass
Présence	Optionnelle N'est pas définissable pour le champs de type <i>cardSwitch</i> ou <i>cardRegistrationConsent</i>
Description	Style CSS à appliquer au champ lorsque celui-ci contient des données invalides
Format	Objet
Valeur(s) possible(s)	

Propriété	activeClass
Présence	Optionnelle Définissable uniquement pour un champ de type <i>cardPreview</i> , <i>cardSwitch</i> ou <i>cardRegistrationConsent</i>
Description	Style CSS à appliquer au champ ou à certains éléments du champ lorsqu'on clique dessus
Format	Objet
Valeur(s) possible(s)	

Propriété	hoveredClass
Présence	Optionnelle Définissable uniquement pour un champs de type <i>cardSwitch</i> ou <i>cardRegistrationConsent</i>
Description	Style CSS à appliquer au champ ou à certains éléments du champ lorsqu'on passe la souris dessus
Format	Objet
Valeur(s) possible(s)	

Liste des champs définissables pour les styles des champs de type *cardNumber*, *cardExpDate*, *cardCvx* et *cardHolderName* :

Propriété	Format Description	Propriété CSS correspondante
backgroundColorProperty	Chaîne de caractères Couleur appliquée au fond du champ	background-color
backgroundProperty	Chaîne de caractères Propriétés liées à la gestion de l'arrière-plan appliquées au champ	background
colorProperty	Chaîne de caractères Couleur appliquée au texte saisi dans le champ ou au <i>placeholder</i> si le champ est vide	color
iconColorProperty	Chaîne de caractères Couleur appliquée à l'icône de carte de paiement, uniquement dans le cas du champ de type <i>cardNumber</i> . Cet icône indique l'emplacement du menu pour le choix du réseau de la carte	color
arrowColorProperty	Chaîne de caractères Couleur appliquée à la flèche à côté de l'icône de carte de paiement, uniquement dans le cas du champ de type <i>cardNumber</i> . Cette flèche indique la possibilité de dérouler le menu pour le choix du réseau de la carte	color
borderProperty	Chaîne de caractères Bordure du champ. Cette propriété peut être utilisée pour définir l'épaisseur, le style et la couleur de la bordure	border
fontFamilyProperty	Chaîne de caractères Police appliquée au texte saisi dans le champ ou au <i>placeholder</i> si le champ est vide	font-family
fontSizeProperty	Chaîne de caractères Taille appliquée au texte saisi dans le champ ou au <i>placeholder</i> si le champ est vide	font-size
fontStyleProperty	Chaîne de caractères Style de police italique appliqué au texte saisi dans le champ ou au <i>placeholder</i> si le champ est vide, sélectionné parmi les polices listées dans <i>fontFamilyProperty</i> .	font-style
fontWeightProperty	Chaîne de caractères	font-weight

	Graisse appliquée au texte saisi dans le champ ou au <i>placeholder</i> si le champ est vide, en fonction des polices listées dans <i>fontFamilyProperty</i> .	
textAlignProperty	Chaîne de caractères Alignement du texte saisi dans le champ ou du <i>placeholder</i> si le champ est vide	text-align
paddingProperty	Chaîne de caractères Ecart appliqué entre la bordure et le contenu du champ	padding
textDecorationProperty	Chaîne de caractères Décoration sous forme de ligne (ex : soulignement), appliquée au texte saisi dans le champ ou au <i>placeholder</i> si le champ est vide. Cette propriété peut être utilisée pour définir le style et la couleur de la décoration	text-decoration
textShadowProperty	Chaîne de caractères Ombre appliquée au texte saisi dans le champ ou au <i>placeholder</i> si le champ est vide	text-shadow

Liste des champs définissables pour les styles du champ de type *cardPreview* :

Propriété	Format Description	Propriété CSS correspondante
iconColorProperty	Chaîne de caractères Couleur appliquée à l'icône indiquant l'emplacement de la fonctionnalité de suppression de l'enregistrement de la carte	color
backgroundColorProperty	Chaîne de caractères Couleur appliquée au fond du champ	background-color
backgroundProperty	Chaîne de caractères Propriétés liées à la gestion de l'arrière-plan appliquées au champ	background
colorProperty	Chaîne de caractères Couleur appliquée au texte affiché à l'intérieur du champ	color
borderProperty	Chaîne de caractères Bordure du champ Cette propriété peut être utilisée pour définir l'épaisseur, le style et la couleur de la bordure	border
fontFamilyProperty	Chaîne de caractères	font-family

	Police appliquée au texte affiché à l'intérieur du champ	
fontSizeProperty	Chaîne de caractères Taille appliquée au texte affiché à l'intérieur du champ	font-size
fontStyleProperty	Chaîne de caractères Style de police italique au texte affiché à l'intérieur du champ, sélectionné parmi les polices listées dans <i>fontFamilyProperty</i> .	font-style
fontWeightProperty	Chaîne de caractères Graisie appliquée au texte affiché à l'intérieur du champ, en fonction des polices listées dans <i>fontFamilyProperty</i> .	font-weight
textAlignProperty	Chaîne de caractères Alignement du texte affiché à l'intérieur du champ Seules les valeurs <i>left</i> , <i>right</i> , <i>center</i> et <i>justify</i> sont acceptées	text-align
paddingProperty	Chaîne de caractères Ecart appliqué entre la bordure et le contenu du champ	padding
textDecorationProperty	Chaîne de caractères Décoration sous forme de ligne (ex : soulignement), appliquée au texte affiché à l'intérieur du champ Cette propriété peut être utilisée pour définir le style et la couleur de la décoration	text-decoration
textShadowProperty	Chaîne de caractères Ombre appliquée au texte affiché à l'intérieur du champ	text-shadow
borderRadiusProperty	Chaîne de caractères Arrondi appliqué aux coins du bord du champ	border-radius
cardNumberProperties	Objet Propriétés CSS à appliquer à la zone d'affichage du numéro au format PCIDSS de la carte à l'intérieur du champ de type <i>cardPreview</i> Définissable uniquement pour la configuration d'un champ de type <i>cardPreview</i>	
cardSchemeProperties	Objet Propriétés CSS à appliquer à la zone d'affichage du menu pour le choix du réseau de la carte	

	Définissable uniquement pour la configuration d'un champ de type <i>cardPreview</i>	
cardExpDateProperties	Objet Propriétés CSS à appliquer à la zone d'affichage de la date d'expiration de la carte à l'intérieur du champ de type <i>cardPreview</i> Définissable uniquement pour la configuration d'un champ de type <i>cardPreview</i>	
cardHolderNameProperties	Objet Propriétés CSS à appliquer à la zone d'affichage du nom du porteur de la carte à l'intérieur du champ de type <i>cardPreview</i> Définissable uniquement pour la configuration d'un champ de type <i>cardPreview</i>	
cardRemovalProperties	Objet Propriétés CSS à appliquer à l'encart de confirmation de suppression de l'enregistrement d'une carte à l'intérieur du champ de type <i>cardPreview</i> Définissable uniquement pour la configuration d'un champ de type <i>cardPreview</i>	

Liste des champs définissables pour l'objet *cardSchemeProperties* :

Propriété	Format Description	Propriété CSS correspondante
iconColorProperty	Chaîne de caractères Couleur appliquée à l'icône indiquant l'emplacement du menu pour le choix du réseau de la carte	color
backgroundColorProperty	Chaîne de caractères Couleur appliquée au fond de la zone d'affichage	background-color
backgroundProperty	Chaîne de caractères Propriétés liées à la gestion de l'arrière-plan appliquées au champ	background
borderProperty	Chaîne de caractères Bordure de la zone d'affichage Cette propriété peut être utilisée pour définir l'épaisseur, le style et la couleur de la bordure	border

Liste des champs définissables pour l'objet *cardRemovalProperties* :

Propriété	Format Description	Propriété CSS correspondante
iconColorProperty	Chaîne de caractères Couleur appliquée à l'icône illustrant la fonctionnalité de suppression de l'enregistrement de la carte	color
titleProperties	Objet Propriétés CSS à appliquer au titre du message de l'encart de confirmation de suppression de l'enregistrement de la carte	color
messageProperties	Objet Propriétés CSS à appliquer au contenu du message de l'encart de confirmation de suppression de l'enregistrement de la carte	padding
confirmButtonProperties	Objet Propriétés CSS à appliquer au bouton de confirmation de suppression de l'enregistrement de la carte	height
cancelButtonProperties	Objet Propriétés CSS à appliquer au bouton d'annulation de suppression de l'enregistrement de la carte	width

Liste des champs définissables pour les objets suivants :

- *cardNumberProperties*
- *cardExpDateProperties*
- *cardHolderNameProperties*
- *titleProperties*
- *messageProperties*
- *confirmButtonProperties*
- *cancelButtonProperties* :

Propriété	Format Description	Propriété CSS correspondante
backgroundColorProperty	Chaîne de caractères Couleur appliquée au fond de la zone d'affichage	background-color
backgroundProperty	Chaîne de caractères Propriétés liées à la gestion de l'arrière-plan appliquées au champ	background
colorProperty	Chaîne de caractères Couleur appliquée au texte affiché dans la zone	color
borderProperty	Chaîne de caractères	border

	Bordure de la zone d'affichage Cette propriété peut être utilisée pour définir l'épaisseur, le style et la couleur de la bordure	
fontFamilyProperty	Chaîne de caractères Police appliquée au texte indiquant le numéro au format PCIDSS de la carte	font-family
fontSizeProperty	Chaîne de caractères Taille appliquée au texte affiché dans la zone	font-size
fontStyleProperty	Chaîne de caractères Style de police italique appliqué au texte affiché dans la zone, sélectionné parmi les polices listées dans <i>fontFamilyProperty</i>	font-style
fontWeightProperty	Chaîne de caractères Graisie appliquée au texte affiché dans la zone, en fonction des polices listées dans <i>fontFamilyProperty</i>	font-weight
textAlignProperty	Chaîne de caractères Alignement du texte affiché dans la zone	text-align
paddingProperty	Chaîne de caractères Ecart appliqué entre la bordure et le contenu de la zone d'affichage	padding
textDecorationProperty	Chaîne de caractères Décoration sous forme de ligne (ex : soulignement), appliquée au texte affiché dans la zone Cette propriété peut être utilisée pour définir le style et la couleur de la décoration	text-decoration
textShadowProperty	Chaîne de caractères Ombre appliquée au texte affiché dans la zone	text-shadow

Liste des champs définissables pour les styles du champ de type *cardRegistrationConsent*:

Propriété	Format Description	Propriété CSS correspondante
backgroundColorProperty	Chaîne de caractères Couleur appliquée au fond de la case à cocher ou de l'interrupteur	background-color
backgroundProperty	Chaîne de caractères Propriétés définissant le fond de la case à cocher ou de l'interrupteur	background
iconColorProperty	Chaîne de caractères	color

	Couleur appliquée à l'icône ✓ de la case à cocher lorsqu'elle est cochée ou au bouton de l'interrupteur	
paddingProperty	Chaîne de caractères Ecart appliqué entre le bord du champ et la case à cocher ou l'interrupteur	padding
borderProperty	Chaîne de caractères Bordure de la case à cocher ou de l'interrupteur Cette propriété peut être utilisée pour définir l'épaisseur, le style et la couleur de la bordure	border

Note : les syntaxes contenant le caractère « / » **ne sont pas autorisées** pour la valorisation des propriétés CSS décrites ci-dessus.

Exemple 1 :

```

1  var style = {
2      baseClass: {
3          backgroundColorProperty: '#329ba8',
4          colorProperty: 'white',
5          borderProperty: '2px solid #d2d2f7',
6          paddingProperty: '10px',
7          fontFamilyProperty: 'Roboto, Open Sans, Segoe UI, sans-
8  serif',
9          textAlignProperty: 'end',
10         fontSizeProperty: '0.8em'
11     },
12     focusedClass: {
13         borderProperty: '2px solid #00c5cd',
14         fontWeightProperty: 'bolder'
15     },
16     completedClass: {
17         borderProperty: '2px solid #ae0000',
18         backgroundColorProperty: '#b6fcd5',
19         colorProperty: '#ae0000',
20         textShadowProperty: '1px 1px 2px #d546a6'
21     },
22     invalidClass: {
23         borderProperty: '2px solid #43ab7f',
24         backgroundColorProperty: '#ecdddd',
25         colorProperty: '#595843'
26     }
27 };
28
29 var style2 = {
30     baseClass: {
31         iconColorProperty: '#d7d7d7',
32         backgroundColorProperty: 'white'

```

```
33     },
34     activeClass: {
35         iconColorProperty: 'white',
36         backgroundColorProperty: '#1b8dff'
37     },
38 };
39
40 if(hostedFields.initialize('1234567', '90d3fe3b-e82e-4633-a67c-
41 70ec967c0852')) {
42     var hostedField1 = hostedFields.createField('cardNumber', {
43 style: style, placeholder: ['PAN'] });
44
45     var hostedField2 = hostedFields.createField('cardExpDate', {
46 style: style, placeholder: ['MM', 'AA'] });
47
48     var hostedField3 = hostedFields.createField('cardCvx', {
49 style: style, placeholder: ['CVX'] });
50
51     var hostedField4 = hostedFields.createField('cardHolderName',
52 { style: style, placeholder: ['Nom sur la carte'] });
53
54     var hostedField5 =
55 hostedFields.createField('cardRegistrationConsent', { style:
56 style2, displayType: 'toggleSwitch' });
57 }
```

Exemple 2 :

```
1  var cvxStyle = {
2      baseClass: {
3          backgroundColorProperty: '#329ba8',
4          colorProperty: 'white',
5          borderProperty: '2px solid #d2d2f7',
6          paddingProperty: '10px',
7          fontFamilyProperty: 'Roboto, Open Sans, Segoe UI, sans-
8 serif',
9          textAlignProperty: 'end',
10         fontSizeProperty: '0.8em'
11     },
12     focusedClass: {
13         borderProperty: '2px solid #00c5cd',
14         fontWeightProperty: 'bolder'
15     },
16     completedClass: {
17         borderProperty: '2px solid #ae0000',
18         backgroundColorProperty: '#b6fcd5',
19         colorProperty: '#ae0000',
20         textShadowProperty: '1px 1px 2px #d546a6'
21     },
22     invalidClass: {
```

```

23         borderProperty: '2px solid #43ab7f',
24         backgroundColorProperty: '#ecddd',
25         colorProperty: '#595843'
26     }
27 };
28
29 var previewStyle = {
30     baseClass: {
31         borderProperty: 'none',
32         backgroundProperty: 'linear-gradient(146deg,
33 rgba(91,155,213,1) 0%, rgba(9,9,121,1) 50%, rgba(112,48,160,1)
34 100%)',
35         paddingProperty: '10px',
36         fontFamilyProperty: 'Roboto, Open Sans, Segoe UI, sans-
37 serif',
38         textAlignProperty: 'end',
39         fontSizeProperty: '16px',
40         fontWeightProperty: '500',
41         borderRadiusProperty: '10px',
42         cardSchemeProperties: {
43             colorProperty: '#a6cdf',
44             arrowColorProperty: '#42f5e6'
45         },
46         cardNumberProperties: {
47             colorProperty: 'white'
48         },
49         cardExpDateProperties: {
50             colorProperty: '#5b9bd5',
51             textAlignProperty: 'right'
52         },
53         cardHolderNameProperties: {
54             colorProperty: '#ff9999'
55         },
56     },
57     cardRemovalProperties: {
58         iconColorProperty: '#950300',
59         colorProperty: '#484848',
60         backgroundColorProperty: 'white',
61         borderProperty: 'yellow',
62         titleProperties: {
63             textAlignProperty: 'center',
64             fontWeightProperty: 'bold'
65         },
66         messageProperties: {
67             textAlignProperty: 'center',
68             fontStyleProperty: 'italic'
69         },
70         confirmButtonProperties: {
71             backgroundColorProperty: '#950245',
72             colorProperty: '#ffdee3',
73             borderRadiusProperty: '8px'
74         },

```

```

75         cancelButtonProperties: {
76             backgroundColorProperty: 'white',
77             colorProperty: 'grey',
78             borderRadiusProperty: '8px'
79         }
80     },
81 },
82
83 },
84 hoveredClass: {
85     iconColorProperty: '#ff6633',
86     cardRemovalProperties: {
87         confirmButtonProperties: {
88             backgroundColorProperty: '#d0005f'
89         },
90         cancelButtonProperties: {
91             colorProperty: '#b9b9b9'
92         }
93     },
94 },
95 activeClass: {
96     cardRemovalProperties: {
97         confirmButtonProperties: {
98             borderProperty: '2px solid white'
99         },
100         cancelButtonProperties: {
101             colorProperty: 'blue'
102         }
103     },
104 },
105 };
106
107 var buttonStyle = {
108     baseClass: {
109         borderRadiusProperty: '10px',
110         borderProperty: '2px solid #5FB696',
111         colorProperty: 'white',
112         backgroundColorProperty: '#5FB696',
113         paddingProperty: '10px',
114         textAlignProperty: 'right'
115     },
116     hoveredClass: {
117         backgroundColorProperty: '#75A593',
118         borderProperty: '2px solid #75A593'
119     },
120     activeClass: {
121         backgroundColorProperty: '#B5F0DA',
122         borderProperty: '2px solid white'
123     }
124 };
125
126 };
127
128 var switchText = {

```

```
129     cardSwitchButtonText: 'Utiliser une autre carte'
130 };
131 var removalTexts = {
132     cardRemovalTitleText: 'Etes-vous sûr(e) ?',
133     cardRemovalMessageText: 'Votre carte devra être ressaisie
134 lors de paiements ultérieurs...',
135     cardRemovalConfirmButtonText: 'Supprimer',
136     cardRemovalCancelButtonText: 'Conserver'
137 };
138 };
139
140 if(hostedFields.initialize('1234567', '90d3fe3b-e82e-4633-a67c-
141 70ec967c0852')) {
142     var hostedField1 = hostedFields.createField('cardCvx', {
143     style: cvxStyle, placeholder: ['CVX'] });
144     var hostedField2 = hostedFields.createField('cardPreview', {
145     style: previewStyle, texts: removalTexts });
146     var hostedField2 = hostedFields.createField('cardSwitch', {
147     style: buttonStyle, texts: switchText });
148 }
```

field.generate(domElementId)

Cette méthode permet de générer l'élément graphique et de l'associer à l'objet *hosted field* sur lequel elle est appelée.

Paramètres :

Champ	domElementId
Présence	Obligatoire
Description	Id de l'élément HTML <div> qui doit contenir le champ généré
Format	Chaîne de caractères
Valeur(s) possible(s)	

Exemple :

```
1 <div id="card-number"></div>
2
3 <script>
4   hostedfield1.generate('card-number');
5 </script>
```

addEventListener(eventType, handler(event))

Cette méthode permet de gérer les événements liés aux *hosted fields* instanciés sur la page.

Paramètres :

Champ	eventType
Présence	Obligatoire
Description	Type de l'évènement à gérer
Format	Chaîne de caractères
Valeur(s) possible(s)	change, switch, delete

Champ	handler(event)
Présence	Obligatoire
Description	Fonction de callback qui sera appelée lorsqu'un évènement de type 'eventType' est lancé
Format	Fonction
Valeur(s) possible(s)	

Liste des types d'évènement :

Type	Parcours et présence	Description
change	Parcours de paiement sans enregistrement de carte : obligatoire Parcours avec enregistrement de carte : obligatoire Parcours de paiement avec réutilisation d'une carte enregistrée : obligatoire	Indique que le statut de la saisie a évolué et/ou qu'une erreur de saisie est survenue
switch	Parcours de paiement sans enregistrement de carte : interdit Parcours avec enregistrement de carte : interdit Parcours de paiement avec réutilisation d'une carte enregistrée : conditionnelle (si le champ de type <i>cardSwitch</i> est instancié)	Indique que le bouton de changement de carte de paiement en cours de parcours (correspond au champ de type <i>cardSwitch</i>) a été utilisé
delete	Parcours de paiement sans enregistrement de carte : interdit Parcours avec enregistrement de carte : interdit Parcours de paiement avec réutilisation d'une carte enregistrée : conditionnelle (si le champ de type <i>cardPreview</i> est instancié)	Indique que la fonctionnalité de suppression de d'enregistrement de carte (incluse dans le champ de type <i>cardPreview</i>) a été utilisée
consent	Parcours de paiement sans enregistrement de carte : interdit Parcours avec enregistrement de carte : obligatoire	Indique que le consentement a été soit donné par le porteur de la carte pour l'enregistrement de celle-ci, soit retiré

	Parcours de paiement avec réutilisation d'une carte enregistrée : interdit	
--	---	--

Objet « event.detail » :

Propriété	Type	Présence
complete	Booléen	Conditionnelle
error	Objet	Conditionnelle
success	Booléen	Conditionnelle
granted	Booléen	Conditionnelle
timestamp	Date	Conditionnelle

Propriété	complete
Présence	Conditionnelle Présent seulement si l'event est de type « change »
Description	Indique si l'ensemble des champs de la page ont été saisis et complétés avec succès
Format Valeur(s) possible(s)	Booléen

Propriété	error
Présence	Conditionnelle Présent seulement si l'event est de type « change »
Description	Erreur remontée lors de la saisie d'un des champs
Format Valeur(s) possible(s)	Objet undefined si aucune erreur de saisie n'a été remontée

Propriété	success
Présence	Conditionnelle Présent seulement si l'event est de type « delete »
Description	Indique si la suppression de l'enregistrement de la carte a été effectuée correctement
Format Valeur(s) possible(s)	Booléen

Propriété	granted
Présence	Conditionnelle

	Présent seulement si l'event est de type « consent »
Description	Indique si le consentement pour l'enregistrement de la carte de paiement a été donné ou retiré par le porteur de celle-ci
Format Valeur(s) possible(s)	Booléen

Propriété	timestamp
Présence	Conditionnelle Présent seulement si l'event est de type « consent » et que le consentement pour l'enregistrement de la carte de paiement a été donné par le porteur de celle-ci (paramètre « granted » valorisé à « true »)
Description	Indique si le moment où le consentement pour l'enregistrement de la carte de paiement a été donné par le porteur de celle-ci
Format Valeur(s) possible(s)	Date AAAA-MM-DD hh:mm:ss

Objet « error » :

Propriété	Type	Présence
cardNumber	Objet de type <i>errorDetails</i>	Optionnelle
cardExpDate	Objet de type <i>errorDetails</i>	Optionnelle
cardCvx	Objet de type <i>errorDetails</i>	Optionnelle
cardHolderName	Objet de type <i>errorDetails</i>	Optionnelle
cardPreview	Objet de type <i>cardPreviewErrorDetails</i>	Optionnelle

Propriété	cardNumber
Présence	Optionnelle
Description	Indique qu'une erreur a été levée lors de la saisie du champ de type <i>cardNumber</i>
Format Valeur(s) possible(s)	Objet de type <i>errorDetails</i>

Propriété	cardExpDate
Présence	Optionnelle
Description	Indique qu'une erreur a été levée lors de la saisie du champ de type <i>cardExpDate</i>

Format	Objet de type <i>errorDetails</i>
Valeur(s) possible(s)	

Propriété	cardCvx
Présence	Optionnelle
Description	Indique qu'une erreur a été levée lors de la saisie du champ de type <i>cardCvx</i>
Format	Objet de type <i>errorDetails</i>
Valeur(s) possible(s)	

Propriété	cardHolderName
Présence	Optionnelle
Description	Indique qu'une erreur a été levée lors de la saisie du champ de type <i>cardHolderName</i>
Format	Objet de type <i>errorDetails</i>
Valeur(s) possible(s)	

Propriété	cardPreview
Présence	Optionnelle
Description	Indique qu'une erreur a été levée lors de la génération du champ de type <i>cardPreview</i>
Format	Objet de type <i>errorDetails</i>
Valeur(s) possible(s)	

Objet « *errorDetails* » :

Propriété	Type	Présence
code	Entier	Obligatoire
description	Chaîne	Obligatoire
message	Chaîne	Obligatoire

Propriété	code
Présence	Obligatoire
Description	Code technique correspondant à l'erreur remontée
Format	Entier
Valeur(s) possible(s)	

Propriété	description
Présence	Obligatoire

Description	Description de l'erreur remontée
Format	Chaîne de caractères
Valeur(s) possible(s)	

Propriété	message
Présence	Obligatoire
Description	Message à afficher sur la page correspondant à l'erreur remontée
Format	Chaîne de caractères
Valeur(s) possible(s)	

Objet « cardPreviewErrorDetails » :

Propriété	Type	Présence
code	Entier	Obligatoire
description	Chaîne	Obligatoire
message	Chaîne	Obligatoire
cardExpDate	Objet de type errorDetails	Optionnelle
cardNumber	Objet de type errorDetails	Optionnelle
cardHolderName	Objet de type errorDetails	Optionnelle

Propriété	code
Présence	Obligatoire
Description	Code technique correspondant à l'erreur remontée sur l'ensemble du champ de type <i>cardPreview</i>
Format	Entier
Valeur(s) possible(s)	

Propriété	description
Présence	Obligatoire
Description	Description de l'erreur remontée sur l'ensemble du champ de type <i>cardPreview</i>
Format	Chaîne de caractères
Valeur(s) possible(s)	

Propriété	message
Présence	Obligatoire
Description	Message à afficher sur la page correspondant à l'erreur remontée sur l'ensemble du champ de type <i>cardPreview</i>
Format	Chaîne de caractères

Valeur(s) possible(s)	
-----------------------	--

Propriété	cardNumber
Présence	Optionnelle
Description	Indique que l'erreur levée lors de la génération du champ de type <i>cardPreview</i> est liée à la valeur du numéro de la carte enregistrée et précise en quoi cette valeur est problématique
Format	Objet de type <i>errorDetails</i>
Valeur(s) possible(s)	

Propriété	cardExpDate
Présence	Optionnelle
Description	Indique que l'erreur levée lors de la génération du champ de type <i>cardPreview</i> est liée à la valeur de la date d'expiration de la carte enregistrée et précise en quoi cette valeur est problématique
Format	Objet de type <i>errorDetails</i>
Valeur(s) possible(s)	

Propriété	cardHolderName
Présence	Optionnelle
Description	Indique que l'erreur levée lors de la génération du champ de type <i>cardPreview</i> est liée à la valeur du nom du porteur de la carte enregistrée et précise en quoi cette valeur est problématique
Format	Objet de type <i>errorDetails</i>
Valeur(s) possible(s)	

Liste des erreurs possibles :

Code	Description	Message	Condition(s)
-1	technical_error	Votre demande de paiement ne peut aboutir	Erreur technique
1	invalid_request	Votre demande de paiement ne peut aboutir	Format des paramètres token/pointOfSale invalide ou token/pointOfSale inexistant
2	expired_request	Votre demande de paiement a expiré	Le jeton a expiré
101	cardnumber_too_long	Le numéro de carte doit être composé de 13 à 19 chiffres	Le numéro de carte saisi est plus long que 19 caractères
102	cardnumber_too_short	Le numéro de carte doit être composé de 13 à 19 chiffres	Le numéro de carte saisi est plus court que 13 caractères
103	cardnumber_invalid_format	Le numéro de carte ne peut contenir que des chiffres	Le numéro de carte saisi contient des lettres ou des caractères spéciaux

104	cardnumber_invalid_pan	Le numéro de carte est invalide	Le numéro de carte saisi ne respecte pas l'algorithme de Luhn
105	cardnumber_incorrect_pan	Le numéro de carte est erroné	Le numéro de carte saisi n'est pas accepté par Monetico online
106	cardnumber_not_accepted_pan	Ce type de carte n'est pas accepté par la société, veuillez saisir une nouvelle carte OU Ce type de carte n'est pas accepté par la société	Le numéro de carte saisi n'est pas accepté par le commerçant OU Le numéro de carte enregistré n'est plus accepté par le commerçant (cas du champ de type <i>cardPreview</i> uniquement)
201	cardexpdate_invalid_month	Le numéro du mois est invalide	Le mois de la date d'expiration saisie n'est pas compris entre 1 et 12 ou ne contient pas que des chiffres
202	cardexpdate_invalid_year	Le numéro de l'année est invalide	L'année de la date d'expiration saisie ne contient pas que des chiffres
203	cardexpdate_invalid_date	La date d'expiration de la carte doit être supérieure à la date courante OU La durée de validité de la carte doit être supérieure à N mois	La date d'expiration saisie est dans le passé OU Des restrictions concernant la durée de validité d'une carte sont configurées au niveau du TPE
301	cardcvx_too_long	Le cryptogramme visuel de la carte doit être composé de 3 ou 4 chiffres	Le cryptogramme visuel saisi contient plus de 4 caractères
302	cardcvx_too_short	Le cryptogramme visuel de la carte doit être composé de 3 ou 4 chiffres	Le cryptogramme visuel saisi contient moins de 3 caractères
303	cardcvx_invalid_format	Le cryptogramme visuel de la carte ne peut contenir que des chiffres	Le cryptogramme visuel saisi ne contient pas que des chiffres
401	cardholdername_too_long	Le nom du porteur de la carte ne peut contenir plus de 45 caractères	Le nom du porteur saisi contient plus de 45 caractères
402	cardholdername_too_short	Le nom du porteur de la carte doit contenir au moins 2 caractères	Le nom du porteur saisi contient moins de 2 caractères
403	cardholdername_invalid	Le nom du porteur de la carte est invalide	Le nom du porteur saisi contient des chiffres ou des caractères spéciaux

Exemple 1 :

```
1 hostedFields.addEventListener('change', function(event) {
2     if (event.detail.complete) {
3         // enable payment button
4     } else if (event.detail.error) {
5         // show error message to customer
6     }
7 });
```

Exemple 2 :

```
1 hostedFields.addEventListener('delete', function(event) {
2     if (event.detail.success === true) {
3         // generate cardNumber, cardExpDate and cardHolderName
4     hosted fields
5     }
6 });
7 hostedFields.addEventListener('switch', function(event) {
8     // generate cardNumber, cardExpDate and cardHolderName hosted
9     fields
10 });
```

Note : lorsqu'un évènement de type « *delete* » ou « *switch* » est détecté, il n'est pas nécessaire de réinitialisé le SDK avec un nouveau jeton pour pouvoir générer les champs de saisie de la nouvelle carte. La seule contrainte est de n'instancier qu'une seule fois chaque type de champ pour un jeton donné.

Note 2 : si un champ d'un certain type a déjà été généré et que vous souhaitez le générer à nouveau pour un même jeton, vous pouvez faire appel aux fonction « *removeField* » et « *removeFields* » décrites ci-dessous.

Exemples d'objet « event.detail » pour l'événement de type « change » :

```
1 {
2     complete: true,
3     error: undefined
4 }
```

```
1 {
2     complete: false,
3     error: {
4         cardCvx: {
5             code: 301,
6             description: 'cardcvx_too_long',
7             message: 'Le cryptogramme visuel de la carte doit
8 être composé de 3 chiffres exactement'
9         },
10        cardNumber: {
11            code: 102,
12            description: 'cardnumber_too_short',
13        }
```

```
14         message: 'Le numéro de carte doit être composé 15 à
15 19 chiffres'
16     }
17 }
18 }
```

```
1 {
2     complete: false,
3     error: {
4         cardPreview: {
5             cardExpDate: {
6                 code: 203,
7                 description: 'cardexpdate_invalid_date',
8                 message: 'La durée de validité de la carte doit
9 être supérieure à 4 mois'
10            },
11            cardNumber: {
12                code: 102,
13                description: 'cardnumber_not_accepted_pan',
14                message: 'Ce type de carte n'est pas accepté par
15 la société'
16            },
17            code: 1,
18            description: 'invalid_request',
19            message: 'Votre demande de paiement ne peut aboutir'
20        }
21    }
22 }
```

Exemples d'objet « event.detail » pour l'événement de type « delete » :

```
1 { success: true }
```

Exemples d'objet « event.detail » pour l'événement de type « consent » :

```
1 { granted: true, timestamp: '2022-03-07 10:00:52' }
```

```
1 { granted: false }
```

removeField(fieldType)

Cette méthode permet de supprimer une instance d'un champ *hosted field* créé via la fonction « *createField* ».

Chaque champ ne pouvant être instancié qu'une seule fois par session, cela permet par exemple de réinstancier le champ du type correspondant à un autre emplacement du DOM ou sur une autre page web. Pour cela, les fonctions « *createField* » et « *generate* » devront toutes deux à nouveau être appelées pour réinstancier le champ en question.

Paramètres :

Champ	fieldType
Présence	Obligatoire
Description	Type du champ à supprimer
Format	Chaîne de caractères
Valeur(s) possible(s)	cardNumber, cardExpDate, cardCvx, cardHolderName, cardPreview, cardSwitch

removeFields()

Cette méthode permet de supprimer les instances de tous les champs *hosted fields* générés via la fonction « *createField* » et associé au jeton fourni à la méthode « *initialize* ».

Chaque champ ne pouvant être instancié qu'une seule fois par session, cela permet par exemple de réinstancier l'ensemble des champs à un autre emplacement du DOM ou sur une autre page web. Pour cela, les fonctions « *createField* » et « *generate* » devront toutes deux à nouveau être appelées pour chacun des champs à instancier.

6. Informations et remarques pour l'intégration

Un exemple d'intégration des *hosted fields* Monetico online est disponible aux URL suivantes :

Production - Sandbox	https://p.monetico-services.com/test/hostedfields/hostedfields_example.cgi https://p.monetico-services.com/test/hostedfields/hostedfields_example2.cgi
Production - Live	https://p.monetico-services.com/hostedfields/hostedfields_example.cgi https://p.monetico-services.com/hostedfields/hostedfields_example2.cgi

Note : l'exemple 2 illustre un parcours de paiement avec réutilisation d'une carte enregistrée, il ne fonctionne que si une carte a été enregistrée au préalable

Remarques :

- La largeur d'un *hosted field* s'adapte à la largeur de la balise `<div>` où il a été généré même si celle-ci varie après génération de la page html.
- La hauteur d'un *hosted field* s'adapte à la hauteur de la balise `<div>` où il a été généré, mais celle-ci doit être explicitement fixée au préalable.
- La hauteur de la `<div>` du *hosted field* doit être en cohérence avec la configuration demandée pour celui-ci : la hauteur doit être assez grande pour contenir le texte avec la taille de police demandée, pour prendre en compte le padding et l'épaisseur de la bordure souhaitée.
- La hauteur de la balise `<div>` du champ de type *cardNumber* doit être de 20px minimum pour contenir le menu du choix du réseau de la carte. Il est néanmoins recommandé de la fixer à au moins 30px.
- Certaines variantes de styles peuvent s'appliquer au même moment sur un champ (ex : un champ peut être à la fois vide et à la fois *focused*). Dans ce cas, si les styles appliqués redéfinissent les mêmes propriétés CSS, ils seront priorisés dans l'ordre suivant :

base < empty < focused < completed < invalid