



API de Passerelle Moneris – Guide d'intégration – .NET

Version : 1.6.8

Copyright © Moneris Solutions, 2025

All rights reserved. No part of this **publication** may be reproduced, stored in retrieval systems, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Moneris Solutions Corporation.

## Sécurité et conformité

Il est possible que votre solution doive se conformer aux exigences PCI, CISP et PABP des associations de cartes de crédit. Pour en savoir plus sur la façon de rendre votre application conforme aux normes PCI DSS, veuillez communiquer avec le centre des ventes de Moneris. Vous pouvez également consulter le site <https://developer.moneris.com> pour télécharger le guide d'implantation des normes PCI DSS.

Tous les commerçants et fournisseurs de services qui enregistrent, traitent et transmettent les données des titulaires de cartes doivent respecter les normes de sécurité des données de l'industrie des cartes de paiement (PCI DSS) et les programmes de conformité de l'association de cartes de crédit. Cependant, les exigences de certification varient en fonction des entreprises, et elles dépendent de votre niveau de commerçant ou de votre niveau de fournisseur de service.

L'association de cartes de crédit a certaines normes de sécurité des données qui précisent les exigences que toutes les entreprises stockant, traitant ou transférant les données des titulaires de carte doivent respecter. En tant que client ou partenaire de Moneris utilisant ce mode d'intégration, votre solution doit respecter les normes de sécurité des données de l'industrie des cartes de paiement (PCI DSS) ou les normes de sécurité des données des applications de paiement (PA DSS). Ces normes sont conçues pour aider les titulaires de carte et les commerçants à faire en sorte que les numéros de carte de crédit soient chiffrés lorsqu'ils sont transmis à une base de données ou enregistrés dans celle-ci, ainsi qu'à veiller à ce que les commerçants mettent en place de bonnes mesures de contrôle.

Des solutions non conformes peuvent empêcher les commerçants de faire affaire avec Moneris. Un commerçant non conforme peut également être soumis à des amendes, des frais, des évaluations ou la résiliation de ses services de traitement des paiements.

Pour en savoir plus sur les exigences des normes PCI DSS et PA DSS, veuillez consulter le site <http://www.pcisecuritystandards.org>.

## Confidentialité

Vous devez protéger les renseignements confidentiels des titulaires de carte et des commerçants. Vous ne pouvez en aucun cas envoyer des renseignements confidentiels par courriel lorsque vous tentez de régler un problème d'intégration ou de production. Lorsque vous envoyez des exemples de fichiers ou de code aux employés de Moneris à des fins d'analyse, toute référence à des numéros de carte valides, à des comptes de commerçant et à des jetons de transaction doit être retirée ou masquée. Vous ne pouvez en aucun cas utiliser un compte de titulaire de carte actif dans l'environnement de test.

## Changements apportés à la v1.6.8

- Ajout de la demande Vault Tokenize Credit Card à l'ensemble des transactions utilisant la chambre forte. Cette transaction permet de transformer en jeton une carte utilisée lors d'une précédente transaction financière sans avoir à soumettre à nouveau les données de la carte.
- Ajout du champ return\_issuer\_id à la définition des champs de la demande pour la chambre forte.
- Ajout de nouvelles valeurs pour le champ request\_type dans les demandes d'authentification 3DS.
- Mise à jour de la prise en charge de Visa Secure (3DS) pour le champ <ri\_indicator> afin d'accepter les codes 01, 02, 06, 07 et 11 pour les transactions de paiement et 03, 04, 05 et 10 pour les transactions sans paiement.

### Changements dans la version 1.6.7

- Suppression du chapitre Interac en ligne et des sujets connexes dans l'annexe concernant la certification avec Interac en ligne. Le soutien pour Interac en ligne a pris fin le 31 octobre 2024.
- Changements dans la version 1.6.6
- Ajout de la préautorisation par incréments dans les transactions de base.
- Ajout du champ is\_incremental aux transactions de préautorisation de base et de préautorisation avec la chambre forte.

### Changements dans la version 1.6.5

- Ajout d'un nouvel objet et de nouveaux champs de demande pour prendre en charge la vérification du nom du compte Visa en tant qu'option dans la transaction de base de vérification de la carte. Voir account\_name\_verificationet ses sous-champs first\_name, middle\_name, and last\_name.
- Ajout du champ de réponse AccountNameVerificationResultCode aux définitions des champs de réponse de base dans le cadre de la nouvelle fonction Visa Account Name Verification.

### Changements apportés par la version 1.6.4

- Ajout du champ <browser\_ip> à la demande d'authentification 3DS pour les modules d'extension des commerçantes et commerçants – navigateur
- Ajout des champs <work\_phone>, <home\_phone> et <mobile\_phone> à la demande d'authentification 3DS pour les modules d'extension des commerçantes et commerçants – navigateur
- Ajout de trois nouveaux types de transaction : Ajout d'un jeton temporaire de Google Pay, Achat d'un jeton de Google Pay et Jeton de préautorisation de Google Pay (Ces transactions permettent d'envoyer des données utiles et chiffrées de Google Pay et de recevoir un jeton temporaire Moneris en échange de l'authentification 3DS.)
- Ajout de nouveaux objets et champs de requête pour prendre en charge les transactions par jeton Google Pay tels que PaymentToken et ses sous-champs signature, version du protocole, et message signé
- Ajout du champ de réponse GooglePaymentMethod
- Ajout du code d'avis du commerçant dans les exemples de réponses à l'achat, à la préautorisation, à un achat avec le code de vérification d'authentification du titulaire de carte (CAVV) et à une préautorisation avec le CAVV
- Mise à jour du type et des limites du DS trans ID et de la description dans B.1 Définition des champs de réponse - 3-D Secure
- Suppression du champ requestType de Demande d'authentification MPI 3DS – Processus d'authentification 3RI périodique
- Modification de la note pour les champs recurring\_frequency et recurring\_expiry
- Ajout des méthodes Set dans la demande d'authentification MPI 3DS
- Suppression de la vérification d'état dans la demande d'authentification MPI 3DS

- Suppression de threedsCompletionInd dans la demande d'authentification MPI 3DS – Processus d'authentification 3RI périodique et non périodique
- Déplacement du champ prior\_request\_auth\_info vers la section facultative pour MPI 3DS Prior Authentication Info for MPI 3DS Authentication Request - 3RI with recurring
- Modification de la note relative au champ Courriel
- Ajout d'un champ « code d'avis du commerçant » dans l'Annexe B : Définitions des champs de réponse

### **Changements apportés à la v1.6.2**

- Passage de la version 3-D à la version 2.2
- Ajout du diagramme du processus de transaction avec l'authentification pour 3-D Secure - canal 3RI
- Passage du champ courriel de facultatif à obligatoire
- Ajout d'une note au champ Courriel
- Ajout de l'état "D" dans le code TransStatus
- Modification du code de raison d'un état de transaction Codes de refus
- Suppression de la section Visa Checkout (en raison du retrait de Visa Checkout, qui a eu lieu le 1er juin 2023)
- Modification de la note relative au champ dsTransId
- Modification de la note relative au champ threedsServerTransId
- Ajout d'un commentaire à Visa Secure
- Modification de la note relative au champ ri\_indicator
- Ajout d'une note à ds\_trans\_id sur le fait de ne le soumettre que dans les transactions financières en cas d'utilisation d'un service 3DS Secure tiers
- Ajout de nouveaux champs 3DS à l'authentification 3DS pour prendre en charge 3RI, tels que les champs messageCategory, device\_channel et ri\_indicator
- Ajout de nouveaux champs 3DS à l'authentification 3DS pour prendre en charge l'authentification découpée 3RI tels que les champs decoupled\_request\_indicator, decoupled\_request\_max\_time et decoupled\_request\_async\_url
- Ajout d'un nouvel objet 3DS prior\_request\_auth\_data à l'authentification 3DS pour prendre en charge 3RI, y compris ses champs prior\_request\_auth\_info, prior\_request\_auth\_method, prior\_request\_auth\_ref et prior\_request\_auth\_timestamp
- Ajout de champs supplémentaires dans les réponses 3DS : threedsVersion et AuthenticationType
- Ajout de rubriques supplémentaires sur l'authentification 3DS pour les scénarios 3RI avec et sans fonctionnalités récurrentes
- Scénario d'authentification 3DS existant renommé afin de préciser qu'il est destiné uniquement au canal du navigateur
- Ajout d'une rubrique sur le flux d'authentification du canal 3RI et modification du titre du flux 3DS précédent pour préciser qu'il est destiné uniquement au canal du navigateur
- Ajout d'une rubrique sur la gestion du flux d'authentification découpée 3RI pour expliquer la gestion des réponses asynchrones
- Ajout d'une rubrique sur les points de terminaison serveur à serveur afin de couvrir l'URL distincte pour l'authentification 3DS

### **Changements apportés à la v1.6.1**

- Ajout d'une note à ds\_trans\_id sur le fait de ne le soumettre que dans les transactions financières en cas d'utilisation d'un service 3DS Secure d'un tiers

### **Changements apportés à la v1.6.0**

- Ajout d'un nouveau champ foreign\_indicator à l'ensemble des transactions de base : purchase et preauth
- Ajout d'un nouveau champ foreign\_indicator à l'ensemble de transactions 3-D Secure : cavvPurchase et cavvPreauth
- Ajout de nouvelles valeurs foreign\_indicator dans l'annexe A.2 Définition des champs de demande - Champs principaux
- Modification de l'exemple de code pour inclure le nouveau champ foreign\_indicator

- Modification du tableau des codes de réponse du SVA

### **Changements apportés à la v1.5.0**

- Ajout d'une nouvelle définition pour les transactions de demande d'authentification 3DS pour les modules d'extension pour les commerçants
- Ajout d'une nouvelle valeur V pour le champ de demande **payment indicator** pour tenir compte des commerçants qui offrent des paiements périodiques variables
- Modification des valeurs permises pour le champ **e-commerce indicator** afin de tenir compte de la nouvelle valeur **payment indicator**, et modification de la rubrique correspondante dans la section Renseignements d'identification au dossier
- Ajout d'une note concernant les limites du champ de demande **cardholder name** pour indiquer les caractères accentués ne sont pas autorisés
- Ajout d'une section et de rubriques concernant la solution Versements Visa
- Ajout d'une rubrique au sujet de la chambre forte et des versements dans la section Chambre forte

### **Changements apportés à la v1.4.4**

- Ajout d'une nouvelle rubrique dans la section sur la chambre forte indiquant les transactions de la chambre forte qui prennent en charge les jetons temporaires
- Limite du champ de réponse **convenience fee status** des frais de commodité corrigée à trois caractères alphanumériques
- Renseignement ajouté dans la section À propos des frais de commodité et à la définition des champs de réponse pour préciser que les transactions utilisant des frais de commodité ne prennent pas en charge la TMD ou les portefeuilles électroniques
- Ajout ou modification des descriptions des types de transactions dans les sections Ajout d'un jeton à la chambre forte, Suppression d'un jeton dans la chambre forte, Détection de carte d'entreprise dans la chambre forte et Ajout d'une carte de crédit à la chambre forte
- Correction des méthodes Get pour les champs de réponse Numéro de carte de crédit dans la chambre forte et Numéro de carte de crédit masqué par la chambre forte

### **Changements apportés à la v1.4.3**

- Ajout de nouveaux sujets sur les transactions traitées avec la TMD : Achat avec la TMD et 3-D Secure, Achat avec la TMD, 3-D Secure et la chambre forte, Préautorisation d'une transaction avec la TMD et 3-D Secure, et Préautorisation d'une transaction avec la TMD et 3-D Secure
- Ajout de rubriques sur les types de transactions dans les sections Achats avec 3-D Secure et la chambre forte, Préautorisations avec 3-D Secure et la chambre forte, Achats avec 3-D Secure et Facturation périodique avec 3-D Secure 2.0
- Ajout d'un nouveau champ de demande **DS transaction ID**
- Champs de demande liés à la facturation devenus obligatoires pour la transaction de demande d'authentification 3DS

### **Changements apportés à la v1.4.2**

- Correction des limites pour le champ de demande **start date**
- Correction de la présentation des ensembles de méthodes pour les champs de demande **browser language**, **browser java enabled**, **browser screen height** et **browser screen width**

### **Changements apportés à la v1.4.1**

- Ajout de renseignements sur les champs de demande 3DS version et 3DS server indicator dans les rubriques sur les transaction Achats avec 3-D Secure et Préautorisations avec 3-D Secure
- Correction du nom de variable pour le champ de demande 3DS server indicator

## **Changements apportés à la v1.4.0**

- La section 3-D Secure 2.0 a remplacé la section Modules d'extension pour les commerçants (concernant l'outil 3-D Secure 1.0)
- Ajout de nouveaux champs liés à l'outil 3-D Secure 2.0 dans les sections Achat avec 3-D Secure et Préautorisation avec 3-D Secure
- Ajout de références concernant l'outil 3-D Secure 2.0, y compris les codes CAVV Results pour les cartes Visa, Mastercard et American Express, et les codes TransStatus pour les transactions traitées avec l'outil 3-D Secure 2.0
- Ajout de la section Définition des champs de réponse liés à 3-D Secure
- Ajout de la section Définition des champs de demande – 3-D Secure 2.0 et retrait de l'ancienne section Définition des champs de demande – Modules d'extension pour les commerçants

## **Modification de la version 1.3.1**

- Modification du code de réponse 959 dans la section « Autres codes de réponse »

## **Modification de la version 1.3.0**

- Retrait de la mention de tests par carte Visa seulement dans la section Au sujet des renseignements d'identification au dossier
- Retrait des différences du SVA et du NVC entre les marques de carte prises en charge dans la section Vérification de carte
- Ajout de renseignements au sujet de la prise en charge des cartes American Express dans les sections Vérification de carte avec le SVA et le NVC et Vérification de carte avec la chambre forte
- Retrait du type de transaction Réautorisation et des renseignements connexes
- Retrait de la section sur les transactions par glissement de la bande magnétique
- Ajout de référence sur les codes de réponse
- Ajout des nouvelles devises prises en charge à la section Codes de devises de la TMD
- Ajout de définitions concernant les types de transactions aux rubriques liées à Visa Checkout
- Ajout de renseignements concernant le comportement du descripteur dynamique lié aux transactions de préautorisation dans les rubriques Préautorisation et Conclusion de préautorisation
- Ajout des renseignements manquants sur le champ de descripteur dynamique à la rubrique Achat avec 3-D Secure
- Ajout de renseignements sur les valeurs des champs liés aux indicateurs de paiements et de commerce électronique, ainsi que l'ajout des rubriques Indicateur de paiement et Valeurs des indicateurs de commerce électronique
- Réorganisation des renseignements sur les frais de commodité et ajout des nouvelles rubriques Transactions prenant en charge les frais de commodité et Objet d'information sur les frais de commodité
- Modification du nom des rubriques sur les types de transactions concernant les frais de commodité pour mettre l'accent sur le fait que l'achat est la transaction de base et que les frais de commodité sont une fonction additionnelle
- Réorganisation des sections Définition des champs de demande selon les ensembles de fonction et retrait des champs de demande Mag Swipe
- Nouvelles rubriques sur la définition des champs de demande pour les champs de connexion, les principaux champs, la chambre forte, les modules d'extension pour les commerçants et les frais de commodité

## **Changements apportés aux versions précédentes**

### **Changements apportés à la v1.2.12**

- Ajout d'avertissements concernant la mise en place de l'outil 3-D Secure à l'aide de cadres dans la section Modules d'extension pour les commerçants et de rubriques concernant les types de transactions liés à 3-D Secure

## **Changements apportés à la v1.2.11**

- Ajout de renseignements au sujet de Google Pay<sup>MC</sup> et retrait des mentions d'Android Pay
- Séparation des flux de traitement des transactions en deux sujets distincts pour Apple Pay et Google Pay<sup>MC</sup>, soit Sommaire du processus de transaction avec Apple Pay et Sommaire du processus de transaction avec Google Pay<sup>MC</sup>

## **Changements apportés à la v1.2.10**

- Ajout de champs de demande manquants dans la section Achat avec la TMD et la chambre forte

## **Changements apportés à la v1.2.9**

- Correction de l'exemple de code pour certaines transactions financières (retrait de l'ancienne méthode de traitement de la TMD)

## **Changements apportés à la v1.2.8**

- Ajout d'une section sur les transactions traitées avec la tarification multidevise (TMD)
- Ajout de nouvelles rubriques dans la section Renseignements d'identification enregistrés au dossier :
- Transactions nécessitant des renseignements d'identification au dossier entamées par le commerçant et entamées par le titulaire de carte
- Renseignements d'identification enregistrés au dossier avec des renseignements d'identification précédemment stockés
- Modification du numéro de carte test de Discover (maintenant 6011000992927602)
- Modification de la description des indicateurs de commerce électronique dans la section Définition des champs de réponse afin de retirer les valeurs autorisées dépréciées (8 et 9)

## **Changements apportés à la v1.2.7**

- Ajout des limites manquantes pour les variables de demande Montant, Montant de conclusion et Montant de transaction

## **Changements apportés à la v1.2.6**

- Modification de la valeur « 7 » de la variable de demande Indicateur de commerce électronique (crypt\_type) pour indiquer que cette valeur représente également les transactions American Express SafeKey non authentifiées

## **Changements apportés à la v1.2.5**

- Modification de la section Transaction d'achat pour inclure la variable Customer ID

## **Changements apportés à la v1.2.4**

- Modifications des limites des variables de demandes Montant, Montant de la transaction et Montant de la conclusion pour inclure 10 décimales

## **Modification de la version 1.3.2**

- Ajout de renseignements concernant le transfert de données Offlin<sup>MC</sup> pour le pixel invisible Card Match des transactions traitées par l'API unifiée

# Table des matières

<b>Sécurité et conformité</b>	<b>2</b>	
Confidentialité	2	
<b>Changements apportés à la v1.6.8</b>	<b>3</b>	
<b>Obtenir de l'aide</b>	<b>14</b>	
<b>1.</b>	<b>À propos du présent document</b>	<b>15</b>
1.1	Objectif	15
1.2	À qui ce guide s'adresse-t-il?	15
<b>2.</b>	<b>Ensemble de transactions de base</b>	<b>16</b>
2.1	Achat	17
2.2	Préautorisation	22
2.3	Préautorisation incrémentale	28
2.4	Conclusion de préautorisation	30
2.5	Transaction forcée	36
2.6	Correction d'achat	39
2.7	Remboursement	42
2.8	Remboursement indépendant	44
2.9	Vérification de carte avec le SVA et le NVC	48
2.10	Fermeture de lot	53
2.11	Ouverture des totaux	55
<b>3</b>	<b>Renseignements d'identification au dossier</b>	<b>57</b>
3.1	À propos des renseignements d'identification au dossier	57
3.2	Objet Credential on File Info et variables	57
3.3	Types de transactions nécessitant des renseignements	58
3.4	Transactions nécessitant des renseignements d'identification au dossier entamées par le commerçant et entamées par le titulaire de carte	58
3.5	Transactions initiales nécessitant des renseignements	59
	Renseignements d'identification enregistrés au dossier avec des	59
3.6	renseignements d'identification précédemment stockés	59
3.7	Valeurs des indicateurs de paiement (payment indicator) et de commerce électronique (e-commerce indicator)	60
3.8	Transformation en jetons des cartes de crédit et des renseignements d'identification au dossier	60
3.9	Renseignements d'identification au dossier et conversion de jetons temporaires	61
3.10	Transactions de vérification de carte et de renseignements d'identification au dossier	61
3.10.1	Quand effectuer une vérification de carte avec les renseignements d'identification au dossier	62
3.10.2	Renseignements d'identification au dossier et ajout de jeton à la chambre forte	62
3.10.3	Renseignements d'identification au dossier et mise à jour d'une carte de crédit dans la chambre forte	62
3.10.4	Renseignements d'identification au dossier et ajout d'une carte de crédit	63
3.10.5	Renseignements d'identification au dossier et facturation périodique	63
<b>4</b>	<b>Chambre forte</b>	<b>64</b>
4.1	À propos des transactions utilisant la chambre forte	64
4.2	Types de transactions utilisant la chambre forte	64
4.2.1	Types de transactions administratives utilisant la chambre forte	64

4.2.2	Types de transactions financières utilisant la chambre forte	66
4.3	Transactions utilisant la chambre forte qui prennent en charge les jetons temporaires	66
4.4	Chambre forte et versements	68
4.5	Transactions administratives utilisant la chambre forte	68
4.5.1	Ajout d'une carte de crédit à la chambre forte (ResAddCC)	68
4.5.2	Ajout d'un jeton temporaire à la chambre forte (ResTempAdd)	75
4.5.3	Mise à jour d'une carte de crédit dans la chambre forte (ResUpdateCC)	77
4.5.4	Suppression d'un profil de la chambre forte (ResDelete)	84
4.5.5	Recherche d'un numéro complet dans la chambre forte (ResLookupFull)	86
4.5.6	Recherche d'un numéro masqué dans la chambre forte	88
	(ResLookupMasked)	88
4.5.7	Obtention des cartes expirées dans la chambre forte (ResGetExpiring)	89
4.5.8	Détection de carte d'entreprise dans la chambre forte	91
4.5.9	Ajout d'un jeton à la chambre forte (ResAddToken)	93
4.5.10	Transformation en jeton d'une carte de crédit dans la chambre forte (ResTokenizeCC)	96
4.6	Transactions financières utilisant la chambre forte	100
4.6.1	Changements apportés à l'ID du client	100
4.6.2	Achat avec la chambre forte (ResPurchaseCC)	101
4.6.3	Préautorisation avec la chambre forte (ResPreauthCC)	106
4.6.4	Transaction de remboursement indépendant avec la chambre forte (ResIndRefundCC)	113
4.6.5	Transaction forcée avec la chambre forte (ResForcePostCC)	116
4.6.6	Vérification de la carte avec la chambre forte (ResCardVerificationCC)	119
4.7	Transformation en jetons hébergée	123
<b>5</b>	<b>Les transactions de niveaux 2 et 3</b>	<b>124</b>
5.1	À propos des transactions de niveaux 2 et 3	124
5.2	Transactions Visa de niveaux 2 et 3	124
5.2.1	Types de transactions de niveaux 2 et 3 pour Visa	124
5.2.2	Flux de transaction de niveaux 2 et 3 par carte Visa	126
5.2.3	Conclusion par carte Visa	128
5.2.4	Correction d'achat par carte Visa	133
5.2.5	Transaction forcée par carte Visa	135
5.2.6	Remboursement par carte Visa	140
5.2.7	Remboursement indépendant par carte Visa	145
5.2.8	Transaction VS Corpais	150
5.3	Transactions de niveau 2 et 3 de Mastercard	161
5.3.1	Types de transaction de niveaux 2 et 3 par carte Mastercard	161
5.3.2	Flux de transaction de niveaux 2 et 3 par carte Mastercard	164
5.3.3	Conclusion par carte Mastercard	166
5.3.4	Transaction forcée par carte Mastercard	167
5.3.5	Correction d'achat par carte Mastercard	170
5.3.6	Remboursement par carte Mastercard	172
5.3.7	Remboursement indépendant par carte Mastercard	174
5.3.8	MC Corpais – Données communes de carte d'entreprise avec les détails de la ligne d'article	177
5.4	Transactions American Express de niveaux 2 et 3	194
5.4.1	Types de transaction de niveaux 2 et 3 par carte Amex	194
5.4.2	Flux de transaction de niveaux 2 et 3 par carte Amex	196
5.4.3	Objets de données de niveaux 2 et 3 pour les transactions par carte Amex	196
5.4.4	Conclusion par carte Amex	216
5.4.5	Transaction forcée par carte Amex	219
5.4.6	Correction d'achat par carte Amex	223
5.4.7	Remboursement par carte Amex	225
5.4.8	Remboursement indépendant par carte Amex	228

<b>6</b>	<b>3-D Secure 2.2</b>	<b>232</b>
6.1	À propos de la solution 3-D Secure 2.2	232
6.1.1	Mises en place de l'outil 3-D Secure	232
6.1.2	Hors de portée ou non pris en charge	233
6.1.3	Compatibilité des versions	233
6.1.4	Passage de l'outil 3-D Secure 2.0 à 2.2	234
6.2	Créer votre intégration 3-D Secure 2.2	234
6.2.1	Activation de la fonction 3-D Secure	234
6.2.2	Flux des transactions avec la solution 3-D Secure	234
6.2.3	Flux de transactions pour 3-D Secure : Canal 3RI	236
6.3	Mise en œuvre de la demande de recherche de carte (CardLookup)	237
6.3.1	Demande de recherche de carte (Card Lookup)	237
6.4	Gestion de la méthode 3DS pour l'empreinte digitale des appareils	239
6.5	Mise en œuvre de la demande d'authentification 3DS	240
6.5.1	Mise en œuvre de la demande d'authentification 3DS pour les modules d'extension des commerçants	240
6.5.2	Demande d'authentification MPI 3DS - Authentification 3RI avec un processus d'authentification périodique	251
6.5.3	Demande d'authentification MPI 3DS – Authentification 3RI sans processus périodique	261
6.6	Traitement du flux de contestation	270
6.6.1	Demande de recherche de code de vérification d'authentification du	270
6.7	Gestion du processus d'authentification découpée	272
6.8	Effectuer l'autorisation	275
6.8.1	Achat avec la solution 3-D Secure (cavv_Purchase)	275
6.8.2	Préautorisation avec la solution 3-D Secure (cavv_Preauth)	283
6.8.3	Achat avec la chambre forte et la solution 3-D Secure	293
6.8.4	Préautorisation avec la chambre forte et la solution 3-D Secure	298
6.8.5	Achat avec la solution 3-D Secure et facturation périodique	302
6.9	Tester votre intégration 3-D Secure 2.2	303
6.10	Passage à la phase de production avec 3-D Secure 2.2	304
6.11	Codes TransStatus de la solution 3-D Secure 2.2	304
6.12	3-D Secure 2.2 Codes de refus TransStatusReason communs	305
6.13	Codes de résultat liés au code de vérification d'authentification du titulaire de carte (CAVV)	306
	Codes de résultat liés au code de vérification d'authentification du	307
6.13.1	titulaire de carte (CAVV) de Visa	307
	Codes de résultat liés au code de vérification d'authentification du	308
6.13.2	titulaire de carte (CAVV) de Mastercard	308
6.13.3	Codes de résultat liés au code de vérification d'authentification du titulaire de carte (CAVV) de American Express	309
<b>7</b>	<b>Tarification multidevise (TMD)</b>	<b>310</b>
7.1	À propos de la tarification multidevise (TMD)	310
7.2	Méthode de traitement des transactions avec la TMD	310
7.3	Obtention du taux de la TMD	311
7.4	Achat utilisant la TMD	313
7.5	Achat utilisant la TMD avec 3-D Secure	317
7.6	Achat utilisant la TMD avec 3-D Secure et la chambre forte	323
7.7	Préautorisation utilisant la TMD328	
7.8	Préautorisation utilisant la TMD avec 3-D Secure	333
7.9	Préautorisation utilisant la TMD avec 3-D Secure et la chambre forte	338
7.10	Conclusion de préautorisation utilisant la TMD	344
7.11	Correction d'achat utilisant la TMD	347

7.12	Remboursement utilisant la TMD	350
7.13	Remboursement indépendant utilisant la TMD	353
7.14	Achat utilisant la TMD avec la chambre forte	357
7.15	Préautorisation utilisant la TMD avec la chambre forte	361
7.16	Remboursement indépendant utilisant la TMD avec la chambre forte	366
7.17	Codes de devise de la TMD	369
7.18	Codes d'erreur de la TMD	377
<b>8</b>	<b>Versements Visa</b>	<b>378</b>
8.1	À propos de la solution Versements Visa	378
8.2	Types de transaction Versements Visa	378
8.3	Envoi de transactions avec la solution Versements Visa	379
8.4	Recherche de plan de versements	379
8.5	Recherche de plan de versements dans la chambre forte	383
8.6	Objet Installment Info	386
<b>9</b>	<b>Outils de protection contre la fraude en ligne</b>	<b>387</b>
9.1	Service de vérification d'adresse	388
9.1.1	À propos du service de vérification d'adresse (SVA)	388
9.1.2	Objet AVS Info	389
9.1.3	Codes de réponse du SVA	389
9.1.4	Exemple d'une transaction utilisant le SVA	392
9.2	Numéro de vérification de carte (NVC)	393
9.2.1	À propos du numéro de vérification de carte (NVC)	393
9.2.2	Transactions nécessitant le NCV	393
9.2.3	Objet CVD Information	394
9.2.4	Codes de résultat liés au NVC	395
9.2.5	Exemple d'une transaction d'achat avec l'objet CVD Info	396
9.3	Outil de gestion des risques transactionnels	397
9.3.1	À propos de l'outil de gestion des risques transactionnels	397
9.3.2	Introduction aux demandes d'information (queries)	397
9.3.3	Demande d'information sur la session (Session Query)	398
9.3.4	Demande d'information sur les attributs (Attribute Query)	405
9.3.5	Gérer les réponses	409
9.3.6	Ajout de balises de profilage à votre site Web	424
9.4	Intégration de tous les outils de protection contre la fraude offerts	425
9.4.1	Options de mise en œuvre de l'outil de gestion des risques transactionnels (TRMT)	425
9.4.2	Liste de vérification de mise en œuvre	426
9.4.3	Prise de décision	427
<b>10</b>	<b>Intégration d'Apple Pay et de Google PayMC</b>	<b>428</b>
10.1	À propos de l'intégration d'Apple Pay et de Google PayMC	428
10.2	Sommaire du processus de transaction Apple Pay	428
10.3	Sommaire du processus de transaction Google PayMC	429
10.4	À propos de l'intégration de l'API d'Apple Pay et de Google PayMC	434
10.4.1	Types de transaction utilisées pour Apple Pay et Google PayMC	434
	Achat utilisant le code de vérification d'authentification du	435
10.5	titulaire de carte – Apple Pay et Google PayMC	435
	Préautorisation utilisant le code de vérification d'authentification	440
10.6	du titulaire de carte – Apple Pay et Google PayMC	440
10.7	Ajout d'un jeton temporaire de Google Pay - GooglePayTokenTempAdd	445
10.8	Transaction de préautorisation par jeton Google PayMD	448
10.9	Achat de jetons Google PayMD	453

<b>11</b>	<b>OfflinxMC</b>	<b>459</b>
11.1	Qu'est-ce qu'un pixel invisible?	459
11.2	OfflinxMC et les transactions API	459
<b>12</b>	<b>Frais de commodité</b>	<b>460</b>
12.1	À propos des frais de commodité	460
12.2	Objet Convenience Fee Information	460
12.3	Achat avec frais de commodité	461
12.4	Achat avec renseignements sur le client et frais commodité	464
12.5	Achat avec frais de commodité utilisant 3-D Secure	470
<b>13</b>	<b>Facturation périodique</b>	<b>474</b>
13.1	À propos de la facturation périodique	474
13.2	Achat avec la facturation périodique	474
13.3	Mise à jour de la facturation périodique	477
13.4	Codes et champs de réponse liés à la facturation périodique	481
13.5	Renseignements d'identification au dossier et facturation périodique	482
<b>14</b>	<b>Renseignements du client</b>	<b>483</b>
14.1	Utiliser l'objet Customer Information	483
14.1.1	Objet Customer Info – Propriétés diverses	484
14.1.2	Objet Customer Info – Renseignements de facturation et d'expédition	484
14.1.3	Objet Customer Info – Information sur les articles	486
14.2	Exemple d'une transaction incluant l'objet Customer Information	487
<b>15</b>	<b>Vérification d'état</b>	<b>490</b>
15.1	À propos de la vérification d'état	490
15.2	Utiliser les champs de réponse liés à la vérification d'état	490
15.3	Exemple d'achat avec la vérification d'état	491
<b>16</b>	<b>Mise à l'essai d'une solution</b>	<b>492</b>
16.1	À propos du centre de ressources pour commerçants	492
16.2	Se connecter à l'environnement d'assurance de la qualité du centre de ressources pour commerçants	492
16.3	Renseignements d'identification test du centre de ressources pour commerçants	493
16.4	Obtenir un ID de commerce et un jeton API uniques pour l'environnement de test	494
16.5	Traiter une transaction	495
16.5.1	Sommaire	495
16.5.2	Objet HttpsPostRequest	497
16.5.3	Objet Receipt	498
16.6	Mise à l'essai d'un module d'extension pour les commerçants	498
16.7	Numéro de carte de test	501
16.7.1	Numéro de carte de test pour les transactions de niveaux 2 et 3	501
16.7.2	Cartes de Test pour Visa Checkout	501
16.8	Simulation du serveur de traitement	502
<b>17</b>	<b>Passage à l'environnement de production</b>	<b>503</b>
17.1	Activation d'un compte de magasin dans l'environnement de production	503
17.2	Configuration d'un commerce pour l'environnement de production	503
17.3	Exigences relatives aux reçus	504
17.3.1	Exigences de certification	505
<b>Annexe A Définition des champs de demande</b>		
A.1	Définition des champs de demande – Champs de connection	506
A.2	Définition des champs de demande – Champs de base	507

A.3	Définition des champs de demande – Renseignements	513
A.4	Définition des champs de demande - Ajout du jeton temporaire de Google Pay	516
A.5	Définition des champs de demande – Chambre forte	517
A.6	Définition des champs de demande – Transactions de niveaux 2 et 3 de Visa	519
A.7	Définition des champs de demande – Transactions de niveaux 2 et 3 de Mastercard	529
A.8	Définition des champs de demande – Transactions de niveaux 2 et 3 de Amex	541
A.9	Définition des champs de demande – 3-D Secure 2.2	552
A.10	Définition des champs de demande – TMD	558
A.11	Définition des champs de demande – Offlinx <sup>MC</sup>	560
A.12	Définition des champs de demande – Frais de commodité	560
A.13	Définition des champs de demande – Transaction périodiques	561
A.14	Définition des champs de demande – Renseignements	562
A.15	Définition des champs de la demande – Objet Account Name Verification	562

**Annexe B Définition des champs de réponse 564**

B.1	Définition des champs de réponse – 3-D Secure	578
B.2	Définition des champs de réponse – TMD	581
B.3	Définition des champs de réponse – Versements Visa	583

**Annexe C Codes de réponse 590**

**Annexe D Messages d'erreur 605**

**Mention des droits d'auteur 607**

**Marques de commerce 608**

## Obtenir de l'aide

Moneris peut vous aider à chaque étape du processus d'intégration.

pour commencer	Durant le développement	Phase de production
Communiquez avec nos spécialistes de l'intégration des clients :  clientintegrations@moneris.com	Si vous collaborez déjà avec un spécialiste de l'intégration et que vous avez besoin de soutien technique, communiquez avec nos conseillers techniques pour les produits en ligne :  1 866 319-7450  <a href="mailto:api@moneris.com">api@moneris.com</a>	Si votre application est déjà fonctionnelle et que vous avez besoin d'aide concernant l'environnement de production, communiquez avec le service à la clientèle de Moneris :  onlinepayments@moneris.com  1 866 319-7450  Vous pouvez appeler en tout temps.

Pour obtenir d'autres ressources, vous pouvez consulter nos forums communautaires à la page

<http://community.moneris.com/product-forums/>.

# 1. À propos du présent document

## 1.1 Objectif

Le présent document décrit les renseignements de transaction nécessaires pour utiliser l'API .NET de Moneris afin de traiter des transactions par carte de crédit. Plus précisément, il décrit le format d'envoi des transactions et les réponses correspondantes.

## 1.2 À qui ce guide s'adresse-t-il?

Le guide d'intégration de l'API de Passerelle Moneris a été conçu pour les développeurs qui doivent travailler avec Passerelle Moneris.

Ce guide tient pour acquis que le système que vous intégrez répond aux exigences décrites ci-dessous et que vous avez une certaine connaissance du langage de programmation .NET.

### Configuration requise

- .NET version 4.7.2 ou plus récente
- Port 443 ouvert pour une communication bidirectionnelle
- Serveur Web avec un certificat SSL

## 2. Ensemble de transactions de base

- 2.1 Achat
- 2.2 Préautorisation
- 2.3 Conclusion de préautorisation
- 2.4 Transaction forcée
- 2.5 Correction d'achat
- 2.6 Remboursement
- 2.7 Remboursement indépendant
- 2.8 Vérification de carte avec le SVA et le NVC
- 2.9 Fermeture de lot
- 2.10 Ouverture des totaux

## 2.1 Achat

Une transaction d'achat vérifie que les fonds sont présents sur la carte du client, retire ces fonds de la carte et les prépare à être déposés dans le compte du commerçant.

```
Purchase purchase = new Purchase();
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setTransaction(purchase);
```

### Principaux champs d'objet de connexion (toutes les transactions API)

Variable	Type et limites	
ID de commerce	<i>Chaîne</i> S. O.	mpgReq.setstoreId(store_id);
Jeton API	<i>Chaîne</i> S. O.	mpgReq.setApiToken(api_token);

### Champs de demande liés aux transactions d'achat (obligatoires)

Variable	Type et limites	
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques	purchase.setOrderId(order_id);
Montant	<i>Chaîne</i> 10 caractères décimaux Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal  <b>EXEMPLE : 1 234 567,89</b>	purchase.setAmount(amount);
Numéro de carte de crédit	<i>Chaîne</i>	purchase.setPan(pan);

Variable	Type et limites	
	20 caractères alphanumériques	
Date d'expiration	<p><i>Chaîne</i></p> <p>4 caractères alphanumériques (format AAMM)</p>	<pre>purchase.setExpDate(expiry_date);</pre>
Indicateur de commerce électronique	<p><i>Chaîne</i></p> <p>1 caractère alphanumérique</p>	<pre>purchase.setCryptType(crypt);</pre>

### Champs de demande liés aux transactions d'achat (facultatifs)

Variable	Type et limites	
ID de client	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p>	<pre>purchase.setCustId(cust_id);</pre>
ID de correspondance de carte	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p>	<pre>purchase.setCmId(transaction_id);</pre>
<b>REMARQUE :</b> Applicables à Offlinx <sup>MC</sup> seulement, chaque transaction doit avoir une valeur unique		
Renseignements du client	<p><i>Objet</i></p> <p>S. O.</p>	<pre>purchase.setCustInfo(customer);</pre>
Renseignements du SVA	<p><i>Objet</i></p> <p>S. O.</p>	<pre>purchase.setAvsInfo(avscCheck);</pre>
Renseignements du NVC	<p><i>Objet</i></p> <p>S. O.</p>	<pre>purchase.setCvdInfo(cvdCheck);</pre>
<b>REMARQUE :</b> Lorsque les renseignements d'identification sont stockés lors de la première		

Variable	Type et limites	
transaction, l'objet CVD (numéro de vérification de carte, ou NVC) doit être envoyé. Pour les transactions subséquentes utilisant les renseignements d'identification stockés, le NVC peut être envoyé avec les transactions entamées par le titulaire de carte seulement. <b>Les commerçants ne doivent pas enregistrer le NVC.</b>		
Information sur les frais de commodité  <b>REMARQUE :</b> Cette variable ne s'applique pas aux transactions traitées avec des renseignements d'identification enregistrés au dossier.	<i>Objet</i> S. O.	<code>purchase.setConvenienceFee(convFeeInfo);</code>
Facturation périodique	<i>Objet</i> S. O.	<code>purchase.SetRecur(recurring_cycle);</code>
Descripteur dynamique	<i>Chaîne</i> 20 caractères alphanumériques	<code>purchase.SetDynamicDescriptor(dynamic_descriptor);</code>
Indicateur étranger	<i>Valeur booléenne</i> true/false	<code>purchase.SetForeignIndicator(foreign_indicator);</code>
Indicateur de portefeuille électronique	<i>Chaîne</i> 3 caractères alphanumériques	<code>purchase.SetWalletIndicator(wallet_indicator);</code>
Renseignements d'identification au dossier cof	<i>Objet</i> S. O.	<code>purchase.SetCofInfo(cof);</code>

<p><b>REMARQUE :</b> Cette variable est un objet imbriqué au sein d'une transaction et elle est requise lors du stockage des renseignements d'identification d'un client ou lors de l'utilisation de tels renseignements stockés. Pour plus de renseignements sur les champs de l'objet Credential on File Info, consultez la section Objet Credential File Info et variables.</p>		
<p><b>Information sur le versement</b></p> <p>Pour les champs liés à cet objet, consultez la section 8.6 Objet Installment Info</p> <p><b>REMARQUE :</b> N'envoyez pas l'objet Installment Info lors d'une transaction qui n'offre pas la fonction Versements Visa, car la transaction pourrait échouer</p>	<i>Objet</i> S. O.	<pre>purchase.setInstallmentInfo(i nstallmentInfo);</pre>

## Champs de demande pour l'objet Credential on File Info

Variable	Type et limites	
ID de l'émetteur  <b>REMARQUE :</b> Cette variable est requise pour chaque transaction entamée par le commerçant à la suite de la première transaction. Après l'envoi de la première transaction, l'ID d'émetteur (issuer ID) est reçu dans la réponse de transaction, puis utilisé dans les demandes de transaction subséquentes.	<i>Chaîne</i> 15 caractères alphanumériques Longueur variable	<pre>cof.setIssuerId("VALUE_FOR_ISSUER_ID");</pre> <p><b>REMARQUE :</b> Pour obtenir une liste et une explication des valeurs possibles à envoyer pour cette variable, consultez la section Définition des champs de demande – Renseignements d'identification au dossier (cof)</p>
Indicateur de paiement  <b>REMARQUE :</b> Lors d'une transaction utilisant les renseignements d'identification au dossier pour laquelle le champ de demande e-commerce indicator est également inclus, les valeurs acceptables de ce champ dépendent de la valeur envoyée pour l'indicateur de paiement. Pour plus d'information, consultez les définitions des champs de réponse.	<i>Chaîne</i> 1 caractère alphabétique	<pre>cof.setPaymentIndicator("PAYMENT_INDICATOR_VALUE");</pre> <p><b>REMARQUE :</b> Pour obtenir une liste et une explication des valeurs possibles à envoyer pour cette variable, consultez la section Définition des champs de demande – Renseignements d'identification au dossier (cof)</p>

<b>Information de paiement</b>	<b>Chaîne</b> <b>1 caractère numérique</b>	<pre>cof.setPaymentInformation("PAYMENT_INFO_VALUE");</pre> <p><b>REMARQUE :</b> Pour obtenir une liste et une explication des valeurs possibles à envoyer pour cette variable, consultez la section Définition des champs de demande – Renseignements d'identification au dossier (cof)</p>
--------------------------------	---	--

## Exemple de transaction d'achat

```
public class TestCanadaPurchase
{
    public static void main(String[] args)
    {
        java.util.Date createDate = new java.util.Date();
        String order_id = "Test"+createDate.getTime();
        String store_id = "monca03650";
        String api_token = "7Yw0MPTlhjBRcZiE6837";
        String amount = "6000.0";
        String pan = "4622943127023886";
        String expdate = "2212"; //YYMM
        String crypt = "7";
        String processing_country_code = "CA";
        boolean foreign_indicator= true; //New Foreign Indicator field
        boolean status_check = false;

        Purchase purchase = new Purchase();
        purchase.setOrderId(order_id);
        purchase.setAmount(amount);
        purchase.setPan(pan);
        purchase.setExpdate(expdate);
        purchase.setCryptType(crypt);
        purchase.setForeignIndicator(foreign_indicator);
        //purchase.setDynamicDescriptor("123456");
        //purchase.setWalletIndicator(""); //Refer documentation for possible values
        //purchase.setCmId("8nAK8712sGaAkls56"); //set only for usage with Offlinx - Unique max 50 alphanumeric characters transaction id generated by merchant

        // TrId and TokenCryptogram are optional, refer documentation for more details.
        //purchase.setTrId("50189815682");
        //purchase.setTokenCryptogram("APmbM/411e0uAAH+s6xMAAADFA==");

        //optional - Installment Info
        InstallmentInfo installmentInfo = new InstallmentInfo();
        installmentInfo.setPlanId("ae59ef1-eb91-b708-8b80-1dd481746401");
        installmentInfo.setPlanIdRef("0000000065");
        installmentInfo.setTacVersion("2");
        purchase.setInstallmentInfo(installmentInfo);

        //optional - Credential on File details
        CofInfo cof = new CofInfo();
        cof.setPaymentIndicator("U");
        cof.setPaymentInformation("2");
        cof.setIssuerId("168451306048014");
        purchase.setCofInfo(cof);

        HttpsPostRequest mpgReq = new HttpsPostRequest();
        mpgReq.setProcCountryCode(processing_country_code);
        mpgReq.setTestMode(true); //false for production transactions
        mpgReq.setstoreId(store_id);
        mpgReq.setApiToken(api_token);
        mpgReq.setTransaction(purchase);
        mpgReq.setStatusCheck(status_check);

        //Optional - Proxy
        mpgReq.setProxy(false); //true to use proxy
        mpgReq.setProxyHost("proxyURL");
        mpgReq.setProxyPort("proxyPort");
        mpgReq.setProxyUser("proxyUser"); //optional - domainName\user
        mpgReq.setProxyPassword("proxyPassword"); //optional
        mpgReq.send();
        try
    }
}
```

```

{
Receipt receipt = mpgReq.getReceipt();
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = " + receipt.getBankTotals());
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedout());
System.out.println("IsVisaDebit = " + receipt.getIsVisaDebit());
System.out.println("HostId = " + receipt.getHostId());
System.out.println("IssuerId = " + receipt.getIssuerId());
System.out.println("SourcePanLast4 = " + receipt.getSourcePanLast4());
System.out.println("Advice Code= "+ receipt.getAdviceCode());

// InstallmentResults installmentResults = receipt.getInstallmentResults();
// System.out.println("\nPlanId = " + installmentResults.getPlanId() +"\n");
// System.out.println("PlanIDRef = " + installmentResults.getPlanIDRef());
// System.out.println("TacVersion = " + installmentResults.getTacVersion());
// System.out.println("PlanAcceptanceId = " + installmentResults.getPlanAcceptanceId());
// System.out.println("PlanStatus = " + installmentResults.getPlanStatus());
// System.out.println("PlanResponse = " + installmentResults.getPlanResponse());
}
catch (Exception e)
{
e.printStackTrace();
}
}
}

```

## 2.2 Préautorisation

Une transaction de préautorisation vérifie que les fonds sont présents sur la carte de crédit du client et les bloque. Les fonds sont bloqués pour une durée prédéterminée qui varie en fonction de l'émetteur de carte.

Pour récupérer les fonds bloqués par une transaction de préautorisation et les déposer dans le compte du commerçant, une transaction de conclusion de préautorisation doit être effectuée. On ne peut conclure une préautorisation qu'une seule fois.

### Éléments dont il faut tenir compte :

Si une transaction de préautorisation n'est pas suivie d'une transaction de conclusion de préautorisation, la préautorisation doit être annulée par l'entremise d'une transaction de conclusion de préautorisation d'une valeur de 0,00 \$. Consultez la section 2.3Préautorisation incrémentale Augmente le montant des fonds bloqués dans une préautorisation existante en vue d'un règlement ultérieur par une seule conclusion de préautorisation. Il n'y a pas de limite au nombre de transactions de préautorisation incrémentale sur l'autorisation initiale estimative et chaque nouvelle préautorisation

```

PreAuth preauth = new PreAuth();

HttpsPostRequest mpgReq = new HttpsPostRequest();

mpgReq.setTransaction(preauth);

```

## Principaux champs d'objet de connexion (toutes les transactions API)

Variable	Type et limites	
ID de commerce	<i>Chaîne</i> S. O.	mpgReq.setstoreId(store_id);
Jeton API	<i>Chaîne</i> S. O.	mpgReq.setApiToken(api_token);

## Champs de demande pour les transactions de préautorisation (obligatoires)

Variable	Type et limites	
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques a-Z A-Z 0-9 _ - : . @ espaces	preauth.setorderId(order_id);
Montant	<i>Chaîne</i> 10 caractères décimaux Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal	preauth.setAmount(amount);
	<b>EXAMPLE : 1 234 567,89</b>	
Numéro de carte de crédit	<i>Chaîne</i> Maximum 20 caractères alphanumériques	preauth.setPan(pan);
Date d'expiration	<i>Chaîne</i> 4 caractères alphanumériques AAMM	preauth.setExpDate(expiry_date);
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	preauth.setCryptType(crypt);

## Champs de demande pour les transactions de préautorisation ( facultatifs)

Variable	Type et limites
Descripteur dynamique	<p><i>Chaîne</i> 20 caractères alphanumériques Total de 22 caractères incluant votre nom de commerçant et un séparateur</p> <p><b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : &lt;&gt; \$ % = ? ^ { } [ ] \</p>
<b>Pour les transactions de préautorisation</b> <b>REMARQUE :</b> La valeur du champ Dynamic descriptor est uniquement transférée à une conclusion de préautorisation lorsque cette dernière est effectuée par l'entremise du centre de ressources pour commerçants. Autrement, la valeur du champ doit être renvoyée durant la conclusion de préautorisation.	

is incremental is_incremental	<p><i>Valeur booléenne</i> true/false</p> <p>Indique si cette préautorisation utilise un montant estimé. Les estimations permettent d'incrémenter le montant retenu au moyen de demandes incrementalAuth subséquentes. La valeur par défaut est « false ».</p> <p><b>{b}REMARQUE :</b> veuillez noter que si ce champ contient la valeur « true », la préautorisation n'est valable que pour une seule conclusion de préautorisation. Toute conclusion soumise à titre de conclusion partielle est traitée comme une conclusion complète (ship_indicator= P est traité comme ship_indicator= F lorsque, dans la préautorisation originale, le paramètre is_incremental= true).</p>

Indicateur étranger	<i>Valeur booléenne true/false</i>	<code>preauth.setForeignIndicator( foreign_indicator);</code>
ID de correspondance de carte	<i>Chaîne</i>  50 caractères alphanumériques	<code>preauth.setCmId(transaction_id );</code>
<b>REMARQUE :</b> Applicables à Offlinx <sup>MC</sup> seulement, chaque transaction doit avoir une valeur unique		
Renseignements du client	<i>Objet</i>  S. O.	<code>preauth.setCustInfo(customer);</code>
Renseignements du SVA	<i>Objet</i>  S. O.	<code>preauth.setAvsInfo(avsCheck);</code>
Renseignements du NVC	<i>Objet</i>  <b>REMARQUE :</b> Lorsque les renseignements d'identification sont stockés lors de la première transaction, l'objet CVD (numéro de vérification de carte, ou NVC) doit être envoyé. Pour les transactions subséquentes utilisant les renseignements d'identification stockés, le NVC peut être envoyé avec les transactions entamées par le titulaire de carte seulement. <b>Les commerçants ne doivent pas enregistrer le NVC.</b>	<code>preauth.setCvdInfo(cvdCheck);</code>  S. O.
ID de client	<i>Chaîne</i>  50 caractères alphanumériques	<code>preauth.setCustId(cust_id);</code>
<b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : <> \$ % = ? ^ { } [ ] \		
Indicateur de portefeuille électronique	<i>Chaîne</i>  3 caractères alphanumériques	<code>preauth.setWalletIndicator(wallet_indicator);</code>

<b>REMARQUE :</b> Pour les transactions d'achat et de préautorisation de base, l'indicateur de portefeuille s'applique uniquement à Visa Checkout et à Masterpass de Mastercard. Pour plus de renseignements, consultez l'annexe A Définition des champs de demande.		
Autorisation finale	<i>Chaîne</i>	<code>preauth.setFinalAuth("true");</code>
<b>REMARQUE :</b> Applicable uniquement aux transactions par carte Mastercard	true/false	
Renseignements d'identification au dossier  cof	<i>Objet</i>  S. O.	<code>cof.setCofInfo(cof);</code>
<b>REMARQUE :</b> Cette variable est un objet imbriqué au sein d'une transaction et elle est requise lors du stockage des renseignements d'identification d'un client ou lors de l'utilisation de tels renseignements stockés. Pour plus de renseignements sur les champs de l'objet Credential on File Info, consultez la section Objet Credential File Info et variables.		

## Champs de demande pour l'objet Credential on File Info

Variable	Type et limites	
ID de l'émetteur	<i>Chaîne</i>  15 caractères alphanumériques  Longueur variable	<code>cof.setIssuerId("VALUE_FOR_ISSUER_ID");</code>  <b>REMARQUE :</b> Pour obtenir une liste et une explication des valeurs possibles à envoyer pour cette variable, consultez la section Définition des champs de demande – Renseignements d'identification au dossier (cof)
Indicateur de paiement	<i>Chaîne</i>	<code>cof.setPaymentIndicator("PAYMENT_INDICATOR_VALUE");</code>

<p><b>REMARQUE :</b> Lors d'une transaction utilisant les renseignements d'identification au dossier pour laquelle le champ de demande e-commerce indicator est également inclus, les valeurs acceptables de ce champ dépendent de la valeur envoyée pour l'indicateur de paiement. Pour plus d'information, consultez les <a href="#">définitions des champs de réponse</a></p>	<p>1 caractère alphabétique</p>	<p><b>REMARQUE :</b> Pour obtenir une liste et une explication des valeurs possibles à envoyer pour cette variable, consultez la section Définition des champs de demande – Renseignements d'identification au dossier (cof)</p>
<p>Information de paiement</p>	<p><i>Chaîne</i> 1 caractère numérique</p>	<pre>cof.setPaymentInformation ("PAYMENT_INFO_VALUE");</pre> <p><b>REMARQUE :</b> Pour obtenir une liste et une explication des valeurs possibles à envoyer pour cette variable, consultez la section Définition des champs de demande – Renseignements d'identification au dossier (cof)</p>

### Exemple de transaction de préautorisation

```
package Canada;
import JavaAPI.*;
public class TestCanadaPreauth
{
    public static void main(String[] args)
    {
        String store_id = "store5";
        String api_token = "yesguy";
        java.util.Date createDate = new java.util.Date();
        String order_id = "Test"+createDate.getTime();
        String amount = "5.00";
        String pan = "4242424242424242";
        String expdate = "1902";
        String crypt = "7";
        String processing_country_code = "CA";
        boolean foreign_indicator= true;
        boolean status_check = false;
        boolean is_incremental = true;

        PreAuth preauth = new PreAuth();
        preauth.setOrderId(order_id);
        preauth.setAmount(amount);
        preauth.setPan(pan);
        preauth.setExpdate(expdate);
        preauth.setCryptType(crypt);
        preauth.setForeignIndicator(foreign_indicator);
        //preauth.setWalletIndicator(""); //Refer documentation for possible values
        //preauth.setCmId("8nAK8712sGaAkls56"); //set only for usage with Offlinx - Unique max 50 alphanumeric characters transaction id generated by merchant
        //preauth.setFinalAuth("true");

        // TrId and TokenCryptogram are optional, refer documentation for more details.
        preauth.setTrId("50189815682");
        preauth.setTokenCryptogram("APmbM/411e0uAAH+s6xMAAADFA==");

        //optional - Installment Info
        InstallmentInfo installmentInfo = new InstallmentInfo();
        installmentInfo.setPlanId("ae859ef1-eb91-b708-8b80-1dd481746401");
        installmentInfo.setPlanIdRef("0000000065");
        installmentInfo.setTacVersion("2");
        //preauth.setInstallmentInfo(installmentInfo);

        //optional - Credential on File details
        CofInfo cof = new CofInfo();
        cof.setPaymentIndicator("U");
        cof.setPaymentInformation("2");
    }
}
```

```

cof.setIssuerId("139X3130ASCXAS9");

preauth.setCofInfo(cof);

HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false for production transactions
mpgReq.setStoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(preauth);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = " + receipt.getBankTotals());
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("IsVisaDebit = " + receipt.getIsVisaDebit());
//System.out.println("StatusCode = " + receipt.getStatusCode());
//System.out.println("StatusMessage = " + receipt.getStatusMessage());
System.out.println("IssuerId = " + receipt.getIssuerId());
System.out.println("SourcePanLast4 = " + receipt.getSourcePanLast4());
System.out.println("Advice Code= " + receipt.getAdviceCode());

InstallmentResults installmentResults = receipt.getInstallmentResults();
System.out.println("\nPlanId = " + installmentResults.getPlanId() +"\n");
System.out.println("PlanIDRef = " + installmentResults.getPlanIDRef());
System.out.println("TacVersion = " + installmentResults.getTacVersion());
System.out.println("PlanAcceptanceId = " + installmentResults.getPlanAcceptanceId());
System.out.println("PlanStatus = " + installmentResults.getPlanStatus());
System.out.println("PlanResponse = " + installmentResults.getPlanResponse());
}

}
catch (Exception e)
{
e.printStackTrace();
}
}
}

```

## 2.3 Préautorisation incrémentale

Augmente le montant des fonds bloqués dans une préautorisation existante en vue d'un règlement ultérieur par une seule conclusion de préautorisation. Il n'y a pas de limite au nombre de transactions de préautorisation incrémentale sur l'autorisation initiale estimative et chaque nouvelle préautorisation incrémentale augmente la retenue sur la carte de crédit du client.

Les préautorisations incrémentales requièrent un montant estimé dans la préautorisation initiale. Il est défini en paramétrant le champ <is\_incremental> à « true ».

Pour Mastercard uniquement, une préautorisation incrémentale peut être soumise avec un montant d'une valeur de 0 \$ afin de demander une prolongation du délai admissible pour la conclusion de la préautorisation (par exemple, 30 jours).

Pour plus de détails sur l'utilisation des montants estimés dans les préautorisations et sur l'utilisation des préautorisations incrémentales pour augmenter le montant des fonds bloqués, voir 1 Règles d'autorisation incrémentale.

## Définition de l'objet de transaction Incremental Pre-Authorization

```
IncrementalPreauth incrementalPreauth = new IncrementalPreauth();
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setTransaction(incrementalPreauth);
```

## Principaux champs d'objet de connexion (toutes les transactions API)

Variable	Type et limites	Description
store_id	<i>Chaîne</i> S.O.	mpgReq.setstoreId(store_id); ID de commerce
API token	<i>Chaîne</i> S.O.	mpgReq.setApiToken(api_token); Jeton API

## Champs de la demande de transaction Incremental Preauthorization – obligatoires

Variable	Type et limites	
order ID	<i>Chaîne</i> 50 caractères alphanumériques a-Z A-Z 0-9 _ - : . @ espaces	incrementalPreauth.setOrderId(order_id);
amount	<i>Chaîne</i> 10 caractères décimaux Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (cents) après le point	incrementalPreauth.setAmount(amount);
	{b}EXEMPLE: {/b}1234567.89	
transaction number	<i>Chaîne</i> 255 caractères (alphanumériques, trait d'union ou trait de soulignement) Longueur variable	incrementalPreauth.setTxnNumber(txn_number);

## Exemple de préautorisation incrémentale

```
package Canada;
import JavaAPI.*;
public class TestCanadaIncrementalPreauth
{
    public static void main(String[] args)
    {
        String store_id = "store5";
        String api_token = "yesguy";
        String order_id = "Test1683565649745";
        String amount = "8.00";
        String txn_number = "85401-0_473";
        String processing_country_code = "CA";
        IncrementalPreauth incrementalPreauth = new IncrementalPreauth();
        incrementalPreauth.setOrderId(order_id);
        incrementalPreauth.setAmount(amount);
        incrementalPreauth.setTxnNumber(txn_number);
        HttpsPostRequest mpgReq = new HttpsPostRequest();
        mpgReq.setProcCountryCode(processing_country_code);
        mpgReq.setTestMode(true); //false or comment out this line for production transactions
        mpgReq.setstoreId(store_id);
        mpgReq.setApiToken(api_token);
        mpgReq.setTransaction(incrementalPreauth);
        mpgReq.send();
        try
        {
            Receipt receipt = mpgReq.getReceipt();
            System.out.println("CardType = " + receipt.getCardType());
            System.out.println("TransAmount = " + receipt.getTransAmount());
            System.out.println("TxnNumber = " + receipt.getTxnNumber());
            System.out.println("ReceiptId = " + receipt.getReceiptId());
            System.out.println("TransType = " + receipt.getTransType());
            System.out.println("ReferenceNum = " + receipt.getReferenceNum());
            System.out.println("ResponseCode = " + receipt.getResponseCode());
            System.out.println("ISO = " + receipt.getISO());
            System.out.println("BankTotals = " + receipt.getBankTotals());
            System.out.println("Message = " + receipt.getMessage());
            System.out.println("AuthCode = " + receipt.getAuthCode());
            System.out.println("Complete = " + receipt.getComplete());
            System.out.println("TransDate = " + receipt.getTransDate());
            System.out.println("TransTime = " + receipt.getTransTime());
            System.out.println("Ticket = " + receipt.getTicket());
            System.out.println("TimedOut = " + receipt.getTimedOut());
            System.out.println("IsVisaDebit = " + receipt.getIsVisaDebit());
            System.out.println("SourcePanLast4 = " + receipt.getSourcePanLast4());
            // InstallmentResults installmentResults = receipt.getInstallmentResults();
            // System.out.println("\nPlanId = " + installmentResults.getPlanId() +"\n");
            // System.out.println("PlanIDRef = " + installmentResults.getPlanIDRef());
            // System.out.println("TacVersion = " + installmentResults.getTacVersion());
            // System.out.println("PlanAcceptanceId = " +
            installmentResults.getPlanAcceptanceId());
            // System.out.println("PlanStatus = " + installmentResults.getPlanStatus());
            // System.out.println("PlanResponse = " + installmentResults.getPlanResponse());
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

## 2.4 Conclusion de préautorisation

Une conclusion de préautorisation récupère les fonds bloqués par une transaction de préautorisation et les prépare à être déposés dans le compte du commerçant.

### Éléments dont il faut tenir compte :

On ne peut conclure une préautorisation qu'une seule fois.

- Pour annuler entièrement une transaction de préautorisation, effectuez une conclusion de préautorisation d'une valeur de 0,00

- Pour traiter cette transaction vous avez besoin de l'ID de commande (order ID) et du numéro de transaction (transaction number) de la transaction de préautorisation d'origine.

```
Completion completion = new Completion();
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setTransaction(completion);
```

### Principaux champs d'objet de connexion (toutes les transactions API)

Variable	Type et limites	
ID de commerce	<i>Chaîne</i> S. O.	<code>mpgReq.setstoreId(store_id);</code> Identifiant unique fourni par Moneris lors de la configuration du compte de commerçant
Jeton API	<i>Chaîne</i> S. O.	<code>mpgReq.setApiToken(api_token);</code>

Variable	Type et limites
	<p>Chaîne de caractères alphanumériques unique créée par Moneris lors de l'activation du compte de commerçant Pour obtenir votre jeton API, accédez aux paramètres Admin de l'environnement de test ou de production de votre magasin dans le Centre de ressources pour commerçants, qui se trouve aux liens URL qui suivent :</p> <p>Test :  <a href="https://esqa.moneris.com/mpg/?chlang=fr">https://esqa.moneris.com/mpg/?chlang=fr</a></p> <p>Production :  <a href="https://www3.moneris.com/mpg/?chlang=fr">https://www3.moneris.com/mpg/?chlang=fr</a></p>

### Champs de demande pour les transactions de conclusion de préautorisation (obligatoires)

Variable	Type et limites
ID de commande	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p> <p>a-Z A-Z 0-9 _ - : . @ espaces</p>
Montant de conclusion	<p><i>Chaîne</i></p> <p>10 caractères décimaux</p> <p>Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal</p> <p><b>EXEMPLE : 1 234 567,89</b></p>
Numéro de transaction	<p><i>Chaîne</i></p> <p>255 caractères</p>

Variable	Type et limites	
	(alphanumériques, trait d'union ou trait de soulignement)  Longueur variable	
Indicateur de commerce électronique	<i>Chaîne</i>  1 caractère alphanumérique	completion.setCryptType(crypt);

**Champs de demande pour les transactions de conclusion de préautorisation (facultatifs)**

Variable	Type et limites	Description
ID de client	<i>Chaîne</i>  50 caractères alphanumériques  <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\	completion.setCustId(cust_id);
Descripteur dynamique	<i>Chaîne</i>  20 caractères alphanumériques  Total de 22 caractères incluant votre nom de commerçant et un séparateur  <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\	completion.setDynamicDescriptor(dynamic_descriptor);
Indicateur d'expédition	<i>Chaîne</i>  1 caractère alphanumérique	completion.setShipIndicator(ship_indicator);

## Exemple d'une transaction de conclusion de préautorisation

```
package Canada;
import JavaAPI.*;
public class TestCanadaPreauth
{
    public static void main(String[] args)
    {
        String store_id = "store5";
        String api_token = "yesguy";
        java.util.Date createDate = new java.util.Date();
        String order_id = "Test"+createDate.getTime();
        String amount = "5.00";
        String pan = "4242424242424242";
        String expdate = "1902";
        String crypt = "7";
        String processing_country_code = "CA";
        boolean status_check = false;
        PreAuth preauth = new PreAuth();
        preauth.setOrderId(order_id);
        preauth.setAmount(amount);
        preauth.setPan(pan);
        preauth.setExpdate(expdate);
        preauth.setCryptType(crypt);
        //preauth.setWalletIndicator(""); //Refer documentation for possible values
        //preauth.setCmId("8nAK8712sGaAkls56"); //set only for usage with Offlinx - Unique max 50
        alphanumeric characters transaction id generated by merchant
        //optional - Credential on File details
        CofInfo cof = new CofInfo();
        cof.setPaymentIndicator("U");
        cof.setPaymentInformation("2");
        cof.setIssuerId("139X3130ASCXAS9");

        preauth.setCofInfo(cof);

        HttpsPostRequest mpgReq = new HttpsPostRequest();
        mpgReq.setProcCountryCode(processing_country_code);
        mpgReq.setTestMode(true); //false for production transactions
        mpgReq.setstoreId(store_id);
        mpgReq.setApiToken(api_token);
        mpgReq.setTransaction(preauth);
        mpgReq.setStatusCheck(status_check);
        mpgReq.send();
        try
        {
            Receipt receipt = mpgReq.getReceipt();
            System.out.println("CardType = " + receipt.getCardType());
            System.out.println("TransAmount = " + receipt.getTransAmount());
            System.out.println("TxnNumber = " + receipt.getTxnNumber());
            System.out.println("ReceiptId = " + receipt.getReceiptId());
            System.out.println("TransType = " + receipt.getTransType());
            System.out.println("ReferenceNum = " + receipt.getReferenceNum());
            System.out.println("ResponseCode = " + receipt.getResponseCode());
            System.out.println("ISO = " + receipt.getISO());
            System.out.println("BankTotals = " + receipt.getBankTotals());
            System.out.println("Message = " + receipt.getMessage());
            System.out.println("AuthCode = " + receipt.getAuthCode());
            System.out.println("Complete = " + receipt.getComplete());
            System.out.println("TransDate = " + receipt.getTransDate());
            System.out.println("TransTime = " + receipt.getTransTime());
            System.out.println("Ticket = " + receipt.getTicket());
            System.out.println("TimedOut = " + receipt.getTimedOut());
            System.out.println("IsVisaDebit = " + receipt.getIsVisaDebit());
            //System.out.println("StatusCode = " + receipt.getStatusCode());
            //System.out.println("StatusMessage = " + receipt.getStatusMessage());
            System.out.println("IssuerId = " + receipt.getIssuerId());

            InstallmentResults installmentResults = receipt.getInstallmentResults();
            System.out.println("\nPlanId = " + installmentResults.getPlanId() +"\n");
            System.out.println("PlanIDRef = " + installmentResults.getPlanIDRef());
            System.out.println("TacVersion = " + installmentResults.getTacVersion());
            System.out.println("PlanAcceptanceId = " + installmentResults.getPlanAcceptanceId());
            System.out.println("PlanStatus = " + installmentResults.getPlanStatus());
            System.out.println("PlanResponse = " + installmentResults.getPlanResponse());
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

}

## 2.5 Transaction forcée

Une transaction forcée récupère les fonds bloqués et les prépare à être déposés dans le compte du commerçant.

Un commerçant effectue cette transaction lorsqu'il obtient le numéro d'autorisation directement de l'émetteur par l'entremise d'une autorisation tierce (p. ex. au téléphone).

### Éléments dont il faut tenir compte :

Cette transaction est une conclusion indépendante utilisée lorsque la transaction de préautorisation d'origine n'a pas été traitée par le même compte de commerçant de Passerelle Moneris.

Vous pouvez donc conclure des transactions de préautorisation n'ayant pas été traitée par l'entremise de Passerelle Moneris. Cependant, vous avez besoin d'un numéro de carte de crédit, d'une date d'expiration et du numéro d'autorisation original.

Les transactions traitées par carte UnionPay ne peuvent pas être forcées.

```
ForcePost forcepost = new ForcePost();
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setTransaction(forcepost);
```

### Principaux champs d'objet de connexion (toutes les transactions API)

Variable	Type et limites	
ID de commerce	<i>Chaîne</i> S. O.	mpgReq.setstoreId(store_id);
Jeton API	<i>Chaîne</i> S. O.	mpgReq.setApiToken(api_token);

### Champs de demande pour les transactions forcées (obligatoires)

Variable	Type et limites	
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques	forcepost.setorderId(order_id);

Variable	Type et limites	
	a-Z A-Z 0-9 _ - : . @ espaces	
Montant	<p><i>Chaîne</i></p> <p>10 caractères décimaux</p> <p>Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal</p> <p><b>EXEMPLE : 1 234 567,89</b></p>	forcepost.setAmount(amount);
Numéro de carte de crédit	<p><i>Chaîne</i></p> <p>Maximum 20 caractères alphanumériques</p>	forcepost.setPan(pan);
Date d'expiration	<p><i>Chaîne</i></p> <p>4 caractères alphanumériques</p> <p>AAMM</p>	forcepost.setExpDate(expiry_date);
Code d'autorisation	<p><i>Chaîne</i></p> <p>8 caractères alphanumériques</p>	forcepost.setAuthCode(auth_code);
Indicateur de commerce électronique	<p><i>Chaîne</i></p> <p>1 caractère alphanumérique</p>	forcepost.setCryptType(crypt);

### Champs de demande pour les transactions forcées (facultatifs)

Variable	Type et limites	
ID de client	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p>	forcepost.setCustId(cust_id);

Variable	Type et limites	
	<p><b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : &lt;&gt;\$ % = ? ^{ } [ ] \</p>	
Descripteur dynamique	<p><i>Chaîne</i></p> <p>20 caractères alphanumériques</p> <p>Total de 22 caractères incluant votre nom de commerçant et un séparateur</p> <p><b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : &lt;&gt;\$ % = ? ^{ } [ ] \</p>	forcepost.setDynamicDescriptor(dynamic_descriptor);

### Exemple de transaction forcée

```

package Canada;
import JavaAPI.*;
public class TestCanadaForcePost
{
    public static void main(String[] args)
    {
        java.util.Date createDate = new java.util.Date();
        String order_id = "Test"+createDate.getTime();
        String cust_id = "my customer id";
        String store_id = "moneris";
        String api_token = "hurgle";
        String amount = "1.00";
        String pan = "4242424242424242";
        String expdate = "1901"; //YYMM format
        String auth_code = "88864";
        String crypt = "7";
        String dynamic_descriptor = "my descriptor";
        String processing_country_code = "CA";
        boolean status_check = false;
        ForcePost forcepost = new ForcePost();
        forcepost.setOrderId(order_id);
        forcepost.setCustomerId(cust_id);
        forcepost.setAmount(amount);
        forcepost.setPan(pan);
        forcepost.setExpdate(expdate);
        forcepost.setAuthCode(auth_code);
        forcepost.setCryptType(crypt);
        forcepost.setDynamicDescriptor(dynamic_descriptor);

        HttpsPostRequest mpgReq = new HttpsPostRequest();
        mpgReq.setProcCountryCode(processing_country_code);
        mpgReq.setTestMode(true); //false for production transactions
        mpgReq.setstoreId(store_id);
        mpgReq.setApiToken(api_token);
        mpgReq.setTransaction(forcepost);
        mpgReq.setStatusCheck(status_check);
        mpgReq.send();
        try
        {
    
```

```

Receipt receipt = mpgReq.getReceipt();
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = " + receipt.getBankTotals());
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("CorporateCard = " + receipt.getCorporateCard());
//System.out.println("MessageId = " + receipt.getMessageId());
System.out.println("IssuerId = " + receipt.getIssuerId());
}
catch (Exception e)
{
e.printStackTrace();
}
}
}

```

## 2.6 Correction d'achat

Une correction d'achat rembourse l'entièreté du montant d'un achat, d'une conclusion de préautorisation ou d'une transaction forcée sur la carte d'un titulaire de carte et supprime toute mention de la transaction du relevé du titulaire de carte.

Cette transaction peut être effectuée à la suite d'un achat ou d'une conclusion de préautorisation traitée la même journée, à condition que le lot contenant la transaction original soit toujours ouvert.

### Éléments dont il faut tenir compte :

Pour traiter cette transaction, vous avez besoin de l'ID de commande (order ID) et du numéro de transaction (transaction number) de la transaction de conclusion, d'achat ou forcée d'origine.

```

PurchaseCorrection purchasecorrection = new PurchaseCorrection();

HttpsPostRequest mpgReq = new HttpsPostRequest();

mpgReq.setTransaction(purchasecorrection);

```

### Principaux champs d'objet de connexion (toutes les transactions API)

Variable	Type et limites	
ID de commerce	<i>Chaîne</i> S. O.	mpgReq.setstoreId(store_id);
Jeton API	<i>Chaîne</i> S. O.	mpgReq.setApiToken(api_token);

## Champs de demande liés aux transactions de correction d'achat (obligatoires)

Variable	Type et limites	
ID de commande	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p> <p>a-Z A-Z 0-9 _ - : . @ espaces</p>	<code>purchasecorrection.setOrderId(order_id);</code>
Numéro de transaction	<p><i>Chaîne</i></p> <p>255 caractères (alphanumériques, trait d'union ou trait de soulignement)</p> <p>Longueur variable</p>	<code>purchasecorrection.setTxnNumber(txn_number);</code>
Indicateur de commerce électronique	<p><i>Chaîne</i></p> <p>1 caractère alphanumérique</p>	<code>purchasecorrection.setCryptType(crypt);</code>

## Champs de demande liés aux transactions de correction d'achat (facultatifs)

Variable	Type et limites	
ID de client	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p> <p><b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : &lt;&gt; \$ % = ? ^ { } [ ] \</p>	<code>purchasecorrection.setCustId(cust_id);</code>
Descripteur dynamique	<p><i>Chaîne</i></p> <p>20 caractères alphanumériques</p> <p>Total de 22 caractères incluant votre nom de commerçant et un séparateur</p> <p><b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : &lt;&gt; \$ % = ? ^ { } [ ] \</p>	<code>purchasecorrection.setCustId(cust_id);</code>

## Exemple de transaction de correction d'achat

```
package Canada;
import JavaAPI.*;
public class TestCanadaRefund
{
    public static void main(String[] args)
    {
        String store_id = "store1";
        String api_token = "yesguy";
        String amount = "1.00";
        String crypt = "7";
        String dynamic_descriptor = "123456";
        String custid = "mycust9";
        String order_id = "mvt2713618548";
        String txn_number = "911464-0_10";
        String processing_country_code = "CA";
        boolean status_check = false;
        Refund refund = new Refund();
        refund.setTxnNumber(txn_number);
        refund.setOrderId(order_id);
        refund.setAmount(amount);
        refund.setCryptType(crypt);
        refund.setCustId(custid);
        refund.setDynamicDescriptor(dynamic_descriptor);

        //Optional
        PBBInfo pbbInfo= new PBBInfo();
        String consentId="1b5ee10a-5356-4a71-b2cf-874ab134661f";
        String lifeCycleTraceId="A1x7ecRv1YSTEAx";
        String channel="DESKTOP_WEB";
        pbbInfo.setConsentId(consentId);
        pbbInfo.setLifeCycleTraceId(lifeCycleTraceId);
        pbbInfo.setChannel(channel);
        refund.setPbbInfo(pbbInfo);

        HttpsPostRequest mpgReq = new HttpsPostRequest();
        mpgReq.setProcCountryCode(processing_country_code);
        mpgReq.setTestMode(true); //false for production transactions
        mpgReq.setStoreId(store_id);
        mpgReq.setApiToken(api_token);
        mpgReq.setTransaction(refund);
        mpgReq.setStatusCheck(status_check);
        mpgReq.send();
        try
        {
            Receipt receipt = mpgReq.getReceipt();
            System.out.println("CardType = " + receipt.getCardType());
            System.out.println("TransAmount = " + receipt.getTransAmount());
            System.out.println("TxnNumber = " + receipt.getTxnNumber());
            System.out.println("ReceiptId = " + receipt.getReceiptId());
            System.out.println("TransType = " + receipt.getTransType());
            System.out.println("ReferenceNum = " + receipt.getReferenceNum());
            System.out.println("ResponseCode = " + receipt.getResponseCode());
            System.out.println("ISO = " + receipt.getISO());
            System.out.println("BankTotals = " + receipt.getBankTotals());
            System.out.println("Message = " + receipt.getMessage());
            System.out.println("AuthCode = " + receipt.getAuthCode());
            System.out.println("Complete = " + receipt.getComplete());
            System.out.println("TransDate = " + receipt.getTransDate());
            System.out.println("TransTime = " + receipt.getTransTime());
            System.out.println("Ticket = " + receipt.getTicket());
            System.out.println("TimedOut = " + receipt.getTimedOut());

            InstallmentResults installmentResults = receipt.getInstallmentResults();
            System.out.println("\nPlanId = " + installmentResults.getPlanId() +"\n");
            System.out.println("PlanIDRef = " + installmentResults.getPlanIDRef());
            System.out.println("TacVersion = " + installmentResults.getTacVersion());
            System.out.println("PlanAcceptanceId = " + installmentResults.getPlanAcceptanceId());
            System.out.println("PlanStatus = " + installmentResults.getPlanStatus());
            System.out.println("PlanResponse = " + installmentResults.getPlanResponse());
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

## 2.7 Remboursement

Un remboursement rembourse entièrement ou en partie les fonds obtenus à la suite d'un achat, d'une conclusion de préautorisation ou d'une transaction forcée et les renvoie sur la carte du titulaire de carte.

Contrairement à une correction d'achat, la transaction initiale et le remboursement apparaîtront tous les deux sur le relevé du titulaire de carte.

Si les fonds doivent être remis sur une carte différente de celle utilisée lors de la transaction d'origine, une transaction de remboursement indépendant doit plutôt être effectué.

Pour traiter cette transaction, vous avez besoin de l'ID de commande et du numéro de transaction de la conclusion, de l'achat ou de la transaction forcée d'origine.

```
Refund refund = new Refund();  
  
HttpsPostRequest mpgReq = new HttpsPostRequest();  
  
mpgReq.SetTransaction(refund);
```

### Principaux champs d'objet de connexion (toutes les transactions API)

Variable	Type et limites	
ID de commerce	<i>Chaîne</i> S. O.	mpgReq.setstoreId(store_id);
Jeton API	<i>Chaîne</i> S. O.	mpgReq.setApiToken(api_token);

### Champs de demande liés aux transactions de remboursement (obligatoires)

Variable	Type et limites	
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques a-Z A-Z 0-9 _ - : . @ espaces	refund.setorderId(order_id);

Montant	<i>Chaîne</i>	refund.setAmount(amount);
	10 caractères décimaux Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal	
	<b>EXEMPLE : 1 234 567,89</b>	
Numéro de transaction	<i>Chaîne</i>  255 caractères (alphanumériques, trait d'union ou trait de soulignement)  Longueur variable	refund.setTxnNumber(txn_number);
Indicateur de commerce électronique	<i>Chaîne</i>  1 caractère alphanumérique	refund.setCryptType(crypt);

#### Champs de demande de transaction Pre-Authorization Completion (facultatifs)

Variable	Type et limites	Méthode de configuration
Pay By Bank Info  PBB_info	<i>Objet</i>  S.O.	Nécessaire afin d'effectuer une transaction de remboursement pour un paiement par compte (transaction ABP) avec l'application Pay By Bank. Utilisez Get Transaction Data pour obtenir le cryptogramme. Crée un lien entre le consentement de l'application PBB et cette transaction.  <div style="border: 1px solid black; padding: 5px; background-color: #e0f2e0;"> <b>{b}REMARQUE :{/b}interne uniquement. Cet objet est lié au FID 7P « Pay By Bank Elements » dans Host Messaging.         </b></div>

#### Exemple d'une transaction de remboursement

```
package Canada;
import JavaAPI.*;
public class TestCanadaRefund
{
    public static void main(String[] args)
    {
        String store_id = "store1";
```

```

String api_token = "yesguy";
String amount = "1.00";
String crypt = "7";
String dynamic_descriptor = "123456";
String custid = "mycust9";
String order_id = "mvt2713618548";
String txn_number = "911464-0_10";
String processing_country_code = "CA";
boolean status_check = false;
Refund refund = new Refund();
refund.setTxnNumber(txn_number);
refund.setOrderId(order_id);
refund.setAmount(amount);
refund.setCryptType(crypt);
refund.setCustId(custid);
refund.setDynamicDescriptor(dynamic_descriptor);
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false for production transactions
mpgReq.setStoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(refund);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = " + receipt.getBankTotals());
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedOut());

InstallmentResults installmentResults = receipt.getInstallmentResults();
System.out.println("\nPlanId = " + installmentResults.getPlanId() +"\n");
System.out.println("PlanIDRef = " + installmentResults.getPlanIDRef());
System.out.println("TacVersion = " + installmentResults.getTacVersion());
System.out.println("PlanAcceptanceId = " + installmentResults.getPlanAcceptanceId());
System.out.println("PlanStatus = " + installmentResults.getPlanStatus());
System.out.println("PlanResponse = " + installmentResults.getPlanResponse());
}
catch (Exception e)
{
e.printStackTrace();
}
}
}

```

## 2.8 Remboursement indépendant

Une transaction de remboursement indépendant crédite un montant précis à la carte de crédit du titulaire de carte. Le numéro de la carte de crédit et la date d'expiration sont requis.

Vous pouvez donc rembourser des transactions n'ayant pas été traitées par l'entremise de Passerelle Moneris.

### **Éléments dont il faut tenir compte :**

En raison du potentiel de fraude, ce ne sont pas tous les comptes qui peuvent traiter ces transactions par défaut. Si vous avez besoin de ce type de transactions dans votre entreprise, vous devez en faire la demande auprès de votre gestionnaire de compte.

**REMARQUE :** Les versements ne sont pas pris en charge par les transactions de remboursement indépendant.

```
IndependentRefund indrefund = new IndependentRefund();  
HttpsPostRequest mpgReq = new HttpsPostRequest();  
mpgReq.setTransaction(indrefund);
```

### **Principaux champs d'objet de connexion (toutes les transactions API)**

Variable	Type et limites	
ID de commerce	<i>Chaîne</i> S. O.	mpgReq.setstoreId(store_id);

Variable	Type et limites	
Jeton API	<i>Chaîne</i> S. O.	mpgReq.setApiToken(api_token);

**Champs de demande liés aux transactions de remboursement indépendant (obligatoires)**

Variable	Type et limites	
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques a-Z A-Z 0-9 _ - : . @ espaces	indrefund.setOrderId(order_id);
Montant	<i>Chaîne</i> 10 caractères décimaux Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal  <b>EXEMPLE : 1 234 567,89</b>	indrefund.setAmount(amount);
Numéro de carte de crédit	<i>Chaîne</i> Maximum 20 caractères alphanumériques	indrefund.setPan(pan);
Date d'expiration	<i>Chaîne</i> 4 caractères alphanumériques AAMM	indrefund.setExpDate(expiry_date);
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	indrefund.setCryptType(crypt);

## Champs de demande liés aux transactions de remboursement indépendant (facultatifs)

Variable	Type et limites	
ID de client	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p> <div style="background-color: #e0f2e0; padding: 5px;"> <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : &lt;&gt;\$%=?^{}[]\ </div>	indrefund.setCustomerId(cust_id);
Descripteur dynamique	<p><i>Chaîne</i></p> <p>20 caractères alphanumériques</p> <p>Total de 22 caractères incluant votre nom de commerçant et un séparateur</p> <div style="background-color: #e0f2e0; padding: 5px;"> <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : &lt;&gt;\$%=?^{}[]\ </div>	indrefund.setDynamicDescriptor(dynamic_descriptor);

### Exemple d'une transaction de remboursement indépendant

```

package Canada;
import JavaAPI.*;
public class TestCanadaIndependentRefund
{
public static void main(String[] args)
{
java.util.Date createDate = new java.util.Date();
String order_id = "Test"+createDate.getTime();
String store_id = "store5";
String api_token = "yesguy";
String cust_id = "my customer id";
String amount = "20.00";
String pan = "4242424242424242";
String expdate = "1901"; //YYMM
String crypt = "7";
String processing_country_code = "CA";
boolean status_check = false;
IndependentRefund indrefund = new IndependentRefund();
indrefund.setOrderId(order_id);
indrefund.setCustomerId(cust_id);
indrefund.setAmount(amount);
indrefund.setPan(pan);
indrefund.setExpdate(expdate);
indrefund.setCryptType(crypt);
indrefund.setDynamicDescriptor("123456");
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false for production transactions
mpgReq.setstoreId(store_id);
}
}

```

```

mpgReq.setApiToken(api_token);
mpgReq.setTransaction(indrefund);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = " + receipt.getBankTotals());
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("IsVisaDebit = " + receipt.getIsVisaDebit());
}
catch (Exception e)
{
e.printStackTrace();
}
}
}

```

## 2.9 Vérification de carte avec le SVA et le NVC

Une transaction de vérification de carte permet de vérifier la validité d'une carte de crédit, sa date d'expiration et d'autres renseignements additionnels en utilisant le numéro de vérification de carte ou le service de vérification d'adresse. Cette transaction ne vérifie pas la disponibilité des fonds ni ne bloque ces fonds sur la carte de crédit.

### Éléments dont il faut tenir compte :

Une transaction de vérification de carte peut être utilisée seulement pour les cartes Visa, Mastercard, Discover et American Express.

Pour certaines transactions utilisant les renseignements d'identification au dossier, une vérification de carte au moyen du SVA et du NVC a lieu afin d'obtenir l'ID de l'émetteur (issuer ID), qui sera utilisé dans les transactions subséquentes.

Lors de la mise à l'essai des transactions de vérification de carte, veuillez utiliser les numéros de carte test Visa et Mastercard fournis dans les tableaux Vérification de carte Mastercard et Vérification de carte Visa qui se trouvent dans le simulateur de NVC et de SVA (fraude électronique).

Pour obtenir une liste complète des résultats du SVA et du NVC possible, consultez les tableaux Codes de résultat du NVC et du SVA.

```

CardVerification cardVerification = new CardVerification();

HttpsPostRequest mpgReq = new HttpsPostRequest();

mpgReq.setTransaction(cardVerification );

```

## Principaux champs d'objet de connexion (toutes les transactions API)

Variable	Type et limites	
ID de commerce	<i>Chaîne</i> S. O.	mpgReq.setstoreId(store_id);
Jeton API	<i>Chaîne</i> S. O.	mpgReq.setApiToken(api_token);

## Champs de demande de transaction de vérification de carte (obligatoires)

Variable	Type et limites	
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques a-Z A-Z 0-9 _ - : . @ espaces	cardVerification.setOrderId(order_id);
Numéro de carte de crédit	<i>Chaîne</i> Maximum 20 caractères alphanumériques	cardVerification.setPan(pan);
Date d'expiration	<i>Chaîne</i> 4 caractères alphanumériques AAMM	cardVerification.setExpDate(expiry_date);
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	cardVerification.setCryptType(crypt);
Renseignements du SVA	<i>Objet</i> S. O.	cardVerification.setAvsInfo(avscCheck);
Renseignements du NVC	<i>Objet</i>	cardVerification.setCvdInfo(c

**REMARQUE :** Lorsque les renseignements d'identification sont stockés lors de la première transaction, l'objet CVD (numéro de vérification de carte, ou NVC) doit être envoyé. Pour les transactions subséquentes utilisant les renseignements d'identification stockés, le NVC peut être envoyé avec les transactions entamées par le titulaire de carte seulement. **Les commerçants ne doivent pas enregistrer le NVC.**

S. O.

vdCheck) ;

## Champs de demande pour l'objet Credential on File Info

Variable	Type et limites	
ID de l'émetteur	<p><i>Chaîne</i></p> <p>15 caractères alphanumériques</p> <p>Longueur variable</p>	<pre>cof.setIssuerId("VALUE_FOR_ISSUER_ID");</pre> <p><b>REMARQUE :</b> Pour obtenir une liste et une explication des valeurs possibles à envoyer pour cette variable, consultez la section Définition des champs de demande – Renseignements d'identification au dossier (cof)</p>
Indicateur de paiement	<p><i>Chaîne</i></p> <p>1 caractère alphabétique</p>	<pre>cof.setPaymentIndicator("PAYMENT_INDICATOR_VALUE");</pre> <p><b>REMARQUE :</b> Pour obtenir une liste et une explication des valeurs possibles à envoyer pour cette variable, consultez la section Définition des champs de demande – Renseignements d'identification au dossier (cof)</p>
Information de paiement	<p><i>Chaîne</i></p> <p>1 caractère numérique</p>	<pre>cof.setPaymentInformation("PAYMENT_INFO_VALUE");</pre> <p><b>REMARQUE :</b> Pour obtenir une liste et une explication des valeurs possibles à envoyer pour cette variable, consultez la section Définition des champs de demande – Renseignements d'identification au dossier (cof)</p>

## Champs de la demande d'objet Account Name Verification

Champs de demande dans l'objet Account Name Verification. L'objet ne peut être inclus que dans les transactions de type Card Verification. La vérification du nom du compte ne s'applique qu'aux cartes de crédit Visa.

Variable	Type et limites	
First Name	<i>Chaîne</i> 32 caractères alphanumériques	.setFirstName ("FIRST");  Prénom du titulaire de la carte
Middle Name	<i>Chaîne</i> 32 caractères alphanumériques	.setMiddleName ("MIDDLE");  Deuxième prénom du titulaire de la carte
Last Name	<i>Chaîne</i> 32 caractères alphanumériques	.setLastName ("LAST");  Nom de famille du titulaire de la carte

## Exemple de transaction de vérification de carte

```
package Canada;
import JavaAPI.*;
public class TestCanadaCardVerification
{
public static void main(String[] args)
{
String store_id = "store5";
String api_token = "yesguy";
java.util.Date createDate = new java.util.Date();
String order_id = "Test"+createDate.getTime();
String pan = "4242424242424242";
String expdate = "1901"; //YYMM format
String crypt = "7";
String processing_country_code = "CA";
boolean status_check = false;

AvsInfo avsCheck = new AvsInfo();
avsCheck.setAvsStreetNumber("212");
avsCheck.setAvsStreetName("Payton Street");
avsCheck.setAvsZipCode("M1M1M1");
CvdInfo cvdCheck = new CvdInfo();
cvdCheck.setCvdIndicator("1");
cvdCheck.setCvdValue("099");
CardVerification cardVerification = new CardVerification();
cardVerification.setOrderId(order_id);
cardVerification.setPan(pan);
cardVerification.setExpdate(expdate);
cardVerification.setCryptType(crypt);
cardVerification.setAvsInfo(avsCheck);
```

```

cardVerification.setCvdInfo(cvdCheck);

// TrId and TokenCryptogram are optional, refer documentation for more details.
cardVerification.setTrId("50189815682");
cardVerification.setTokenCryptogram("APmbM/411e0uAAH+s6xMAAADFA==");

//optional - Credential on File details
CofInfo cof = new CofInfo();
cof.setPaymentIndicator("U");
cof.setPaymentInformation("2");
cof.setIssuerId("139X3130ASCXAS9");

cardVerification.setCofInfo(cof);

//optional - Visa Account Name Verification
AccountNameVerification anf = new AccountNameVerification();
anf.setFirstName("FIRST");
anf.setMiddleName("MIDDLE");
anf.setLastName("LAST");

cardVerification.setAccountNameVerification(anf);

HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false for production transactions
mpgReq.setstoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(cardVerification);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = " + receipt.getBankTotals());
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("IsVisaDebit = " + receipt.getIsVisaDebit());
System.out.println("IssuerId = " + receipt.getIssuerId());
System.out.println("SourcePanLast4 = " + receipt.getSourcePanLast4());
System.out.println("AccountNameVerificationResult = " +
receipt.getAccountNameVerificationResult());
}
catch (Exception e)
{
e.printStackTrace();
}
}
}

```

## 2.10 Fermeture de lot

Une transaction de fermeture de lot récupère les fonds de toutes les transactions d'achat, de conclusion, de remboursement et forcées afin qu'ils soient déposés ou débités le jour ouvrable suivant.

Pour que les fonds soient déposés le jour ouvrable suivant, le lot doit être fermé avant 23 h, heure de l'Est.

```
BatchClose batchclose = new BatchClose();

HttpsPostRequest mpgReq = new HttpsPostRequest();

mpgReq.setTransaction(batchclose);
```

### Principaux champs d'objet de connexion (toutes les transactions API)

Variable	Type et limites	
ID de commerce	<i>Chaîne</i> S. O.	mpgReq.setstoreId(store_id);
Jeton API	<i>Chaîne</i> S. O.	mpgReq.setApiToken(api_token);

### Champs de demande de transaction de fermeture de lot (obligatoires)

Variable	Type et limites	
Numéro de caisse enregistreuse électronique	<i>Chaîne</i> Aucune limite (valeur fournie par Moneris)	batchclose.setEcrno(ecr_no);

### Exemple de transaction de fermeture de lot

```
package Canada;
import JavaAPI.*;
public class TestCanadaBatchClose
{
    public static void main(String[] args)
    {
        String store_id = "store5";
        String api_token = "yesguy";
        String ecr_no = "66013455"; //ecr within store
        String processing_country_code = "CA";
        boolean status_check = false;
        BatchClose batchclose = new BatchClose();
```



## 2.11 Ouverture des totaux

Une transaction d'ouverture des totaux permet d'obtenir des renseignements sur le lot actuellement ouvert.

Cette transaction est semblable à la fermeture de lot, à la différence qu'elle ne ferme pas le lot à des fins de règlement.

```
OpenTotals opentotals = new OpenTotals();

HttpsPostRequest mpgReq = new HttpsPostRequest();

mpgReq.setTransaction(opentotals);
```

### Principaux champs d'objet de connexion (toutes les transactions API)

Variable	Type et limites	
ID de commerce	<i>Chaîne</i> S. O.	mpgReq.setstoreId(store_id);
Jeton API	<i>Chaîne</i> S. O.	mpgReq.setApiToken(api_token);

### Champs de demande de transaction d'ouverture des totaux

Pour obtenir une description de toutes les valeurs obligatoires et facultatives, consultez l'annexe A Définition des champs de demande.

Variable	Type et limites	
Numéro de caisse enregistreuse électronique	Aucune limite (valeur fournie par Moneris)	opentotals.setEcrno(ecr_no);

### Exemple d'une transaction d'ouverture des totaux

```
package Canada;
import JavaAPI.*;
public class TestCanadaOpenTotals
{
public static void main(String[] args)
{
String store_id = "store5";
String api_token = "yesguy";
String ecr_no = "66013455";
//String ecr_no = "66011091";
String processing_country_code = "CA";
OpenTotals opentotals = new OpenTotals();
opentotals.setEcrno(ecr_no);
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
```

```

mpgReq.setTestMode(true); //false for production transactions
mpgReq.setstoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(opentotals);
mpgReq.send();

try
{
Receipt receipt = mpgReq.getReceipt();
if ((receipt.getReceiptId()).equals("Global Error Receipt") ||
receipt.getReceiptId().equals("null") ||
receipt.getReceiptId().equals(""))
{
System.out.println("CardType = null");
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = null");
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedOut());
}
else
{
for (String ecr : receipt.getTerminalIDs())
{
System.out.println("ECR: " + ecr);

for (String cardType : receipt.getCreditCards(ecr))
{
System.out.println("\tCard Type: " + cardType);
System.out.println("\t\tPurchase: Count = "
+ receipt.getPurchaseCount(ecr, cardType)
+ " Amount = "
+ receipt.getPurchaseAmount(ecr,
cardType));
System.out.println("\t\tRefund: Count = "
+ receipt.getRefundCount(ecr, cardType)
+ " Amount = "
+ receipt.getRefundAmount(ecr, cardType));
System.out.println("\t\tCorrection: Count = "
+ receipt.getCorrectionCount(ecr, cardType)
+ " Amount = "
+ receipt.getCorrectionAmount(ecr,
cardType));
}
}
}
}
catch (Exception e)
{
e.printStackTrace();
}
}
}
}

```

## 3 Renseignements d'identification au dossier

- 3.1 À propos des renseignements d'identification au dossier
- 3.2 Objet Credential on File Info et variables
- 3.3 Types de transactions nécessitant des renseignements d'identification au dossier
- 3.5 Transactions initiales nécessitant des renseignements d'identification au dossier
- 3.9 Renseignements d'identification au dossier et conversion de jetons temporaires
- 3.8 Transformation en jetons des cartes de crédit et des renseignements d'identification au dossier
- 3.10 Transactions de vérification de carte et de renseignements d'identification au dossier

### 3.1 À propos des renseignements d'identification au dossier

Les marques de carte exigent désormais que les commerçants qui enregistrent les renseignements d'identification des cartes de crédit de leurs clients pour des transactions subséquentes, ou qui utilisent ces renseignements dans des transactions subséquentes, l'indiquent dans la demande de transaction.

Dans l'API de Moneris, ceci est géré par la Passerelle Moneris et l'inclusion de l'objet Credential on File Info et de ses variables dans les demandes de transaction.

Bien que seules Visa, Mastercard et Discover ont des exigences concernant la gestion des transactions d'enregistrement des données d'identification au dossier, afin de prévenir toute confusion et erreur, veuillez appliquer ces changements pour tous les types de cartes, et le système de Moneris transférera les données de cartes pertinentes au besoin.

**REMARQUE :** Si la première transaction ou une autorisation de vérification de carte est refusée lors de l'enregistrement des renseignements d'identification du titulaire de carte, ces derniers ne peuvent pas être stockés. Par conséquent, le commerçant ne doit pas utiliser ces renseignements pour le traitement de transactions ultérieures.

### 3.2 Objet Credential on File Info et variables

L'objet Credential on File Info est imbriqué dans la demande pour les types de transactions applicables.

Objet Credential on File Info :

1. cof

Variables de l'objet cof :

1. Indicateur de paiement
2. Information de paiement
3. ID de l'émetteur

Pour plus de renseignements, consultez la section Définitions des champ de demande – Renseignements d'identification au dossier (cof).

### 3.3 Types de transactions nécessitant des renseignements d'identification au dossier

L'objet Credential on File Info s'applique aux transactions suivantes :

- Achat
- Préautorisation
- Achat avec 3-D Secure (cavv\_purchase)
- Achat avec 3-D Secure et facturation périodique
- Préautorisation avec 3-D Secure (cavv\_preatuh)
- Achat avec la chambre forte (ResPurchaseCC)
- Préautorisation avec la chambre forte (ResPreatuhCC)
- Vérification de la carte avec le SVA et le NVC
- Vérification de la carte avec la chambre forte (ResCardVerificationCC)
- Ajout d'une carte de crédit à la chambre forte (ResAddCC)
- Mise à jour d'une carte de crédit dans la chambre forte (ResUpdateCC)
- Ajout d'un jeton à la chambre forte (ResAddToken)
- Transformation en jeton d'une carte de crédit dans la chambre forte (ResTokenizeCC)
- Facturation périodique
- Achat avec la TMD
- Préautorisation avec la TMD
- Conclusion de préautorisation avec la TMD
- Achat avec la TMD et la chambre forte
- Préautorisation avec la TMD et la chambre forte
- Achat
- Préautorisation
- Achat avec 3-D Secure (cavv\_purchase)
- Préautorisation avec 3-D Secure (cavv\_preatuh)
- Achat avec la chambre forte
- Préautorisation avec la chambre forte
- Vérification de la carte
- Vérification de la carte avec la chambre forte
- Ajout d'une carte de crédit à la chambre forte
- Mise à jour d'une carte de crédit dans la chambre forte
- Transactions de facturation périodique

### 3.4 Transactions nécessitant des renseignements d'identification au dossier entamées par le commerçant et entamées par le titulaire de carte

Les transactions nécessitant des renseignements d'identification au dossier peuvent être entamées de deux manières : par un commerçant ou par un titulaire de carte. La personne qui effectue la transaction est importante, car elle détermine les champs de la variable Indicateurs des renseignements d'identification au dossier (Credential on File Indicator) doivent être envoyés dans la demande de transaction.

**Transactions nécessitant des renseignements d'identification au dossier entamées par le commerçant :** Il s'agit des transactions pour lesquelles le commerçant a l'intention d'enregistrer les renseignements d'identification du titulaire de la carte ou d'utiliser des renseignements d'identification qui ont déjà été enregistrés. Cela inclut l'envoi de l'objet Credential on File Info dans la demande de transaction ainsi que de ses trois champs **issuer ID**, **payment indicator** et **payment information**.

**Transactions avec renseignements d'identification au dossier entamées par le titulaire de carte:** ces transactions sont déclenchées par une action du titulaire de carte. Pour les transactions entamées par le titulaire de carte, seuls les champs **payment indicator** et **payment information** sont requis.

Pour simplifier le développement de votre intégration, Passerelle Moneris permet également aux transactions entamées par le titulaire de carte d'être traitées selon les mêmes règles en matière de renseignements d'identification au dossier que celles qui s'appliquent aux transactions entamées par le commerçant. Techniquement, l'**ID de l'émetteur** (issuer ID) n'est pas requis pour les transactions initiées par le titulaire de carte, mais pour des raisons pratiques, s'il est inclus dans la demande de transaction, Passerelle Moneris l'ignorera lorsqu'elle transmettra la demande au serveur.

### 3.5 Transactions initiales nécessitant des renseignements d'identification au dossier

Lors de l'envoi d'une transaction *initiale* avec un objet Credential on File Info (c.-à-d. une demande de transaction au cours de laquelle les renseignements d'identification du titulaire de carte sont enregistrés pour la *première* fois), il est important de comprendre ce qui suit :

- Vous devez envoyer le numéro de vérification de la carte (NVC) du titulaire de carte.
- L'**ID de l'émetteur** (issuer ID) sera envoyé sans valeur pour la transaction initiale, car il est reçu en réponse à cette transaction. Pour chaque transaction *subséquente* entamée par le commerçant ainsi que pour toutes les transactions administratives, vous devez inclure cet **ID de l'émetteur**.
- Le champ **Information de paiement** (payment information) doit toujours avoir une valeur de 0 lors de la première transaction.
- Le champ **Indicateur de paiement** (payment indicator) doit être rempli selon la valeur appropriée pour la transaction.

### 3.6 Renseignements d'identification enregistrés au dossier avec des renseignements d'identification précédemment stockés

Lorsque vous traitez une transaction avec des renseignements d'un titulaire de carte qui étaient déjà enregistrés **avant** la mise en œuvre des exigences relatives aux renseignements d'identification enregistrés au dossier, vous devez :

- inclure l'objet Credential on File Info dans la demande de transaction;
- envoyer la variable **payment information** avec une valeur de 2;
- enregistrer l'**ID de l'émetteur** (issuer ID) envoyé dans la transaction et l'associer aux renseignements d'identification du titulaire pour une utilisation ultérieure.

Lorsque l'**ID de l'émetteur** (issuer ID) a été enregistré et associé aux renseignements d'identification du titulaire de carte, envoyez-le dans toutes les transactions qui suivront. L'**ID de l'émetteur** (issuer ID) est requis seulement lors de l'envoi de transactions entamées par le commerçant.

### 3.7 Valeurs des indicateurs de paiement (payment indicator) et de commerce électronique (e-commerce indicator)

Lors de l'envoi de renseignements d'identification au dossier dans des demandes de transaction qui comprennent également le champ de demande **e-commerce indicator** (en code, appelé `crypt_type`), les valeurs acceptables pour ce champ dépendent de la valeur envoyée pour le champ **payment indicator**.

Si la valeur de l'indicateur de paiement (payment indicator) est :	Les valeurs de l'indicateur de commerce électronique (e-commerce indicator) sont :
R	2 = Commande postale/téléphonique – Périodique 5 = Transaction électronique authentifiée (3-D Secure) 6 = Transaction électronique non authentifiée (3-D Secure)
V	2 = Commande postale/téléphonique – Périodique (variable) 5 = Transaction électronique authentifiée (3-D Secure) 6 = Transaction électronique non authentifiée (3-D Secure)
C	1 = Commande postale/téléphonique – Unique 5 = Transaction électronique authentifiée (3-D Secure) 6 = Transaction électronique non authentifiée (3-D Secure) 7 = Commerçant prenant en charge le SSL
U	1 = Commande postale/téléphonique – Unique 7 = Commerçant prenant en charge le SSL
Z	1 = Commande postale/téléphonique – Unique 5 = Transaction électronique authentifiée (3-D Secure) 6 = Transaction électronique non authentifiée (3-D Secure) 7 = Commerçant prenant en charge le SSL

### 3.8 Transformation en jetons des cartes de crédit et des renseignements d'identification au dossier

Lorsque vous voulez enregistrer dans la chambre forte les renseignements d'identification d'un titulaire de carte provenant d'une transaction antérieure, vous devez effectuer une demande de transaction de transformation en jeton d'une carte de crédit dans la chambre forte. Selon les règles des

renseignements d'identification au dossier, seules les transactions antérieures ayant l'objet Credential on File Info peuvent être transformées en jeton dans la chambre forte.

Pour plus de renseignements sur cette transaction, consultez la section 4.5.10 Transformation en jeton d'une carte de crédit dans la chambre forte

### 3.9 Renseignements d'identification au dossier et conversion de jetons temporaires

Si vous décidez de transformer un jeton temporaire représentant les renseignements d'identification d'un titulaire de carte en jeton permanent, ces renseignements sont enregistrés au dossier et, par conséquent, vous devez envoyer les renseignements d'identification au dossier.

Pour les transactions d'ajout d'un jeton temporaire à la chambre forte que vous décidez par la suite de convertir en jeton permanent (renseignements d'identification au dossier) :

1. Envoyez une demande de transaction incluant l'objet Credential on File Info afin d'obtenir l'ID de l'émetteur (issuer ID); il peut s'agir d'une demande de vérification de carte, d'achat ou de préautorisation.
2. Une fois la transaction conclue, envoyez une demande d'ajout de jeton à la chambre forte en incluant l'objet Credential on File Info (champ Issuer ID seulement) afin de transformer le jeton temporaire en jeton permanent.

Pour plus de renseignements sur les transactions d'ajout de jeton temporaire à la chambre forte, consultez la section 4.5.1 Ajout d'une carte de crédit à la chambre forte (ResAddCC).

### 3.10 Transactions de vérification de carte et de renseignements d'identification au dossier

En l'absence d'une transaction d'achat ou de préautorisation, une vérification de carte est utilisée afin d'obtenir la valeur unique de l'ID de l'émetteur (**issuerId**) qui sera utilisée dans les transactions subséquentes nécessitant les renseignements d'identification au dossier. L'ID de l'émetteur est une variable inclue dans l'objet Credential on File Info imbriqué.

Pour chaque transaction initiale, y compris les transactions de vérification de carte, vous devez également demander le numéro de vérification de carte (NVC) du titulaire de carte. Pour plus de renseignements sur les NVC, consultez la section 9.2 Numéro de vérification de carte (NVC).

Pour obtenir une liste complète de ces variables, consultez chaque type de transaction ou la section Définition des champs de demande – Renseignements d'identification au dossier (cof).

La demande de vérification de carte, incluant l'objet Credential on File Info, doit être envoyé juste avant l'enregistrement des renseignements d'identification du titulaire de carte.

Pour plus de renseignements sur les transactions de vérification de carte, consultez la section 2.9 Vérification de carte avec le SVA et le NVC.

### 3.10.1 Quand effectuer une vérification de carte avec les renseignements d'identification au dossier

Si vous n'envoyez pas de transaction d'achat ou de préautorisation (c.-à-d. que vous ne facturez pas immédiatement de frais au client), vous devez effectuer une transaction de vérification de carte (ou dans le cas d'un ajout de jeton à la chambre forte, une transaction de vérification de carte avec la chambre forte) avant d'effectuer la transaction afin d'obtenir l'ID de l'émetteur (issuer ID).

Ces transactions s'appliquent à ce qui suit :

1. Ajout d'une carte de crédit à la chambre forte (ResAddCC)
2. Mise à jour d'une carte de crédit dans la chambre forte (ResUpdateCC)
3. Ajout d'un jeton à la chambre forte (ResAddToken)
4. Ajout d'une carte de crédit à la chambre forte (Res\_Add\_CC)
5. Mise à jour d'une carte de crédit dans la chambre forte (Res\_Update\_CC)
6. Transactions de facturation périodique, si :
  - la première transaction aura lieu ultérieurement

### 3.10.2 Renseignements d'identification au dossier et ajout de jeton à la chambre forte

Pour les transactions d'ajout de jeton à la chambre forte :

1. Envoyez une transaction de vérification de carte avec la chambre forte incluant l'objet Credential on File Info afin d'obtenir l'ID de l'émetteur (issuer ID).
2. Envoyez une demande d'ajout de jeton à la chambre forte en incluant l'objet Credential on File Info (champ Issuer ID seulement; les autres champs ne s'appliquent pas dans cette situation).

Pour en savoir plus sur ce type de transaction, consultez la section 4.5.9 Ajout d'un jeton à la chambre forte (ResAddToken).

### 3.10.3 Renseignements d'identification au dossier et mise à jour d'une carte de crédit dans la chambre forte

Pour les transactions de mise à jour d'une carte de crédit dans la chambre forte permettant de mettre à jour le numéro de la carte de crédit :

1. Envoyez une transaction de vérification de carte incluant l'objet Credential on File Info afin d'obtenir l'ID de l'émetteur (issuer ID).
2. Envoyez la demande de mise à jour de la carte de crédit dans la chambre forte en incluant l'objet Credential on File Info (champ Issuer ID seulement).

Pour en savoir plus sur ce type de transaction, consultez la section 4.5.3 Mise à jour d'une carte de crédit dans la chambre forte (ResUpdateCC).

### 3.10.4 Renseignements d'identification au dossier et ajout d'une carte de crédit à la chambre forte

Pour les transactions d'ajout d'une carte de crédit à la chambre forte :

1. Envoyez une transaction de vérification de carte incluant l'objet Credential on File Info afin d'obtenir l'ID de l'émetteur (issuer ID).
2. Envoyez la demande d'ajout d'une carte de crédit à la chambre forte en incluant l'objet Credential on File Info (champ Issuer ID seulement).

Pour en savoir plus sur ce type de transaction, consultez la section 4.5.1 Ajout d'une carte de crédit à la chambre forte (ResAddCC).

### 3.10.5 Renseignements d'identification au dossier et facturation périodique

**REMARQUE :** La valeur du champ **payment indicator** doit être **R** lors de l'envoi de transactions de facturation périodique.

Pour les transactions de facturation périodique qui commencent **immédiatement** :

1. Envoyez une demande de transaction d'achat en incluant les objets Recurring Billing et Credential on File Info. Assurez-vous que le champ **start now** de l'objet Recurring Billing indique « true ».

Pour les transactions de facturation périodique qui commencent à une date **ultérieure** :

1. Envoyez une transaction de vérification de carte incluant l'objet Credential on File Info afin d'obtenir l'ID de l'émetteur (issuer ID).
2. Envoyez une demande de transaction d'achat en incluant les objets Recur et Credential on File Info.

Pour mettre à jour le numéro de carte de crédit d'une série de transactions de facturation périodique (ne s'applique pas si vous modifiez seulement le calendrier de paiement ou le montant de cette série de transactions) :

1. Envoyez une demande de transaction de vérification de carte incluant l'objet Credential on File Info afin d'obtenir l'ID de l'émetteur (issuer ID).
2. Envoyez une transaction de mise à jour de facturation périodique.

Pour plus de renseignements sur l'objet Recurring Billing, consultez la section Définition des champs de demande – Facturation périodique (recurring).

## 4 Chambre forte

- 4.1 À propos des transactions utilisant la chambre forte
- 4.2 Types de transactions utilisant la chambre forte
- 4.3 Transactions utilisant la chambre forte qui prennent en charge les jetons temporaires
- 4.6 Transactions administratives utilisant la chambre forte
- 4.7 Transactions financières utilisant la chambre forte
- 4.8 Transformation en jetons hébergée

### 4.1 À propos des transactions utilisant la chambre forte

La fonction de chambre forte permet aux commerçants de créer des profils de client, de modifier ces profils et de les utiliser afin de traiter des transactions sans qu'il soit nécessaire d'ajouter les renseignements financiers. Les profils de client enregistrent les données des clients requises pour traiter des transactions, y compris les renseignements sur les cartes de crédit et de débit.

La chambre forte complète le module de facturation périodique. Elle stocke les renseignements des clients en toute sécurité sur les serveurs sécuritaires de Moneris. Ainsi, les commerçants peuvent facturer leurs clients pour des produits ou services routiniers lorsqu'une facture est envoyée.

### 4.2 Types de transactions utilisant la chambre forte

L'API de la chambre forte prend en charge les transactions administratives et financières.

#### 4.2.1 Types de transactions administratives utilisant la chambre forte

##### ResAddCC

Cette transaction crée un nouveau profil de carte et génère une clé de données unique, que vous pouvez obtenir par l'entremise de l'objet Receipt.

Cette clé de données est l'identifiant du profil que toutes les transactions financières ultérieures utilisant la chambre forte utiliseront pour accéder aux renseignements enregistrés.

##### EncResAddCC

Cette transaction crée un nouveau profil de carte de crédit, mais nécessite que la carte soit glissée ou que ses données soient saisies manuellement par l'entremise d'un lecteur de carte à bande magnétique chiffré fourni par Moneris.

##### ResTempAdd

Cette transaction crée un nouveau profil de carte de crédit avec un jeton temporaire. Une durée doit être configurée afin d'indiquer la durée de stockage du jeton temporaire.

Pendant sa durée de vie, le jeton temporaire peut être utilisé pour conclure d'autres transactions utilisant la chambre forte avant qu'il ne soit supprimé de façon permanente du système.

##### ResUpdateCC

Cette transaction met à jour les renseignements de carte de crédit d'un profil de chambre forte (basé sur la clé de données).

Tous les renseignements inclus dans un profil de carte de crédit sont mis à jour selon les champs remplis.

### **EncResUpdateCC**

Cette transaction met à jour les renseignements de carte de crédit d'un profil de carte de crédit (basé sur la clé de données). La version chiffrée de cette transaction nécessite que la carte soit glissée ou que ses données soient saisies manuellement par l'entremise d'un lecteur de carte à bande magnétique chiffré fourni par Moneris.

### **ResDelete**

Cette transaction supprime un profil de chambre forte existant, peu importe son type, en fonction de la clé de données unique attribuée à ce profil lors de sa création.

Il est important de noter qu'après la suppression d'un profil, les renseignements enregistrés dans ce profil ne peuvent plus être récupérés.

### **ResLookupFull**

Cette transaction vérifie les renseignements enregistrés dans le profil de chambre forte associé à la clé de données fournie. La réponse de cette transaction contient les plus récentes données actives de ce profil.

Contrairement à la transaction ResLookupMasked, qui renvoie un numéro de carte de crédit masqué, cette transaction renvoie le numéro de carte de crédit masqué et le numéro de carte de crédit non masqué.

### **ResLookupMasked**

Cette transaction vérifie les renseignements enregistrés dans le profil de chambre forte associé à la clé de données fournie. La réponse de cette transaction contient les plus récentes données actives de ce profil.

Contrairement à la transaction ResLookupFull, qui renvoie le numéro de carte de crédit masqué et le numéro de carte de crédit non masqué, cette transaction renvoie uniquement le numéro de carte de crédit masqué.

### **ResGetExpiring**

Cette transaction vérifie les profils dont les cartes de crédit expirent durant les mois civils actuel et prochain. Par exemple, si vous traitez cette transaction le 30 septembre, toutes les cartes dont la date d'expiration est en septembre ou en octobre s'afficheront.

Lors de la génération d'un liste de profils avec des cartes de crédit expirées ou qui expireront prochainement, seuls les numéros de carte de crédit **masqués** s'affichent.

Cette transaction ne peut être effectuée plus de deux fois par jour civil, et elle s'applique uniquement aux profils de carte de crédit.

### **ResIsCorporateCard**

Cette transaction détermine si une carte d'entreprise est enregistrée dans un profil.

Après l'envoi de la transaction, le champ de réponse getCorporateCard de l'objet Receipt indique soit true ou false, selon si la carte associée est une carte d'entreprise.

### **ResAddToken**

Cette transaction transforme un jeton temporaire obtenu par l'entremise de la transformation en jeton hébergée en jeton permanent de la chambre forte.

Un jeton temporaire est valide pour une durée de 15 minutes après sa création.

#### **ResTokenizeCC**

Cette transaction crée un nouveau profil de carte de crédit en utilisant le numéro de carte de crédit, la date d'expiration et l'indicateur de commerce électronique fournis dans une transaction financière antérieure. Une transaction effectuée précédemment dans Passerelle Moneris est prise, et les données de la carte liées à cette transaction sont enregistrées dans la chambre forte de Moneris.

Tout comme la transaction ResAddCC, une clé de données unique est générée et renvoyée au commerçant par l'entremise de l'objet Receipt. Cette clé de données est l'identifiant du profil que toutes les transactions financières ultérieures utilisant la chambre forte utiliseront pour accéder aux renseignements enregistrés.

#### **4.2.2 Types de transactions financières utilisant la chambre forte**

#### **ResPurchaseCC**

Cette transaction utilise la clé de données afin de trouver un profil de carte de crédit enregistré précédemment. Les renseignements enregistrés dans le profil sont ensuite utilisés pour effectuer une transaction d'achat.

#### **ResPreauthCC**

Cette transaction utilise la clé de données afin de trouver un profil de carte de crédit enregistré précédemment. Les renseignements du profil sont ensuite utilisés pour effectuer une transaction de préautorisation.

#### **ResIndRefundCC**

Cette transaction utilise la clé de données unique afin de trouver un profil de carte de crédit précédemment enregistré, puis crédite un montant précis sur cette carte de crédit.

#### **ResMpiTxn**

Cette transaction utilise la clé de données (et non pas un numéro de carte de crédit) dans une transaction MpiTxn utilisant Vérifié par Visa ou Mastercard SecureCode. Le commerçant utilise la clé de données avec la demande ResMpiTxn, puis lit les champs de réponse afin de vérifier si la carte est inscrite au programme Vérifié par Visa ou Mastercard SecureCode. Cette transaction récupère la valeur de la transaction utilisant la chambre forte et la transfère à Visa ou à Mastercard.

Après avoir validé que la clé de données est inscrite à 3-D Secure, une fenêtre dans laquelle le client peut entrer son mot de passe 3-D Secure s'affiche. Le commerçant peut entamer la création du formulaire de validation `getMpiInLineForm()`.

Pour plus de renseignements sur l'intégration des modules d'extension pour les commerçants de Moneris, consultez la section 1 Modules d'extension pour les commerçants.

#### **4.3 Transactions utilisant la chambre forte qui prennent en charge les jetons temporaires**

Les transactions suivantes prennent en charge les jetons temporaires et les jetons permanents :

- Achat avec la chambre forte (ResPurchaseCC)
- Préautorisation avec la chambre forte (ResPreauthCC)
- Vérification de la carte avec la chambre forte (ResCardVerificationCC)
- Achat avec la chambre forte et 3-D Secure
- Préautorisation avec la chambre forte et 3-D Secure

- Transaction forcée avec la chambre forte (ResForcePostCC)
- Remboursement indépendant avec la chambre forte (ResIndRefundCC)
- Détection de carte d'entreprise dans la chambre forte (ResIsCorporateCard)

## 4.4 Chambre forte et versements

La fonction de versements est également offerte pour les transactions traitées à l'aide des renseignements d'identification d'un titulaire de carte enregistrés dans la chambre forte de Moneris. Pour offrir cette fonction au client, envoyez la transaction Recherche d'un plan de versements dans la chambre forte (Vault Installment Plan Lookup) avant d'effectuer une transaction d'achat avec la chambre forte ou une transaction de préautorisation avec la chambre forte.

Pour en savoir plus sur les versements, consultez la section 8 Versements Visa.

## 4.5 Transactions administratives utilisant la chambre forte

Les transactions administratives vous permettent d'effectuer des tâches comme la création de nouveaux profils de chambre forte, la suppression de profils de chambre forte existants et la mise à jour des renseignements des profils.

Certaines transactions administratives utilisant la chambre forte nécessitent l'objet Credential on File, qui doit être envoyé avec le champ **issuer ID** seulement.

### 4.5.1 Ajout d'une carte de crédit à la chambre forte (ResAddCC)

Cette transaction crée un nouveau profil de carte et génère une clé de données unique, que vous pouvez obtenir par l'entremise de l'objet Receipt.

Cette clé de données est l'identifiant du profil que toutes les transactions financières ultérieures utilisant la chambre forte utiliseront pour accéder aux renseignements enregistrés.

#### Définition de l'objet de transaction Vault Add Credit Card

```
ResAddCC resaddcc = new ResAddCC();
```

#### Objet HttpsPostRequest pour les transactions d'ajout d'une carte de crédit à la chambre forte

```
HttpsPostRequest mpgReq = new HttpsPostRequest();  
mpgReq.setTransaction(resaddcc);
```

#### Champs de demande pour les transactions d'ajout de carte de crédit à la chambre forte (obligatoires)

Pour obtenir une description de toutes les valeurs obligatoires et facultatives, consultez l'annexe A Définition des champs de demande.

Variable	Type et limites	Méthode Set
Numéro de carte de crédit	<p><i>Chaîne</i></p> <p>Maximum 20 caractères alphanumériques</p>	resaddcc.setPan(pan);
Date d'expiration	<p><i>Chaîne</i></p> <p>4 caractères alphanumériques</p> <p>AAMM</p>	resaddcc.setExpDate(expiry_date);
Indicateur de commerce électronique	<p><i>Chaîne</i></p> <p>1 caractère alphanumérique</p>	resaddcc.setCryptType(crypt);
Renseignements d'identification au dossier cof	<p><i>Objet</i></p> <p>S. O.</p> <p><b>REMARQUE :</b> Cette variable est un objet imbriqué au sein d'une transaction et elle est requise lors du stockage des renseignements d'identification d'un client ou lors de l'utilisation de tels renseignements stockés. Pour plus de renseignements sur les champs de l'objet Credential on File Info, consultez la section Objet Credential File Info et variables.</p>	resaddcc.setCofInfo(cof);

Champs de demande pour les transactions d'ajout de carte de crédit à la chambre forte (**facultatifs**)

Variable	Type et limites	Méthode Set
ID de client	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p> <p><b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : &lt;&gt;\$%=?^{}[]\</p>	resaddcc.setCustId(cust_id);

Adresse courriel	<i>Chaîne</i> 30 caractères alphanumériques	resaddcc.setEmail(email);
Numéro de téléphone	<i>Chaîne</i> 30 caractères alphanumériques	resaddcc.setPhone(phone);
Remarque	<i>Chaîne</i> 30 caractères alphanumériques	resaddcc.setNote(note);
Format de la clé de données	<i>Chaîne</i> 2 caractères alphanumériques	resaddcc.setDataKeyFormat(data_key_format);
Renseignements du SVA	<i>Objet</i> S. O.	resaddcc.setAvsInfo(avsCheck);

#### Champs de demande pour l'objet Credential on File Info

Variable	Type et limites	Méthode Set
ID de l'émetteur  <b>REMARQUE :</b> Cette variable est requise pour chaque transaction entamée par le commerçant à la suite de la première transaction. Après l'envoi de la première transaction, l'ID d'émetteur (issuer ID) est reçu dans la réponse de transaction, puis utilisé dans les demandes de transaction subséquentes.	<i>Chaîne</i> 15 caractères alphanumériques Longueur variable	cof.setIssuerId("VALUE_FOR_ISSUER_ID");  <b>REMARQUE :</b> Pour obtenir une liste et une explication des valeurs possibles à envoyer pour cette variable, consultez la section Définition des champs de demande – Renseignements d'identification au dossier (cof)

### Exemple de transaction d'ajout de carte de crédit à la chambre forte

```
package Canada;
import JavaAPI.*;
public class TestCanadaResAddCC
{
    public static void main(String[] args)
    {
        String store_id = "store5";
        String api_token = "yesguy";
        String pan = "4242424242424242";
        String expdate = "1912";
        String phone = "0000000000";
        String email = "bob@smith.com";
        String note = "my note";
        String cust_id = "customer1";
        String crypt_type = "7";
        String data_key_format = "0";
        String processing_country_code = "CA";
        boolean status_check = false;
        AvsInfo avsCheck = new AvsInfo();
        avsCheck.setAvsStreetNumber("212");
        avsCheck.setAvsStreetName("Payton Street");
        avsCheck.setAvsZipCode("M1M1M1");
        ResAddCC resaddcc = new ResAddCC();
        resaddcc.setPan(pan);
        resaddcc.setExpdate(expdate);
        resaddcc.setCryptType(crypt_type);
        resaddcc.setCustId(cust_id);
        resaddcc.setPhone(phone);
        resaddcc.setEmail(email);
        resaddcc.setNote(note);
        resaddcc.setAvsInfo(avsCheck);
        //resaddcc.setDataKeyFormat(data_key_format); //optional
        //Mandatory - Credential on File details
        CofInfo cof = new CofInfo();
        cof.setIssuerId("139X3130ASCXAS9"); //can be obtained by performing card verification

        resaddcc.setCofInfo(cof);

        HttpsPostRequest mpgReq = new HttpsPostRequest();
        mpgReq.setProcCountryCode(processing_country_code);
        mpgReq.setTestMode(true); //false for production transactions
        mpgReq.setstoreId(store_id);
        mpgReq.setApiToken(api_token);
        mpgReq.setTransaction(resaddcc);
        mpgReq.setStatusCheck(status_check);
        mpgReq.send();
        try
        {
            Receipt receipt = mpgReq.getReceipt();
            System.out.println("DataKey = " + receipt.getDataKey());
            System.out.println("ResponseCode = " + receipt.getResponseCode());
            System.out.println("Message = " + receipt.getMessage());
            System.out.println("TransDate = " + receipt.getTransDate());
            System.out.println("TransTime = " + receipt.getTransTime());
            System.out.println("Complete = " + receipt.getComplete());
            System.out.println("TimedOut = " + receipt.getTimedOut());
            System.out.println("ResSuccess = " + receipt.getResSuccess());
            System.out.println("PaymentType = " + receipt.getPaymentType());
            System.out.println("Cust ID = " + receipt.getResCustId());
            System.out.println("Phone = " + receipt.getResPhone());
            System.out.println("Email = " + receipt.getResEmail());
            System.out.println("Note = " + receipt.getResNote());
            System.out.println("MaskedPan = " + receipt.getResMaskedPan());
            System.out.println("Exp Date = " + receipt.getResExpdate());
            System.out.println("Crypt Type = " + receipt.getResCryptType());
            System.out.println("Avs Street Number = " + receipt.getResAvsStreetNumber());
            System.out.println("Avs Street Name = " + receipt.getResAvsStreetName());
            System.out.println("Avs Zipcode = " + receipt.getResAvsZipcode());
            System.out.println("IssuerId = " + receipt.getIssuerId());
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

## **Champs de réponse liés à la chambre forte**

Pour obtenir une liste et une explication des champs de réponse de l'objet Receipt pouvant apparaître après l'envoi d'une transaction utilisant la chambre forte, consultez l'Annexe B Définition des champs de réponse (page 564).

### **4.5.1.1 Clé de données de la chambre forte**

L'exemple de code ResAddCC inclut l'instruction suivante provenant de l'objet Receipt :

```
System.out.println("DataKey = " + receipt.getDataKey());
```

Le champ de réponse data key (clé de données) est rempli lorsque vous envoyez les transactions suivantes : Ajout d'une carte de crédit à la chambre forte (ResAddCC) (page 68), Ajout d'une carte de crédit chiffrée à la chambre forte – EncResAddCC (page 72), Transformation en jeton d'une carte de crédit dans la chambre forte – ResTokenizeCC (page 94), Ajout d'un jeton temporaire à la chambre forte – ResTempAdd (page 72) ou Ajout d'un jeton à la chambre forte – ResAddToken (page 90). La clé de données est l'identifiant du profil que toutes les transactions financières ultérieures utilisant la chambre forte utiliseront pour accéder aux renseignements enregistrés.

La clé de données est une chaîne alphanumérique formée d'un maximum de 28 caractères.

### **4.5.1.2 Ajout d'une carte de crédit chiffrée à la chambre forte – EncResAddCC**

#### **Définition de l'objet de transaction Vault Encrypted Add Credit Card**

```
EncResAddCC encresaddcc = new EncResAddCC();
```

#### **Objet HttpsPostRequest pour les transactions d'ajout d'une carte de crédit chiffrée à la chambre forte**

```
HttpsPostRequest mpgReq = new HttpsPostRequest();  
  
mpgReq.setTransaction(encresaddcc);
```

#### **Champs de demande pour les transactions d'ajout de carte de crédit chiffrée à la chambre forte (obligatoires)**

Pour obtenir une description de toutes les valeurs obligatoires et facultatives, consultez l'annexe A Définition des champs de demande.

Variable	Type et limites	Méthode Set
Piste de données chiffrée	<i>Chaîne</i> 40 caractère numérique	encresaddcc.setEncTrack2 (enc_track2);

Type d'appareil	<i>Chaîne</i> 30 caractères alphanumériques Sensible à la casse	encresaddcc.setDeviceType(device_type);
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	encresaddcc.setCryptType(crypt);

**Champs de demande pour les transactions d'ajout de carte de crédit chiffrée à la chambre forte (facultatifs)**

Variable	Type et limites	Méthode Set
ID de client	<i>Chaîne</i> 50 caractères alphanumériques  <div style="border: 1px solid black; padding: 5px;"><b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : &lt;&gt;\$%=?^{}[]\</div>	encresaddcc.setDeviceType(device_type);
Renseignements du SVA	<i>Objet</i> S. O.	encresaddcc.setCryptType(crypt);
Adresse courriel	<i>Chaîne</i> 30 caractères alphanumériques	encresaddcc.setDeviceType(device_type);
Numéro de téléphone	<i>Chaîne</i> 30 caractères alphanumériques	encresaddcc.setCryptType(crypt);
Remarque	<i>Chaîne</i> 30 caractères alphanumériques	encresaddcc.setDeviceType(device_type);
Format de la clé de données	<i>Chaîne</i> 2 caractères alphanumériques	encresaddcc.setDataKeyFormat(data_key_format);

## Exemple de transaction d'ajout de carte de crédit chiffrée à la chambre forte

```
package Canada;
import JavaAPI.*;
public class TestCanadaEncResAddCC
{
    public static void main(String args[])
    {
        String store_id = "moneris";
        String api_token = "hurgle";
        String enc_track2 = "ENCRYPTEDTRACK2DATA";
        String device_type = "idtech_bdk";
        String phone = "55555555555";
        String email = "test.user@moneris.com";
        String note = "my note";
        String cust_id = "customer2";
        String crypt = "7";
        String processing_country_code = "CA";

        AvsInfo avsCheck = new AvsInfo();
        avsCheck.setAvsStreetNumber("212");
        avsCheck.setAvsStreetName("Payton Street");
        avsCheck.setAvsZipcode("M1M1M1");
        EncResAddCC enc_res_add_cc = new EncResAddCC ();
        enc_res_add_cc.setEncTrack2(enc_track2);
        enc_res_add_cc.setDeviceType(device_type);
        enc_res_add_cc.setCryptType(crypt);
        enc_res_add_cc.setCustId(cust_id);
        enc_res_add_cc.setPhone(phone);
        enc_res_add_cc.setEmail(email);
        enc_res_add_cc.setNote(note);
        //enc_res_add_cc.setAvsInfo(avsCheck);
        HttpsPostRequest mpgReq = new HttpsPostRequest();
        mpgReq.setProcCountryCode(processing_country_code);
        mpgReq.setTestMode(true); //false for production transactions
        mpgReq.setstoreId(store_id);
        mpgReq.setApiToken(api_token);
        mpgReq.setTransaction(enc_res_add_cc);
        mpgReq.send();

        try
        {
            Receipt receipt = mpgReq.getReceipt();
            System.out.println("DataKey = " + receipt.getDataKey());
            System.out.println("ResponseCode = " + receipt.getResponseCode());
            System.out.println("Message = " + receipt.getMessage());
            System.out.println("TransDate = " + receipt.getTransDate());
            System.out.println("TransTime = " + receipt.getTransTime());
            System.out.println("Complete = " + receipt.getComplete());
            System.out.println("TimedOut = " + receipt.getTimedOut());
            System.out.println("ResSuccess = " + receipt.getResSuccess());
            System.out.println("PaymentType = " + receipt.getPaymentType() + "\n");
            //Contents of ResolveData
            System.out.println("Cust ID = " + receipt.getResCustomerId());
            System.out.println("Phone = " + receipt.getResPhone());
            System.out.println("Email = " + receipt.getResEmail());
            System.out.println("Note = " + receipt.getResNote());
            System.out.println("MaskedPan = " + receipt.getResMaskedPan());
            System.out.println("Exp Date = " + receipt.getResExpDate());
            System.out.println("Crypt Type = " + receipt.getResCryptType());
            System.out.println("Avs Street Number = " + receipt.getResAvsStreetNumber());
            System.out.println("Avs Street Name = " + receipt.getResAvsStreetName());
            System.out.println("Avs Zipcode = " + receipt.getResAvsZipcode());
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

## Champs de réponse liés à la chambre forte

Pour obtenir une liste et une explication des champs de réponse de l'objet `Receipt` pouvant apparaître après l'envoi d'une transaction utilisant la chambre forte, consultez l'Annexe B Définition des champs de réponse (page 547).

## 4.5.2 Ajout d'un jeton temporaire à la chambre forte (ResTempAdd)

Cette transaction crée un nouveau profil de carte de crédit avec un jeton temporaire. Une durée doit être configurée afin d'indiquer la durée de stockage du jeton temporaire.

Pendant sa durée de vie, le jeton temporaire peut être utilisé pour conclure d'autres transactions utilisant la chambre forte avant qu'il ne soit supprimé de façon permanente du système.

### Éléments dont il faut tenir compte :

Le jeton temporaire peut avoir une durée de vie maximale de 15 minutes.

#### Définition de l'objet de transaction Vault Temporary Token Add

```
ResTempAdd resTempAdd = new ResTempAdd();
```

#### Objet HttpsPostRequest pour les transactions d'ajout d'un jeton temporaire à la chambre forte

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
```

```
mpgReq.setTransaction(resTempAdd);
```

#### Champs de demande pour les transactions d'ajout d'un jeton temporaire à la chambre forte (obligatoires)

Pour obtenir une description de toutes les valeurs obligatoires et facultatives, consultez l'annexe A Définition des champs de demande.

Variable	Type et limites	Méthode Set
Numéro de carte de crédit	<i>Chaîne</i> Maximum 20 caractères alphanumériques	resTempAdd.setPan(pan);
Date d'expiration	<i>Chaîne</i> 4 caractères alphanumériques AAMM	resTempAdd.setExpDate(expiry_date);
Durée	<i>Chaîne</i> 3 caractère numérique Maximum de 900 secondes	resTempAdd.setDuration(duration);
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	resTempAdd.setCryptType(crypt);

## Champs de demande pour les transactions d'ajout d'un jeton temporaire à la chambre forte (facultatifs)

Variable	Type et limites	Méthode Set
Format de la clé de données	<i>Chaîne</i> 2 caractères alphanumériques	restTempAdd.setDataKeyFormat (data_key_format);

### Exemple de transaction d'ajout d'un jeton temporaire à la chambre forte

```

package Canada;
import JavaAPI.*;
public class TestCanadaResTempAdd
{
  public static void main(String[] args)
  {
    String store_id = "store1";
    String api_token = "yesguy";
    String pan = "5454545454545454";
    String expdate = "1901"; //YYMM format
    String crypt_type = "7";
    String duration = "900";
    String processing_country_code = "CA";
    boolean status_check = false;
    ResTempAdd resTempAdd = new ResTempAdd();
    resTempAdd.setPan(pan);
    resTempAdd.setExpdate(expdate);
    resTempAdd.setDuration(duration);
    resTempAdd.setCryptType(crypt_type);
    // resTempAdd.setDataKeyFormat("OU");

    HttpsPostRequest mpgReq = new HttpsPostRequest();
    mpgReq.setProcCountryCode(processing_country_code);
    mpgReq.setTestMode(true); //false for production transactions
    mpgReq.setstoreId(store_id);
    mpgReq.setApiToken(api_token);
    mpgReq.setTransaction(resTempAdd);
    mpgReq.setStatusCheck(status_check);
    mpgReq.send();
    try
    {
      Receipt receipt = mpgReq.getReceipt();
      System.out.println("DataKey = " + receipt.getDataKey());
      System.out.println("ResponseCode = " + receipt.getResponseCode());
      System.out.println("Message = " + receipt.getMessage());
      System.out.println("TransDate = " + receipt.getTransDate());
      System.out.println("TransTime = " + receipt.getTransTime());
      System.out.println("Complete = " + receipt.getComplete());
      System.out.println("TimedOut = " + receipt.getTimedOut());
      System.out.println("ResSuccess = " + receipt.getResSuccess());
      System.out.println("PaymentType = " + receipt.getPaymentType());
      System.out.println("MaskedPan = " + receipt.getResMaskedPan());
      System.out.println("Exp Date = " + receipt.getResExpdate());
    }
    catch (Exception e)
    {
      e.printStackTrace();
    }
  }
}

```

## Champs de réponse liés à la chambre forte

Pour obtenir une liste et une explication des champs de réponse de l'objet Receipt pouvant apparaître après l'envoi d'une transaction utilisant la chambre forte, consultez l'Annexe B Définition des champs de réponse (page 547).

#### 4.5.3 Mise à jour d'une carte de crédit dans la chambre forte (ResUpdateCC)

La mise à jour d'une carte de crédit dans la chambre forte permet de mettre à jour les renseignements d'un titulaire de carte enregistrés dans un profil de chambre forte au moyen de la **clé de données** unique du profil.

Les renseignements du profil de carte de crédit sont mis à jour en fonction des champs soumis. Si un champ représentant un renseignement du titulaire de carte n'est pas inclus dans la demande, celui-ci demeurera inchangé dans le profil du titulaire de carte.

Si un nouveau numéro de carte de crédit est ajouté au profil de chambre forte, vous devez tout d'abord envoyer une transaction d'achat, de préautorisation ou de vérification de carte, en incluant l'objet Credential on File Info, avant d'effectuer la transaction de mise à jour d'une carte de crédit dans la chambre forte. Si le numéro de carte de crédit n'est pas mis à jour, cette étape n'est pas requise.

##### Éléments dont il faut tenir compte :

Pour mettre à jour un élément particulier du profil, configurez cet élément au moyen de la méthode Set correspondante.

Lorsque vous mettez à jour un numéro de carte de crédit, et avant d'effectuer cette transaction, envoyez une transaction d'achat, de préautorisation ou de vérification de carte en incluant l'objet Credential on File Info. Ensuite, envoyez l'ID de l'émetteur reçu dans la réponse de cette transaction dans la demande subséquente de mise à jour d'une carte de crédit dans la chambre forte.

Si le numéro de carte de crédit n'est pas mis à jour, l'objet Credential on File Info n'est pas requis.

##### Définition de l'objet de transaction Vault Update Credit Card

```
ResUpdateCC resUpdateCC = new ResUpdateCC();
```

##### Objet HttpsPostRequest pour les transactions de mise à jour d'une carte de crédit dans la chambre forte

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
```

```
mpgReq.setTransaction(resUpdateCC);
```

## Champs de demande pour les transactions de mise à jour d'une carte de crédit dans la chambre forte (obligatoires)

Pour obtenir une description de toutes les valeurs obligatoires et facultatives, consultez l'annexe A Définition des champs de demande.

Variable	Type et limites	Méthode Set
Clé de données	<i>Chaîne</i> 25 caractères alphanumériques	<code>resUpdateCC.setData (data_key);</code>

Les valeurs facultatives soumises à l'objet ResUpdateCC sont mises à jour. Les valeurs facultatives non incluses (à une exception près) demeurent inchangées. Vous pouvez ainsi modifier uniquement les champs voulus.

Si un profil contient des renseignements de SVA, mais qu'une transaction de mise à jour d'une carte de crédit dans la chambre forte est soumises sans l'objet AVS Info, les renseignements existants liés au SVA sont désactivés et les renseignements de la nouvelle carte de crédit sont enregistrés sans le SVA.

## Champs de demande pour les transactions de mise à jour d'une carte de crédit dans la chambre forte (facultatifs)

Variable	Type et limites	Méthode Set
Numéro de carte de crédit	<i>Chaîne</i> 20 caractères alphanumériques	<code>resUpdateCC.setPan (pan);</code>
Date d'expiration	<i>Chaîne</i> 4 caractères alphanumériques AAMM	<code>resUpdateCC.setExpDate (expiry_date);</code>
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	<code>resUpdateCC.setCryptType (crypt);</code>
ID de client	<i>Chaîne</i> 50 caractères alphanumériques  <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : <> \$ % = ? ^ { } [ ] \ \	<code>resUpdateCC.setCustId (cust_id);</code>
Adresse courriel	<i>Chaîne</i> 30 caractères alphanumériques	<code>resUpdateCC.setEmail (email);</code>

Numéro de téléphone	<i>Chaîne</i> 30 caractères alphanumériques	resUpdateCC.setPhone(phone);
Remarque	<i>Chaîne</i> 30 caractères alphanumériques	resUpdateCC.setNote(note);
Renseignements du SVA	<i>Objet</i> S. O.	resUpdateCC.setAvsInfo(avscCheck);
Renseignements d'identification au dossier cof	<i>Objet</i> S. O.	resUpdateCC.setCofInfo(cof);

**REMARQUE :** Cette variable est un objet imbriqué au sein d'une transaction et elle est requise lors du stockage des renseignements d'identification d'un client ou lors de l'utilisation de tels renseignements stockés. Pour plus de renseignements sur les champs de l'objet Credential on File Info, consultez la section Objet Credential File Info et variables.

### Champs de demande pour l'objet Credential on File Info

Variable	Type et limites	Méthode Set
ID de l'émetteur	<i>Chaîne</i> 15 caractères alphanumériques Longueur variable	cof.setIssuerId("VALUE_FOR_ISSUER_ID");
<b>REMARQUE :</b> Cette variable est requise pour chaque transaction entamée par le commerçant à la suite de la première transaction. Après l'envoi de la première transaction, l'ID d'émetteur (issuer ID) est reçu dans la réponse de transaction, puis utilisé dans les demandes de transaction subséquentes.		<b>REMARQUE :</b> Pour obtenir une liste et une explication des valeurs possibles à envoyer pour cette variable, consultez la section Définition des champs de demande – Renseignements d'identification au dossier (cof)

#### Exemple de transaction de mise à jour d'une carte de crédit dans la chambre forte

```
package Canada;
import JavaAPI.*;
public class TestCanadaResUpdateCC
```

```

{
    public static void main(String[] args)
    {
        String store_id = "moneris";
        String api_token = "hurgle";
        String data_key = "vthBjyN1BicbRkdWFZ9flyDP2";
        String pan = "4242424242424242";
        String expdate = "1901";
        String phone = "0000000000";
        String email = "bob@smith.com";
        String note = "my note";
        String cust_id = "customer1";
        String crypt_type = "7";
        String processing_country_code = "CA";
        boolean status_check = false;
        AvsInfo avsCheck = new AvsInfo();
        avsCheck.setAvsStreetNumber("212");
        avsCheck.setAvsStreetName("Payton Street");
        avsCheck.setAvsZipCode("M1M1M1");
        //Credential on File details
        CofInfo cof = new CofInfo();
        cof.setIssuerId("139X3130ASCXAS9");

        ResUpdateCC resUpdateCC = new ResUpdateCC();
        resUpdateCC.setData(data_key);
        resUpdateCC.setAvsInfo(avsCheck);
        resUpdateCC.setCustId(cust_id);
        resUpdateCC.setPan(pan);
        resUpdateCC.setExpdate(expdate);
        resUpdateCC.setPhone(phone);
        resUpdateCC.setEmail(email);
        resUpdateCC.setNote(note);
        resUpdateCC.setCryptType(crypt_type);
        resUpdateCC.setCofInfo(cof);
        HttpsPostRequest mpgReq = new HttpsPostRequest();
        mpgReq.setProcCountryCode(processing_country_code);
        mpgReq.setTestMode(true); //false for production transactions
        mpgReq.setstoreId(store_id);
        mpgReq.setApiToken(api_token);
        mpgReq.setTransaction(resUpdateCC);
        mpgReq.setStatusCheck(status_check);
        mpgReq.send();
        try
        {
            Receipt receipt = mpgReq.getReceipt();
            System.out.println("DataKey = " + receipt.getDataKey());
            System.out.println("ResponseCode = " + receipt.getResponseCode());
            System.out.println("Message = " + receipt.getMessage());
            System.out.println("TransDate = " + receipt.getTransDate());
            System.out.println("TransTime = " + receipt.getTransTime());
            System.out.println("Complete = " + receipt.getComplete());
            System.out.println("TimedOut = " + receipt.getTimedOut());
            System.out.println("ResSuccess = " + receipt.getResSuccess());
            System.out.println("PaymentType = " + receipt.getPaymentType());
            System.out.println("Cust ID = " + receipt.getResCustId());
            System.out.println("Phone = " + receipt.getResPhone());
            System.out.println("Email = " + receipt.getResEmail());
            System.out.println("Note = " + receipt.getResNote());
            System.out.println("MaskedPan = " + receipt.getResMaskedPan());
            System.out.println("Exp Date = " + receipt.getResExpdate());
            System.out.println("Crypt Type = " + receipt.getResCryptType());
            System.out.println("Avs Street Number = " + receipt.getResAvsStreetNumber());
            System.out.println("Avs Street Name = " + receipt.getResAvsStreetName());
            System.out.println("Avs Zipcode = " + receipt.getResAvsZipcode());
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}

```

## Champs de réponse liés à la chambre forte

Pour obtenir une liste et une explication des champs de réponse de l'objet Receipt pouvant apparaître après l'envoi d'une transaction utilisant la chambre forte, consultez l'Annexe B Définition des champs de réponse (page 547).

#### 4.5.3.1 Mise à jour d'une carte de crédit chiffrée dans la chambre forte (EncResUpdateCC)

##### Définition de l'objet de transaction Vault Encrypted Update CC

```
EncResUpdateCC enc_res_update_cc = new EncResUpdateCC ();
```

##### Objet HttpsPostRequest pour les transactions de mise à jour d'une carte de crédit chiffrée dans la chambre forte

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
```

```
mpgReq.setTransaction(enc_res_update_cc);
```

##### Champs de demande pour les transactions de mise à jour d'une carte de crédit chiffrée dans la chambre forte (obligatoires)

Pour obtenir une description de toutes les valeurs obligatoires et facultatives, consultez l'annexe A Définition des champs de demande.

Variable	Type et limites	Méthode Set
Clé de données	<i>Chaîne</i> 25 caractères alphanumériques	enc_res_update_cc.setData(data_key);
Piste de données chiffrée	<i>Chaîne</i> 40 caractère numérique	enc_res_update_cc.setEncTrack2(enc_track2);
Type d'appareil	<i>Chaîne</i> 30 caractères alphanumériques Sensible à la casse	enc_res_update_cc.setDeviceType(device_type);

Les valeurs facultatives soumises à l'objet ResUpdateCC sont mises à jour, alors que les valeurs facultatives non incluses (à une exception près) demeurent inchangées. Vous pouvez ainsi modifier uniquement les champs voulus.

L'exception est que si vous modifiez le type de paiement, **toutes** les variables indiquées dans le tableau des valeurs facultatives ci-dessous doivent être soumises.

Si vous modifiez le type de paiement du profil, ce dernier sera automatiquement désactivé et un nouveau profil de carte de crédit sera créé, puis attribué à la clé de données. Seuls l'ID de client, le numéro de téléphone, l'adresse courriel et les remarques du profil précédent demeureront inchangés.

**EXEMPLE :** Si un profil contient des renseignements de SVA, mais qu'une transaction ResUpdateCC est soumises sans l'objet AVSInfo, les renseignements existants liés à l'objet AVSInfo sont désactivés et les renseignements de la nouvelle carte de crédit sont enregistrés sans le SVA.

## la chambre forte (facultatifs)

Variable	Type et limites	Méthode Set
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	enc_res_update_cc.setCryptType(crypt);
ID de client	<i>Chaîne</i> 50 caractères alphanumériques <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : <> \$ % = ? ^ { } [ ] \	enc_res_update_cc.setCustId(cust_id);
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	enc_res_update_cc.setCryptType(crypt);
Adresse courriel	<i>Chaîne</i> 30 caractères alphanumériques	enc_res_update_cc.setEmail(email);
Numéro de téléphone	<i>Chaîne</i> 30 caractères alphanumériques	enc_res_update_cc.setPhone(phone);
Remarque	<i>Chaîne</i> 30 caractères alphanumériques	enc_res_update_cc.setNote(note);

## Exemple de transaction de mise à jour d'une carte de crédit chiffrée dans la chambre forte

```
package Canada;
import JavaAPI.*;
public class TestCanadaEncResUpdateCC
{
    public static void main(String args[])
    {
        String store_id = "store1";
        String api_token = "yesguy";
        String data_key = "PHTMLpun7VOaSCFM2xdeP2Sim";
        String enc_track2 = "ENCRYPTEDTRACK2DATA";
        String device_type = "idtech_bdk";
        String phone = "55555555555";
        String email = "test.user@moneris.com";
        String note = "my note";
        String cust_id = "customer2";
        String crypt = "7";
        String processing_country_code = "CA";
        AvsInfo avsinfo = new AvsInfo();
        avsinfo.setAvsStreetNumber("212");
        avsinfo.setAvsStreetName("Smith Street");
        avsinfo.setAvsZipcode("M1M1M1");
        EncResUpdateCC enc_res_update_cc = new EncResUpdateCC ();
        enc_res_update_cc.setDataKey(data_key);
        enc_res_update_cc.setAvsInfo(avsinfo);
        enc_res_update_cc.setCustId(cust_id);
        enc_res_update_cc.setEncTrack2(enc_track2);
        enc_res_update_cc.setDeviceType(device_type);
        enc_res_update_cc.setPhone(phone);
        enc_res_update_cc.setEmail(email);
        enc_res_update_cc.setNote(note);
        enc_res_update_cc.setCryptType(crypt);
        HttpsPostRequest mpgReq = new HttpsPostRequest();
        mpgReq.setProcCountryCode(processing_country_code);
        mpgReq.setTestMode(true); //false for production transactions
        mpgReq.setstoreId(store_id);
        mpgReq.setApiToken(api_token);
        mpgReq.setTransaction(enc_res_update_cc);
        mpgReq.send();
        try
        {
            Receipt receipt = mpgReq.getReceipt();
            System.out.println("DataKey = " + receipt.getDataKey());
            System.out.println("ResponseCode = " + receipt.getResponseCode());
            System.out.println("Message = " + receipt.getMessage());
            System.out.println("TransDate = " + receipt.getTransDate());
            System.out.println("TransTime = " + receipt.getTransTime());
            System.out.println("Complete = " + receipt.getComplete());
            System.out.println("TimedOut = " + receipt.getTimedOut());
            System.out.println("ResSuccess = " + receipt.getResSuccess());
            System.out.println("PaymentType = " + receipt.getPaymentType() + "\n");
            //Contents of ResolveData
            System.out.println("Cust ID = " + receipt.getResCustomerId());
            System.out.println("Phone = " + receipt.getResPhone());
            System.out.println("Email = " + receipt.getResEmail());
            System.out.println("Note = " + receipt.getResNote());
            System.out.println("MaskedPan = " + receipt.getResMaskedPan());
            System.out.println("Exp Date = " + receipt.getResExpDate());
            System.out.println("Crypt Type = " + receipt.getResCryptType());
            System.out.println("Avs Street Number = " + receipt.getResAvsStreetNumber());
            System.out.println("Avs Street Name = " + receipt.getResAvsStreetName());
            System.out.println("Avs Zipcode = " + receipt.getResAvsZipcode());
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

4 Chambre forte

## Champs de réponse liés à la chambre forte

Pour obtenir une liste et une explication des champs de réponse de l'objet Receipt pouvant apparaître après l'envoi d'une transaction utilisant la chambre forte, consultez l'Annexe B Définition des champs de réponse (page 547).

#### 4.5.4 Suppression d'un profil de la chambre forte (ResDelete)

Cette transaction supprime un profil de chambre forte existant, peu importe son type, en fonction de la clé de données unique attribuée à ce profil lors de sa création.

API de Passerelle Moneris – Guide d'intégration

**REMARQUE :** Une fois un profil supprimé, les renseignements enregistrés dans ce profil ne peuvent plus être récupérés.

##### Définition de l'objet de transaction Vault Delete

```
ResDelete resDelete = new ResDelete (data_key);
```

##### Objet HttpsPostRequest pour les transactions de suppression d'un profil de la chambre forte

```
HttpsPostRequest mpgReq = new HttpsPostRequest();  
mpgReq.setTransaction(resDelete);
```

##### Champs de demande pour les transactions de suppression d'un profil de la chambre forte (obligatoires)

Pour obtenir une description de toutes les valeurs obligatoires et facultatives, consultez l'annexe A Définition des champs de demande.

Variable	Type et limites	Méthode Set
Clé de données	<i>Chaîne</i> 25 caractères alphanumériques	resDelete.setData(data_key);

### Exemple de transaction de suppression de profil de la chambre forte

```
package Canada;
import JavaAPI.*;
public class TestCanadaResDelete
{
public static void main(String[] args)
{
String store_id = "moneris";
String api_token = "hurgle";
String data_key = "DxwdemrvfnoXO1HhmRikfw3gA";
String processing_country_code = "CA";
boolean status_check = false;
ResDelete resDelete = new ResDelete(data_key);
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false for production transactions
mpgReq.setstoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(resDelete);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("DataKey = " + receipt.getDataKey());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("Message = " + receipt.getMessage());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("ResSuccess = " + receipt.getResSuccess());
System.out.println("PaymentType = " + receipt.getPaymentType());
//ResolveData
System.out.println("Cust ID = " + receipt.getResCustomerId());
System.out.println("Phone = " + receipt.getResPhone());
System.out.println("Email = " + receipt.getResEmail());
System.out.println("Note = " + receipt.getResNote());
System.out.println("MaskedPan = " + receipt.getResMaskedPan());
System.out.println("Exp Date = " + receipt.getResExpdate());
System.out.println("Crypt Type = " + receipt.getResCryptType());
System.out.println("Avs Street Number = " + receipt.getResAvsStreetNumber());
System.out.println("Avs Street Name = " + receipt.getResAvsStreetName());
System.out.println("Avs Zipcode = " + receipt.getResAvsZipcode());
}
catch (Exception e)
{
e.printStackTrace();
}
}
}
```

### Champs de réponse liés à la chambre forte

Pour obtenir une liste et une explication des champs de réponse de l'objet Receipt pouvant apparaître après l'envoi d'une transaction utilisant la chambre forte, consultez l'Annexe B Définition des champs de réponse (page 547).

#### 4.5.5 Recherche d'un numéro complet dans la chambre forte (ResLookupFull)

Cette transaction vérifie les renseignements enregistrés dans le profil de chambre forte associé à la clé de données fournie. La réponse de cette transaction contient les plus récentes données actives de ce profil.

Contrairement à la transaction de recherche d'un numéro masqué dans la chambre forte, qui renvoie un numéro de carte de crédit masqué, cette transaction renvoie à la fois le numéro de carte de crédit masqué et le numéro de carte de crédit non masqué.

##### Définition de l'objet de transaction Vault Lookup Full

```
ResLookupFull resLookupFull = new ResLookupFull(data_key);
```

##### Objet HttpsPostRequest pour les transactions de recherche d'un numéro complet dans la chambre forte

```
HttpsPostRequest mpgReq = new HttpsPostRequest();  
  
mpgReq.setTransaction(resLookupFull);
```

##### Champs de demande pour les transactions de recherche d'un numéro complet dans la chambre forte (obligatoires)

Variable	Type et limites	Méthode Set
Clé de données	<i>Chaîne</i> 25 caractères alphanumériques	resLookupFull.setData(data_key);

### Exemple de transaction de recherche d'un numéro complet dans la chambre forte

```
package Canada;
import JavaAPI.*;
public class TestCanadaResLookupFull
{
    public static void main(String[] args)
    {
        String store_id = "store1";
        String api_token = "yesguy";
        String data_key = "pi3ZMZOttM8pLM9wuwuws2KBxw";
        String processing_country_code = "CA";
        boolean status_check = false;
        ResLookupFull resLookupFull = new ResLookupFull(data_key);
        HttpsPostRequest mpgReq = new HttpsPostRequest();
        mpgReq.setProcCountryCode(processing_country_code);
        mpgReq.setTestMode(true); //false for production transactions
        mpgReq.setstoreId(store_id);
        mpgReq.setApiToken(api_token);
        mpgReq.setTransaction(resLookupFull);
        mpgReq.setStatusCheck(status_check);
        mpgReq.send();
        try
        {
            Receipt receipt = mpgReq.getReceipt();
            System.out.println("DataKey = " + receipt.getDataKey());
            System.out.println("ResponseCode = " + receipt.getResponseCode());
            System.out.println("Message = " + receipt.getMessage());
            System.out.println("TransDate = " + receipt.getTransDate());
            System.out.println("TransTime = " + receipt.getTransTime());
            System.out.println("Complete = " + receipt.getComplete());
            System.out.println("TimedOut = " + receipt.getTimedOut());
            System.out.println("ResSuccess = " + receipt.getResSuccess());
            System.out.println("PaymentType = " + receipt.getPaymentType());
            System.out.println("Cust ID = " + receipt.getResCustomerId());
            System.out.println("Phone = " + receipt.getResPhone());
            System.out.println("Email = " + receipt.getResEmail());
            System.out.println("Note = " + receipt.getResNote());
            System.out.println("Pan = " + receipt.getResPan());
            System.out.println("MaskedPan = " + receipt.getResMaskedPan());
            System.out.println("Exp Date = " + receipt.getResExpdate());
            System.out.println("Crypt Type = " + receipt.getResCryptType());
            System.out.println("Avs Street Number = " + receipt.getResAvsStreetNumber());
            System.out.println("Avs Street Name = " + receipt.getResAvsStreetName());
            System.out.println("Avs Zipcode = " + receipt.getResAvsZipcode());
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

### Champs de réponse liés à la chambre forte

Pour obtenir une liste et une explication des champs de réponse de l'objet Receipt pouvant apparaître après l'envoi d'une transaction utilisant la chambre forte, consultez l'Annexe B Définition des champs de réponse (page 547).

#### 4.5.6 Recherche d'un numéro masqué dans la chambre forte (ResLookupMasked)

Cette transaction vérifie les renseignements enregistrés dans le profil de chambre forte associé à la clé de données fournie. La réponse de cette transaction contient les plus récentes données actives de ce profil.

Contrairement à la transaction de recherche d'un numéro complet dans la chambre forte, qui renvoie le numéro de carte de crédit masqué et le numéro de carte de crédit non masqué, cette transaction renvoie uniquement le numéro de carte de crédit masqué.

##### Définition de l'objet de transaction Vault Lookup Masked

```
ResLookupMasked resLookupMasked = new ResLookupMasked();
```

##### Objet HttpsPostRequest pour les transactions de recherche d'un numéro masqué dans la chambre forte

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
```

```
mpgReq.setTransaction(resLookupMasked);
```

##### Champs de demande pour les transactions de recherche d'un numéro masqué dans la chambre forte (obligatoires)

Pour obtenir une description de toutes les valeurs obligatoires et facultatives, consultez l'annexe A Définition des champs de demande.

Variable	Type et limites	Méthode Set
Clé de données	<i>Chaîne</i> 25 caractères alphanumériques	resLookupMasked.setData(data_key);

##### Exemple de transaction de recherche d'un numéro masqué dans la chambre forte

```
package Canada;
import JavaAPI.*;
public class TestCanadaResLookupMasked
{
    public static void main(String[] args)
    {
        String store_id = "store1";
        String api_token = "yesguy";
        String data_key = "pi3ZMZoTTM8pLM9wuwss2KBxw";
        String processing_country_code = "CA";
        boolean status_check = false;
        ResLookupMasked resLookupMasked = new ResLookupMasked();
        resLookupMasked.setData(data_key);
        HttpsPostRequest mpgReq = new HttpsPostRequest();
        mpgReq.setProcCountryCode(processing_country_code);
        mpgReq.setTestMode(true); //false for production transactions
        mpgReq.setstoreId(store_id);
        mpgReq.setApiToken(api_token);
        mpgReq.setTransaction(resLookupMasked);
        mpgReq.setStatusCheck(status_check);
        mpgReq.send();
        try
        {
            Receipt receipt = mpgReq.getReceipt();
            System.out.println("DataKey = " + receipt.getDataKey());
            System.out.println("ResponseCode = " + receipt.getResponseCode());
            System.out.println("Message = " + receipt.getMessage());
        }
```

```

System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("ResSuccess = " + receipt.getResSuccess());
System.out.println("PaymentType = " + receipt.getPaymentType());
System.out.println("Cust ID = " + receipt.getResCustomerId());
System.out.println("Phone = " + receipt.getResPhone());
System.out.println("Email = " + receipt.getResEmail());
System.out.println("Note = " + receipt.getResNote());
System.out.println("MaskedPan = " + receipt.getResMaskedPan());
System.out.println("Exp Date = " + receipt.getResExpdate());
System.out.println("Crypt Type = " + receipt.getResCryptType());
System.out.println("Avs Street Number = " + receipt.getResAvsStreetNumber());
System.out.println("Avs Street Name = " + receipt.getResAvsStreetName());
System.out.println("Avs Zipcode = " + receipt.getResAvsZipcode());
}
catch (Exception e)
{
e.printStackTrace();
}
}
}

```

### **Champs de réponse liés à la chambre forte**

Pour obtenir une liste et une explication des champs de réponse de l'objet Receipt pouvant apparaître après l'envoi d'une transaction utilisant la chambre forte, consultez l'Annexe B Définition des champs de réponse (page 547).

### **4.5.7 Obtention des cartes expirées dans la chambre forte (ResGetExpiring)**

Cette transaction vérifie les profils dont les cartes de crédit expirent durant les mois civils actuel et prochain.

**EXEMPLE :** Si vous traitez cette transaction le 30 septembre, toutes les cartes dont la date d'expiration est en septembre ou en octobre s'afficheront.

Lors de la génération d'un liste de profils avec des cartes de crédit expirées ou qui expireront prochainement, seuls les numéros de carte de crédit masqués s'affichent. Cette transaction ne peut être effectuée plus de deux fois par jour civil.

### **Définition de l'objet de transaction Vault Get Expiring**

```
ResGetExpiring resGetExpiring = new ResGetExpiring();
```

### **Objet HttpsPostRequest pour les transactions d'obtention des cartes expirées dans la chambre forte**

```
HttpsPostRequest mpgReq = new HttpsPostRequest();

mpgReq.setTransaction(resGetExpiring);
```

### **Champs de demande pour les transactions d'obtention des cartes expirées dans la chambre forte (obligatoires)**

Cette transaction n'a aucun champ de demande obligatoire.

### **Exemple de transaction d'obtention des cartes expirées dans la chambre forte**

```

package Canada;
import JavaAPI.*;
public class TestCanadaResGetExpiring
{
    public static void main(String[] args)
    {
        String store_id = "store1";
        String api_token = "yesguy";
        String processing_country_code = "CA";
        boolean status_check = false;
        ResGetExpiring resGetExpiring = new ResGetExpiring();
        HttpsPostRequest mpgReq = new HttpsPostRequest();
        mpgReq.setProcCountryCode(processing_country_code);
        mpgReq.setTestMode(true); //false for production transactions
        mpgReq.setstoreId(store_id);
        mpgReq.setApiToken(api_token);
        mpgReq.setTransaction(resGetExpiring);
        mpgReq.setStatusCheck(status_check);
        mpgReq.send();
        try
        {
            Receipt receipt = mpgReq.getReceipt();
            System.out.println("DataKey = " + receipt.getDataKey());
            System.out.println("ResponseCode = " + receipt.getResponseCode());
            System.out.println("Message = " + receipt.getMessage());
            System.out.println("TransDate = " + receipt.getTransDate());
            System.out.println("TransTime = " + receipt.getTransTime());
            System.out.println("Complete = " + receipt.getComplete());
            System.out.println("TimedOut = " + receipt.getTimedOut());
            System.out.println("ResSuccess = " + receipt.getResSuccess());
            System.out.println("PaymentType = " + receipt.getPaymentType());
            //ResolveData
            for (int index = 0; index < receipt.getExpiredCardCount(); index++)
            {
                System.out.println("\nDataKey = " + index);
                System.out.println("Payment Type = " + receipt.getExpPaymentType(index));
                System.out.println("Cust ID = " + receipt.getExpCustId(index));
                System.out.println("Phone = " + receipt.getExpPhone(index));
                System.out.println("Email = " + receipt.getExpEmail(index));
                System.out.println("Note = " + receipt.getExpNote(index));
                System.out.println("Masked Pan = " + receipt.getExpMaskedPan(index));
                System.out.println("Exp Date = " + receipt.getExpExdate(index));
                System.out.println("Crypt Type = " + receipt.getExpCryptType(index));
                System.out.println("Avs Street Number = " + receipt.getExpAvsStreetNumber(index));
                System.out.println("Avs Street Name = " + receipt.getExpAvsStreetName(index));
                System.out.println("Avs Zipcode = " + receipt.getExpAvsZipCode(index));
            }
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}

```

## Champs de réponse liés à la chambre forte

Pour obtenir une liste et une explication des champs de réponse de l'objet Receipt pouvant apparaître après l'envoi d'une transaction utilisant la chambre forte, consultez l'Annexe B Définition des champs de réponse (page 547).

#### 4.5.8 Détection de carte d'entreprise dans la chambre forte (ResIsCorporateCard)

Cette transaction détermine si une carte d'entreprise est enregistrée dans un profil.

Après l'envoi de la transaction, le champ de réponse `getCorporateCard` de l'objet `Receipt` indique soit true ou false, selon si la carte associée est une carte d'entreprise.

**REMARQUE :** Cette transaction prend en charge les jetons temporaires et permanents.

#### Définition de l'objet de transaction Vault Is Corporate Card

```
ResIsCorporatecard resIsCorporatecard = new ResIsCorporatecard();
```

#### Objet HttpsPostRequest pour les transactions de détection de carte d'entreprise dans la chambre forte

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setTransaction(ResIsCorporateCard);
```

#### Champs de demande pour les transactions de détection de carte d'entreprise dans la chambre forte (obligatoires)

Variable	Type et limites	Méthode Set
Clé de données	<i>Chaîne</i> 25 caractères alphanumériques	<code>resIsCorporatecard.setData(data_key);</code>

### Exemple de transaction de détection d'une carte d'entreprise dans la chambre forte

```

package Canada;
import JavaAPI.*;
public class TestCanadaResIs corporatecard
{
    public static void main(String[] args)
    {
        String store_id = "store1";
        String api_token = "yesguy";
        String data_key = "eLqsADfwqHDxIpJG9vLnELx01";
        String processing_country_code = "CA";
        boolean status_check = false;
        ResIs corporatecard resIs corporatecard = new ResIs corporatecard();
        resIs corporatecard.setdata(data_key);
        HttpsPostRequest mpgReq = new HttpsPostRequest();
        mpgReq.setProcCountryCode(processing_country_code);
        mpgReq.setTestMode(true); //false for production transactions
        mpgReq.setstoreId(store_id);
        mpgReq.setApiToken(api_token);
        mpgReq.setTransaction(resIs corporatecard);
        mpgReq.setStatusCheck(status_check);
        mpgReq.send();
        try
        {
            Receipt receipt = mpgReq.getReceipt();
            System.out.println("DataKey = " + receipt.getDataKey());
            System.out.println("CorporateCard = " + receipt.getCorporateCard());
            System.out.println("ResponseCode = " + receipt.getResponseCode());
            System.out.println("Message = " + receipt.getMessage());
            System.out.println("TransDate = " + receipt.getTransDate());
            System.out.println("TransTime = " + receipt.getTransTime());
            System.out.println("Complete = " + receipt.getComplete());
            System.out.println("TimedOut = " + receipt.getTimedOut());
            System.out.println("ResSuccess = " + receipt.getResSuccess());
            System.out.println("PaymentType = " + receipt.getPaymentType());
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}

```

### Champs de réponse liés à la chambre forte

Pour obtenir une liste et une explication des champs de réponse de l'objet Receipt pouvant apparaître après l'envoi d'une transaction utilisant la chambre forte, consultez l'Annexe B Définition des champs de réponse (page 547).

## 4.5.9 Ajout d'un jeton à la chambre forte (ResAddToken)

Cette transaction transforme un jeton temporaire obtenu par l'entremise de la transformation en jeton hébergée en jeton permanent de la chambre forte.

Un jeton temporaire est valide pour une durée de 15 minutes après sa création. Cette transaction doit être effectuée dans ce délai si le jeton doit être transformé en jeton permanent à des fins d'utilisation ultérieure.

À l'aide du jeton temporaire, envoyez une demande de transaction Achat avec la chambre forte, Préautorisation avec la chambre forte ou Vérification de carte avec la chambre forte en incluant l'objet Credential on File pour obtenir l'ID de l'émetteur.

### Définition de l'objet de transaction Vault Add Token

```
ResAddToken resAddToken = new ResAddToken();
```

### Objet HttpsPostRequest pour les transactions d'ajout d'un jeton à la chambre forte

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
```

```
mpgReq.setTransaction(resAddToken);
```

### Champs de demande pour les transactions d'ajout d'un jeton à la chambre forte (obligatoires)

Pour obtenir une description de toutes les valeurs obligatoires et facultatives, consultez l'annexe A Définition des champs de demande.

Variable	Type et limites	Méthode Set
Clé de données	<i>Chaîne</i> 25 caractères alphanumériques	resAddToken.setData(data_key);
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	resAddToken.setCryptType(crypt);
Renseignements d'identification au dossier cof	<i>Objet</i> S. O.	cof.setCofInfo(cof);
<b>REMARQUE :</b> Cette variable est un objet imbriqué au sein d'une transaction et elle est requise lors du stockage des renseignements d'identification d'un client ou lors de l'utilisation de tels renseignements stockés. Pour plus de renseignements sur les champs de l'objet Credential on File Info, consultez la section Objet Credential File Info et variables.		

**Champs de demande pour les transactions d'ajout d'un jeton à la chambre forte (facultatifs)****Tableau 1 : Valeurs facultatives pour les transaction d'ajout d'un jeton à la chambre forte**

Variable	Type et limites	Méthode Set
ID de client	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p> <div style="background-color: #e0f2f1; padding: 5px;"> <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : &lt; &gt; \$ % = ? ^ { } [ ] \          </div>	<code>resAddToken.setCustomerId(cust_id);</code>
Renseignements du SVA	<p><i>Objet</i></p> <p>S. O.</p>	<code>resAddToken.setAvsInfo(avsCheck);</code>
Adresse courriel	<p><i>Chaîne</i></p> <p>30 caractères alphanumériques</p>	<code>resAddToken.setEmail(email);</code>
Numéro de téléphone	<p><i>Chaîne</i></p> <p>30 caractères alphanumériques</p>	<code>resAddToken.setPhone(phone);</code>
Remarque	<p><i>Chaîne</i></p> <p>30 caractères alphanumériques</p>	<code>resAddToken.setNote(note);</code>
Format de la clé de données	<p><i>Chaîne</i></p> <p>2 caractères alphanumériques</p>	<code>resAddToken.setDataKeyFormat(data_key_format);</code>

## Champs de demande pour l'objet Credential on File Info

Variable	Type et limites	Méthode Set
ID de l'émetteur  <b>REMARQUE :</b> Cette variable est requise pour chaque transaction entamée par le commerçant à la suite de la première transaction. Après l'envoi de la première transaction, l'ID d'émetteur (issuer ID) est reçu dans la réponse de transaction, puis utilisé dans les demandes de transaction subséquentes.	<i>Chaîne</i> 15 caractères alphanumériques Longueur variable	cof.setIssuerId("VALUE_FOR_ISSUER_ID");  <b>REMARQUE :</b> Pour obtenir une liste et une explication des valeurs possibles à envoyer pour cette variable, consultez la section Définition des champs de demande – Renseignements d'identification au dossier (cof)

### Exemple de transaction d'ajout d'un jeton à la chambre forte

```

package Canada;
import JavaAPI.*;
public class TestCanadaResAddToken
{
    public static void main(String[] args)
    {
        String store_id = "store1";
        String api_token = "yesguy";
        String data_key = "ot-545454ucx87A5454";
        String expdate = "2001";
        String phone = "0000000000";
        String email = "bob@smith.com";
        String note = "my note";
        String cust_id = "customer1";
        String crypt_type = "7";
        String data_key_format = "0";
        String processing_country_code = "CA";
        boolean status_check = false;
        AvsInfo avsCheck = new AvsInfo();
        avsCheck.setAvsStreetNumber("212");
        avsCheck.setAvsStreetName("Payton Street");
        avsCheck.setAvsZipCode("M1M1M1");

        //Credential on File details
        CofInfo cof = new CofInfo();
        cof.setIssuerId("139X3130ASCXAS9");
        ResAddToken resAddToken = new ResAddToken();
        resAddToken.setDataKey(data_key);
        resAddToken.setCryptType(crypt_type);
        resAddToken.setExpdate(expdate);
        resAddToken.setCustomerId(cust_id);
        resAddToken.setPhone(phone);
        resAddToken.setEmail(email);
        resAddToken.setNote(note);
        resAddToken.setAvsInfo(avsCheck);
        resAddToken.setCofInfo(cof);
        //resAddToken.setaDataKeyFormat(data_key_format); //optional
        HttpsPostRequest mpgReq = new HttpsPostRequest();
        mpgReq.setProcCountryCode(processing_country_code);
        mpgReq.setTestMode(true); //false for production transactions
        mpgReq.setstoreId(store_id);
        mpgReq.setApiToken(api_token);
        mpgReq.setTransaction(resAddToken);
        mpgReq.setStatusCheck(status_check);
        mpgReq.send();
        try
        {
            Receipt receipt = mpgReq.getReceipt();
        }
    }
}

```

```

System.out.println("DataKey = " + receipt.getDataKey());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("Message = " + receipt.getMessage());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("ResSuccess = " + receipt.getResSuccess());
System.out.println("PaymentType = " + receipt.getPaymentType());
System.out.println("Cust ID = " + receipt.getResCustomerId());
System.out.println("Phone = " + receipt.getResPhone());
System.out.println("Email = " + receipt.getResEmail());
System.out.println("Note = " + receipt.getResNote());
System.out.println("MaskedPan = " + receipt.getResMaskedPan());
System.out.println("Exp Date = " + receipt.getResExpdate());
System.out.println("Crypt Type = " + receipt.getResCryptType());
System.out.println("Avs Street Number = " + receipt.getResAvsStreetNumber());
System.out.println("Avs Street Name = " + receipt.getResAvsStreetName());
System.out.println("Avs Zipcode = " + receipt.getResAvsZipcode());
}
catch (Exception e)
{
e.printStackTrace();
}
}
}

```

### Champs de réponse liés à la chambre forte

Pour obtenir une liste et une explication des champs de réponse de l'objet Receipt pouvant apparaître après l'envoi d'une transaction utilisant la chambre forte, consultez l'Annexe B Définition des champs de réponse (page 547).

### 4.5.10 Transformation en jeton d'une carte de crédit dans la chambre forte (ResTokenizeCC)

Cette transaction crée un nouveau profil de carte de crédit en utilisant le numéro de carte de crédit, la date d'expiration et l'indicateur de commerce électronique fournis dans une transaction financière antérieure. Les transactions antérieures qui doivent être transformées en jeton doivent avoir inclus l'objet Credential on File Info.

L'ID de l'émetteur reçu dans la réponse de ces transactions est envoyé dans la demande de transformation en jeton d'une carte de crédit dans la chambre forte afin d'indiquer qu'il s'agit de renseignements d'identification au dossier. Si vous avez besoin de cet identifiant de l'émetteur dans la réponse à cette demande, incluez **return\_issuer\_ID** avec la valeur `true`; cela permet de récupérer l'identifiant de l'émetteur à partir de la transaction financière précédente.

Voici les transactions de base pouvant être utilisées pour la transformation en jeton :

- Achat
- Préautorisation
- Vérification de la carte

Le processus de transformation en jeton est illustré ci-dessous :

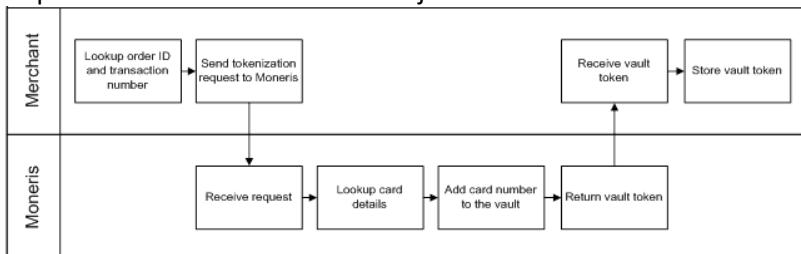


Image 1 : Processus de transformation en jeton

### Définition de l'objet de transaction Vault Tokenize Credit Card

```
ResTokenizeCC resTokenizeCC = new ResTokenizeCC();
```

### Objet HttpsPostRequest pour les transactions de transformation en jeton d'une carte dans la chambre forte

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
```

```
mpgReq.setTransaction(resTokenizeCC);
```

### Champs de demande pour les transactions de transformation en jeton d'une carte dans la chambre forte (obligatoires)

Ces valeurs obligatoires font référence à une transaction financière par carte de crédit traitée précédemment. Le numéro de carte de crédit, la date d'expiration et l'indicateur de commerce électronique de la transaction d'origine sont enregistrés dans la chambre forte afin d'être utilisés lors d'une transaction ultérieure utilisant la chambre forte.

Variable	Type et limites	Méthode Set
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques a-Z A-Z 0-9 _ - : . @ espaces	resTokenizeCC.setOrderId(order_id);
Numéro de transaction	<i>Chaîne</i> 255 caractères (alphanumériques, trait d'union ou trait de soulignement) Longueur variable	resTokenizeCC.SetTxnNumber(txn_number);  resTokenizeCC.setTxnNumber(txn_number);

### Champs de demande pour les transactions de transformation en jeton d'une carte dans la chambre forte (facultatifs)

Variable	Type et limites	Méthode Set
ID de client	<i>Chaîne</i> 50 caractères alphanumériques <div style="background-color: #e0f2e0; padding: 10px; margin-top: 10px;"> <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés :            &lt;&gt;\$%=?^{}[]\         </div>	resTokenizeCC.setCustId(cust_id);

return issuer ID	<i>Valeur booléenne</i> true/false	resTokenizeCC.setReturnIssuerId( false);
Adresse courriel	<i>Chaîne</i>  30 caractères alphanumériques	resTokenizeCC.setEmail(email);
Numéro de téléphone	<i>Chaîne</i>  30 caractères alphanumériques	resTokenizeCC.setPhone(phone);
Remarque	<i>Chaîne</i>  30 caractères alphanumériques	resTokenizeCC.setNote(note);
Renseignements du SVA	<i>Objet</i>  S. O.	resTokenizeCC.setAvsInfo(avscH eck);
Format de la clé de données	<i>Chaîne</i>  2 caractères alphanumériques	resTokenizeCC.setDataKeyFormat (data_key_format);
Renseignements d'identification au dossier  cof	<i>Objet</i>  S. O.	cof.setCofInfo(cof);
<b>REMARQUE :</b> Cette variable est un objet imbriqué au sein d'une transaction et elle est requise lors du stockage des renseignements d'identification d'un client ou lors de l'utilisation de tels renseignements stockés. Pour plus de renseignements sur les champs de l'objet Credential on File Info, consultez la section Objet Credential File Info et variables.		

## Champs de demande pour l'objet Credential on File Info

Variable	Type et limites	Méthode Set
ID de l'émetteur  <b>REMARQUE :</b> Cette variable est requise pour chaque transaction entamée par le commerçant à la suite de la première transaction. Après l'envoi de la première transaction, l'ID d'émetteur (issuer ID) est reçu dans la réponse de transaction, puis utilisé dans les demandes de transaction subséquentes.	<i>Chaîne</i> 15 caractères alphanumériques Longueur variable	cof.setIssuerId("VALUE_FOR_ISSUER_ID");  <b>REMARQUE :</b> Pour obtenir une liste et une explication des valeurs possibles à envoyer pour cette variable, consultez la section Définition des champs de demande – Renseignements d'identification au dossier (cof)

Les champs non inclus dans la demande de transformation en jeton ne seront pas enregistrés avec la transaction. En d'autres mots, Passerelle Moneris ne prend pas automatiquement les renseignements facultatifs inclus dans la transaction d'origine.

L'objet ResolveData inclus dans les champs de réponse indique les valeurs enregistrées pour le profil.

### Exemple de transaction de transformation en jeton d'une carte dans la chambre forte

```
package Canada;
import JavaAPI.*;
public class TestCanadaResTokenizeCC
{
    public static void main(String[] args)
    {
        String store_id = "moneris";
        String api_token = "hungle";
        String order_id = "mvt3212954335";
        String txn_number = "199999-0_10";
        String phone = "0000000000";
        String email = "bob@smith.com";
        String note = "my note";
        String cust_id = "customer1";
        String data_key_format = "0";
        String processing_country_code = "CA";
        boolean status_check = false;
        AvsInfo avsCheck = new AvsInfo();
        avsCheck.setAvsStreetNumber("212");
        avsCheck.setAvsStreetName("Payton Street");
        avsCheck.setAvsZipCode("M1M1M1");

        //Credential on File details
        CofInfo cof = new CofInfo();
        cof.setIssuerId("139X3130ASCXAS9");
        ResTokenizeCC resTokenizeCC = new ResTokenizeCC();
        resTokenizeCC.setOrderId(order_id);
        resTokenizeCC.setTxnNumber(txn_number);
        resTokenizeCC.setCustId(cust_id);
        resTokenizeCC.setPhone(phone);
        resTokenizeCC.setEmail(email);
        resTokenizeCC.setNote(note);
        resTokenizeCC.setAvsInfo(avsCheck);
        resTokenizeCC.setCofInfo(cof);
        //resTokenizeCC.setDataKeyFormat(data_key_format); //optional

        HttpsPostRequest mpgReq = new HttpsPostRequest();
        mpgReq.setProcCountryCode(processing_country_code);
        mpgReq.setTestMode(true); //false for production transactions
        mpgReq.setstoreId(store_id);
```

```

mpgReq.setApiToken(api_token);
mpgReq.setTransaction(resTokenizeCC);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("DataKey = " + receipt.getDataKey());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("Message = " + receipt.getMessage());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("ResSuccess = " + receipt.getResSuccess());
System.out.println("PaymentType = " + receipt.getPaymentType());
//ResolveData
System.out.println("Cust ID = " + receipt.getResCustomerId());
System.out.println("Phone = " + receipt.getResPhone());
System.out.println("Email = " + receipt.getResEmail());
System.out.println("Note = " + receipt.getResNote());
System.out.println("MaskedPan = " + receipt.getResMaskedPan());
System.out.println("Exp Date = " + receipt.getResExpdate());
System.out.println("Crypt Type = " + receipt.getResCryptType());
System.out.println("Avs Street Number = " + receipt.getResAvsStreetNumber());
System.out.println("Avs Street Name = " + receipt.getResAvsStreetName());
System.out.println("Avs Zipcode = " + receipt.getResAvsZipcode());
}
catch (Exception e)
{
e.printStackTrace();
}
}
}

```

## Champs de réponse liés à la chambre forte

Pour obtenir une liste et une explication des champs de réponse de l'objet Receipt pouvant apparaître après l'envoi d'une transaction utilisant la chambre forte, consultez l'Annexe B Définition des champs de réponse (page 547).

## 4.6 Transactions financières utilisant la chambre forte

Après la conclusion d'une transaction financière, les champs de réponse indiquent toutes les valeurs actuellement enregistrées dans le profil utilisé.

### 4.6.1 Changements apportés à l'ID du client

Certaines transactions financières considèrent l'ID de client comme une valeur facultative. Il se peut que l'ID de client soit déjà enregistré, ou non, dans le profil de la chambre forte lors de l'envoi de la transaction. Il est donc possible de modifier la valeur de l'ID de client en traitant une transaction financière.

Le tableau ci-dessous indique la valeur de l'ID de client dans le champ de réponse après le traitement d'une transaction financière.

**Tableau 2 : ID de client utilisé dans les champs de réponse**

Déjà dans le profil?	Transféré?	Version utilisée dans la réponse
Non	Non	ID de client non utilisé dans la transaction
Non	Oui	Transféré
Oui	Non	Profil
Oui	Oui	Transféré

#### 4.6.2 Achat avec la chambre forte (ResPurchaseCC)

**REMARQUE :** Cette transaction prend en charge les jetons temporaires et permanents.

##### Définition de l'objet de transaction Purchase with Vault

```
ResPurchaseCC resPurchaseCC = new ResPurchaseCC();
```

##### Objet HttpsPostRequest pour les transactions d'achat avec la chambre forte

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
```

```
mpgReq.setTransaction(resPurchaseCC);
```

##### Champs de demande liés aux transactions d'achat avec la chambre forte (obligatoires)

Pour obtenir une description de toutes les valeurs obligatoires et facultatives, consultez l'annexe A

Définition des champs de demande.

Variable	Type et limites	Méthode Set
Clé de données	<i>Chaîne</i> 25 caractères alphanumériques	resPurchaseCC.setData(data_key);
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques a-Z A-Z 0-9 _ - : . @ espaces	resPurchaseCC.setOrderId(order_id);

Montant	<i>Chaîne</i>	resPurchaseCC.setAmount(amount);
	10 caractères décimaux Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal	
	<b>EXEMPLE : 1 234 567,89</b>	
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	resPurchaseCC.setCryptType(crypt);
Renseignements d'identification au dossier cof	<i>Objet</i> S. O.	cof.setCofInfo(cof);
<b>REMARQUE :</b> Cette variable est un objet imbriqué au sein d'une transaction et elle est requise lors du stockage des renseignements d'identification d'un client ou lors de l'utilisation de tels renseignements stockés. Pour plus de renseignements sur les champs de l'objet Credential on File Info, consultez la section Objet Credential File Info et variables.		

#### Champs de demande liés aux transactions d'achat avec la chambre forte (facultatifs)

Variable	Type et limites	Méthode Set
Vérification d'état	<i>Valeur booléenne</i> true/false	mpgReq.setStatusCheck(status_check);
Date d'expiration	<i>Chaîne</i> 4 caractères numériques Format AAMM (Remarque : Ce format est différent de la date affichée sur la carte, qui est affichée au format MMAA)	resPurchaseCC.setExpDate(expiry_date);

ID de client	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p> <div style="background-color: #e0f7fa; padding: 5px;"> <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés :            &lt;&gt;\$%=?^{}[]\         </div>	resPurchaseCC.setCustId(cust_id);
Descripteur dynamique	<p><i>Chaîne</i></p> <p>20 caractères alphanumériques</p> <p>Total de 22 caractères incluant votre nom de commerçant et un séparateur</p> <div style="background-color: #e0f7fa; padding: 5px;"> <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés :            &lt;&gt;\$%=?^{}[]\         </div>	resPurchaseCC.setDynamicDescriptor(dynamic_descriptor);
Renseignements du client	<p><i>Objet</i></p> <p>S. O.</p>	resPurchaseCC.setCustInfo(customer);
Renseignements du SVA	<p><i>Objet</i></p> <p>S. O.</p>	resPurchaseCC.setAvsInfo(avscCheck);
Renseignements du NVC	<p><i>Objet</i></p> <p>S. O.</p>	resPurchaseCC.setCvdInfo(cvdCheck);
REMARQUE : Lorsque les renseignements d'identification sont stockés lors de la première transaction, l'objet CVD (numéro de vérification de carte, ou NVC) doit être envoyé. Pour les transactions subséquentes utilisant les renseignements d'identification stockés, le NVC peut être envoyé avec les transactions entamées par le titulaire de carte seulement. <b>Les commerçants ne doivent pas enregistrer le NVC.</b>		
Facturation périodique	<p><i>Objet</i></p> <p>S. O.</p>	resPurchaseCC.setRecurInfo(recurInfo);

Information sur le versement  Pour les champs liés à cet objet, consultez la section 8.6 Objet Installment Info	<i>Objet</i>  S. O.	resPurchaseCC.setInstallmentInfo(installmentInfo);
<b>REMARQUE :</b> N'envoyez pas l'objet Installment Info lors d'une transaction qui n'offre pas la fonction Versements Visa, car la transaction pourrait échouer		

### Champs de demande pour l'objet Credential on File Info

Variable	Type et limites	
ID de l'émetteur  <b>REMARQUE :</b> Cette variable est requise pour chaque transaction entamée par le commerçant à la suite de la première transaction. Après l'envoi de la première transaction, l'ID d'émetteur (issuer ID) est reçu dans la réponse de transaction, puis utilisé dans les demandes de transaction subséquentes.	<i>Chaîne</i>  15 caractères alphanumériques  Longueur variable	cof.setIssuerId("VALUE_FOR_ISSUER_ID");  <b>REMARQUE :</b> Pour obtenir une liste et une explication des valeurs possibles à envoyer pour cette variable, consultez la section Définition des champs de demande – Renseignements d'identification au dossier (cof)
Indicateur de paiement  <b>REMARQUE :</b> Lors d'une transaction utilisant les renseignements d'identification au dossier pour laquelle le champ de demande e-commerce indicator est également inclus, les valeurs acceptables de ce champ dépendent de la valeur envoyée pour l'indicateur de paiement. Pour plus d'information, consultez les définitions des champs de réponse.	<i>Chaîne</i>  1 caractère alphabétique	cof.setPaymentIndicator("PAYMENT_INDICATOR_VALUE");  <b>REMARQUE :</b> Pour obtenir une liste et une explication des valeurs possibles à envoyer pour cette variable, consultez la section Définition des champs de demande – Renseignements d'identification au dossier (cof)
Information de paiement	<i>Chaîne</i>  1 caractère numérique	cof.setPaymentInformation("PAYMENT_INFO_VALUE");  <b>REMARQUE :</b> Pour obtenir une liste et une explication des valeurs possibles à envoyer pour cette variable, consultez la section Définition des champs de demande – Renseignements d'identification au dossier (cof)

### Exemple de transaction d'achat avec la chambre forte

```

package Canada;
import JavaAPI.*;
public class TestCanadaResPurchaseCC
{
    public static void main(String[] args)
    {
        java.util.Date createDate = new java.util.Date();
        String order_id = "Test"+createDate.getTime();
        String store_id = "store5";
        String api_token = "yesguy";
        String data_key = "800XGiwxgvfbZngigVFeld9d2";
        String amount = "1.00";
        String cust_id = "customer1"; //if sent will be submitted, otherwise cust_id from profile will
        be used
        String crypt_type = "1";
        String descriptor = "my descriptor";
        String processing_country_code = "CA";
        String expdate = "1512"; //For Temp Token
        boolean status_check = false;

        ResPurchaseCC resPurchaseCC = new ResPurchaseCC();
        resPurchaseCC.setDataKey(data_key);
        resPurchaseCC.setOrderId(order_id);
        resPurchaseCC.setCustId(cust_id);
        resPurchaseCC.setAmount(amount);
        resPurchaseCC.setCryptType(crypt_type);
        //resPurchaseCC.setDynamicDescriptor(descriptor);
        resPurchaseCC.setExpDate(expdate); //Temp Tokens only

        //NT Response Option
        boolean get_nt_response = true;
        resPurchaseCC.setGetNtResponse(get_nt_response);

        //optional - Installment Info
        // InstallmentInfo installmentInfo = new InstallmentInfo();
        // installmentInfo.setPlanId("ae859ef1-eb91-b708-8b80-1dd481746401");
        // installmentInfo.setPlanIdRef("0000000065");
        // installmentInfo.setTacVersion("2");
        // resPurchaseCC.setInstallmentInfo(installmentInfo);

        //Mandatory - Credential on File details
        CofInfo cof = new CofInfo();
        cof.setPaymentIndicator("U");
        cof.setPaymentInformation("2");
        cof.setIssuerId("139X3130ASCXAS9");
        resPurchaseCC.setCofInfo(cof);

        HttpsPostRequest mpgReq = new HttpsPostRequest();
        mpgReq.setProcCountryCode(processing_country_code);
        mpgReq.setTestMode(true); //false for production transactions
        mpgReq.setstoreId(store_id);
        mpgReq.setApiToken(api_token);
        mpgReq.setTransaction(resPurchaseCC);
        mpgReq.setStatusCheck(status_check);
        mpgReq.send();
        try
        {
            Receipt receipt = mpgReq.getReceipt();
            System.out.println("DataKey = " + receipt.getDataKey());
            System.out.println("ReceiptId = " + receipt.getReceiptId());
            System.out.println("ReferenceNum = " + receipt.getReferenceNum());
            System.out.println("ResponseCode = " + receipt.getResponseCode());
            System.out.println("AuthCode = " + receipt.getAuthCode());
            System.out.println("Message = " + receipt.getMessage());
            System.out.println("TransDate = " + receipt.getTransDate());
            System.out.println("TransTime = " + receipt.getTransTime());
            System.out.println("TransType = " + receipt.getTransType());
            System.out.println("Complete = " + receipt.getComplete());
            System.out.println("TransAmount = " + receipt.getTransAmount());
            System.out.println("CardType = " + receipt.getCardType());
            System.out.println("TxnNumber = " + receipt.getTxnNumber());
            System.out.println("TimedOut = " + receipt.getTimedOut());
            System.out.println("ResSuccess = " + receipt.getResSuccess());
            System.out.println("PaymentType = " + receipt.getPaymentType());
        }
    }
}

```

```

System.out.println("IsVisaDebit = " + receipt.getIsVisaDebit());
System.out.println("Cust ID = " + receipt.getResCustomerId());
System.out.println("Phone = " + receipt.getResPhone());
System.out.println("Email = " + receipt.getResEmail());
System.out.println("Note = " + receipt.getResNote());
System.out.println("Masked Pan = " + receipt.getResMaskedPan());
System.out.println("Exp Date = " + receipt.getResExpdate());
System.out.println("Crypt Type = " + receipt.getResCryptType());
System.out.println("Avs Street Number = " + receipt.getResAvsStreetNumber());
System.out.println("Avs Street Name = " + receipt.getResAvsStreetName());
System.out.println("Avs Zipcode = " + receipt.getResAvsZipcode());
System.out.println("IssuerId = " + receipt.getIssuerId());
System.out.println("SourcePanLast4 = " + receipt.getSourcePanLast4());

if(get_nt_response) {
System.out.println("\nNTResponseCode = " + receipt.getNTResponseCode());
System.out.println("NTMessage = " + receipt.getNTMessage());
System.out.println("NTUsed = " + receipt.getNTUsed());
System.out.println("NTTokenBin = " + receipt.getNTTokenBin());
System.out.println("NTTokenLast4 = " + receipt.getNTTokenLast4());
System.out.println("NTTokenExpDate = " + receipt.getNTTokenExpDate());
}

// InstallmentResults installmentResults = receipt.getInstallmentResults();
// System.out.println("\nPlanId = " + installmentResults.getPlanId() +"\n");
// System.out.println("PlanIDRef = " + installmentResults.getPlanIDRef());
// System.out.println("TacVersion = " + installmentResults.getTacVersion());
// System.out.println("PlanAcceptanceId = " + installmentResults.getPlanAcceptanceId());
// System.out.println("PlanStatus = " + installmentResults.getPlanStatus());
// System.out.println("PlanResponse = " + installmentResults.getPlanResponse());
}

catch (Exception e)
{
e.printStackTrace();
}
}
}
}

```

## Champs de réponse liés à la chambre forte

Pour obtenir une liste et une explication des champs de réponse de l'objet Receipt pouvant apparaître après l'envoi d'une transaction utilisant la chambre forte, consultez l'Annexe B Définition des champs de réponse (page 547).

### 4.6.3 Préautorisation avec la chambre forte (ResPreauthCC)

**REMARQUE :** Cette transaction prend en charge les jetons temporaires et permanents.

#### Définition de l'objet de transaction Pre-Authorization with Vault

```
ResPreauthCC resPreauthCC = new ResPreauthCC();
```

#### Objet HttpsPostRequest pour les transactions de préautorisation avec la chambre forte

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setTransaction(resPreauthCC);
```

**Champs de demande pour les transactions de préautorisation avec la chambre forte (obligatoires)**

Variable	Type et limites	Méthode Set
Clé de données	<i>Chaîne</i> 25 caractères alphanumériques	<code>resPreauthCC.setData(data_key);</code>
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques a-Z A-Z 0-9 _ - : . @ espaces	<code>resPreauthCC.setOrderId(order_id);</code>
Montant	<i>Chaîne</i> 10 caractères décimaux Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal	<code>resPreauthCC.setAmount(amount);</code>
	<b>EXEMPLE : 1 234 567,89</b>	
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	<code>resPreauthCC.setCryptType(crypt);</code>
Renseignements d'identification au dossier cof	<i>Objet</i> S. O.	<code>resPreauthCC.setCofInfo(cof);</code>
<b>REMARQUE :</b> Cette variable est un objet imbriqué au sein d'une transaction et elle est requise lors du stockage des renseignements d'identification d'un client ou lors de l'utilisation de tels renseignements stockés. Pour plus de renseignements sur les champs de l'objet Credential on File Info, consultez la section Objet Credential File Info et variables.		

**Champs de demande pour les transactions de préautorisation avec la chambre forte ( facultatifs )**

Valeur	Limites	Méthode Set
Vérification d'état	<i>Valeur booléenne</i> true/false	<code>mpgReq.setStatusCheck(status_check);</code>
Descripteur dynamique	<i>Chaîne</i>  20 caractères alphanumériques  Total de 22 caractères incluant votre nom de commerçant et un séparateur	<code>resPreauthCC.setDynamicDescriptor(dynamic_descriptor);</code>
<b>Pour les transactions de préautorisation REMARQUE :</b> La valeur du champ Dynamic descriptor est uniquement transférée à une conclusion de préautorisation lorsque cette dernière est effectuée par l'entremise du centre de ressources pour commerçants. Autrement, la valeur du champ doit être renvoyée durant la conclusion de préautorisation.	<b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : <code>&lt;&gt;\$%=?^{}[]\</code>	
is incremental  is_incremental	<i>Valeur booléenne</i> true/false	<code>resPreauthCC.setIsIncremental(is_incremental);</code>  Indique si cette préautorisation utilise un montant estimé. Les estimations permettent d'incrémenter le montant retenu au moyen de demandes subséquentes d'autorisation incrémentale. La valeur par défaut est « false ».
		{b}REMARQUE : {/b}veuillez noter que si ce champ contient la valeur « true », la préautorisation n'est valable que pour une seule conclusion de préautorisation. Toute conclusion soumise à titre de conclusion partielle est traitée comme une conclusion complète (ship_indicator= P est traité comme ship_indicator= F lorsque, dans la préautorisation originale, le paramètre is_incremental= true).
Date d'expiration	<i>Chaîne</i>	<code>resPreauthCC.setExpDate(expiry_date);</code>

<b>REMARQUE :</b> Ce champ est requis lors du référencement d'un jeton temporaire lors d'une transaction Versements Visa.	4 caractères alphanumériques  AAMM	
ID de client	<i>Chaîne</i> 50 caractères alphanumériques <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés :            &lt;&gt;\$%=?^{}[]\         </div>	resPreauthCC.setCustId(cust_id);
Autorisation finale  <b>REMARQUE :</b> Applicable uniquement aux transactions par carte Mastercard	<i>Chaîne</i> true/false	resPreauthCC.setFinalAuth("true");

Valeur	Limites	Méthode Set
Renseignements du client	<i>Objet</i> S. O.	<code>resPreauthCC.setCustInfo(cus tomer);</code>
Renseignements du SVA	<i>Objet</i> S. O.	<code>resPreauthCC.setAvsInfo(avsc heck);</code>
Renseignements du NVC	<i>Objet</i> S. O.	<code>resPreauthCC.setCvdInfo(cvdC heck);</code>
<b>REMARQUE :</b> Lorsque les renseignements d'identification sont stockés lors de la première transaction, l'objet CVD (numéro de vérification de carte, ou NVC) doit être envoyé. Pour les transactions subséquentes utilisant les renseignements d'identification stockés, le NVC peut être envoyé avec les transactions entamées par le titulaire de carte seulement. <b>Les commerçants ne doivent pas enregistrer le NVC.</b>		
Information sur le versement  Pour les champs liés à cet objet, consultez la section 8.6 Objet Installment Info	<i>Objet</i> S. O.	<code>resPreauthCC.setInstallmentInf o(installmentInfo);</code>
<b>REMARQUE :</b> N'envoyez pas l'objet Installment Info lors d'une transaction qui n'offre pas la fonction Versements Visa, car la transaction pourrait échouer		

#### Champs de demande pour l'objet Credential on File Info

Variable	Type et limites	
ID de l'émetteur	<i>Chaîne</i> 15 caractères alphanumériques	<code>cof.setIssuerId("VALUE_FOR_ISSU ER_ID");</code>
<b>REMARQUE :</b> Cette variable est requise pour chaque transaction entamée par le commerçant à la suite de la première transaction. Après l'envoi de la première	Longueur variable	<b>REMARQUE :</b> Pour obtenir une liste et une explication des valeurs possibles à envoyer pour cette variable, consultez la section Définition des champs de demande – Renseignements

Variable	Type et limites	
transaction, l'ID d'émetteur (issuer ID) est reçu dans la réponse de transaction, puis utilisé dans les demandes de transaction subséquentes.		d'identification au dossier (cof)
Indicateur de paiement	<p><i>Chaîne</i> 1 caractère alphabétique</p> <p><b>REMARQUE :</b> Lors d'une transaction utilisant les renseignements d'identification au dossier pour laquelle le champ de demande e-commerce indicator est également inclus, les valeurs acceptables de ce champ dépendent de la valeur envoyée pour l'indicateur de paiement. Pour plus d'information, consultez les définitions des champs de réponse.</p>	<pre>cof.setPaymentIndicator ("PAYMENT_INDICATOR_VALUE");</pre> <p><b>REMARQUE :</b> Pour obtenir une liste et une explication des valeurs possibles à envoyer pour cette variable, consultez la section Définition des champs de demande – Renseignements d'identification au dossier (cof)</p>
Information de paiement	<p><i>Chaîne</i> 1 caractère numérique</p>	<pre>cof.setPaymentInformation ("PAYMENT_INFO_VALUE");</pre> <p><b>REMARQUE :</b> Pour obtenir une liste et une explication des valeurs possibles à envoyer pour cette variable, consultez la section Définition des champs de demande – Renseignements d'identification au dossier (cof)</p>

#### Exemple de transaction de préautorisation avec la chambre forte

```

package Canada;
import JavaAPI.*;
public class TestCanadaResPreauthCC
{
    public static void main(String[] args)
    {
        java.util.Date createDate = new java.util.Date();
        String order_id = "Test"+createDate.getTime();
        String store_id = "store5";
        String api_token = "yesguy";
        String data_key = "rS7DbroQHJmJxdBfXFXiauQc4";
        String amount = "1.00";
        String cust_id = "customer1"; //if sent will be submitted, otherwise cust_id from profile will be used
        String crypt_type = "1";
        String dynamic_descriptor = "my descriptor";
        String processing_country_code = "CA";
        String expdate = "1712"; //For Temp Token
        boolean status_check = false;

        ResPreauthCC resPreauthCC = new ResPreauthCC();
        resPreauthCC.setDataKey(data_key);
        resPreauthCC.setOrderId(order_id);
        resPreauthCC.setCustomerId(cust_id);
        resPreauthCC.setAmount(amount);
        resPreauthCC.setCryptType(crypt_type);
    }
}

```

```

resPreauthCC.setInstallmentInfo(installmentInfo);
//resPreauthCC.setDynamicDescriptor(dynamic_descriptor);
//resPreauthCC.setExpDate(expdate); //Temp Tokens only

//NT Response Option
boolean get_nt_response = true;
resPreauthCC.setGetNtResponse(get_nt_response);

//optional - Installment Info
// InstallmentInfo installmentInfo = new InstallmentInfo();
// installmentInfo.setPlanId("ae859ef1-eb91-b708-8b80-1dd481746401");
// installmentInfo.setPlanIdRef("0000000065");
// installmentInfo.setTacVersion("2");
// resPreauthCC.setInstallmentInfo(installmentInfo);

//Mandatory - Credential on File details
CofInfo cof = new CofInfo();
cof.setPaymentIndicator("U");
cof.setPaymentInformation("2");
cof.setIssuerId("139X3130ASCXAS9");

resPreauthCC.setCofInfo(cof);

HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false for production transactions
mpgReq.setstoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(resPreauthCC);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("DataKey = " + receipt.getDataKey());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Message = " + receipt.getMessage());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("ResSuccess = " + receipt.getResSuccess());
System.out.println("PaymentType = " + receipt.getPaymentType());
System.out.println("IsVisaDebit = " + receipt.getIsVisaDebit());
System.out.println("IsCorporate = " + receipt.getCorporateCard());
System.out.println("Cust ID = " + receipt.getResCustId());
System.out.println("Phone = " + receipt.getResPhone());
System.out.println("Email = " + receipt.getResEmail());
System.out.println("Note = " + receipt.getResNote());
System.out.println("Masked Pan = " + receipt.getResMaskedPan());
System.out.println("Exp Date = " + receipt.getResExpdate());
System.out.println("Crypt Type = " + receipt.getResCryptType());
System.out.println("Avs Street Number = " + receipt.getResAvsStreetNumber());
System.out.println("Avs Street Name = " + receipt.getResAvsStreetName());
System.out.println("Avs Zipcode = " + receipt.getResAvsZipcode());
System.out.println("IssuerId = " + receipt.getIssuerId());
System.out.println("SourcePanLast4 = " + receipt.getSourcePanLast4());

if(get_nt_response) {
System.out.println("\nNTResponseCode = " + receipt.getNtResponseCode());
System.out.println("NTMessage = " + receipt.getNtMessage());
System.out.println("NTUsed = " + receipt.getNtUsed());
System.out.println("NTTokenBin = " + receipt.getNtTokenBin());
System.out.println("NTTokenLast4 = " + receipt.getNtTokenLast4());
System.out.println("NTTokenExpDate = " + receipt.getNtTokenExpDate());
}

// InstallmentResults installmentResults = receipt.getInstallmentResults();
// System.out.println("\nPlanId = " + installmentResults.getPlanId());
// System.out.println("PlanIDRef = " + installmentResults.getPlanIDRef());
// System.out.println("TacVersion = " + installmentResults.getTacVersion());
// System.out.println("PlanAcceptanceId = " + installmentResults.getPlanAcceptanceId());
// System.out.println("PlanStatus = " + installmentResults.getPlanStatus());
// System.out.println("PlanResponse = " + installmentResults.getPlanResponse());
}

```

```

        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}

```

### **Champs de réponse liés à la chambre forte**

Pour obtenir une liste et une explication des champs de réponse de l'objet Receipt pouvant apparaître après l'envoi d'une transaction utilisant la chambre forte, consultez l'Annexe B Définition des champs de réponse (page 547).

#### **4.6.4 Transaction de remboursement indépendant avec la chambre forte (ResIndRefundCC)**

**REMARQUE :** Cette transaction prend en charge les jetons temporaires et permanents.

#### **Définition de l'objet de transaction Vault Independent Refund**

```
ResIndRefundCC resIndRefundCC = new ResIndRefundCC();
```

#### **Objet HttpsPostRequest pour les transactions de remboursement indépendant avec la chambre forte**

```
HttpsPostRequest mpgReq = new HttpsPostRequest();

mpgReq.setTransaction(resIndRefundCC);
```

#### **Valeurs des transactions de remboursement indépendant avec la chambre forte**

Pour obtenir une description de toutes les valeurs obligatoires et facultatives, consultez l'annexe A Définition des champs de demande.

Variable	Type et limites	Méthode Set
Clé de données	<i>Chaîne</i> 25 caractères alphanumériques	resIndRefundCC.setData(data_key);
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques a-Z A-Z 0-9 _ - : . @ espaces	resIndRefundCC.setOrderId(order_id);

Variable	Type et limites	Méthode Set
Montant	<p><i>Chaîne</i></p> <p>10 caractères décimaux</p> <p>Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal</p> <p><b>EXEMPLE : 1 234 567,89</b></p>	<code>resIndRefundCC.setAmount(amount);</code>
Indicateur de commerce électronique	<p><i>Chaîne</i></p> <p>1 caractère alphanumérique</p>	<code>resIndRefundCC.setCryptType(crypt);</code>

#### Champs de demande liés aux transactions de remboursement indépendant avec la chambre forte (facultatifs)

Valeur	Limites	Méthode Set
ID de client	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p> <p><b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : &lt; &gt; \$ % = ? ^ { } [ ] \</p>	<code>resIndRefundCC.setCustomerId(cust_id);</code>
Date d'expiration	<p><i>Chaîne</i></p> <p>4 caractères alphanumériques</p> <p>AAMM</p>	<code>resIndRefundCC.setExpDate(expiry_date);</code>
Vérification d'état	<p><i>Valeur booléenne</i></p> <p>true/false</p>	<code>mpgReq.setStatusCheck(status_check);</code>
Descripteur dynamique	<p><i>Chaîne</i></p> <p>20 caractères alphanumériques</p> <p>Total de 22 caractères</p>	<code>resIndRefundCC.setDynamicDescriptor(dynamic_descriptor);</code>

Valeur	Limites	Méthode Set
	incluant votre nom de commerçant et un séparateur <div style="background-color: #e0f2fd; padding: 10px; border: 1px solid #80d9ff; width: fit-content; margin-left: auto; margin-right: 0;"> <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés :            &lt; &gt; \$ % = ? ^ { } [ ] \          </div>	

### Exemple de transaction de remboursement indépendant avec la chambre forte

```

package Canada;
import JavaAPI.*;
public class TestCanadaResIndRefundCC
{
  public static void main(String[] args)
  {
    java.util.Date createDate = new java.util.Date();
    String order_id = "Test"+createDate.getTime();
    String store_id = "moneris";
    String api_token = "hurgle";
    String data_key = "eRNr6lU1RD6jmgS9OPqmmbVrk";
    String amount = "1.00";
    String cust_id = "customer1";
    String crypt_type = "1";
    String processing_country_code = "CA";
    boolean status_check = false;

    ResIndRefundCC resIndRefundCC = new ResIndRefundCC();
    resIndRefundCC.setOrderId(order_id);
    resIndRefundCC.setCustId(cust_id);
    resIndRefundCC.setAmount(amount);
    resIndRefundCC.setCryptType(crypt_type);
    resIndRefundCC.setData(data_key);

    //NT Response Option
    boolean get_nt_response = true;
    resIndRefundCC.setGetNtResponse(get_nt_response

    HttpsPostRequest mpgReq = new HttpsPostRequest();
    mpgReq.setProcCountryCode(processing_country_code);
    mpgReq.setTestMode(true); //false for production transactions
    mpgReq.setstoreId(store_id);
    mpgReq.setApiToken(api_token);
    mpgReq.setTransaction(resIndRefundCC);
    mpgReq.setStatusCheck(status_check);
    mpgReq.send();
    try
    {
      Receipt receipt = mpgReq.getReceipt();
      System.out.println("DataKey = " + receipt.getDataKey());
      System.out.println("ReceiptId = " + receipt.getReceiptId());
      System.out.println("ReferenceNum = " + receipt.getReferenceNum());
      System.out.println("ResponseCode = " + receipt.getResponseCode());
      System.out.println("AuthCode = " + receipt.getAuthCode());
      System.out.println("Message = " + receipt.getMessage());
      System.out.println("TransDate = " + receipt.getTransDate());
      System.out.println("TransTime = " + receipt.getTransTime());
      System.out.println("TransType = " + receipt.getTransType());
      System.out.println("Complete = " + receipt.getComplete());
      System.out.println("TransAmount = " + receipt.getTransAmount());
      System.out.println("CardType = " + receipt.getCardType());
      System.out.println("TxnNumber = " + receipt.getTxnNumber());
      System.out.println("TimedOut = " + receipt.getTimedOut());
      System.out.println("ResSuccess = " + receipt.getResSuccess());
      System.out.println("PaymentType = " + receipt.getPaymentType());
    }
  }
}

```

```

System.out.println("IsVisaDebit = " + receipt.getIsVisaDebit());
System.out.println("Cust ID = " + receipt.getResCustomerId());
System.out.println("Phone = " + receipt.getResPhone());
System.out.println("Email = " + receipt.getResEmail());
System.out.println("Note = " + receipt.getResNote());
System.out.println("Masked Pan = " + receipt.getResMaskedPan());
System.out.println("Exp Date = " + receipt.getResExpdate());
System.out.println("Crypt Type = " + receipt.getResCryptType());
System.out.println("Avs Street Number = " + receipt.getResAvsStreetNumber());
System.out.println("Avs Street Name = " + receipt.getResAvsStreetName());
System.out.println("Avs Zipcode = " + receipt.getResAvsZipcode());
System.out.println("SourcePanLast4 = " + receipt.getSourcePanLast4());

if(get_nt_response) {
    System.out.println("\nNTResponseCode = " + receipt.getNTResponseCode());
    System.out.println("NTMessage = " + receipt.getNTMessage());
    System.out.println("NTUsed = " + receipt.getNTUsed());
    System.out.println("NTTokenBin = " + receipt.getNTTokenBin());
    System.out.println("NTTokenLast4 = " + receipt.getNTTokenLast4());
    System.out.println("NTTokenExpDate = " + receipt.getNTTokenExpDate());
}
}
catch (Exception e)
{
e.printStackTrace();
}
}
}

```

### **Champs de réponse liés à la chambre forte**

Pour obtenir une liste et une explication des champs de réponse de l'objet Receipt pouvant apparaître après l'envoi d'une transaction utilisant la chambre forte, consultez l'Annexe B Définition des champs de réponse (page 547).

### **4.6.5 Transaction forcée avec la chambre forte (ResForcePostCC)**

**REMARQUE :** Cette transaction prend en charge les jetons temporaires et permanents.

#### **Définition de l'objet de Force Post with Vault**

```
ResForcePostCC resForcePostCC = new ResForcePostCC();
```

#### **Objet HttpsPostRequest pour les transactions forcées avec la chambre forte**

```
HttpsPostRequest mpgReq = new HttpsPostRequest();

mpgReq.setTransaction(resForcePostCC);
```

#### **Champs de demande liés aux transactions forcées avec la chambre forte (obligatoires)**

Variable	Type et limites	Méthode Set
Montant	<i>Chaîne</i> 10 caractères décimaux Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres	resForcePostCC.setAmount(amount);

Variable	Type et limites	Méthode Set
	(sous) après le point décimal  <b>EXEMPLE : 1 234 567,89</b>	
Clé de données	<i>Chaîne</i>  25 caractères alphanumériques	<code>resForcePostCC.setData(data_key);</code>
Code d'autorisation	<i>Chaîne</i>  8 caractères alphanumériques	<code>resForcePostCC.setAuthCode(auth_code);</code>
Indicateur de commerce électronique	<i>Chaîne</i>  1 caractère alphanumérique	<code>resForcePostCC.setCryptType(crypt);</code>

#### Définition de l'objet de Force Post with Vault

Variable	Type et limites	Méthode Set
ID de client	<i>Chaîne</i>  50 caractères alphanumériques  <div style="border: 1px solid #ccc; padding: 5px; background-color: #e0f2fd;"> <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés :            &lt;&gt;\$%=?^{}[]\         </div>	<code>resForcePostCC.setCustId(cust_id);</code>
Descripteur dynamique	<i>Chaîne</i>  20 caractères alphanumériques  Total de 22 caractères incluant votre nom de commerçant et un séparateur  <div style="border: 1px solid #ccc; padding: 5px; background-color: #e0f2fd;"> <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés :         </div>	<code>resForcePostCC.setDynamicDescriptor(dynamic_descriptor);</code>

Variable	Type et limites	Méthode Set
	<>\$%=?^{}[]\`	
Vérification d'état	<i>Valeur booléenne</i> true/false	mpgReq.setStatusCheck(status_check);

Exemple de transaction forcée avec la chambre forte
<pre> package Canada; import JavaAPI.*; public class TestCanadaResForcePostCC {     public static void main(String[] args)     {         java.util.Date createDate = new java.util.Date();         String order_id = "Test"+createDate.getTime();         String store_id = "store5";         String api_token = "yesguy";         String data_key = "uroyVNSxzjk5hHoT0kpQDBCw4";         String amount = "1.00";         String cust_id = "customer1"; //if sent will be submitted, otherwise cust_id from profile will be used         String crypt_type = "7";         String auth_code = "124424";         String descriptor = "my descriptor";         String processing_country_code = "CA";         boolean status_check = false;          ResForcePostCC resForcePostCC = new ResForcePostCC();         resForcePostCC.setOrderId(order_id);         resForcePostCC.setCustId(cust_id);         resForcePostCC.setAmount(amount);         resForcePostCC.setDataKey(data_key);         resForcePostCC.setAuthCode(auth_code);         resForcePostCC.setCryptType(crypt_type);         resForcePostCC.setDynamicDescriptor(descriptor);          //NT Response Option         boolean get_nt_response = true;         resForcePostCC.setGetNtResponse(get_nt_response);          HttpsPostRequest mpgReq = new HttpsPostRequest();         mpgReq.setProcCountryCode(processing_country_code);         mpgReq.setTestMode(true); //false for production transactions         mpgReq.setStoreId(store_id);         mpgReq.setApiToken(api_token);         mpgReq.setTransaction(resForcePostCC);         mpgReq.setStatusCheck(status_check);         mpgReq.send();         try         {             Receipt receipt = mpgReq.getReceipt();             System.out.println("DataKey = " + receipt.getDataKey());             System.out.println("ReceiptId = " + receipt.getReceiptId());             System.out.println("ReferenceNum = " + receipt.getReferenceNum());             System.out.println("ResponseCode = " + receipt.getResponseCode());             System.out.println("AuthCode = " + receipt.getAuthCode());             System.out.println("Message = " + receipt.getMessage());             System.out.println("TransDate = " + receipt.getTransDate());             System.out.println("TransTime = " + receipt.getTransTime());             System.out.println("TransType = " + receipt.getTransType());             System.out.println("Complete = " + receipt.getComplete());             System.out.println("TransAmount = " + receipt.getTransAmount());             System.out.println("CardType = " + receipt.getCardType());             System.out.println("TxnNumber = " + receipt.getTxnNumber());         }     } } </pre>

```

System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("ResSuccess = " + receipt.getResSuccess());
System.out.println("PaymentType = " + receipt.getPaymentType());
System.out.println("IsVisaDebit = " + receipt.getIsVisaDebit());
System.out.println("Cust ID = " + receipt.getResCustId());
System.out.println("Phone = " + receipt.getResPhone());
System.out.println("Email = " + receipt.getResEmail());
System.out.println("Note = " + receipt.getResNote());
System.out.println("Masked Pan = " + receipt.getResMaskedPan());
System.out.println("Exp Date = " + receipt.getResExpdate());
System.out.println("Crypt Type = " + receipt.getResCryptType());
System.out.println("Avs Street Number = " + receipt.getResAvsStreetNumber());
System.out.println("Avs Street Name = " + receipt.getResAvsStreetName());
System.out.println("Avs Zipcode = " + receipt.getResAvsZipcode());
System.out.println("SourcePanLast4 = " + receipt.getSourcePanLast4());

if(get_nt_response) {
    System.out.println("\nNTResponseCode = " + receipt.getNTResponseCode());
    System.out.println("NTMessage = " + receipt.getNTMessage());
    System.out.println("NTUsed = " + receipt.getNTUsed());
    System.out.println("NTTokenBin = " + receipt.getNTTokenBin());
    System.out.println("NTTokenLast4 = " + receipt.getNTTokenLast4());
    System.out.println("NTTokenExpDate = " + receipt.getNTTokenExpDate());
}
}
catch (Exception e)
{
e.printStackTrace();
}
}
}

```

#### 4.6.6 Vérification de la carte avec la chambre forte (ResCardVerificationCC)

**REMARQUE :** Cette transaction prend en charge les jetons temporaires et permanents.

##### Éléments dont il faut tenir compte :

Ce type de transactions s'applique uniquement aux transactions par cartes Visa, Mastercard, American Express et Discover.

Le numéro de carte et la date d'expiration utilisée pour cette transaction sont transférés au moyen d'un jeton, représenté par la valeur data key.

Lorsque vous utilisez un jeton temporaire (p. ex. avec la transformation en jetons hébergée) **et** que vous prévoyez enregistrer les renseignements d'identification du titulaire de carte, cette transaction doit être effectuée avant la transaction d'ajout d'un jeton à la chambre forte.

##### Définition de l'objet Card Verification with Vault

```
ResCardVerificationCC rescardverify = new ResCardVerificationCC();
```

##### Objet HttpsPostRequest pour les transactions de vérification de la carte avec la chambre forte

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
```

```
mpgReq.setTransaction(rescardverify);
```

## Champs de demande pour les transactions de vérification de la carte avec la chambre forte (obligatoires)

Pour obtenir une description de toutes les valeurs obligatoires et facultatives, consultez l'annexe A

Définition des champs de demande.

Variable	Type et limites	Méthode Set
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques a-Z A-Z 0-9 _ - : . @ espaces	<code>rescardverify.setOrderId(order_id);</code>
Clé de données	<i>Chaîne</i> 25 caractères alphanumériques	<code>rescardverify.setDataKeyFormat(data_key_format);</code>
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	<code>rescardverify.setCryptType(crypt);</code>
Renseignements du SVA	<i>Objet</i> S. O.	<code>rescardverify.setAvsInfo(avsCheck);</code>
Renseignements du NVC	<i>Objet</i> S. O.	<code>rescardverify.setCvdInfo(cvdCheck);</code>
<b>REMARQUE :</b> Lorsque les renseignements d'identification sont stockés lors de la première transaction, l'objet CVD (numéro de vérification de carte, ou NVC) doit être envoyé. Pour les transactions subséquentes utilisant les renseignements d'identification stockés, le NVC peut être envoyé avec les transactions entamées par le titulaire de carte seulement. <b>Les commerçants ne doivent pas enregistrer le NVC.</b>		
Renseignements d'identification au dossier cof	<i>Objet</i> S. O.	<code>rescardverify.setCofInfo(cof);</code>
<b>REMARQUE :</b> Cette variable est un objet imbriqué au sein d'une		

Variable	Type et limites	Méthode Set
transaction et elle est requise lors du stockage des renseignements d'identification d'un client ou lors de l'utilisation de tels renseignements stockés. Pour plus de renseignements sur les champs de l'objet Credential on File Info, consultez la section Objet Credential File Info et variables.		

### Champs de demande pour l'objet Credential on File Info

Variable	Type et limites	
ID de l'émetteur  <b>REMARQUE :</b> Cette variable est requise pour chaque transaction entamée par le commerçant à la suite de la première transaction. Après l'envoi de la première transaction, l'ID d'émetteur (issuer ID) est reçu dans la réponse de transaction, puis utilisé dans les demandes de transaction subséquentes.	<i>Chaîne</i> 15 caractères alphanumériques Longueur variable	<code>cof.setIssuerId("VALUE_FOR_ISSUER_ID");</code>  <b>REMARQUE :</b> Pour obtenir une liste et une explication des valeurs possibles à envoyer pour cette variable, consultez la section Définition des champs de demande – Renseignements d'identification au dossier (cof)
Indicateur de paiement  <b>REMARQUE :</b> Lors d'une transaction utilisant les renseignements d'identification au dossier pour laquelle le champ de demande e-commerce indicator est également inclus, les valeurs acceptables de ce champ dépendent de la valeur envoyée pour l'indicateur de paiement. Pour plus d'information, consultez les définitions des champs de réponse.	<i>Chaîne</i> 1 caractère alphabétique	<code>cof.setPaymentIndicator("PAYMENT_INDICATOR_VALUE");</code>  <b>REMARQUE :</b> Pour obtenir une liste et une explication des valeurs possibles à envoyer pour cette variable, consultez la section Définition des champs de demande – Renseignements d'identification au dossier (cof)
Information de paiement	<i>Chaîne</i> 1 caractère numérique	<code>cof.setPaymentInformation("PAYMENT_INFO_VALUE");</code>  <b>REMARQUE :</b> Pour obtenir une liste et une explication des valeurs possibles à envoyer pour cette variable, consultez la section Définition des champs de demande – Renseignements d'identification au dossier (cof)

### Exemple de transaction de vérification de la carte avec la chambre forte

```

package Canada;
import java.io.*;
import JavaAPI.*;
public class TestCanadaResCardVerificationCC
{
    public static void main(String args[]) throws IOException
    {
        String store_id = "store5";
        String api_token = "yesguy";
        String data_key = "AoG4zAFzlFFFxcVmzWAZVQuhj";
        java.util.Date createDate = new java.util.Date();
        String order_id = "Test"+createDate.getTime();
        String crypt_type = "7";
        String processing_country_code = "CA";
        boolean status_check = false;

        /***** Efraud Variables *****/
        AvsInfo avs = new AvsInfo ();
        avs.setAvsStreetName("test ave");
        avs.setAvsStreetNumber("123");
        avs.setAvsZipcode("123456");
        CvdInfo cvd = new CvdInfo ("1", "099");
        /***** Transaction Object *****/
        ResCardVerificationCC resCardVerificationCC = new ResCardVerificationCC();
        resCardVerificationCC.setDataKey(data_key);
        resCardVerificationCC.setOrderId(order_id);
        resCardVerificationCC.setCryptType(crypt_type);
        resCardVerificationCC.setAvsInfo(avs);
        resCardVerificationCC.setCvdInfo(cvd);
        //NT Response Option
        boolean get_nt_response = true;
        resCardVerificationCC.setGetNtResponse(get_nt_response);

        //resCardVerificationCC.setExppdate("1412"); //For Temp Tokens only

        //Mandatory - Credential on File details
        CofInfo cof = new CofInfo();
        cof.setPaymentIndicator("U");
        cof.setPaymentInformation("2");
        cof.setIssuerId("139X3130ASCXAS9");

        resCardVerificationCC.setCofInfo(cof);

        HttpsPostRequest mpgReq = new HttpsPostRequest();
        mpgReq.setProcCountryCode(processing_country_code);
        mpgReq.setTestMode(true); //false for production transactions
        mpgReq.setstoreId(store_id);
        mpgReq.setApiToken(api_token);
        mpgReq.setTransaction(resCardVerificationCC);
        mpgReq.setStatusCheck(status_check);
        mpgReq.send();
        /***** Receipt Object *****/
        try
        {
            Receipt resreceipt = mpgReq.getReceipt();
            System.out.println("DataKey = " + resreceipt.getDataKey());
            System.out.println("ReceiptId = " + resreceipt.getReceiptId());
            System.out.println("ReferenceNum = " + resreceipt.getReferenceNum());
            System.out.println("ResponseCode = " + resreceipt.getResponseCode());
            System.out.println("AuthCode = " + resreceipt.getAuthCode());
            System.out.println("ISO = " + resreceipt.getISO());
            System.out.println("Message = " + resreceipt.getMessage());
            System.out.println("TransDate = " + resreceipt.getTransDate());
            System.out.println("TransTime = " + resreceipt.getTransTime());
            System.out.println("TransType = " + resreceipt.getTransType());
            System.out.println("Complete = " + resreceipt.getComplete());
            System.out.println("TransAmount = " + resreceipt.getTransAmount());
            System.out.println("CardType = " + resreceipt.getCardType());
            System.out.println("TxnNumber = " + resreceipt.getTxnNumber());
            System.out.println("TimedOut = " + resreceipt.getTimedOut());
            System.out.println("ResSuccess = " + resreceipt.getResSuccess());
            System.out.println("PaymentType = " + resreceipt.getPaymentType() + "\n");
            System.out.println("IssuerId = " + resreceipt.getIssuerId());
        }
    }
}

```

```

System.out.println("SourcePanLast4 = " + resreceipt.getSourcePanLast4());
System.out.println();

//Contents of ResolveData
System.out.println("Cust ID = " + resreceipt.getResCustomerId());
System.out.println("Phone = " + resreceipt.getResPhone());
System.out.println("Email = " + resreceipt.getResEmail());
System.out.println("Note = " + resreceipt.getResNote());
System.out.println("Masked Pan = " + resreceipt.getResMaskedPan());
System.out.println("Exp Date = " + resreceipt.getResExpdate());
System.out.println("Crypt Type = " + resreceipt.getResCryptType());
System.out.println("Avs Street Number = " + resreceipt.getResAvsStreetNumber());
System.out.println("Avs Street Name = " + resreceipt.getResAvsStreetName());
System.out.println("Avs Zipcode = " + resreceipt.getResAvsZipcode());

if(get_nt_response) {
System.out.println("\nNTResponseCode = " + resreceipt.getNTResponseCode());
System.out.println("NTMessage = " + resreceipt.getNTMessage());
System.out.println("NTUsed = " + resreceipt.getNTUsed());
System.out.println("NTTokenBin = " + resreceipt.getNTTokenBin());
System.out.println("NTTokenLast4 = " + resreceipt.getNTTokenLast4());
System.out.println("NTTokenExpDate = " + resreceipt.getNTTokenExpDate());
}
}
catch (Exception e)
{
e.printStackTrace();
}
}
} // end TestResCardVerificationCC

```

## 4.7 Transformation en jetons hébergée

La transformation en jetons hébergée de Moneris est une solution en ligne pour les commerçants électroniques qui ne veulent pas gérer de numéro de carte de crédit directement sur leur site Web, mais qui veulent tout de même pouvoir personnaliser pleinement l'apparence de leur page de paiement.

Lorsqu'une transaction de transformation en jetons hébergée est entamée, Passerelle Moneris affiche (au nom du commerçant) une boîte de texte sur la page de paiement du commerçant. Le titulaire de carte entre ensuite les renseignements de sa carte de crédit dans la boîte de texte. Lors de l'envoi des renseignements de paiement sur la page de paiement, Passerelle Moneris renvoie au commerçant un jeton temporaire qui représente le numéro de carte de crédit. Ce jeton est ensuite utilisé dans un appel API pour traiter une transaction financière directement auprès de Moneris afin que le montant de la transaction soit porté à la carte de crédit. Après avoir reçu une réponse pour la transaction financière, le commerçant produit un reçu et permet au titulaire de carte de poursuivre son magasinage en ligne.

Pour en savoir plus sur la façon d'intégrer la fonction de transformation en jetons hébergée de Moneris, consultez le guide d'intégration des solutions hébergées. Vous pouvez télécharger ce guide dans le portail pour développeurs de Moneris à l'adresse

[developer.moneris.com](http://developer.moneris.com).

## 5 Les transactions de niveaux 2 et 3

- 5.1 À propos des transactions de niveaux 2 et 3
- 5.2 Les transactions Visa de niveaux 2 et 3
- 5.3 Les transactions Mastercard de niveaux 2 et 3
- 5.4 Les transactions American Express de niveaux 2 et 3

### 5.1 À propos des transactions de niveaux 2 et 3

L'API de Passerelle Moneris prend en charge le transfert des données de transaction par carte d'achat de niveaux 2 et 3 pour les cartes d'entreprise Visa, Mastercard et American Express.

Toutes les transactions de niveaux 2 et 3 utilisent la même transaction de préautorisation, comme indiqué dans la section Préautorisation (page 22).

### 5.2 Transactions Visa de niveaux 2 et 3

- 5.2.1 Types de transactions de niveaux 2 et 3 par carte Visa
- 5.2.2 Flux de transaction de niveaux 2 et 3 par carte Visa
- 5.2.3 Conclusion par carte Visa
- 5.2.5 Transaction forcée par carte Visa
- 5.2.4 Correction d'achat par carte Visa
- 5.2.6 Remboursement par carte Visa
- 5.2.7 Remboursement indépendant par carte Visa
- 5.2.8 Transaction VS Corpais
- 1 Facture VS Corpais
- 1 Facture VS Corpais – Itinéraire de passager

#### 5.2.1 Types de transactions de niveaux 2 et 3 pour Visa

Cet ensemble de transactions comprend une série de transactions financières par carte d'entreprise ainsi qu'une transaction qui permet de transmettre des données de niveaux 2 et 3. Veuillez vous assurer les données de niveaux 2 et 3 sont prises en charge pour les cartes Visa dans votre compte de commerçant. Les transactions de fermeture de lot, d'ouverture des totaux et de préautorisation sont identiques aux transactions indiquées dans la section Ensemble de transactions de base (page 16).

- Lorsque la réponse à la préautorisation contient une valeur CorporateCard égale à true, vous pouvez soumettre les transactions Visa.
- Si la valeur CorporateCard est false, la carte ne prend pas en charge les données de niveau 2 et 3, il faut donc utiliser des transactions autres que celles de niveau 2 et 3. Si la carte n'est pas une carte d'entreprise, veuillez consulter la section 2. Ensemble de transactions de base pour connaître les transactions appropriées pour les cartes autres que les cartes d'entreprise.

**REMARQUE :** Cette série de transactions est destinée aux transactions où la valeur Corporate Card est true pour lesquelles des données de niveaux 2 et 3 sont soumises. S'il s'avère que la carte de

crédit est une carte d'entreprise, mais que vous ne souhaitez pas envoyer de données de niveaux 2 et 3, vous pouvez soumettre des transactions Visa en utilisant l'ensemble de transactions de base de la section 2. Ensemble de transactions de base.

### **Préautorisation – (autorisation/préautorisation)**

Une transaction de préautorisation vérifie que les fonds sont présents sur la carte de crédit du client et les bloque. Ces fonds sont bloqués pour une durée prédéterminée, en fonction de l'émetteur de carte. Pour récupérer les fonds provenant d'une préautorisation afin qu'ils puissent être déposés dans le compte du commerçant, une conclusion de préautorisation doit être effectuée. La valeur CorporateCard sera true si la carte prend en charge les données de niveaux 2 et 3.

### **Conclusion par carte Visa – (Conclusion de préautorisation)**

Une fois la préautorisation obtenue, les fonds bloqués doivent être récupérés sur la carte de crédit du client. La transaction de conclusion récupère les fonds bloqués, puis les prépare à des fins de règlement dans le compte du commerçant. Avant de réaliser une conclusion par carte Visa, une préautorisation doit avoir eu lieu. Une fois la transaction conclue, une transaction VS Corpais doit être effectuée pour traiter les données de niveaux 2 et 3.

### **Transaction forcée par carte Visa – (Transaction de conclusion forcée)**

Cette transaction est une autre façon d'obtenir les fonds bloqués lors d'une transaction de préautorisation effectuée par RVI ou par un terminal équivalent. La transaction forcée par carte Visa récupère les fonds bloqués et les prépare à être déposés dans le compte du commerçant. Une fois la transaction conclue, une transaction VS Corpais doit être effectuée pour traiter les données de niveaux 2 et 3.

### **Correction d'achat par carte Visa – (Annulation, correction)**

Les transactions de conclusion et la transaction forcée par carte Visa peuvent être annulées le jour même\* où elles se produisent. Une correction d'achat par carte Visa doit annuler le montant total de la transaction et supprimera toute trace de celle-ci sur le relevé du titulaire de la carte.

### **Remboursement par carte Visa – (Crédit)**

Un remboursement par carte Visa peut être effectué pour une transaction de conclusion par carte Visa afin de rembourser une partie ou la totalité du montant de la transaction. Une fois la transaction conclue, une transaction VS Corpais doit être effectuée pour traiter les données de niveaux 2 et 3.

### **Remboursement indépendant par carte Visa – (Crédit)**

Un remboursement indépendant par carte Visa peut être effectué pour un achat ou une conclusion afin de rembourser une partie ou la totalité du montant de la transaction. Un remboursement indépendant est utilisé lorsque la transaction d'origine n'a pas été effectuée par l'entremise de Passerelle Moneris. Une fois la transaction conclue, une transaction VS Corpais doit être effectuée pour traiter les données de niveaux 2 et 3.

**REMARQUE :** votre compte peut se prévaloir ou non de la fonctionnalité de transaction de remboursement indépendant. Si vous recevez un message d'erreur indiquant que la transaction n'est pas autorisée lorsque vous tentez un remboursement indépendant, cela peut signifier que la transaction n'est pas prise en charge par votre compte. Si vous souhaitez activer (ou réactiver) temporairement les transactions de remboursement indépendant, veuillez communiquer avec le centre de services en composant le 1 866 319-7450.

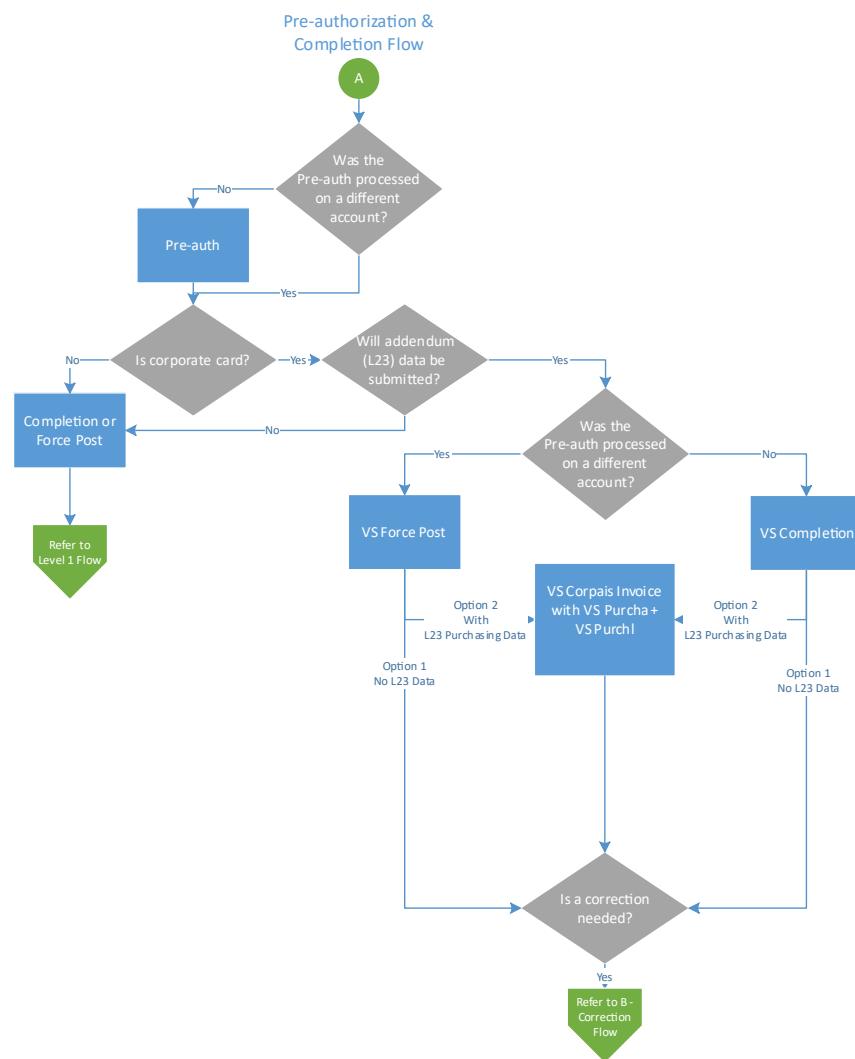
## VS Corpais – (Données de niveaux 2 et 3)

Une transaction VS Corpais contiendra tous les champs de données obligatoires et facultatifs pour les données interentreprises de niveaux 2 et 3. Les données VS Corpais peuvent être envoyées lorsque la carte a été identifiée comme étant une carte d'entreprise dans la demande de transaction de préautorisation.

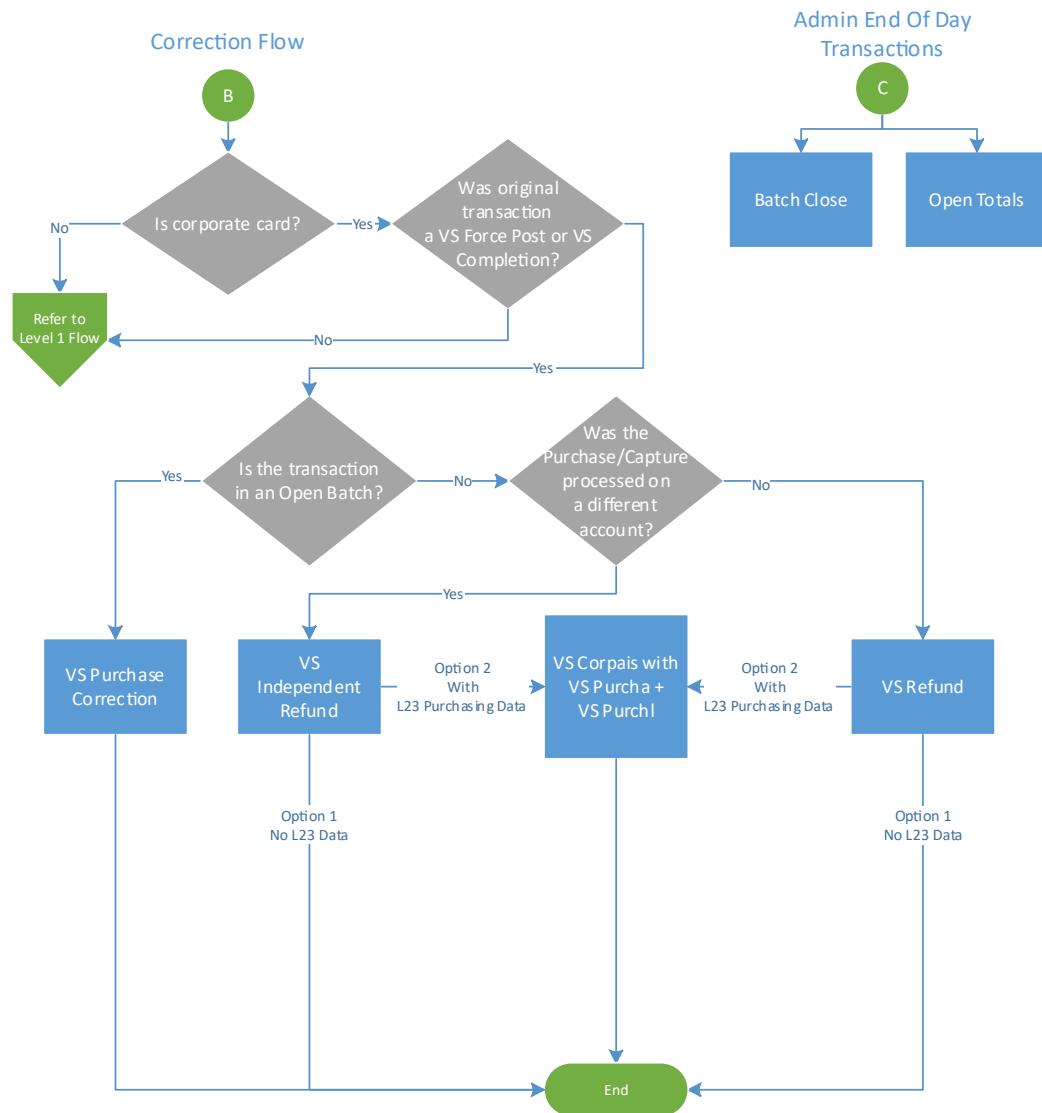
\* Une correction d'achat par carte Visa peut être effectuée pour annuler une transaction tant que le lot qui contient la transaction originale reste ouvert. Lorsque la fonction de fermeture automatique est utilisée, le lot est fermé tous les jours entre 22 h et 23 h HNE.

### 5.2.2 Flux de transaction de niveaux 2 et 3 par carte Visa

#### Flux d'une transaction de conclusion et de préautorisation



## Flux d'une transaction de correction d'achat



## 5.2.3 Conclusion par carte Visa

Une fois la préautorisation obtenue, les fonds bloqués doivent être récupérés sur la carte de crédit du client. Cette transaction de conclusion par carte Visa est utilisée pour sécuriser les fonds bloqués par une transaction de préautorisation, puis les prépare à être déposés dans le compte du commerçant.

**REMARQUE :** Lorsque vous aurez terminé cette transaction avec succès, pour soumettre toutes les données supplémentaires de niveaux 2 et 3, veuillez vous effectuer une transaction VS Corpais.

### Définition de l'objet de transaction VS Completion

```
VsCompletion vsCompletion = new VsCompletion();
```

### Objet HttpsPostRequest pour les transactions de conclusion par carte Visa

```
HttpsPostRequest mpgReq = new HttpsPostRequest();  
  
mpgReq.setTransaction(vsCompletion);
```

### Champs de demande pour les transactions de conclusion par carte Visa (obligatoires)

Variable	Limites	Méthode Set
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques	vsCompletion.setOrderId(order_id);
Montant de la conclusion	<i>Chaîne</i> 10 caractères décimaux Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal	vsCompletion.setCompAmount(comp_amount);
<b>EXEMPLE : 1 234 567,89</b>		
Numéro de transaction	<i>Chaîne</i> 255 caractères alphanumériques	vsCompletion.setTxnNumber(txn_number);
Indicateur de commerce électronique (e-commerce indicator)	<i>Chaîne</i> 1 caractère alphanumérique	vsCompletion.setCryptType(crypt);

**Tableau 1 Visa – Données communes des carte d'entreprise – Champs de demande de niveau 2**

<b>Requis*</b>	<b>Valeur</b>	<b>Limites</b>	<b>Méthode Set</b>	<b>Description</b>
Y	Taxe nationale	12 caractères décimaux	<code>vsCompletion.setNationalTax(national_tax);</code>	<p>Doit être identique à la taxe nationale (TPS ou TVH) apparaissant sur la facture</p> <p>Valeur minimale de 0,01 et maximale de 999 999,99 Doit comporter 2 décimales.</p>
Y	Numéro de TVA du commerçant/Référence d'entreprise unique	20 caractères alphanumériques	<code>vsCompletion.setMerchantVatNo(merchant_vat_no);</code>	<p>Numéro d'enregistrement des taxes du client</p> <p>Doit être fourni si des taxes sont incluses dans la facture</p> <p><b>REMARQUE :</b> Il ne doit pas y avoir que des espaces ou que des zéros.</p>

C	Taxe locale	12 caractères décimaux	<pre>vsCompletion .setLocalTax(local_ tax);</pre>	<p>Doit être identique à la taxe locale (TVP ou TVQ) apparaissant sur la facture</p> <p>Si la taxe locale est inclue, elle ne doit pas être composée uniquement d'espaces ou de zéros. Elle doit être fournie si la taxe locale (TVP ou TVQ) s'applique.</p> <p>Valeur minimale de 0,01</p> <p>Valeur maximale de 999 999,99</p> <p>Doit comporter 2 décimales</p>
C	Numéro d'enregistrement des taxes locales (TVP ou TVQ) du commerçant	15 caractères alphanumériques	<pre>vsCompletion .setLocalTaxNo(loc a l_tax_no);</pre>	<p>Numéro d'enregistrement des taxes locales (TVP ou TVQ) du commerçant</p> <p>Il doit être fourni si des taxes sont incluses dans la facture et si la taxe locale est inclue , il ne doit pas être composé uniquement d'espaces ou zéros.</p> <p>Doit être fourni si la taxe locale (TVP ou TVQ) s'applique</p>

C	Numéro de TVA du client	13 caractères alphanumériques	<pre>vsCompletion .setCustomerVatNo(c ustomer_vat_no);</pre>	Si le numéro d'enregistrement des taxes du client figure sur la facture pour justifier des transactions exemptées de taxes, il doit être indiqué ici.
C	Code de client/Code de référence du client	16 caractères alphanumériques	<pre>vsCompletion .setCri(cri);</pre>	Valeur que le client peut fournir au fournisseur au point de vente (il doit être entré si le client le fournit)
N	Code de client	17 caractères alphanumériques	<pre>vsCompletion .setCustomerCode(cu stomer_code);</pre>	Champ facultatif qui ne sera pas envoyé à Visa, mais qui se trouvera dans les rapports de Moneris
N	Numéro de facture	17 caractères alphanumériques	<pre>vsCompletion .setInvoiceNumber(i nvoice_number);</pre>	Champ facultatif qui ne sera pas envoyé à Visa, mais qui se trouvera dans les rapports de Moneris

\* Y = Obligatoire, N = Facultatif, C = Conditionnel

### Exemple de conclusion par carte Visa

```

package Level23;
import JavaAPI.*;
public class TestVsCompletion
{
    public static void main(String[] args)
    {
        String store_id = "moneris";
        String api_token = "hurgle";
        String processing_country_code = "CA";
        boolean status_check = false;

        String order_id="ord-210916-15:14:46";
        String comp_amount="5.00";
        String txn_number = "19002-0_11";
        String crypt="7";
        String national_tax = "1.23";
        String merchant_vat_no = "gstno111";
        String local_tax = "2.34";
        String customer_vat_no = "gstno999";
        String cri = "CUST-REF-002";
        String customer_code="ccvsfp";
        String invoice_number="invsfp";
        String local_tax_no="ltaxno";
        VsCompletion vsCompletion = new VsCompletion();
        vsCompletion.setOrderId(order_id);
        vsCompletion.setCompAmount(comp_amount);
        vsCompletion.setTxnNumber(txn_number);
        vsCompletion.setCryptType(crypt);
        vsCompletion.setNationalTax(national_tax);
        vsCompletion.setMerchantVatNo(merchant_vat_no);
        vsCompletion.setLocalTax(local_tax);
        vsCompletion.setCustomerVatNo(customer_vat_no);
        vsCompletion.setCri(cri);
        vsCompletion.setCustomerCode(customer_code);
        vsCompletion.setInvoiceNumber(invoice_number);
        vsCompletion.setLocalTaxNo(local_tax_no);
        HttpsPostRequest mpgReq = new HttpsPostRequest();
        mpgReq.setProcCountryCode(processing_country_code);
        mpgReq.setTestMode(true); //false for production transactions
        mpgReq.setstoreId(store_id);
        mpgReq.setApiToken(api_token);
        mpgReq.setTransaction(vsCompletion);
        mpgReq.setStatusCheck(status_check);
        mpgReq.send();
        try
        {
            Receipt receipt = mpgReq.getReceipt();
            System.out.println("CardType = " + receipt.getCardType());
            System.out.println("TransAmount = " + receipt.getTransAmount());
            System.out.println("TxnNumber = " + receipt.getTxnNumber());
            System.out.println("ReceiptId = " + receipt.getReceiptId());
            System.out.println("TransType = " + receipt.getTransType());
            System.out.println("ReferenceNum = " + receipt.getReferenceNum());
            System.out.println("ResponseCode = " + receipt.getResponseCode());
            System.out.println("ISO = " + receipt.getISO());
            System.out.println("BankTotals = " + receipt.getBankTotals());
            System.out.println("Message = " + receipt.getMessage());
            System.out.println("AuthCode = " + receipt.getAuthCode());
            System.out.println("Complete = " + receipt.getComplete());
            System.out.println("TransDate = " + receipt.getTransDate());
            System.out.println("TransTime = " + receipt.getTransTime());
            System.out.println("Ticket = " + receipt.getTicket());
            System.out.println("TimedOut = " + receipt.getTimedOut());
            System.out.println("CavvResultCode = " + receipt.getCavvResultCode());
        }
        catch (Exception e)
        {
            System.out.println(e);
        }
    }
}

```

## 5.2.4 Correction d'achat par carte Visa

La correction d'achat (ou annulation) par carte Visa est utilisée pour annuler une transaction effectuée dans le lot en cours. Aucun montant n'est requis, car la correction d'achat annule toujours la totalité du montant de la transaction initiale. Les seules transactions pouvant être annulées à l'aide de la correction d'achat par carte Visa sont les transactions de conclusion forcées par carte Visa. Pour envoyer une annulation, les variables `order_id` et `txn_number` de la transaction de conclusion par carte Visa ou de la transaction forcée par carte Visa sont requis.

### Définition de l'objet de transaction VS Purchase Correction

```
VsPurchaseCorrection vsPurchaseCorrection = new VsPurchaseCorrection();
```

### Objet `HttpsPostRequest` pour les transactions de correction d'achat par carte Visa

```
HttpsPostRequest mpgReq = new HttpsPostRequest();

mpgReq.setTransaction(vsPurchaseCorrection);
```

### Champs de demande liés aux transactions de correction d'achat par carte Visa (obligatoires)

Variable	Type et limites	Méthode Set
No de commande	<i>Chaîne</i> 50 caractères alphanumériques	<code>vsPurchaseCorrection.setOrderI d(order_id);</code>
Numéro de transaction	<i>Chaîne</i> 255 caractères alphanumériques	<code>vsPurchaseCorrection.setTxnNum ber(txn_number);</code>
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	<code>vsPurchaseCorrection.setCryptT ype(crypt);</code>

### Exemple de transaction de correction d'achat par carte Visa

```

package Level123;
import JavaAPI.*;

public class TestVsPurchaseCorrection
{
    public static void main(String[] args)
    {
        String store_id = "moneris";
        String api_token = "hurgle";
        String processing_country_code = "CA";
        boolean status_check = false;

        String order_id="Test1485208113189";
        String txn_number = "39793-0_11";
        String crypt="7";
        VsPurchaseCorrection vsPurchaseCorrection = new VsPurchaseCorrection();
        vsPurchaseCorrection.setOrderId(order_id);
        vsPurchaseCorrection.setTxnNumber(txn_number);
        vsPurchaseCorrection.setCryptType(crypt);
        HttpsPostRequest mpgReq = new HttpsPostRequest();
        mpgReq.setProcCountryCode(processing_country_code);
        mpgReq.setTestMode(true); //false for production transactions
        mpgReq.setstoreId(store_id);
        mpgReq.setApiToken(api_token);
        mpgReq.setTransaction(vsPurchaseCorrection);
        mpgReq.setStatusCheck(status_check);
        mpgReq.send();
        try
        {
            Receipt receipt = mpgReq.getReceipt();
            System.out.println("CardType = " + receipt.getCardType());
            System.out.println("TransAmount = " + receipt.getTransAmount());
            System.out.println("TxnNumber = " + receipt.getTxnNumber());
            System.out.println("ReceiptId = " + receipt.getReceiptId());
            System.out.println("TransType = " + receipt.getTransType());
            System.out.println("ReferenceNum = " + receipt.getReferenceNum());
            System.out.println("ResponseCode = " + receipt.getResponseCode());
            System.out.println("ISO = " + receipt.getISO());
            System.out.println("BankTotals = " + receipt.getBankTotals());
            System.out.println("Message = " + receipt.getMessage());
            System.out.println("AuthCode = " + receipt.getAuthCode());
            System.out.println("Complete = " + receipt.getComplete());
            System.out.println("TransDate = " + receipt.getTransDate());
            System.out.println("TransTime = " + receipt.getTransTime());
            System.out.println("Ticket = " + receipt.getTicket());
            System.out.println("TimedOut = " + receipt.getTimedOut());
            System.out.println("CavvResultCode = " + receipt.getCavvResultCode());
        }
        catch (Exception e)
        {
            System.out.println(e);
        }
    }
}

```

## 5.2.5 Transaction forcée par carte Visa

La transaction forcée par carte Visa est utilisée pour sécuriser les fonds bloqués par une transaction de pré-autorisation traitée par RVI ou par un terminal équivalent. Lors de l'envoi d'une demande de transaction forcée, vous aurez besoin de l'ID de la commande, du montant, du numéro de carte de crédit, de la date d'expiration, de l'indicateur de commerce électronique et du code d'autorisation reçus dans la réponse de pré-autorisation.

**REMARQUE :** Lorsque vous aurez terminé cette transaction avec succès, pour soumettre toutes les données supplémentaires de niveaux 2 et 3, veuillez vous effectuer une transaction VS Corpais.

### Définition d'objet VS Force Post

```
VsForcePost vsForcePost = new VsForcePost();
```

### Objet HttpsPostRequest pour les transactions forcées par carte Visa

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
```

```
mpgReq.setTransaction(vsForcePost);
```

### Champs de demande pour les transactions forcées par carte Visa (obligatoires)

Variable	Type et limites	Méthode Set
No de commande	<i>Chaîne</i> 50 caractères alphanumériques	vsForcePost.setOrderId(order_id);
Montant	<i>Chaîne</i> 10 caractères décimaux Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal <b>EXEMPLE : 1 234 567,89</b>	vsForcePost.setAmount(amount);
Numéro de carte de crédit	<i>Chaîne</i> 20 caractère numérique	vsForcePost.setPan(pan);
Date d'expiration	<i>Chaîne</i> 4 caractères numériques Format AAMM	vsForcePost.setExpDate(expiry_date);

Code d'autorisation	<i>Chaîne</i> 8 caractères alphanumériques	<code>vsForcePost.setAuthCode(auth_code);</code>
Indicateur de commerce électronique (e-commerce indicator)	<i>Chaîne</i> 1 caractère alphanumérique	<code>vsForcePost.setCryptType(crypt);</code>

#### Champs de demande pour les transactions forcées par carte Amex (facultatifs)

Variable	Type et limites	Méthode Set
ID du client	<i>Chaîne</i> 50 caractères alphanumériques	<code>vsForcePost.setCustId(cust_id);</code>

Tableau 1 Visa – Données communes des carte d'entreprise – Champs de demande de niveau 2

Requis*	Valeur	Limites	Méthode Set	Description
Y	Taxe nationale	12 caractères décimaux	<code>vsForcePost.setNationalTax(national_tax);</code>	Doit être identique à la taxe nationale (TPS ou TVH) apparaissant sur la facture  Valeur minimale de 0,01 et maximale de 999 999,99 Doit comporter 2 décimales.
Y	Numéro de TVA du commerçant/Référence d'entreprise unique	20 caractères alphanumériques	<code>vsForcePost.setMerchantVatNo(merchant_vat_no);</code>	Numéro d'enregistrement des taxes du client  Doit être fourni si des taxes sont incluses dans la facture

**REMARQUE :** Il ne doit pas y avoir que des espaces ou que des zéros.

C	Taxe locale	12 caractères décimaux	<pre>vsForcePost .setLocalTax(local _tax);</pre>	<p>Doit être identique à la taxe locale (TVP ou TVQ) apparaissant sur la facture</p> <p>Si la taxe locale est inclue, elle ne doit pas être composée uniquement d'espaces ou de zéros. Elle doit être fournie si la taxe locale (TVP ou TVQ) s'applique.</p> <p>Valeur minimale de 0,01</p> <p>Valeur maximale de 999 999,99</p> <p>Doit comporter 2 décimales</p>
C	Numéro d'enregistrement des taxes locales (TVP ou TVQ) du commerçant	15 caractères alphanumériques	<pre>vsForcePost .setLocalTaxNo(loc al_tax_no);</pre>	<p>Numéro d'enregistrement des taxes locales (TVP ou TVQ) du commerçant</p> <p>Il doit être fourni si des taxes sont incluses dans la facture et si la taxe locale est inclue , il ne doit pas être composé uniquement d'espaces ou zéros.</p> <p>Doit être fourni si la taxe locale (TVP ou TVQ) s'applique</p>
C	Numéro de TVA du client	13 caractères alphanumériques	<pre>vsForcePost .setCustomerVatNo(c ustomer_vat_no);</pre>	<p>Si le numéro d'enregistrement des taxes du client figure sur la facture pour</p>

				justifier des transactions exemptées de taxes, il doit être indiqué ici.
C	Code de client/Code de référence du client	16 caractères alphanumériques	vsForcePost .setCri(cri);	Valeur que le client peut fournir au fournisseur au point de vente (il doit être entré si le client le fournit)
N	Code de client	17 caractères alphanumériques	vsForcePost .setCustomerCode(cu stomer_code);	Champ facultatif qui ne sera pas envoyé à Visa, mais qui se trouvera dans les rapports de Moneris
N	Numéro de facture	17 caractères alphanumériques	vsForcePost .setInvoiceNumber(i nvoice_number);	Champ facultatif qui ne sera pas envoyé à Visa, mais qui se trouvera dans les rapports de Moneris

\* Y = Obligatoire, N = Facultatif, C = Conditionnel

### Exemple de transaction forcée par carte Visa

```
package Level23;
import JavaAPI.*;
public class TestVsForcePost
{
    public static void main(String[] args)
    {
        String store_id = "moneris";
        String api_token = "hurgle";
        String processing_country_code = "CA";
        boolean status_check = false;
        java.util.Date createDate = new java.util.Date();
        String order_id="Test"+createDate.getTime();
        String cust_id="CUST13343";
        String amount="5.00";
        String pan="4242424254545454";
        String expiry_date="2012"; //YYMM
        String auth_code="123456";
        String crypt="7";
        String national_tax = "1.23";
        String merchant_vat_no = "gstno111";
        String local_tax = "2.34";
        String customer_vat_no = "gstno999";
        String cri = "CUST-REF-002";
        String customer_code="ccvsfp";
        String invoice_number="invsfp";
        String local_tax_no="ltaxno";
        VsForcePost vsForcePost = new VsForcePost();
        vsForcePost.setOrderId(order_id);
        vsForcePost.setCustId(cust_id);
        vsForcePost.setAmount(amount);
        vsForcePost.setPan(pan);
        vsForcePost.setExpDate(expiry_date);
        vsForcePost.setAuthCode(auth_code);
        vsForcePost.setCryptType(crypt);
        vsForcePost.setNationalTax(national_tax);
        vsForcePost.setMerchantVatNo(merchant_vat_no);
        vsForcePost.setLocalTax(local_tax);
        vsForcePost.setCustomerVatNo(customer_vat_no);
        vsForcePost.setCri(cri);
        vsForcePost.setCustomerCode(customer_code);
        vsForcePost.setInvoiceNumber(invoice_number);
        vsForcePost.setLocalTaxNo(local_tax_no);
        HttpsPostRequest mpgReq = new HttpsPostRequest();
        mpgReq.setProcCountryCode(processing_country_code);
        mpgReq.setTestMode(true); //false for production transactions
        mpgReq.setstoreId(store_id);
        mpgReq.setApiToken(api_token);
        mpgReq.setTransaction(vsForcePost);
        mpgReq.setStatusCheck(status_check);
        mpgReq.send();
        try
        {
            Receipt receipt = mpgReq.getReceipt();
            System.out.println("CardType = " + receipt.getCardType());
            System.out.println("TransAmount = " + receipt.getTransAmount());
            System.out.println("TxnNumber = " + receipt.getTxnNumber());
            System.out.println("ReceiptId = " + receipt.getReceiptId());
            System.out.println("TransType = " + receipt.getTransType());
            System.out.println("ReferenceNum = " + receipt.getReferenceNum());
            System.out.println("ResponseCode = " + receipt.getResponseCode());
            System.out.println("ISO = " + receipt.getISO());
            System.out.println("BankTotals = " + receipt.getBankTotals());
            System.out.println("Message = " + receipt.getMessage());
            System.out.println("AuthCode = " + receipt.getAuthCode());
            System.out.println("Complete = " + receipt.getComplete());
            System.out.println("TransDate = " + receipt.getTransDate());
            System.out.println("TransTime = " + receipt.getTransTime());
            System.out.println("Ticket = " + receipt.getTicket());
            System.out.println("TimedOut = " + receipt.getTimedOut());
            System.out.println("CavvResultCode = " + receipt.getCavvResultCode());
        }
        catch (Exception e)
        {
            System.out.println(e);
        }
    }
}
```

### 5.2.6 Remboursement par carte Visa

Une transaction de remboursement indépendant par carte Visa crédite un montant précis sur la carte de crédit du titulaire. Une transaction de remboursement d'une valeur allant jusqu'à la valeur totale de la transaction de conclusion ou forcée par carte Visa originale peut être envoyé. Pour envoyer un remboursement par carte Visa, vous aurez besoin de l'ID de commande et du numéro de transaction de conclusion ou forcée par carte Visa originale.

**REMARQUE :** Lorsque vous aurez terminé cette transaction avec succès, pour soumettre toutes les données supplémentaires de niveaux 2 et 3, veuillez vous effectuer une transaction VS Corpais.

**Définition de l'objet de transaction VS Refund**

```
VsRefund vsRefund = new VsRefund();
```

**Objet HttpsPostRequest pour les transactions de remboursement par carte Visa**

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setTransaction(vsRefund);
```

**Valeurs de l'objet de transaction VS Refund**

Variable	Type et limites	Méthode Set
No de commande	<i>Chaîne</i> 50 caractères alphanumériques	vsRefund.setOrderId(order_id);
Numéro de transaction	<i>Chaîne</i> 255 caractères alphanumériques	vsRefund.setTxnNumber(txn_number);
Montant	<i>Chaîne</i> 10 caractères décimaux  Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal  <b>EXEMPLE : 1 234 567,89</b>	vsRefund.setAmount(amount); vsRefund.setOrderId(order_id);
Indicateur de commerce électronique (e-commerce indicator)	<i>Chaîne</i> 1 caractère alphanumérique	vsRefund.setTxnNumber(txn_number);

**Tableau 1 Visa – Données communes des carte d'entreprise – Champs de demande de niveau 2**

Requis*	Valeur	Limites	Méthode Set	Description
Y	Taxe nationale	12 caractères décimaux	vsRefund.setNationalTax(national_tax);	Doit être identique à la taxe nationale (TPS ou TVH) apparaissant sur la

Requis*	Valeur	Limites	Méthode Set	Description
				<p>facture</p> <p>Valeur minimale de 0,01 et maximale de 999 999,99 Doit comporter 2 décimales.</p>
Y	Numéro de TVA du commerçant/Référence d'entreprise unique	20 caractères alphanumériques	<pre>vsRefund .setMerchantVatNo( merchant_vat_no);</pre> <pre>vsRefund .setMerchantVatNo( merchant_vat_no);</pre>	<p>Numéro d'enregistrement des taxes du client</p> <p>Doit être fourni si des taxes sont incluses dans la facture</p> <p><b>REMARQUE :</b> Il ne doit pas y avoir que des espaces ou que des zéros.</p>
C	Taxe locale	12 caractères décimaux	<pre>vsRefund .setLocalTax(local_tax);</pre>	<p>Doit être identique à la taxe locale (TVP ou TVQ) apparaissant sur la facture</p> <p>Si la taxe locale est inclue, elle ne doit pas être composée uniquement d'espaces ou de zéros. Elle doit être fournie si la taxe locale (TVP ou TVQ) s'applique.</p> <p>Valeur minimale de 0,01</p> <p>Valeur maximale de 999 999,99</p>

Requis*	Valeur	Limites	Méthode Set	Description
				Doit comporter 2 décimales
C	Numéro d'enregistrement des taxes locales (TVP ou TVQ) du commerçant	15 caractères alphanumériques	vsRefund .setLocalTaxNo (local_tax_no);	Numéro d'enregistrement des taxes locales (TVP ou TVQ) du commerçant  Il doit être fourni si des taxes sont incluses dans la facture et si la taxe locale est inclue , il ne doit pas être composé uniquement d'espaces ou zéros.  Doit être fourni si la taxe locale (TVP ou TVQ) s'applique
C	Numéro de TVA du client	13 caractères alphanumériques	vsRefund .setCustomerVatNo (customer_vat_no);	Si le numéro d'enregistrement des taxes du client figure sur la facture pour justifier des transactions exemptées de taxes, il doit être indiqué ici.
C	Code de client/Code de référence du client	16 caractères alphanumériques	vsRefund .setCri (cri);	Valeur que le client peut fournir au fournisseur au point de vente (il doit être entré si le client le fournit)
N	Code de client	17 caractères alphanumériques	vsRefund .setCustomerCode (customer_code);	Champ facultatif qui ne sera pas envoyé à Visa, mais qui se trouvera dans les rapports

Requis*	Valeur	Limites	Méthode Set	Description
				de Moneris
N	Numéro de facture	17 caractères alphanumériques	vsRefund.setInvoiceNumber(invoice_number);	Champ facultatif qui ne sera pas envoyé à Visa, mais qui se trouvera dans les rapports de Moneris

\* Y = Obligatoire, N = Facultatif, C = Conditionnel

#### Exemple de remboursement par carte Visa

```

package Level23;
import JavaAPI.*;
public class TestVsRefund
{
public static void main(String[] args)
{
String store_id = "moneris";
String api_token = "hurgle";
String processing_country_code = "CA";
boolean status_check = false;

String order_id="Test1485208133961";
String amount="5.00";
String txn_number = "39795-0_11";
String crypt="7";
String national_tax = "1.23";
String merchant_vat_no = "gstno111";
String local_tax = "2.34";
String customer_vat_no = "gstno999";
String cri = "CUST-REF-002";
String customer_code="ccvsfp";
String invoice_number="invsfp";
String local_tax_no="ltaxno";
VsRefund vsRefund = new VsRefund();
vsRefund.setOrderId(order_id);
vsRefund.setAmount(amount);
vsRefund.setTxnNumber(txn_number);
vsRefund.setCryptType(crypt);
vsRefund.setNationalTax(national_tax);
vsRefund.setMerchantVatNo(merchant_vat_no);
vsRefund.setLocalTax(local_tax);
vsRefund.setCustomerVatNo(customer_vat_no);
vsRefund.setCri(cri);
vsRefund.setCustomerCode(customer_code);
vsRefund.setInvoiceNumber(invoice_number);
vsRefund.setLocalTaxNo(local_tax_no);
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false for production transactions
mpgReq.setStoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(vsRefund);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
}
}

```

```

        System.out.println("ISO = " + receipt.getISO());
        System.out.println("BankTotals = " + receipt.getBankTotals());
        System.out.println("Message = " + receipt.getMessage());
        System.out.println("AuthCode = " + receipt.getAuthCode());
        System.out.println("Complete = " + receipt.getComplete());
        System.out.println("TransDate = " + receipt.getTransDate());
        System.out.println("TransTime = " + receipt.getTransTime());
        System.out.println("Ticket = " + receipt.getTicket());
        System.out.println("TimedOut = " + receipt.getTimedOut());
        System.out.println("CavvResultCode = " + receipt.getCavvResultCode());
    }
    catch (Exception e)
    {
        System.out.println(e);
    }
}
}

```

## 5.2.7 Remboursement indépendant par carte Visa

Une transaction de remboursement indépendant par carte Visa crédite un montant précis sur la carte de crédit du titulaire. Une transaction de remboursement indépendant ne nécessite pas qu'une commande existante soit enregistrée dans Passerelle Moneris, mais le numéro de la carte de crédit et la date d'expiration doivent être transmis. Le format de la transaction est presque identique à celui d'une préautorisation.

**REMARQUE :** Lorsque vous aurez terminé cette transaction avec succès, pour soumettre toutes les données supplémentaires de niveaux 2 et 3, veuillez vous effectuer une transaction VS Corpais.

### Définition de l'objet de transaction VS Independent Refund

```
VsIndependentRefund vsIndependentRefund = new VsIndependentRefund();
```

### Objet HttpsPostRequest pour les transactions de remboursement indépendant par carte Visa

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
```

```
mpgReq.setTransaction(vsIndependentRefund);
```

**Champs de demande liés aux transactions de remboursement indépendant par carte Visa (obligatoires)**

Variable	Type et limites	Méthode Set
No de commande	<i>Chaîne</i> 50 caractères alphanumériques	vsIndependentRefund.setOrderId(order_id);
Montant	<i>Chaîne</i> 10 caractères décimaux Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal  <b>EXEMPLE : 1 234 567,89</b>	vsIndependentRefund.setAmount(amount);
Numéro de carte de crédit	<i>Chaîne</i> 20 caractère numérique	vsIndependentRefund.setPan(pan);
Date d'expiration	<i>Chaîne</i> 4 caractères numériques Format AAMM	vsIndependentRefund.setExpDate(expiry_date);
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	vsIndependentRefund.setCryptType(crypt);

**Champs de demande liés aux transactions de remboursement indépendant par carte Visa (facultatifs)**

Variable	Type et limites	Méthode Set
ID du client	50 caractères alphanumériques	vsIndependentRefund.setCustomerId(cust_id);

**Tableau 1 Visa – Données communes des carte d'entreprise – Champs de demande de niveau 2**

Requis*	Valeur	Limites	Méthode Set	Description
Y	Taxe nationale	12 caractères	vsIndependentRefund.setNationalTax(national_tax);	Doit être

Requis*	Valeur	Limites	Méthode Set	Description
		décimaux		<p>identique à la taxe nationale (TPS ou TVH) apparaissant sur la facture</p> <p>Valeur minimale de 0,01 et maximale de 999 999,99 Doit comporter 2 décimales.</p>
Y	Numéro de TVA du commerçant/Référence d'entreprise unique	20 caractères alphanumériques	<pre>vsIndependentRefund.setMerchantVatNo (merchant_vat_no);</pre>	<p>Numéro d'enregistrement des taxes du client</p> <p>Doit être fourni si des taxes sont incluses dans la facture</p> <p><b>REMARQUE :</b> Il ne doit pas y avoir que des espaces ou que des zéros.</p>
C	Taxe locale	12 caractères décimaux	<pre>vsIndependentRefund.setLocalTax(local_tax);</pre>	<p>Doit être identique à la taxe locale (TVP ou TVQ) apparaissant sur la facture</p> <p>Si la taxe locale est inclue, elle ne doit pas être composée uniquement d'espaces ou de zéros. Elle doit être fournie si la</p>

Requis*	Valeur	Limites	Méthode Set	Description
				<p>taxe locale (TVP ou TVQ) s'applique.</p> <p>Valeur minimale de 0,01</p> <p>Valeur maximale de 999 999,99</p> <p>Doit comporter 2 décimales</p>
C	Numéro d'enregistrement des taxes locales (TVP ou TVQ) du commerçant	15 caractères alphanumériques	vsIndependentRefund.setLocalTaxNo(local_tax_no);	<p>Numéro d'enregistrement des taxes locales (TVP ou TVQ) du commerçant</p> <p>Il doit être fourni si des taxes sont incluses dans la facture et si la taxe locale est inclue , il ne doit pas être composé uniquement d'espaces ou zéros.</p> <p>Doit être fourni si la taxe locale (TVP ou TVQ) s'applique</p>
C	Numéro de TVA du client	13 caractères alphanumériques	vsIndependentRefund.setCustomerVatNo(customer_vat_no);	<p>Si le numéro d'enregistrement des taxes du client figure sur la facture pour justifier des transactions exemptées de taxes, il doit être indiqué ici.</p>

Requis*	Valeur	Limites	Méthode Set	Description
C	Code de client/Code de référence du client	16 caractères alphanumériques	vsIndependentRefund.setCri(cri);	Valeur que le client peut fournir au fournisseur au point de vente (il doit être entré si le client le fournit)
N	Code de client	17 caractères alphanumériques	vsIndependentRefund.setCustomerCode(customer_code);	Champ facultatif qui ne sera pas envoyé à Visa, mais qui se trouvera dans les rapports de Moneris
N	Numéro de facture	17 caractères alphanumériques	vsIndependentRefund.setInvoiceNumber(invoice_number);	Champ facultatif qui ne sera pas envoyé à Visa, mais qui se trouvera dans les rapports de Moneris

\* Y = Obligatoire, N = Facultatif, C = Conditionnel

#### Exemple de transaction de remboursement indépendant par carte Visa

```
package Level23;
import JavaAPI.*;
public class TestVsIndependentRefund
{
    public static void main(String[] args)
    {
        String store_id = "moneris";
        String api_token = "hurgle";
        String processing_country_code = "CA";
        boolean status_check = false;

        java.util.Date createDate = new java.util.Date();
        String order_id="Test"+createDate.getTime();
        String cust_id="CUST13343";
        String amount="5.00";
        String pan="4242424254545454";
        String expiry_date="2012"; //YYMM
        String crypt="7";
        String national_tax = "1.23";
        String merchant_vat_no = "gstno111";
        String local_tax = "2.34";
        String customer_vat_no = "gstno999";
        String cri = "CUST-REF-002";
        String customer_code="ccvsfp";
        String invoice_number="invsfp";
        String local_tax_no="ltaxno";
        VsIndependentRefund vsIndependentRefund = new VsIndependentRefund();
        vsIndependentRefund.setOrderId(order_id);
        vsIndependentRefund.setCustId(cust_id);
        vsIndependentRefund.setAmount(amount);
        vsIndependentRefund.setPan(pan);
        vsIndependentRefund.setExpDate(expiry_date);
```

```

vsIndependentRefund.setCryptType(crypt);
vsIndependentRefund.setNationalTax(national_tax);
vsIndependentRefund.setMerchantVatNo(merchant_vat_no);
vsIndependentRefund.setLocalTax(local_tax);
vsIndependentRefund.setCustomerVatNo(customer_vat_no);
vsIndependentRefund.setCri(cri);
vsIndependentRefund.setCustomerCode(customer_code);
vsIndependentRefund.setInvoiceNumber(invoice_number);
vsIndependentRefund.setLocalTaxNo(local_tax_no);
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false for production transactions
mpgReq.setstoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(vsIndependentRefund);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = " + receipt.getBankTotals());
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("CavvResultCode = " + receipt.getCavvResultCode());
}
catch (Exception e)
{
System.out.println(e);
}
}
}
}

```

## 5.2.8 Transaction VS Corpais

Une transaction VS Corpais contiendra tous les champs de données obligatoires et facultatifs pour les données interentreprises de niveaux 2 et 3. Les données VS Corpais peuvent être envoyées lorsque la carte a été identifiée comme étant une carte d'entreprise dans la demande de transaction de préautorisation.

En plus de l'ID de commande et du numéro de transaction, cette transaction contient également deux objets :

- VS Purcha – Données communes des cartes d'entreprise
- VS Purchl – Détails de la ligne d'article

La demande VS Corpais doit être précédée d'une transaction financière (conclusion par carte Visa, transaction forcée par carte Visa, remboursement par carte Visa, remboursement indépendant par carte Visa) et l'indicateur Corporate Card doit être réglé à « true » dans le champ Pre-authorization response.

#### Définition de l'objet de transaction Vs Corpais

```
VsCorpais vsCorpais = new VsCorpais();
```

#### Objet HttpsPostRequest pour les transactions VS Corpais

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
```

```
mpgReq.setTransaction(vsCorpais);
```

#### Champs de demande pour les transactions VS Corpais (obligatoires)

Variable	Type et limites	Méthode Set
No de commande	<i>Chaîne</i>  50 caractères alphanumériques	vsCorpais.setOrderId(order_id);
Numéro de transaction	<i>Chaîne</i>  255 caractères alphanumériques	vsCorpais.setTxnNumber(txn_number);
vsPurcha	<i>Chaîne</i>  Pour obtenir une liste des variables apparaissant dans cet objet, veuillez consulter le tableau ci-dessous.	VsPurcha vsPurcha = new VsPurcha();  vsCorpais.setVsPurch(vsPurcha, vsPurchl);
vsPurchl	<i>Chaîne</i>  Pour obtenir une liste des variables apparaissant dans cet objet, veuillez consulter le tableau ci-dessous.	VsPurchl vsPurchl = new VsPurchl();  vsCorpais.setVsPurch(vsPurcha, vsPurchl);

\* Y = Obligatoire, N = Facultatif, C = Conditionnel

##### 5.2.8.1 Objet VS Purcha – Données communes des cartes d'entreprise

Les transactions VS Corpais utilisent l'objet VS Purcha pour inclure les données de niveau 2.

Variable	Type et limites	Description
Nom de l'acheteur	<p><i>Chaîne</i></p> <p>30 caractères alphanumériques</p>	<p>Nom de l'acheteur ou du destinataire</p> <p><b>REMARQUE :</b> Le nom est requis par l'ARC pour les transaction de plus de 150 \$.</p>
Taux de taxe locale	<p><i>Chaîne</i></p> <p>4 caractères décimaux</p>	<p>Indique le taux de taxe détaillé appliqué en fonction du montant de taxe locale</p> <p><b>EXEMPLE :</b> Une TVP de 8 % devrait être 8,0.</p> <p>Valeur minimale de 0,01</p> <p>Valeur maximale de 99,99</p> <p><b>REMARQUE :</b> * Doit être fourni si la taxe locale (TVP ou TVQ) s'applique.</p>
Droits de douane	<p><i>Chaîne</i></p> <p>9 caractères décimaux</p>	<p>Montant de douane de l'achat total</p> <p>Un montant avec un symbole négatif signifie que le montant est un crédit, un montant avec un symbole positif ou sans symbole signifie que le montant est un débit.</p> <p>Valeur maximale de 999 999,99 (sans symbole)</p>
Traitemet des rabais sur la facture	<p><i>Chaîne</i></p> <p>1 caractère numérique</p>	<p>Indique la façon dont le commerçant gère les rabais</p> <p>Doit être l'une des valeurs suivantes :</p> <p>0 = Aucun rabais ne s'applique au niveau de la facture</p> <p>1 = La taxe est calculée sur les totaux après l'application des rabais</p> <p>2 = La taxe est calculée sur les totaux avant l'application des rabais</p>
Montant du rabais au niveau de la facture	<i>Chaîne</i>	Montant du rabais (s'il est fourni au niveau de la facture selon le traitement)

Variable	Type et limites	Description
	9 caractères décimaux	des rabais sur la facture) Ne doit pas être zéro si la valeur de traitement des rabais sur la facture est 1 ou 2 Valeur minimale de 0,00 et maximale de 999 999,99
Code postal d'expédition	<i>Chaîne</i> 10 caractères alphanumériques	Le code postal ou le code ZIP auquel la marchandise sera expédiée.  <b>REMARQUE :</b> * Requis s'il y a une expédition.  Code postal alphanumérique complet – Format ANA<space>NAN requis si expédié à une adresse canadienne
Code postal d'origine	<i>Chaîne</i> 10 caractères alphanumériques	Code postal ou code ZIP d'origine de la marchandise  Pour les adresses canadiennes, un code postal alphanumérique complet au format ANA<espace>NAN valide est requis pour le commerçant.
Code du pays de destination	2 caractères alphanumériques	Code du pays où la marchandise achetée sera expédiée  Utiliser le format ISO 3166-1 alpha-2  <b>REMARQUE :</b> Requis s'il apparaît sur la facture d'une transaction internationale.
Numéro de référence unique de la facture d'une TVA	<i>Chaîne</i> 25 caractères alphanumériques	Numéro de référence unique de la facture d'une TVA  Doit être rempli avec le numéro de facture, qui ne peut pas être composé uniquement d'espaces ou de zéros
Traitement fiscal	<i>Chaîne</i>	Doit être l'une des valeurs suivantes : 0 = Prix nets avec taxe calculée au niveau de

Variable	Type et limites	Description
	1 caractère alphanumérique	<p>chaque ligne</p> <p>1 = Prix nets avec taxe calculée au niveau de la facture</p> <p>2 = Prix bruts fournis avec les renseignements sur les taxes à chaque ligne</p> <p>3 = Prix bruts fournis avec les renseignements sur la taxe au niveau de la facture</p> <p>4 = Aucune taxe ne s'applique (petit commerçant) sur la facture de la transaction</p>
Montant des frais de transport/expédition	<i>Chaîne</i> 9 caractères décimaux	<p>Frais de transport de l'achat total</p> <p>Si les frais d'expédition ne sont pas inclus dans une ligne d'article, ils doivent être indiqués ici, le cas échéant.</p> <p>Montant en numéraire signé :</p> <p>Le symbole négatif (-) signifie que le montant est un crédit.</p> <p>Le symbole positif (+) signifie que le montant est un crédit.</p> <p>Valeur maximale de 999 999,99 (sans symbole)</p>
Taux des frais de transport TPS ou TVH	<i>Chaîne</i> 4 caractères décimaux	<p>Taux de la TPS (à l'exclusion de la TVP) ou de la TVH appliqué au montant de l'expédition (conformément au traitement fiscal)</p> <p>Si le montant de transport ou d'expédition est fourni, alors ce taux de taxe (TPS nationale ou TVH) doit être fourni.</p> <p>Montant en numéraire, valeur maximale de 99,99 Par exemple, une TVH de 13 % équivaut à 13,00</p>
Montant des frais de transport TPS ou TVH	<i>Chaîne</i> 9 caractères décimaux	Montant de la TPS (excluant la TVP) ou de la TVH appliqué au montant de

Variable	Type et limites	Description
		<p>l'expédition Si le montant de transport ou d'expédition est fourni, alors ce taux de taxe (TPS nationale ou TVH) doit être fourni si la valeur de traitement fiscal est de 0 ou 2.</p> <p>Montant en numéraire signé : valeur maximale sans symbole de 999 999,99</p>

### 5.2.8.2 Objet VS Purchl – Détails de la ligne d'article

Les transactions VS Corpais utilisent l'objet VS Purchl pour inclure les données de niveau 3.

#### Détails de la ligne d'article pour l'objet VS Purchl

```

String[] item_com_code = {"X3101", "X84802"};

String[] product_code = {"CHR123", "DDSK200"};

String[] item_description = {"Office Chair", "Disk Drive"};

String[] item_quantity = {"3", "1"};

String[] item_uom = {"EA", "EA"};

String[] unit_cost = {"0.20", "0.40"};

String[] vat_tax_amt = {"0.00", "0.00"};

String[] vat_tax_rate = {"13.00", "13.00"};

String[] discount_treatmentL = {"0", "0"};

String[] discount_amtl = {"0.00", "0.00"};

```

#### Configuration des détails de la ligne d'article VS Purchl

```
vsPurchl.setVsPurchl(item_com_code[0], product_code[0], item_description[0], item_quantity[0], item_uom[0], unit_cost[0], vat_tax_amt[0], vat_tax_rate[0], discount_treatmentL[0], discount_amtl[0]);
```

```
vsPurchl.setVsPurchl(item_com_code[1], product_code[1], item_description[1], item_quantity[1], item_uom[1], unit_cost[1], vat_tax_amt[1], vat_tax_rate[1], discount_treatmentL[1], discount_amtl[1]);
```

**Tableau 1 – Données communes de carte d'entreprise – Champs de demande de niveau 3 - VSPurchl**

Requis *	Valeur	Limites	Variable/Champ	Description
C	Code de commodité de	12 caractères alphanumériques	item_com_code	Entrez le code de commodité du

Requis *	Valeur	Limites	Variable/Champ	Description
	l'article			produit de la ligne d'article (si ce champ n'est pas rempli, le code du produit doit l'être).
Y	Code du produit	12 caractères alphanumériques	product_code	<p>Code de produit pour cette ligne d'article – code de produit du commerçant, code de produit du fabricant ou code de produit de l'acheteur</p> <p>Il s'agit généralement de l'UGS ou de l'identifiant utilisé par le commerçant pour faire le suivi de l'article ou du service et en fixer le prix.</p> <p>Il devrait toujours être fourni pour chaque ligne d'article.</p>
Y	Description de l'article	35 caractères alphanumériques	item_description	Description de la ligne d'article
Y	Nombre d'article	12 caractères décimaux	item_quantity	<p>Quantité facturée pour cette ligne d'article</p> <p>Jusqu'à quatre décimales sont supportés, les chiffres entiers sont acceptés</p> <p>Valeur minimale de</p>

Requis *	Valeur	Limites	Variable/Champ	Description
				0,0001  Valeur maximale de 999 999 999 999
Y	Unité de mesure de l'article	2 caractères alphanumériques	item_uom	Unité de mesure  Utilisez les unités de mesure et codes permis par la norme ANSI X-12 EDI.
Y	Coût unitaire de l'article	12 caractères décimaux	unit_cost	Coût unitaire de chaque article  De 2 à 4 décimales sont acceptées  Valeur minimale de 0,0001  Valeur maximale de 999 999,9999
N	Montant de la TVA	12 caractères décimaux	vat_tax_amt	Tout montant de taxe sur la valeur ajoutée ou autre taxe de vente  Doit comporter 2 décimales  Valeur minimale de 0,01  Valeur maximale de 999 999,99
N	Taux de la TVA	4 caractères décimaux	vat_tax_rate	Taux de la taxe de vente
				<b>EXEMPLE :</b> Une TVP de 8 % devrait être 8,0.
				Valeur maximale

Requis *	Valeur	Limites	Variable/Champ	Description
				de 99,99
Y	Traitement des rabais	1 caractère numérique	discount_treatmentL	<p>Doit être l'une des valeurs suivantes :</p> <p>0 = Aucun rabais ne s'applique au niveau de la facture</p> <p>1 = La taxe est calculée sur les totaux après l'application des rabais</p> <p>2 = La taxe est calculée sur les totaux avant l'application des rabais</p>
C	Montant du rabais	12 caractères décimaux	discount_amtl	<p>Montant du rabais, s'il est prévu pour cette ligne d'article selon la ligne d'article du montant du rabais (Discount Treatment)</p> <p>Ne doit pas être zéro si la valeur de la ligne d'article du montant du rabais est de 1 ou 2</p> <p>Doit comporter 2 décimales</p> <p>Valeur minimale de 0,01</p> <p>Valeur maximale de 999 999,99</p>

#### 5.2.8.3 Exemple de code pour les transactions VS Corpais

**Exemple de transaction VS Corpais**

```

package Level23;
import JavaAPI.*;

public class TestVsCorpais
{
    public static void main(String[] args)
    {
        String store_id = "moneris";
        String api_token = "hurgle";
        String processing_country_code = "CA";
        boolean status_check = false;

        String order_id="Test1485208069127";
        String txn_number="39791-0_11";
        String buyer_name = "Buyer Manager";
        String local_tax_rate = "13.00";
        String duty_amount = "0.00";
        String discount_treatment = "0";
        String discount_amt = "0.00";
        String freight_amount = "0.20";
        String ship_to_pos_code = "M8X 2W8";
        String ship_from_pos_code = "M1K 2Y7";
        String des_cou_code = "CAN";
        String vat_ref_num = "VAT12345";
        String tax_treatment = "3";//3 = Gross prices given with tax information provided at invoice level
        String gst_hst_freight_amount = "0.00";
        String gst_hst_freight_rate = "13.00";
        String[] item_com_code = {"X3101", "X84802"};
        String[] product_code = {"CHR123", "DDSK200"};
        String[] item_description = {"Office Chair", "Disk Drive"};
        String[] item_quantity = {"3", "1"};
        String[] item_uom = {"EA", "EA"};
        String[] unit_cost = {"0.20", "0.40"};
        String[] vat_tax_amt = {"0.00", "0.00"};
        String[] vat_tax_rate = {"13.00", "13.00"};
        String[] discount_treatmentL = {"0", "0"};
        String[] discount_amtl = {"0.00", "0.00"};
        //Create and set VsPurcha
        VsPurcha vsPurcha = new VsPurcha();
        vsPurcha.setBuyerName(buyer_name);
        vsPurcha.setLocalTaxRate(local_tax_rate);
        vsPurcha.setDutyAmount(duty_amount);
        vsPurcha.setDiscountTreatment(discount_treatment);
        vsPurcha.setDiscountAmt(discount_amt);
        vsPurcha.setFreightAmount(freight_amount);
        vsPurcha.setShipToPostalCode(ship_to_pos_code);
        vsPurcha.setShipFromPostalCode(ship_from_pos_code);
        vsPurcha.setDesCouCode(des_cou_code);
        vsPurcha.setVatRefNum(vat_ref_num);
        vsPurcha.setTaxTreatment(tax_treatment);
        vsPurcha.setGstHstFreightAmount(gst_hst_freight_amount);
        vsPurcha.setGstHstFreightRate(gst_hst_freight_rate);
        //Create and set VsPurchl
        VsPurchl vsPurchl = new VsPurchl();
        vsPurchl.setVsPurchl(item_com_code[0], product_code[0], item_description[0], item_quantity[0],
        item_uom[0], unit_cost[0], vat_tax_amt[0], vat_tax_rate[0], discount_treatmentL[0],
        discount_amtl[0]);
        vsPurchl.setVsPurchl(item_com_code[1], product_code[1], item_description[1], item_quantity[1],
        item_uom[1], unit_cost[1], vat_tax_amt[1], vat_tax_rate[1], discount_treatmentL[1],
        discount_amtl[1]);

        VsCorpais vsCorpais = new VsCorpais();
        vsCorpais.setOrderId(order_id);
        vsCorpais.setTxnNumber(txn_number);
        vsCorpais.setVsPurch(vsPurcha, vsPurchl);
        HttpsPostRequest mpgReq = new HttpsPostRequest();
        mpgReq.setProcCountryCode(processing_country_code);
        mpgReq.setTestMode(true); //false for production transactions
        mpgReq.setstoreId(store_id);
        mpgReq.setApiToken(api_token);
        mpgReq.setTransaction(vsCorpais);
        mpgReq.setStatusCheck(status_check);
        mpgReq.send();
    }
}

```

```
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = " + receipt.getBankTotals());
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("CavvResultCode = " + receipt.getCavvResultCode());
}
catch (Exception e)
{
System.out.println(e);
}
}
```

## 5.3 Transactions de niveau 2 et 3 de Mastercard

- 5.3.1 Types de transaction de niveaux 2 et 3 par carte Mastercard
- 5.3.2 Flux de transaction de niveaux 2 et 3 par carte Mastercard
- 5.3.3 Conclusion par carte Mastercard
- 5.3.4 Transaction forcée par carte Mastercard
- 5.3.5 Correction d'achat par carte Mastercard
- 5.3.6 Remboursement par carte Mastercard
- 5.3.7 Remboursement indépendant par carte Mastercard
- 1 MC Corpais – Transactions de niveaux 2 et 3

### 5.3.1 Types de transaction de niveaux 2 et 3 par carte Mastercard

Cet ensemble de transactions comprend une série de transactions financières par carte d'entreprise ainsi qu'une transaction qui permet de transmettre des données de niveaux 2 et 3. Veuillez vous assurer les données de niveaux 2 et 3 sont prises en charge pour les cartes Mastercard dans votre compte de commerçant. Les transactions de fermeture de lot, d'ouverture des totaux et de préautorisation sont identiques aux transactions indiquées dans la section Ensemble de transactions de base (page 16).

Lorsque la réponse à la préautorisation contient une valeur CorporateCard égale à true, vous pouvez soumettre les transactions par carte Mastercard.

Si la valeur CorporateCard est false, la carte ne prend pas en charge les données de niveau 2 et 3, il faut donc utiliser des transactions autres que celles de niveau 2 et 3. Si la carte n'est pas une carte d'entreprise, veuillez consulter la section 4 pour connaître les transactions appropriées pour les cartes autres que des cartes d'entreprise.

**REMARQUE :** Cette série de transactions est destinée aux transactions où la valeur Corporate Card est true pour lesquelles des données de niveaux 2 et 3 sont soumises. S'il s'avère que la carte de crédit est une carte d'entreprise, mais que vous ne souhaitez pas envoyer de données de niveaux 2 et 3, vous pouvez soumettre des transactions par carte Mastercard en utilisant l'ensemble de transactions de base de la section .Ensemble de transactions de base( page 16).

### **Préautorisation – (Transaction de préautorisation)**

La transaction de préautorisation vérifie que les fonds sont présents sur la carte de crédit du client et les bloque. Ces fonds sont bloqués pour une durée prédéterminée, en fonction de l'émetteur de carte. Pour récupérer les fonds bloqués par une transaction de préautorisation afin de les déposer dans le compte du commerçant, une transaction de conclusion doit être effectuée. L'envoi de données de niveaux 2 et 3 n'est pas pris en charge dans le cadre d'une préautorisation, car une préautorisation n'est pas réglée. Lorsque la valeur CorporateCard est « true », les données de niveaux 2 et 3 peuvent être envoyées.

### **Conclusion par carte Mastercard – (Conclusion de préautorisation)**

Une fois la préautorisation obtenue, les fonds bloqués doivent être récupérés sur la carte de crédit du client. Cette transaction récupère les fonds bloqués et les prépare à être déposés dans le compte du commerçant. Avant de réaliser une conclusion par carte Mastercard, une préautorisation doit être effectuée.

### **Transaction forcée par carte Mastercard – (Conclusion de préautorisation forcée)**

Cette transaction est une autre façon d'obtenir les fonds bloqués lors d'une transaction de préautorisation traitée par RVI ou par un terminal équivalent. Une transaction forcée par carte Mastercard nécessite le code d'autorisation original et récupère les fonds bloqués et les prépare à être déposés dans le compte du commerçant.

### **Correction d'achat par carte Mastercard – (Annulation, correction)**

Les transactions de conclusion par carte Mastercard peuvent être annulées le jour même\* où elles se produisent. Une annulation doit annuler le montant total de la transaction et supprimera toute trace de celle-ci sur le relevé du titulaire de la carte. \* Une correction d'achat par carte Mastercard peut être effectuée pour annuler une transaction tant que le lot qui contient la transaction originale reste ouvert. Lorsque la fonction de fermeture automatique est utilisée, le lot est fermé tous les jours entre 22 h et 23 h HNE.

### **Remboursement par carte Mastercard – (Crédit)**

Un remboursement par carte Mastercard peut être effectué pour annuler la totalité ou une partie du montant d'une transaction de conclusion ou forcée par carte Mastercard traitée précédemment.

**Remboursement indépendant par carte Mastercard – (Crédit)**

Un remboursement indépendant par carte Mastercard peut être effectué pour rembourser une partie ou la totalité du montant d'une transaction de conclusion. Un remboursement indépendant est utilisé lorsque la transaction d'origine n'a pas été effectuée par l'entremise de Passerelle Moneris. Veuillez noter que votre compte peut se prévaloir ou non de la fonctionnalité de transaction de remboursement indépendant par carte Mastercard. Si vous recevez un message d'erreur indiquant que la transaction n'est pas autorisée lorsque vous tentez de traiter un remboursement indépendant par carte Mastercard, cela peut signifier que la transaction n'est pas prise en charge par votre compte. Si vous souhaitez activer (ou réactiver) temporairement les transactions de remboursement indépendant par carte Mastercard, veuillez communiquer avec le centre de services en composant le 1 866 319-7450.

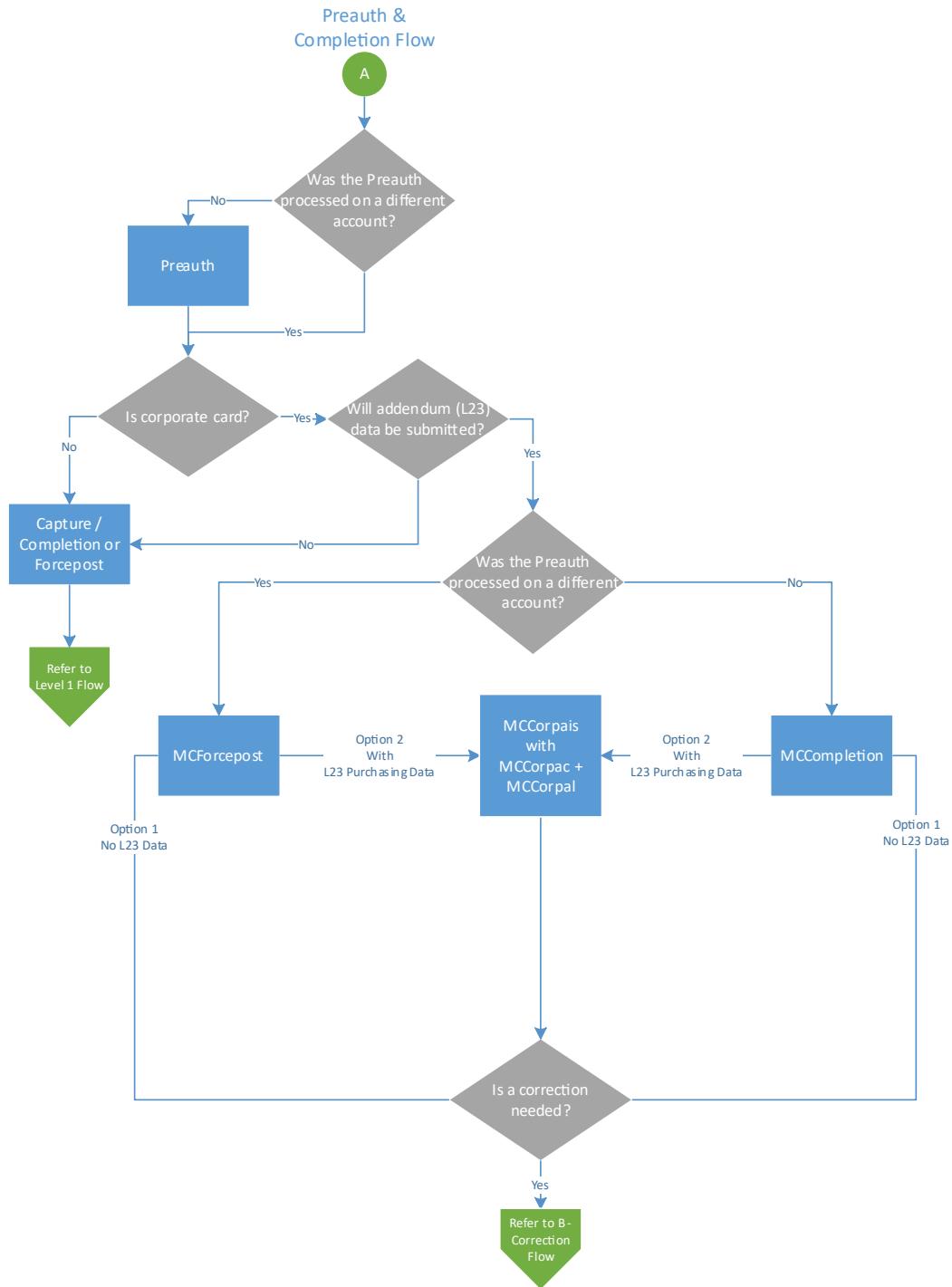
**Élément commun de la ligne d'article MC Corpais – (Données de niveaux 2 et 3)**

Les éléments communs de la ligne d'article MC Corpais contiendront tous les champs de données obligatoires et facultatifs pour les données de niveaux 2 et 3. Les données des éléments communs de la ligne d'article MC Corpais peuvent être envoyées lorsque la carte a été identifiée comme étant une carte d'entreprise dans la demande de transaction de préautorisation. Voici les types de données et de combinaisons que ce type de transaction prend en charge :

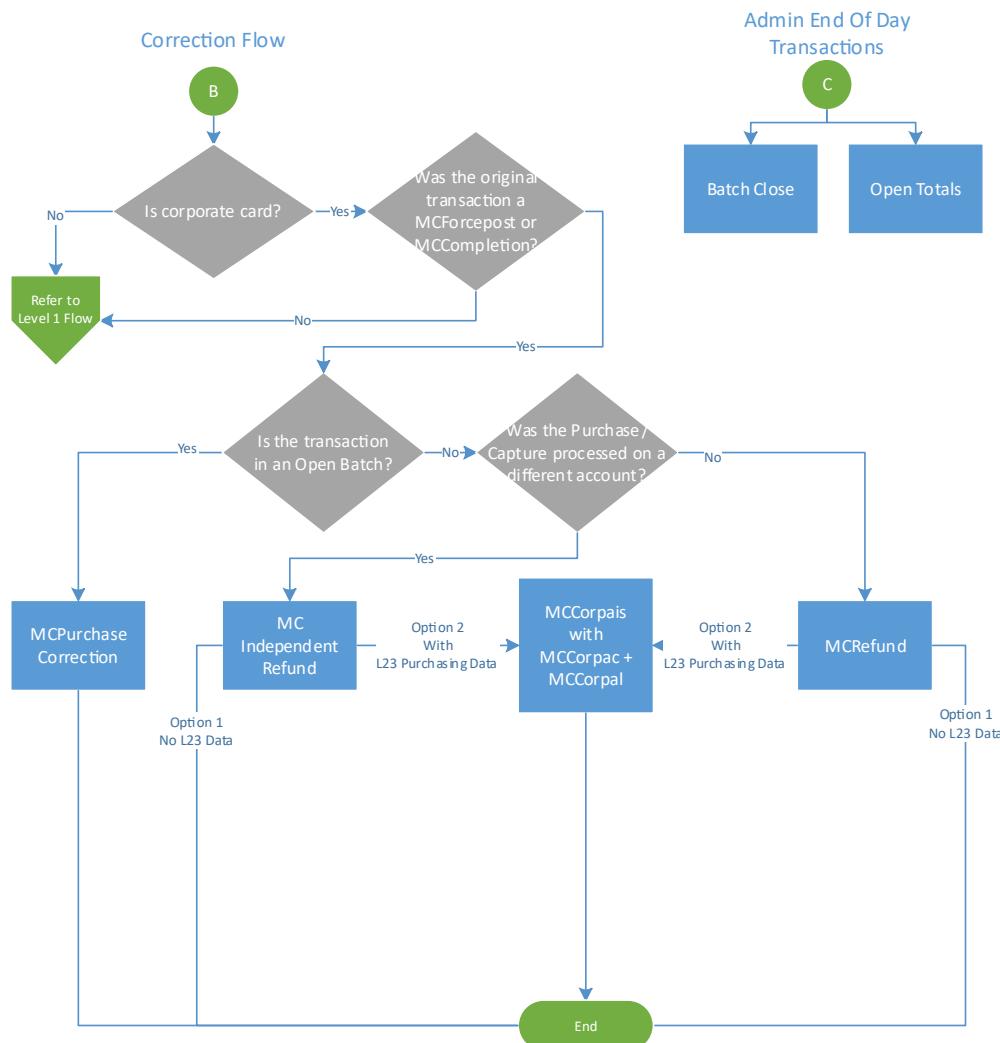
- Données de carte d'achat :
- Données communes de carte d'entreprise avec les détails de l'élément de la ligne d'article

### 5.3.2 Flux de transaction de niveaux 2 et 3 par carte Mastercard

#### Flux de transaction de préautorisation et de conclusion



## Flux de transaction de correction d'achat



### 5.3.3 Conclusion par carte Mastercard

Une transaction de conclusion par carte Mastercard est utilisée pour récupérer les fonds bloqués par une transaction de préautorisation. Lors de l'envoi d'une demande de conclusion, vous aurez besoin de deux renseignements provenant de la réponse de la préautorisation originale, soit l'ID de commande et le numéro de transaction.

Lorsque vous aurez terminé cette transaction avec succès, pour soumettre toutes les données supplémentaires de niveaux 2 et 3, veuillez effectuer une transaction MC Corpais.

#### Définition de l'objet de transaction MC Completion

```
McCompletion mcCompletion = new McCompletion();
```

#### Objet HttpsPostRequest pour les transactions de conclusion par carte Mastercard

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setTransaction(mcCompletion);
```

#### Champs de demande pour les transactions de conclusion par carte Mastercard (obligatoires)

Variable	Limites	Méthode Set
No de commande	<i>Chaîne</i> 50 caractères alphanumériques	mcCompletion.setOrderId(order_id);
Montant de la conclusion	<i>Chaîne</i> 10 caractères décimaux Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal  <b>EXEMPLE : 1 234 567,89</b>	mcCompletion.setCompAmount(comp_amount);
Numéro de transaction	<i>Chaîne</i> 255 caractères alphanumériques	mcCompletion.setTxnNumber(txn_number);
Numéro de référence du commerçant	<i>Chaîne</i> 19 caractères alphanumériques	mcCompletion.setMerchantRefNo(merchant_ref_no);
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	mcCompletion.setCryptType(crypt);

### Exemple de conclusion par carte Mastercard

```
package Level23;
import JavaAPI.*;

public class TestMcCompletion
{
    public static void main(String[] args)
    {
        String store_id = "moneris";
        String api_token = "hurgle";
        String processing_country_code = "CA";
        boolean status_check = false;

        String order_id="Test1485206444761";
        String comp_amount="1.00";
        String txn_number="39777-0_11";
        String crypt="7";
        String merchant_ref_no = "319038";
        McCompletion mcCompletion = new McCompletion();
        mcCompletion.setOrderId(order_id);
        mcCompletion.setCompAmount(comp_amount);
        mcCompletion.setTxnNumber(txn_number);
        mcCompletion.setCryptType(crypt);
        mcCompletion.setMerchantRefNo(merchant_ref_no);
        HttpsPostRequest mpgReq = new HttpsPostRequest();
        mpgReq.setProcCountryCode(processing_country_code);
        mpgReq.setTestMode(true); //false for production transactions
        mpgReq.setstoreId(store_id);
        mpgReq.setApiToken(api_token);
        mpgReq.setTransaction(mcCompletion);
        mpgReq.setStatusCheck(status_check);
        mpgReq.send();
        try
        {
            Receipt receipt = mpgReq.getReceipt();
            System.out.println("CardType = " + receipt.getCardType());
            System.out.println("TransAmount = " + receipt.getTransAmount());
            System.out.println("TxnNumber = " + receipt.getTxnNumber());
            System.out.println("ReceiptId = " + receipt.getReceiptId());
            System.out.println("TransType = " + receipt.getTransType());
            System.out.println("ReferenceNum = " + receipt.getReferenceNum());
            System.out.println("ResponseCode = " + receipt.getResponseCode());
            System.out.println("ISO = " + receipt.getISO());
            System.out.println("BankTotals = " + receipt.getBankTotals());
            System.out.println("Message = " + receipt.getMessage());
            System.out.println("AuthCode = " + receipt.getAuthCode());
            System.out.println("Complete = " + receipt.getComplete());
            System.out.println("TransDate = " + receipt.getTransDate());
            System.out.println("TransTime = " + receipt.getTransTime());
            System.out.println("Ticket = " + receipt.getTicket());
            System.out.println("TimedOut = " + receipt.getTimedOut());
            System.out.println("CavvResultCode = " + receipt.getCavvResultCode());
        }
        catch (Exception e)
        {
            System.out.println(e);
        }
    }
}
```

### 5.3.4 Transaction forcée par carte Mastercard

Une transaction forcée par carte Mastercard est utilisée pour récupérer les fonds bloqués par une transaction de pré-autorisation effectuée par RVI ou par un terminal équivalent. Lors de l'envoi d'une demande de transaction forcée, vous aurez besoin de l'ID de la commande, du montant, du numéro de compte primaire, de la date d'expiration, d'une réponse chiffrée et du code d'autorisation reçus dans la réponse de pré-autorisation.

#### Définition d'objet de transaction MC Force Post

```
McForcePost mcforcepost= new McForcePost();
```

### Objet **HttpsPostRequest** pour les transactions forcées par carte Mastercard

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
```

```
mpgReq.setTransaction(mcforcepost);
```

#### Champs de demande pour les transactions forcées par carte Mastercard (obligatoires)

Variable	Type et limites	Méthode Set
No de commande	<i>Chaîne</i> 50 caractères alphanumériques	mcforcepost.setOrderId(order_id);
Montant	<i>Chaîne</i> 10 caractères décimaux Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal	mcforcepost.setAmount(amount);
	<b>EXEMPLE : 1 234 567,89</b>	
Numéro de carte de crédit	<i>Chaîne</i> 20 caractères alphanumériques	mcforcepost.setPan(pan);
Date d'expiration	<i>Chaîne</i> 4 caractères alphanumériques (format AAMM)	mcforcepost.setExpDate(expiry_date);
Code d'autorisation	<i>Chaîne</i> 8 caractères alphanumériques	mcforcepost.setAuthCode(auth_code);
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	mcforcepost.setCryptType(crypt);
Numéro de référence du commerçant	<i>Chaîne</i> 19 caractères alphanumériques	mcforcepost.setMerchantRefNo(merchant_ref_no);

#### Champs de demande pour les transactions forcées par carte Mastercard (facultatifs)

Variable	Type et limites	Méthode Set
ID du client	<i>Chaîne</i> 50 caractères alphanumériques	<code>mcforcepost.setCustId(cust_id);</code>

### Exemple de transaction forcée par carte Mastercard

```

package Level23;
import JavaAPI.*;

public class TestMcForcePost
{
  public static void main(String[] args)
  {
    String store_id = "moneris";
    String api_token = "hurgle";
    String processing_country_code = "CA";
    boolean status_check = false;

    java.util.Date createDate = new java.util.Date();
    String order_id="Test"+createDate.getTime();
    String cust_id = "CUST13343";
    String amount = "5.00";
    String pan = "5454545442424242";
    String expiry_date = "1912"; //YYMM
    String auth_code = "123456";
    String crypt = "7";
    String merchant_ref_no = "319038";
    McForcePost mcforcepost = new McForcePost();
    mcforcepost.setOrderId(order_id);
    mcforcepost.setCustId(cust_id);
    mcforcepost.setAmount(amount);
    mcforcepost.setPan(pan);
    mcforcepost.setExpDate(expiry_date);
    mcforcepost.setAuthCode(auth_code);
    mcforcepost.setCryptType(crypt);
    mcforcepost.setMerchantRefNo(merchant_ref_no);
    HttpsPostRequest mpgReq = new HttpsPostRequest();
    mpgReq.setProcCountryCode(processing_country_code);
    mpgReq.setTestMode(true); //false for production transactions
    mpgReq.setstoreId(store_id);
    mpgReq.setApiToken(api_token);
    mpgReq.setTransaction(mcforcepost);
    mpgReq.setStatusCheck(status_check);
    mpgReq.send();
    try
    {
      Receipt receipt = mpgReq.getReceipt();
      System.out.println("CardType = " + receipt.getCardType());
      System.out.println("TransAmount = " + receipt.getTransAmount());
      System.out.println("TxnNumber = " + receipt.getTxnNumber());
      System.out.println("ReceiptId = " + receipt.getReceiptId());
      System.out.println("TransType = " + receipt.getTransType());
      System.out.println("ReferenceNum = " + receipt.getReferenceNum());
      System.out.println("ResponseCode = " + receipt.getResponseCode());
      System.out.println("ISO = " + receipt.getISO());
      System.out.println("BankTotals = " + receipt.getBankTotals());
      System.out.println("Message = " + receipt.getMessage());
      System.out.println("AuthCode = " + receipt.getAuthCode());
      System.out.println("Complete = " + receipt.getComplete());
      System.out.println("TransDate = " + receipt.getTransDate());
      System.out.println("TransTime = " + receipt.getTransTime());
      System.out.println("Ticket = " + receipt.getTicket());
      System.out.println("TimedOut = " + receipt.getTimedOut());
      System.out.println("CavvResultCode = " + receipt.getCavvResultCode());
    }
    catch (Exception e)
    {
      System.out.println(e);
    }
  }
}

```

```
}
```

```
}
```

### 5.3.5 Correction d'achat par carte Mastercard

Une correction d'achat par carte Mastercard (annulation) est utilisée pour annuler une transaction effectuée dans le lot en cours. Aucun montant n'est requis, car la correction d'achat annule toujours la totalité du montant de la transaction initiale. La seule transaction pouvant être annulée est la transaction de conclusion. Pour envoyer une annulation, l'ID de commande et le numéro de transaction de la conclusion par carte Mastercard ou de la transaction forcée par carte Mastercard est requis.

#### Définition de l'objet de transaction MC Purchase Correction

```
McPurchaseCorrection mcpurchasecorrection = new McPurchaseCorrection();
```

#### Objet HttpsPostRequest pour les transactions de correction d'achat par carte Mastercard

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
```

```
mpgReq.setTransaction(mcpurchasecorrection);
```

#### Champs de demande liés aux transactions de correction d'achat par carte Mastercard (obligatoires)

Variable	Type et limites	Méthode Set
No de commande	<i>Chaîne</i> 50 caractères alphanumériques	mcpurchasecorrection.setOrderI d(order_id);
Numéro de transaction	<i>Chaîne</i> 255 caractères alphanumériques	mcpurchasecorrection.setTxnNum ber(txn_number);
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	mcpurchasecorrection.setCryptT ype(crypt);

### Exemple de transaction de correction d'achat par carte Mastercard

```

package Level23;
import JavaAPI.*;

public class TestMcPurchaseCorrection
{
    public static void main(String[] args)
    {
        String store_id = "moneris";
        String api_token = "hurgle";
        String processing_country_code = "CA";
        boolean status_check = false;

        String order_id="Test1485207871499";
        String txn_number="660117311902017023164431860-0_11";
        String crypt="7";
        McPurchaseCorrection mcpurchasecorrection = new McPurchaseCorrection();
        mcpurchasecorrection.setOrderId(order_id);
        mcpurchasecorrection.setTxnNumber(txn_number);
        mcpurchasecorrection.setCryptType(crypt);
        HttpsPostRequest mpgReq = new HttpsPostRequest();
        mpgReq.setProcCountryCode(processing_country_code);
        mpgReq.setTestMode(true); //false for production transactions
        mpgReq.setstoreId(store_id);
        mpgReq.setApiToken(api_token);
        mpgReq.setTransaction(mcpurchasecorrection);
        mpgReq.setStatusCheck(status_check);
        mpgReq.send();
        try
        {
            Receipt receipt = mpgReq.getReceipt();
            System.out.println("CardType = " + receipt.getCardType());
            System.out.println("TransAmount = " + receipt.getTransAmount());
            System.out.println("TxnNumber = " + receipt.getTxnNumber());
            System.out.println("ReceiptId = " + receipt.getReceiptId());
            System.out.println("TransType = " + receipt.getTransType());
            System.out.println("ReferenceNum = " + receipt.getReferenceNum());
            System.out.println("ResponseCode = " + receipt.getResponseCode());
            System.out.println("ISO = " + receipt.getISO());
            System.out.println("BankTotals = " + receipt.getBankTotals());
            System.out.println("Message = " + receipt.getMessage());
            System.out.println("AuthCode = " + receipt.getAuthCode());
            System.out.println("Complete = " + receipt.getComplete());
            System.out.println("TransDate = " + receipt.getTransDate());
            System.out.println("TransTime = " + receipt.getTransTime());
            System.out.println("Ticket = " + receipt.getTicket());
            System.out.println("TimedOut = " + receipt.getTimedOut());
            System.out.println("CavvResultCode = " + receipt.getCavvResultCode());
        }
        catch (Exception e)
        {
            System.out.println(e);
        }
    }
}

```

### 5.3.6 Remboursement par carte Mastercard

Une transaction de remboursement par carte Mastercard crédite un montant précis à la carte de crédit du titulaire de carte. Un remboursement d'une valeur allant jusqu'à la valeur totale de la transaction de la transaction de conclusion originale peut être envoyé. Pour effectuer une transaction de remboursement, vous aurez besoin de l'ID de commande et du numéro de la transaction de conclusion ou forcée par carte Mastercard originale.

#### Définition de l'objet de transaction MC Refund

```
McRefund mcRefund = new McRefund();
```

#### Objet HttpsPostRequest pour les transactions de remboursement par carte Mastercard

```
HttpsPostRequest mpgReq = new HttpsPostRequest();

mpgReq.setTransaction(mcRefund);
```

#### Champs de demande liés aux transactions de remboursement par carte Mastercard (obligatoires)

Variable	Type et limites	Méthode Set
No de commande	50 caractères alphanumériques	mcRefund.setOrderId(order_id);
Montant	10 caractères décimaux  Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal	mcRefund.setAmount(amount);
	<b>EXEMPLE : 1 234 567,89</b>	
Numéro de transaction	255 caractères alphanumériques	mcRefund.setTxnNumber(txn_number);

Variable	Type et limites	Méthode Set
Indicateur de commerce électronique	1 caractère alphanumérique	mcRefund.setCryptType(crypt);
Numéro de référence du commerçant	19 caractères alphanumériques	mcRefund.setMerchantRefNo(merchant_ref_no);

#### Exemple de remboursement par carte Mastercard

```

package Level123;
import JavaAPI.*;
public class TestMcRefund
{
    public static void main(String[] args)
    {
        String store_id = "moneris";
        String api_token = "hurgle";
        String processing_country_code = "CA";
        boolean status_check = false;

        String order_id="Test1485207913048";
        String amount="5.00";
        String txn_number="660117311902017023164513403-0_11";
        String crypt="7";
        String merchant_ref_no = "319038";
        McRefund mcRefund = new McRefund();
        mcRefund.setOrderId(order_id);
        mcRefund.setAmount(amount);
        mcRefund.setTxnNumber(txn_number);
        mcRefund.setCryptType(crypt);
        mcRefund.setMerchantRefNo(merchant_ref_no);
        HttpsPostRequest mpgReq = new HttpsPostRequest();
        mpgReq.setProcCountryCode(processing_country_code);
        mpgReq.setTestMode(true); //false for production transactions
        mpgReq.setstoreId(store_id);
        mpgReq.setApiToken(api_token);
        mpgReq.setTransaction(mcRefund);
        mpgReq.setStatusCheck(status_check);
        mpgReq.send();
        try
        {
            Receipt receipt = mpgReq.getReceipt();
            System.out.println("CardType = " + receipt.getCardType());
            System.out.println("TransAmount = " + receipt.getTransAmount());
            System.out.println("TxnNumber = " + receipt.getTxnNumber());
            System.out.println("ReceiptId = " + receipt.getReceiptId());
            System.out.println("TransType = " + receipt.getTransType());
            System.out.println("ReferenceNum = " + receipt.getReferenceNum());
            System.out.println("ResponseCode = " + receipt.getResponseCode());
            System.out.println("ISO = " + receipt.getISO());
            System.out.println("BankTotals = " + receipt.getBankTotals());
            System.out.println("Message = " + receipt.getMessage());
            System.out.println("AuthCode = " + receipt.getAuthCode());
            System.out.println("Complete = " + receipt.getComplete());
            System.out.println("TransDate = " + receipt.getTransDate());
            System.out.println("TransTime = " + receipt.getTransTime());
            System.out.println("Ticket = " + receipt.getTicket());
            System.out.println("TimedOut = " + receipt.getTimedOut());
            System.out.println("CavvResultCode = " + receipt.getCavvResultCode());
        }
        catch (Exception e)
        {
            System.out.println(e);
        }
    }
}

```

### 5.3.7 Remboursement indépendant par carte Mastercard

Une transaction de remboursement indépendant par carte Mastercard est utilisée lorsque la transaction d'origine n'a pas été effectuée par l'entremise de Passerelle Moneris et ne nécessite pas qu'une commande existante soit enregistrée dans Passerelle Moneris, mais le numéro de la carte de crédit et la date d'expiration doivent être transmis. Le format de la transaction est presque identique à celui d'un achat ou d'une préautorisation.

**REMARQUE :** Ce ne sont pas tous les comptes qui prennent en charge les transactions de remboursement indépendant. Si vous recevez un message d'erreur indiquant que la transaction n'est pas autorisée lorsque vous tentez un remboursement indépendant, cela peut signifier que cette fonction n'est pas prise en charge par votre compte. Si vous souhaitez activer (ou réactiver) temporairement les transactions de remboursement indépendant, veuillez communiquer avec le centre de services de Moneris en composant le 1 866 319-7450.

---

Lorsque vous aurez terminé cette transaction avec succès, pour soumettre toutes les données supplémentaires de niveaux 2 et 3, veuillez effectuer une transaction MC Corpais.

---

#### Définition de l'objet de transaction MC Independent Refund

```
McIndependentRefund mcindrefund = new McIndependentRefund();
```

#### Objet HttpsPostRequest pour les transactions de remboursement indépendant par carte Mastercard

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
```

```
mpgReq.setTransaction(mcindrefund);
```

**Champs de demande liés aux transactions de remboursement indépendant par carte Mastercard (obligatoires)**

Variable	Type et limites	Méthode Set
No de commande	50 caractères alphanumériques	mcindrefund.setOrderId(order_id);
Montant	10 caractères décimaux Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal  <b>EXEMPLE : 1 234 567,89</b>	mcindrefund.setAmount(amount);
Indicateur de commerce électronique	1 caractère alphanumérique	mcindrefund.setCryptType(crypt);
Numéro de carte de crédit	20 caractère numérique	mcindrefund.setPan(pan);
Date d'expiration	4 caractères numériques (format AAMM)	mcindrefund.setExpDate(expiry_date);
Numéro de référence du commerçant	19 caractères alphanumériques	mcindrefund.setMerchantRefNo(merchant_ref_no);

**Champs de demande liés aux transactions de remboursement indépendant par carte Mastercard (facultatifs)**

**Tableau 1 Valeurs facultatives de l'objet de transaction MC Independent Refund**

Variable	Type et limites	Méthode Set
ID du client	Chaîne 50 caractères alphanumériques	mcindrefund.setCustId(cust_id);

**Exemple d'une transaction de remboursement indépendant par carte Mastercard**

```
package Level123;
import JavaAPI.*;
public class TestMcIndependentRefund
{
    public static void main(String[] args)
    {
        String store_id = "moneris";
        String api_token = "hurgle";
        String processing_country_code = "CA";
```

```

boolean status_check = false;

java.util.Date createDate = new java.util.Date();
String order_id="Test"+createDate.getTime();
String cust_id = "CUST13343";
String amount = "5.00";
String pan = "5454545442424242";
String expiry_date = "1912"; //YYMM
String crypt = "7";
String merchant_ref_no = "319038";
McIndependentRefund mcindrefund = new McIndependentRefund();
mcindrefund.setOrderId(order_id);
mcindrefund.setCustomerId(cust_id);
mcindrefund.setAmount(amount);
mcindrefund.setPan(pan);
mcindrefund.setExpDate(expiry_date);
mcindrefund.setCryptType(crypt);
mcindrefund.setMerchantRefNo(merchant_ref_no);
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false for production transactions
mpgReq.setstoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(mcindrefund);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = " + receipt.getBankTotals());
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("CavvResultCode = " + receipt.getCavvResultCode());
}
catch (Exception e)
{
System.out.println(e);
}
}
}
}

```

### 5.3.8 MC Corpais – Données communes de carte d'entreprise avec les détails de la ligne d'article

Cet exemple de transaction inclut les éléments suivants pour le traitement des données de carte d'achat d'entreprise de niveaux 2 et 3 :

- Les données communes des cartes d'entreprise (MC Corpac)
- Un seul ensemble de champs MC Corpac peut être soumis.
- Cet ensemble de données comprend des éléments de données qui s'appliquent à la commande globale, p. ex. le total général des taxes.
- Détails de la ligne d'article (MC Corpal)
- Il est possible de soumettre de 1 à 998 lignes d'articles MC Corpal.
- Cet ensemble de données comprend les détails de chaque article ou service acheté.

La demande MC Corpais doit être précédée d'une transaction financière (conclusion par carte Mastercard, transaction forcée par carte Mastercard, remboursement par carte Mastercard, remboursement indépendant par carte Mastercard) et l'indicateur Corporate Card doit indiquer « true » dans le champ Preautorisation response. La demande de transaction MC Corpais doit contenir l'ID de commande de la transaction financière ainsi que le numéro de transaction.

De plus, la transaction MC Corpais dispose d'un objet tax array qui peut être envoyé par les champs liés aux taxes de MC Corpac et MC Corpal. Pour en savoir plus sur l'objet tax array, consultez la section 6.3.8.3 Objet Tax Array – MC Corpais.

Pour obtenir les descriptions des champs de niveaux 2 et 3, veuillez consulter la section Définition des champs de demande pour les transactions de niveaux 2 et 3 liées à Mastercard (à la page 514).

#### Définition de l'objet de transaction MC Corpais

```
McCorpais mcCorpais = new McCorpais();
```

#### Objet HttpsPostRequest pour les transactions MC Corpais

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
```

```
mpgReq.setTransaction(mcCorpais);
```

#### Valeurs de l'objet de transaction MC Corpais

Tableau 1 Valeurs obligatoire de l'objet de transaction MC Corpais

Valeur	Type	Limites	Méthode Set
No de commande	Chaîne	50 caractères alphanumériques	mcCorpais.setOrderId(order_id);
Numéro de transaction	Chaîne	255 caractères alphanumériques	mcCorpais.setTxnNumber(txn_number);
MC Corpac	Objet	S. O.	mcCorpac.setMcCorpac(mcCorpac);

Valeur	Type	Limites	Méthode Set
MC Corpal	Objet	S. O.	mcCorpal.setMcCorpal(mcCorpal);

\* Y = Obligatoire, N = Facultatif, C = Conditionnel

### 5.3.8.1 MC Corpac – Données communes des cartes d'entreprise

Tableau 1 – Données communes de carte d'entreprise – Champs de demande de niveau 2 - MCCorpac

Requis*	Valeur	Limites	Méthode Set	Description
N	Numéro Austin-Tetra	15 caractères alphanumériques	mcCorpac.setAustinTetraNumber(austin_tetra_number);	Numéro Autin-Tetra attribué à l'accepteur de carte
N	Code SCIAN (NAICS code)	15 caractères alphanumériques	mcCorpac.setNaicsCode(naics_code);	Code du système de classification des industries de l'Amérique du Nord (SCIAN) attribué à l'accepteur de la carte
N	Code de client	25 caractères alphanumériques	mcCorpac.setCustomerCode1(customer_code1_c);	Un numéro de contrôle, tel qu'un numéro de bon de commande, un numéro de projet, un numéro de répartition de service ou un nom donné au commerçant par le fournisseur.  Justifié à gauche et peut comporter des espaces
N	Numéro de facture unique	17 caractères alphanumériques	mcCorpac.setUniqueInvoiceNumber(unique_invoice_number_c);	Numéro unique associé à la transaction individuelle fourni par le commerçant
N	Code de marchandise	15 caractères alphanumériques	mcCorpac.setCommodityCode(commodity_code);	Code attribué par le commerçant qui catégorise le mieux l'article acheté

Requis*	Valeur	Limites	Méthode Set	Description
N	Date de la commande	6 caractère numérique Format AAMMJJ	mcCorpac.setOrderDate(order_date_c);	Date d'achat de l'article
				<b>REMARQUE :</b> Si présent, doit être une date valide.
N	Numéro TVA d'entreprise	20 caractères alphanumériques	mcCorpac.setCorporationVatNumber(corporation_vat_number_c);	Contient le numéro de taxe sur la valeur ajoutée (TVA) d'une entreprise
N	Numéro de TVA du client	20 caractères alphanumériques	mcCorpac.setCustomerVatNumber(customer_vat_number_c);	Contient le numéro de TVA du client ou du titulaire de carte qui est utilisé pour identifier le client lors de l'achat de biens et de services vendus par le commerçant
N	Montant des frais de transport	12 caractères décimaux	mcCorpac.setFreightAmount1(freight_amount_c);	Frais d'expédition de l'achat total  Doit contenir 2 décimales  Valeur minimale de 0,00 et maximale de 999 999,99
N	Droits de douane	12 caractères décimaux	mcCorpac.setDutyAmount1(duty_amount_c);	Frais de douane qui s'appliquent au montant total de l'achat  Doit contenir 2 décimales  Valeur minimale de 0,00  Valeur maximale de 999 999,99
N	Code de	3 caractères	mcCorpac.setDestinationProvinceCode(destination_province_code);	Pays, province ou État

Requis*	Valeur	Limites	Méthode Set	Description
	l'État ou de la province de destination	alphanumériques		où la marchandise sera expédiée Justifié à gauche avec des espaces de fin
				<b>EXEMPLE :</b> ONT = Ontario
N	Code du pays de destination	3 caractères alphanumériques  Format ISO 3166-1 alpha-3	<code>mcCorpac.setDestinationCountryCode(destination_country_code);</code>	Code du pays où la marchandise sera expédiée  Justifié à gauche avec des espaces de fin  Format ISO 3166-1 alpha-3
				<b>EXEMPLE :</b> CAN = Canada
N	Code postal d'origine	10 caractères alphanumériques  Format ANA NAN	<code>mcCorpac.setShipFromPosCode(ship_from_pos_code);</code>	Code postal ou code ZIP d'origine de la marchandise  Code postal alphanumérique complet – Format ANA<space>NAN valide
N	Code postal de destination	10 caractères alphanumériques	<code>mcCorpac.setShipToPosCode(ship_to_pos_code_c);</code>	Code postal ou code ZIP auquel la marchandise sera expédiée  Code postal alphanumérique complet – Format ANA<space>NAN si expédié à une adresse canadienne
N	Nom de la	36 caractères	<code>mcCorpac.setAuthorizedContactName(authorized_contact_name_c);</code>	Nom d'une personne

Requis*	Valeur	Limites	Méthode Set	Description
	personne-ressource autorisée	alphanumériques		ou d'une société qui agit à titre de personne-ressource pour les achats autorisés par l'entreprise
N	Numéro de téléphone de la personne-ressource autorisée	17 caractères alphanumériques	mcCorpac.setAuthorizedContactPhone(authorized_contact_phone);	Numéro de téléphone d'une personne ou d'une société avec laquelle il faut communiquer pour les achats autorisés par l'entreprise
N	Données supplémentaires de l'accepteur de carte	40 caractères alphanumériques	mcCorpac.setAdditionalCardAcceptorData(additional_card_acceptor_data);	Renseignements supplémentaires sur l'accepteur de cartes
N	Type d'accepteur de cartes	8 caractères alphanumériques	mcCorpac.setCardAcceptorType(card_acceptor_type);	<p>Différentes classifications des caractéristiques de propriété des entreprises</p> <p>Ce champ prend 8 caractères. Chaque caractère représente un composant différent, soit :</p> <p>Le premier caractère représente le type d'entreprise et contient un code permettant d'identifier la classification ou le type d'entreprise :</p> <ul style="list-style-type: none"> <li>Société par actions</li> <li>Inconnu</li> <li>Individuel ou propriétaire unique</li> <li>Partenariat</li> <li>Association, état ou</li> </ul>

Requ is*	Valeur	Limites	Méthode Set	Description
				<p>fiducie</p> <p>Organisations exonérées d'impôts (501C)</p> <p>Organisation internationale</p> <p>Société à responsabilité limitée (SARL)</p> <p>Agence gouvernementale</p> <p>Le deuxième caractère représente le type de propriétaire d'entreprise. Il contient un code permettant d'identifier les caractéristiques propres au propriétaire de l'entreprise.</p> <p>1 = Aucune classification d'application</p> <p>2 = Propriétaire d'entreprise femme</p> <p>3 = Propriétaire d'entreprise femme avec un handicap physique</p> <p>4 = Propriétaire d'entreprise homme avec un handicap physique</p> <p>0 = Inconnu</p> <p>Le troisième caractère représente le type de certification d'entreprise. Il contient un code permettant d'identifier les caractéristiques relatives au type de certification de l'entreprise, par exemple une certification de petite entreprise, d'entreprise défavorisée ou autre type de certification :</p> <p>1 = Non certifiée</p> <p>2 = Certification de petite entreprise par le Small</p>

Requ is*	Valeur	Limites	Méthode Set	Description
				<p>Business Administration (SBA)</p> <p>3 = Certification SBA de petite entreprise défavorisée</p> <p>4 = Autre certification reconnue par un gouvernement ou une agence (comme le Minority Supplier Development Council)</p> <p>5 = Petite entreprise auto-certifiée</p> <p>6 = Certification de la SBA en tant que petite entreprise et autre certification reconnue par le gouvernement ou une agence</p> <p>7 = Certification de la SBA en tant que petite entreprise défavorisée et autre certification reconnue par le gouvernement ou une agence</p> <p>8 = Autre certification reconnue par un gouvernement ou une agence et certification en tant que petite entreprise auto-certifiée</p> <p>A = Certification de la SBA comme 8(a)</p> <p>B = Petite entreprise défavorisée auto-certifiée (SDB)</p> <p>C = Certification de la SBA comme HUBZone</p> <p>O = Inconnu</p> <p>Le quatrième caractère représente le type racial ou ethnique de l'entreprise. Il contient un code identifiant la race ou l'ethnicité du propriétaire majoritaire de</p>

Requ is*	Valeur	Limites	Méthode Set	Description
				<p>l'entreprise.</p> <p>1 = Afro-américain</p> <p>2 = Américain d'origine asiatique et pacifique</p> <p>3 = Américain d'origine asiatique subcontinentale</p> <p>4 = Américain d'origine hispanique</p> <p>5 = Autochtone de l'Amérique du Nord</p> <p>6 = Autochtone hawaïen</p> <p>7 = Autochtone d'Alaska</p> <p>8 = Caucasiens</p> <p>9 = Autre</p> <p>0 = Inconnu</p> <p>Le cinquième caractère indique si le code du type d'entreprise a été fourni.</p> <p>Y = Le type d'entreprise est fourni</p> <p>N = Le type d'entreprise n'a pas été fourni</p> <p>R = L'accepteur de la carte a refusé de fournir le type d'entreprise</p> <p>Le sixième caractère indique si le code du type de propriétaire de l'entreprise a été fourni.</p> <p>Y = Le type de propriétaire de l'entreprise est fourni</p> <p>N = Le type de propriétaires d'entreprise n'a pas été fourni</p> <p>R = L'accepteur de la carte a refusé de fournir le type d'entreprise</p> <p>Le septième caractère indique si le code du type de</p>

Requis*	Valeur	Limites	Méthode Set	Description
				<p>certification d'entreprise a été fourni.</p> <p>Y = Le type de certification de l'entreprise est fourni</p> <p>N = Le type de certification de l'entreprise n'a pas été fourni</p> <p>R = L'accepteur de la carte a refusé de fournir le type d'entreprise</p> <p>Le huitième caractère indique si le type racial ou ethnique de l'entreprise a été fourni.</p> <p>Y = Le type racial ou ethnique de l'entreprise est fourni</p> <p>N = Le type racial ou ethnique de l'entreprise n'a pas été fourni</p> <p>R = L'accepteur de la carte a refusé de fournir le type racial ou ethnique de l'entreprise</p>
N	Numéro de taxe de l'accepteur de carte	20 caractères alphanumériques	mcCorpac.setCardAcceptorTaxTd(card_acceptor_tax_id_c);	Numéro de taxe fédérale des États-Unis ou numéro de TVA
N	Numéro de référence de l'accepteur de carte	25 caractères alphanumériques	mcCorpac.setCardAcceptorReferenceNumber(card_acceptor_reference_number);	Code qui facilite la communication et la tenue des registres de l'accepteur de cartes ou de l'entreprise
N	Numéro de TVA de l'accepteur de carte	20 caractères alphanumériques	mcCorpac.setCardAcceptorVatNumber(card_acceptor_vat_number_c);	<p>Numéro de taxe sur la valeur ajoutée (TVA) pour l'emplacement de l'accepteur de carte</p> <p>Utilisé pour identifier l'accepteur de la carte</p>

Requis*	Valeur	Limites	Méthode Set	Description
				lors de la collecte et de la déclaration des taxes
C	Taxes	Jusqu'à 6 tableaux	mcCorpac.setTax(tax_c);	Jusqu'à 6 tableaux peuvent contenir des détails de taxes différents
				<b>REMARQUE :</b> Si vous utilisez cette variable, vous devez remplir tous les champs du tableau sur les taxes qui figure ci-dessous.

### 5.3.8.2 Transaction MC Corpal – Détails de ligne d'article

#### Transaction MC Corpal – Détails de ligne d'article

```
mcCorpal.setMcCorpal(customer_code1_l[0], line_item_date_l[0],
ship_date_l[0], order_date1_l[0],
medical_services_ship_to_health_industry_number_l[0],
contract_number_l[0], medical_services_adjustment_l[0],
medical_services_product_number_qualifier_l[0], product_code1_l[0],
item_description_l[0], item_quantity_l[0], unit_cost_l[0],
item_unit_measure_l[0], ext_item_amount_l[0], discount_amount_l[0],
commodity_code_l[0], type_of_supply_l[0], vat_ref_num_l[0], tax_l[0]);
```

Tableau 1 – Détails de ligne d'article – Champs de demande de niveau 3 – MC Corpal

Requis*	Valeur	Limites	Variable	Description
N	Code de client	25 caractères alphanumériques	customer_code1_l	Un numéro de contrôle, tel qu'un numéro de bon de commande, un numéro de projet, un numéro de répartition de service ou un nom donné au commerçant par le fournisseur.
N	Date de la ligne d'article (line item date)	6 caractère numérique Format AAMMJJ	line_item_date_l	Date d'achat de l'article mentionnée dans les détails de la ligne d'article de

Requis *	Valeur	Limites	Variable	Description
				la carte d'entreprise  Numéro à longueur fixe de 6 chiffres, au format AAMMJJ
N	Date d'expédition	6 caractère numérique  Format AAMMJJ	ship_date_I	Date à laquelle la marchandise a été expédiée à sa destination  Numéro à longueur fixe de 6 chiffres, au format AAMMJJ
N	Date de la commande	6 caractère numérique  Format AAMMJJ	order_date1_I	Date d'achat de l'article  Numéro à longueur fixe de 6 chiffres, au format AAMMJJ
Y	Code du produit	12 caractères alphanumériques	product_code1_I	Code de produit pour la ligne de l'article  Indique le code de produit (non lié au carburant) de l'article individuel acheté
Y	Description de l'article	35 caractères alphanumériques	item_description_I	Description de la ligne d'article  Décrit l'article individuel acheté
Y	Nombre d'article	12 caractères	item_quantity_I	Quantité d'article

Requis *	Valeur	Limites	Variable	Description
		alphanumériques		<p>acheté</p> <p>Jusqu'à 5 décimales sont supportées</p> <p>Valeur minimale de 0,0 et maximale de 9 999 999,99999</p>
Y	Coût unitaire	12 caractères décimaux	unit_cost_l	<p>Indique le coût unitaire de chaque article</p> <p>Doit contenir un minimum de 2 décimales (maximum de 5 décimales)</p> <p>Valeur minimale de 0,00001 et maximale de 999 999,99999</p>
Y	Unité de mesure de l'article	12 caractères alphanumériques	item_unit_measure_l	<p>Code de l'unité de mesure de la ligne d'article</p> <p>Unités de mesure et codes permis par la norme ANSI X-12 EDI</p>
Y	Montant prolongé de l'article	9 caractères décimaux	ext_item_amount_l	<p>Indique la valeur de l'article individuel qui est normalement calculée en multipliant le prix par la quantité</p> <p>Doit comporter 2 décimales</p> <p>Valeur minimale de 0,00 et maximale</p>

Requis *	Valeur	Limites	Variable	Description
				de 999 999,99
N	Montant du rabais	9 caractères décimaux	discount_amount_I	Indique le montant du rabais de l'article Doit comporter 2 décimales Valeur minimale de 0,00 et maximale de 999 999,99
N	Code de marchandise	15 caractères alphanumériques	commodity_code_I	Code assigné par le commerçant qui catégorise le mieux les articles achetés
C	Taxes	Jusqu'à 6 tableaux	tax_I	Peut avoir jusqu'à 6 tableaux contenant des détails différents sur les taxes, voir le tableau portant sur les champs de demande de tableaux contenant des détails de taxes pour la description de chaque champ
				<b>REMARQUE :</b> Si vous utilisez cette variable, vous devez remplir tous les champs du tableau sur les taxes qui figure ci-dessous.

### 5.3.8.3 Objet Tax Array – MC Corpais

L'objet tax array est utilisé lorsque vous remplissez le champ Tax de MC Corpac et MC Corpal. Si vous utilisez l'objet tax array, tous les champs du tableau doivent être envoyés.

La configuration du tableau de taxes diffère légèrement entre les deux objets.

## Configuration du tableau de taxes pour la transaction MC Corpac

```
//Tax Details
String[] tax_amount_c = { "1.19", "1.29"};
String[] tax_rate_c = { "6.0", "7.0"};
String[] tax_type_c = { "GST", "PST"};
String[] tax_id_c = { "gst1298", "pst1298"};
String[] tax_included_in_sales_c = { "Y", "N"};
McTax tax_c = new McTax();
tax_c.setTax(tax_amount_c[0], tax_rate_c[0], tax_type_c[0], tax_id_c[0],
tax_included_in_sales_c[0]);
```

## Configuration du tableau de taxes pour la transaction MC Corpal

```
//Tax Details for Items
String[] tax_amount_l = {"0.52", "1.48"};
String[] tax_rate_l = {"13.0", "13.0"};
String[] tax_type_l = {"HST", "HST"};
String[] tax_id_l = {"hst1298", "hst1298"};
String[] tax_included_in_sales_l = {"Y", "Y"};
McTax[] tax_l = new McTax[2];
tax_l[1].setTax(tax_amount_l[1], tax_rate_l[1], tax_type_l[1], tax_id_l[1],
tax_included_in_sales_l[1]);
```

**Tableau 1 Champs de demande du tableau de taxe de la transaction MC Corpais**

Requis *	Valeur	Limites	Variable	Description
Y	Montant des taxes	12 caractères décimaux	tax_amount_c/tax_amount_l	Indique le montant des taxes pour l'achat de biens ou de services  Doit comporter 2 décimales Valeur minimale de 0,00 et maximale de 999 999,99
Y	Taux de taxe	5 caractères décimaux	tax_rate_c/tax_rate_l	Indique le taux de taxe détaillé qui est appliqué en fonction de la taxe

Requis *	Valeur	Limites	Variable	Description
				<b>EXEMPLE :</b> Une TPV de 5 % devrait être « 5,0 », alors qu'une TVP de ou 9,975 % devrait être « 9,975 »
				Peut contenir jusqu'à 3 décimales, avec une valeur minimale de 0,001 ainsi qu'une valeur maximale de 9 999,9
Y	Type de taxe	4 caractères alphanumériques	tax_type_c/tax_type_l	Indique le type de taxe, par exemple TVP, TVQ, TPS, TVH
Y	Numéro de taxe	20 caractères alphanumériques	tax_id_c/tax_id_l	Fournit un numéro d'identification utilisé par l'accepteur de carte avec l'autorité fiscale selon un montant de taxe précis, tel qu'un numéro de TVP ou de TVH.
Y	Taxe incluse dans l'indicateur de vente	1 caractère alphanumérique	tax_included_in_sales_c/tax_included_in_sales_l	<p>Il s'agit de l'indicateur utilisé pour la saisie et la déclaration de taxes supplémentaires.</p> <p>Les valeurs valides sont :</p> <p>Y = Taxe incluse dans le montant total de l'achat</p> <p>N = Taxe non incluse dans le montant total de l'achat</p>

### 5.3.8.4 Exemple de code pour les transactions MC Corpais

#### Exemple de transaction MC Corpais – Données communes de carte d'entreprise avec les détails de la ligne d'article

```

package Level23;
import JavaAPI.*;
public class TestMcCorpaisCommonLineItem
{
    public static void main(String[] args)
    {
        String store_id = "moneris";
        String api_token = "hurgle";
        String processing_country_code = "CA";
        boolean status_check = false;

        String order_id="Test1485206444761";
        String txn_number="39777-1_11";
        String customer_code1_c ="CustomerCode123";
        String card_acceptor_tax_id_c ="UrTaxId";//Merchant tax id which is mandatory
        String corporation_vat_number_c ="cvn123";
        String freight_amount_c ="1.23";
        String duty_amount_c ="2.34";
        String ship_to_pos_code_c ="M1R 1W5";
        String order_date_c ="141211";
        String customer_vat_number_c ="customervn231";
        String unique_invoice_number_c ="uin567";
        String authorized_contact_name_c ="John Walker";
        //Tax Details
        String[] tax_amount_c = { "1.19", "1.29"};
        String[] tax_rate_c = { "6.0", "7.0"};
        String[] tax_type_c = { "GST", "PST"};
        String[] tax_id_c = { "gst1298", "pst1298"};
        String[] tax_included_in_sales_c = { "Y", "N"};
        //Item Details
        String[] customer_code1_l = {"customer code", "customer code2"};
        String[] line_item_date_l = {"150114", "150114"};
        String[] ship_date_l = {"150120", "150122"};
        String[] order_date1_l = {"150114", "150114"};
        String[] medical_services_ship_to_health_industry_number_l = {"", ""};
        String[] contract_number_l = {"", ""};
        String[] medical_services_adjustment_l = {"", ""};
        String[] medical_services_product_number_qualifier_l = {"", ""};
        String[] product_code1_l = {"pc11", "pc12"};
        String[] item_description_l = {"Good item", "Better item"};
        String[] item_quantity_l = {"4", "5"};
        String[] unit_cost_l = {"1.25", "10.00"};
        String[] item_unit_measure_l = {"EA", "EA"};
        String[] ext_item_amount_l = {"5.00", "50.00"};
        String[] discount_amount_l = {"1.00", "50.00"};
        String[] commodity_code_l = {"cCode11", "cCode12"};
        String[] type_of_supply_l = {"", ""};
        String[] vat_ref_num_l = {"", ""};
        //Tax Details for Items
        String[] tax_amount_l = {"0.52", "1.48"};
        String[] tax_rate_l = {"13.0", "13.0"};
        String[] tax_type_l = {"HST", "HST"};
        String[] tax_id_l = {"hst1298", "hst1298"};
        String[] tax_included_in_sales_l = {"Y", "Y"};
        //Create and set Tax for McCorpac
        McTax tax_c = new McTax();
        tax_c.setTax(tax_amount_c[0], tax_rate_c[0], tax_type_c[0], tax_id_c[0],
        tax_included_in_sales_c[0]);
        tax_c.setTax(tax_amount_c[1], tax_rate_c[1], tax_type_c[1], tax_id_c[1],
        tax_included_in_sales_c[1]);
        //Create and set McCorpac for common data - only set values that you know
        McCorpac mcCorpac = new McCorpac();
        mcCorpac.setCustomerCode1(customer_code1_c);
        mcCorpac.setCardAcceptorTaxTd(card_acceptor_tax_id_c);
        mcCorpac.setCorporationVatNumber(corporation_vat_number_c);
        mcCorpac.setFreightAmount1(freight_amount_c);
        mcCorpac.setDutyAmount1(duty_amount_c);
        mcCorpac.setShipToPosCode(ship_to_pos_code_c);
        mcCorpac.setOrderDate(order_date_c);
        mcCorpac.setCustomerVatNumber(customer_vat_number_c);
        mcCorpac.setUniqueInvoiceNumber(unique_invoice_number_c);
    }
}

```

```

mcCorpac.setAuthorizedContactName(authorized_contact_name_c);
mcCorpac.setTax(tax_c);
//Create and set Tax for McCorpal
McTax[] tax_1 = new McTax[2];
tax_1[0] = new McTax();
tax_1[0].setTax(tax_amount_1[0], tax_rate_1[0], tax_type_1[0], tax_id_1[0],
tax_included_in_sales_1[0]);
tax_1[1] = new McTax();
tax_1[1].setTax(tax_amount_1[1], tax_rate_1[1], tax_type_1[1], tax_id_1[1],
tax_included_in_sales_1[1]);
//Create and set McCorpal for each item
McCorpal mcCorpal = new McCorpal();
mcCorpal.setMcCorpal(customer_code1_l[0], line_item_date_1[0], ship_date_1[0],
order_date1_l[0], medical_services_ship_to_health_industry_number_1[0], contract_number_1[0],
medical_services_adjustment_1[0], medical_services_product_number_qualifier_1[0],
product_code1_l[0], item_description_1[0], item_quantity_1[0],
unit_cost_1[0], item_unit_measure_1[0], ext_item_amount_1[0], discount_amount_1[0],
commodity_code_1[0], type_of_supply_1[0], vat_ref_num_1[0], tax_1[0]);
mcCorpal.setMcCorpal(customer_code1_l[1], line_item_date_1[1], ship_date_1[1],
order_date1_l[1], medical_services_ship_to_health_industry_number_1[1], contract_number_1[1],
medical_services_adjustment_1[1], medical_services_product_number_qualifier_1[1],
product_code1_l[1], item_description_1[1], item_quantity_1[1],
unit_cost_1[1], item_unit_measure_1[1], ext_item_amount_1[1], discount_amount_1[1],
commodity_code_1[1], type_of_supply_1[1], vat_ref_num_1[1], tax_1[1]);
McCorpais mcCorpais = new McCorpais();
mcCorpais.setOrderId(order_id);
mcCorpais.setTxnNumber(txn_number);
mcCorpais.setMcCorpac(mcCorpac);
mcCorpais.setMcCorpal(mcCorpal);
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false for production transactions
mpgReq.setStoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(mcCorpais);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = " + receipt.getBankTotals());
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("CavvResultCode = " + receipt.getCavvResultCode());
}
catch (Exception e)
{
System.out.println(e);
}
}
}
}

```

## 5.4 Transactions American Express de niveaux 2 et 3

- 1 Transactions American Express de niveaux 2 et 3
- 1 Transactions L23 Air et Rail American Express

### 5.4.1 Types de transaction de niveaux 2 et 3 par carte Amex

Cet ensemble de transactions comprend une série de transactions financières par carte d'entreprise ainsi qu'une transaction qui permet de transmettre des données de niveaux 2 et 3. Veuillez vous assurer les données de niveaux 2 et 3 sont prises en charge pour les cartes American Express dans votre compte de commerçant. Les transactions de fermeture de lot, d'ouverture des totaux et de préautorisation sont identiques aux transactions indiquées dans la section Ensemble de transactions de base (page 16).

- Lorsque la réponse à la transaction de préautorisation contient une valeur CorporateCard égale à true, vous pouvez soumettre les transactions par carte Amex.
- Si la valeur CorporateCard est false, la carte ne prend pas en charge les données de niveau 2 et 3, il faut donc utiliser des transactions autres que celles de niveau 2 et 3. Si la carte n'est pas une carte d'entreprise, veuillez consulter la section 2. Ensemble de transactions de base pour connaître les transactions appropriées pour les cartes autres que des cartes d'entreprise.

**REMARQUE :** Cette série de transactions est destinée aux transactions où la valeur Corporate Card est true pour lesquelles des données de niveaux 2 et 3 sont soumises. S'il s'avère que la carte de crédit est une carte d'entreprise, mais que vous ne souhaitez pas envoyer de données de niveaux 2 et 3, vous pouvez soumettre des transactions par carte Amex en utilisant l'ensemble de transactions de base de la section Ensemble de transactions de base (page 16).

#### Préautorisation (autorisation)

Une transaction de préautorisation vérifie que les fonds sont présents sur la carte de crédit du client et les bloque. Ces fonds sont bloqués pour une durée prédéterminée, en fonction de l'émetteur de carte. Pour récupérer les fonds bloqués par une transaction de préautorisation afin de les déposer dans le compte du commerçant, une transaction de conclusion doit être effectuée. La valeur CorporateCard sera true si la carte prend en charge les données de niveaux 2 et 3.

**Conclusion par carte Amex – (Conclusion de préautorisation)**

Une fois la préautorisation obtenue, les fonds bloqués doivent être récupérés sur la carte de crédit du client. Cette transaction récupère les fonds bloqués et les prépare à être déposés dans le compte du commerçant. Avant de réaliser une transaction de conclusion par carte Amex, une préautorisation doit être effectuée.

**Transaction forcée par carte Amex – (Transaction forcée de préautorisation)**

Cette transaction est une autre façon d'obtenir les fonds bloqués lors d'une transaction de préautorisation traitée par RVI ou par un terminal équivalent. Cette transaction récupère les fonds bloqués et les prépare à être déposés dans le compte du commerçant.

**Correction d'achat par carte Amex – (Annulation, correction)**

Les transactions de conclusion par carte Amex et les transaction forcée par carte Amex peuvent être annulés le jour même\* où elles se produisent. Une annulation doit annuler le montant total de la transaction et supprimera toute trace de celle-ci sur le relevé du titulaire de la carte. \* Une correction d'achat par carte Amex peut être effectuée pour annuler une transaction tant que le lot qui contient la transaction originale est ouvert. Lorsque la fonction de fermeture automatique est utilisée, le lot est fermé tous les jours entre 22 h et 23 h HNE.

**Remboursement par carte Amex – (Crédit)**

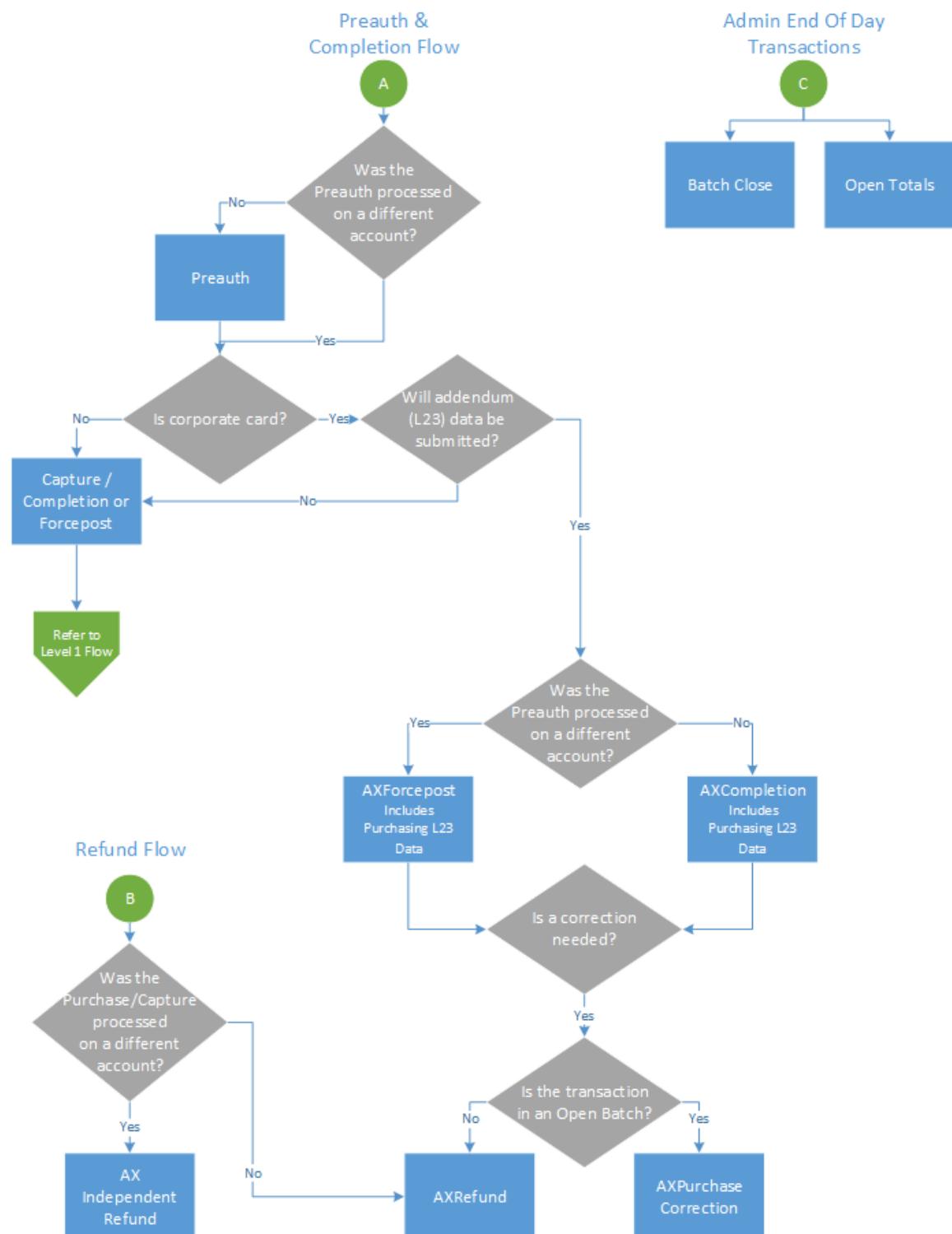
Un remboursement par carte Amex peut être effectué pour rembourser une partie ou la totalité du montant d'une une transaction de conclusion ou d'une transaction forcée par carte Amex.

**Remboursement indépendant par carte Amex – (Crédit)**

Un remboursement indépendant par carte Amex peut être effectué pour rembourser une partie ou la totalité du montant d'une transaction d'achat ou de conclusion. Un remboursement indépendant est utilisé lorsque la transaction d'origine n'a pas été effectuée par l'entremise de Passerelle Moneris.

Veuillez noter que votre compte peut se prévaloir ou non de la fonctionnalité de transaction de remboursement indépendant. Si vous recevez un message d'erreur indiquant que la transaction n'est pas autorisée lorsque vous tentez un remboursement indépendant, cela peut signifier que la transaction n'est pas prise en charge par votre compte. Si vous souhaitez activer (ou réactiver) temporairement les transactions de remboursement indépendant par carte Amex, veuillez communiquer avec le centre de services en composant le 1 866 319-7450.

## 5.4.2 Flux de transaction de niveaux 2 et 3 par carte Amex



## 5.4.3 Objets de données de niveaux 2 et 3 pour les transactions par carte Amex

- 6.4.3.1 Au sujet des objets de données de niveaux 2 et 3 pour les transactions par carte Amex
- 6.4.3.2 Définition de l'objet AxLevel23

- Objet Table1
- Objet Table2
- Objet Table3

#### 5.4.3.1 Au sujet des objets de données de niveaux 2 et 3 pour les transactions par carte Amex

De nombreuses demandes de transaction de niveaux 2 et 3 par carte American Express comprennent également un objet de données obligatoire nommé AxLevel23. L'objet AxLevel23 est également composé d'autres objets, qui sont aussi décrits dans cette section.

Les objets de données de niveaux 2 et 3 de cette section s'appliquent à toutes les transactions et font partie des demandes de transaction suivantes :

- Conclusion par carte Amex
- Transaction forcée par carte Amex
- Remboursement par carte Amex
- Remboursement indépendant par carte Amex

**Éléments dont il faut tenir compte :**

Veuillez vous assurer que les données de l'addenda ci-dessous sont complètes et exactes.  
Veuillez vous assurer que les calculs des quantités, des montants, des rabais, des taxes, etc. correspondent au montant total de la transaction. Des montants incorrects entraîneront le rejet de la transaction.

#### 5.4.3.2 Définition de l'objet AxLevel23

##### Définition de l'objet AxLevel23

```
AxLevel23 level23 = new AxLevel23();
```

L'objet AXLevel23 lui-même comporte trois objets, Table1, Table2 et Table3, qui sont tous obligatoires.

**Tableau 1 – Objet AxLevel23**

Variable	Type et limites	Description	Méthode Set
Table1	Objet	Veuillez vous référer à la section ci-dessous pour obtenir une description et une définition plus détaillées de l'objet table1.	<pre>AxTable1 table1 = new AxTable1(); level23.setTable1(table1);</pre>
Table2	Objet	Veuillez vous référer à la section ci-dessous pour obtenir une description et une définition plus détaillées de l'objet	<pre>AxTable2 table2 = new AxTable2(); level23.setTable2(table2);</pre>

		table2.	
Table3	Objet	Veuillez vous référer à la section ci-dessous pour obtenir une description et une définition plus détaillées de l'objet table3.	<pre>AxTable3 table3 = new AxTable3(); level23.setTable3(table3);</pre>

### Objet Table1

L'objet Table1 contient les renseignements de l'en-tête des données de l'addenda. Il contient des renseignements tels que les éléments d'identification qui identifient de manière unique une facture (transaction), le nom du client et l'adresse de livraison.

#### Définition de l'objet Table1

```
AxTable1 table1 = new AxTable1();
```

#### Objet AxLevel23 Table1 – Champs de l'objet Table1

Requis*	Valeur	Limites	Méthode Set	Description
C	Purchase Order Number	22-character alphanumeric	table1.setBig04(big04);	<p>Numéro du bon de commande fourni par le titulaire de la carte, qui est saisi par le commerçant au point de vente</p> <p>Cette entrée est utilisée dans le processus de déclaration et de production de rapports et elle peut inclure des renseignements comptables propres au client.</p> <p><b>REMARQUE :</b> Cet élément est obligatoire si le client du commerçant fournit un numéro de bon de commande.</p>
N	Release Number	30-character alphanumeric	table1.setBig05(big05);	Numéro qui identifie la libération d'un bon de commande qui a déjà été passé par les parties concernées par la transaction

N	Invoice Number	8-character alphanumeric	table1.setBig10(big10);	Inclut le numéro de facture ou de référence Amex
N	N1Loop	Object	table1.setN1Loop(n1Loop)	Veuillez vous référer à la section ci-dessous pour obtenir une description et une définition plus détaillées de l'objet N1Loop.

\* Y = Obligatoire, N = Facultatif, C = Conditionnel

L'objet Table1 a aussi ses propres objets :

- Objet N1Loop
- Objet AxRef

#### Objet Table1 – Définition de l'objet N1Loop

L'ensemble de données N1Loop indique les noms des demandeurs. Il peut également indiquer, de manière facultative, les détails du groupe d'achat, l'expéditeur, le destinataire et les détails du destinataire.

Au moins une valeur n1Loop doit être définie. Il est possible de définir jusqu'à cinq valeurs n1Loop.

#### Définition de l'objet N1Loop

n1Loop.SetN1Loop(n101, n102, n301, n401, n402, n403, axRef1);

#### Objet AxLevel23 Table1 – Objet Table2 – Champs de l'objet N1Loop

Requis*	Valeur	Limites	Variable ou méthode Set	Description								
Y	Code d'identification d'entité	2 caractères alphanumériques	n101	Valeurs acceptées : R6 = Demandeur (obligatoire) BG = Groupe d'achat (facultatif) SF = Expéditeur (facultatif) ST = Destinataire (facultatif) 40 = Récepteur (facultatif)								
Y	Nom	40 caractères alphanumériques	n102	<table border="1"> <thead> <tr> <th>Code n101</th> <th>Signification n102</th> </tr> </thead> <tbody> <tr> <td>R6</td> <td>Nom du demandeur</td> </tr> <tr> <td>BG</td> <td>Nom du groupe acheteur</td> </tr> <tr> <td>SF</td> <td>Nom de</td> </tr> </tbody> </table>	Code n101	Signification n102	R6	Nom du demandeur	BG	Nom du groupe acheteur	SF	Nom de
Code n101	Signification n102											
R6	Nom du demandeur											
BG	Nom du groupe acheteur											
SF	Nom de											

				<table border="1"> <tr><td></td><td>l'expéditeur</td></tr> <tr><td>ST</td><td>Nom du destinataire</td></tr> <tr><td>En 40</td><td>Nom du récepteur</td></tr> </table>		l'expéditeur	ST	Nom du destinataire	En 40	Nom du récepteur
	l'expéditeur									
ST	Nom du destinataire									
En 40	Nom du récepteur									
N	Adresse	40 caractères alphanumériques	n301	Adresse						
N	Ville	30 caractères alphanumériques	n401	Ville						
N	État ou province	2 caractères alphanumériques	n402	État ou province						
N	Code postal	15 caractères alphanumériques	n403	Code postal						
N	AxRef	Objet	<pre>AxRef axRef1 = new AxRef();</pre>	<p>Veuillez vous référer à la section ci-dessous pour obtenir une description et une définition plus détaillées de l'objet AxRef.</p> <p>Cet objet contient le code postal du client (obligatoire) et le numéro de référence du client (facultatif).</p> <p>Au moins une valeur axRef1 doit être définie; un maximum de deux valeurs axRef1 peuvent être définies.</p>						

\* Y = Obligatoire, N = Facultatif, C = Conditionnel

Tableau 1 – Définition de l'objet AxRef

### Configuration de l'objet AxRef

```
AxRef axRef1 = new AxRef();

String[] ref01 = {"4C", "CR"}; //Reference ID Qualifier

String[] ref02 = {"M5T3A5", "16802309004"}; //Reference ID
```

```

axRef1.setRef(ref01[0], ref02[0]);
axRef1.setRef(ref01[1], ref02[1]);

```

**Objet AxLevel23 Table1 – Objet Table1 – Champs de l'objet AxRef**

<b>Requis*</b>	<b>Valeur</b>	<b>Limites</b>	<b>Variable</b>	<b>Description</b>														
Y	Élément d'identification de la référence	2 caractères alphanumériques	ref01	<p>Cet élément peut contenir les valeurs suivantes pour les occurrences correspondantes de l'objet N1Loop :</p> <table> <thead> <tr> <th><b>Valeur n101</b></th> <th><b>Dénomination</b></th> </tr> </thead> <tbody> <tr> <td>ref01</td> <td></td> </tr> <tr> <td>R6</td> <td>Valeurs acceptées : 4C = Code de destination de la marchandise (obligatoire)</td> </tr> <tr> <td>BG</td> <td>S. O.</td> </tr> <tr> <td>SF</td> <td>S. O.</td> </tr> <tr> <td>ST</td> <td>S. O.</td> </tr> <tr> <td>En 40</td> <td>S. O.</td> </tr> </tbody> </table>	<b>Valeur n101</b>	<b>Dénomination</b>	ref01		R6	Valeurs acceptées : 4C = Code de destination de la marchandise (obligatoire)	BG	S. O.	SF	S. O.	ST	S. O.	En 40	S. O.
<b>Valeur n101</b>	<b>Dénomination</b>																	
ref01																		
R6	Valeurs acceptées : 4C = Code de destination de la marchandise (obligatoire)																	
BG	S. O.																	
SF	S. O.																	
ST	S. O.																	
En 40	S. O.																	
Y	Identification de la référence	15 caractères alphanumériques	ref02	<p>Ce champ doit être rempli pour chaque valeur ref01 fournie.</p> <table> <thead> <tr> <th><b>Valeur ref01</b></th> <th><b>Dénomination ref02</b></th> </tr> </thead> <tbody> <tr> <td>4C (valeur n101 = R6)</td> <td>Cet élément doit contenir le code postal Amex de destination de la marchandise expédiée. Si le code postal de destination n'est pas disponible, le code postal de l'emplacement du commerçant où la transaction a eu lieu peut être utilisé à la place.</td> </tr> <tr> <td>CR (Valeur)</td> <td>Cet élément doit contenir le numéro de</td> </tr> </tbody> </table>	<b>Valeur ref01</b>	<b>Dénomination ref02</b>	4C (valeur n101 = R6)	Cet élément doit contenir le code postal Amex de destination de la marchandise expédiée. Si le code postal de destination n'est pas disponible, le code postal de l'emplacement du commerçant où la transaction a eu lieu peut être utilisé à la place.	CR (Valeur)	Cet élément doit contenir le numéro de								
<b>Valeur ref01</b>	<b>Dénomination ref02</b>																	
4C (valeur n101 = R6)	Cet élément doit contenir le code postal Amex de destination de la marchandise expédiée. Si le code postal de destination n'est pas disponible, le code postal de l'emplacement du commerçant où la transaction a eu lieu peut être utilisé à la place.																	
CR (Valeur)	Cet élément doit contenir le numéro de																	

				n101 = R6) :	référence du membre de la carte Amex (p. ex. bon de commande, centre de coûts, numéro de projet, etc.) qui correspond à cette transaction, s'il est fourni par le titulaire de la carte. Ces renseignements peuvent être affichés dans le processus de déclaration ou de production de rapports et peuvent inclure des renseignements comptables propres au client.
--	--	--	--	--------------	---

\* Y = Obligatoire, N = Facultatif, C = Conditionnel

### Objet Table2

L'objet Table2 comprend les détails de l'addenda de la transaction. Il contient les données de transaction, notamment les codes de référence, les montants des débits ou des crédits et des taxes, les détails de la ligne d'article, les renseignements sur l'expédition, et bien plus encore. Toutes les données transactionnelles d'une facture sont liées à une seule transaction et à un seul numéro de compte de titulaire de carte.

#### Définition de l'objet Table2

```
AxTable2 table2 = new AxTable2();
```

#### Objet AxLevel23 Table1 – Champs de l'objet Table2

Requis*	Valeur	Limites	Méthode Set	Description
N	It1loop	Objet	table2.setIt1Loop(it1Loop);	Veuillez vous référer à la section ci-dessous pour obtenir une description et une définition des détails de l'objet.

\* Y = Obligatoire, N = Facultatif, C = Conditionnel

#### Objet Table2 – Définition de l'objet AxIt1Loop

Les données AxIt1Loop définissent les données de base des articles pour la facture. Ces données sont définies pour chaque article et service acheté et sont incluses dans cette facture. Cet ensemble de

données contient les données de base de la transaction, notamment la quantité, l'unité de mesure, le prix unitaire et les renseignements de référence sur les biens et services.

- Au moins une valeur it1Loop est requise.
- Un maximum de 999 valeurs it1Loop sont acceptées.

#### Définition de l'objet AxIt1Loop

```
AxIt1Loop it1Loop = new AxIt1Loop();

it1Loop.setIt1Loop(it102[0], it103[0], it104[0], it105[0], it106s[0], txi[0],
pam05[0], pid05[0]);

it1Loop.setIt1Loop(it102[1], it103[1], it104[1], it105[1], it106s[1], txi[1],
pam05[1], pid05[1]);
```

**Objet AxLevel23 Table1 – Objet Table2 – Champs de l'objet AxIt1Loop**

Requis*	Valeur	Limites	Variable	Description
Y	Quantité facturée pour la ligne d'article	10 caractères décimaux	it102	<p>Quantité d'article acheté</p> <p>Jusqu'à 2 décimales sont supportées</p> <p>Valeur minimale de 0,0 et maximale de 9 999 999 999</p>
Y	Code de l'unité ou de la base de mesure	2 caractères alphanumériques	it103	<p>Code de l'unité de mesure de la ligne d'article</p> <p>Doit contenir un code qui indique l'unité de mesure de la valeur ou la manière dont une mesure est prise</p> <p><b>EXEMPLE :</b> EA = chaque, E5 = pouces</p> <p>Consultez le site <a href="#">ANSI X-12 EDI Allowable Units of Measure and Codes</a> pour</p>

				connaître la liste des codes.
Y	Prix unitaire	15 caractères décimaux	it104	<p>Coût unitaire de chaque article</p> <p>Doit comporter 2 décimales</p> <p>Valeur minimale de 0,00 et maximale de 999 999,99</p>
N	Code tarifaire de la base ou de l'unité	2 caractères alphanumériques	it105	<p>Code identifiant le type de prix unitaire d'un article</p> <p><b>EXEMPLE :</b> DR = vendeur (dealer), AP = prix conseillé (advise price)</p>
				Consultez le site <a href="#">ASC X12 004010 Element 639</a> pour connaître la liste des codes.
N	AxIt106s	Objet	it106s	Veuillez vous référer à la section ci-dessous pour obtenir une description et une définition des détails de l'objet.
N	AxTxi	Objet	txi	Veuillez vous référer à la section ci-dessous pour obtenir une description et une définition des détails de l'objet. Un maximum de 12 valeurs AxTxi

				(ensembles de données sur les renseignements sur les taxes) peuvent être définies.
				<b>REMARQUE :</b> Si les renseignements sur les taxes au niveau des lignes d'article sont remplis au format AxTxi dans l'objet Table2, les totaux des taxes pour l'ensemble de la facture (transaction) doivent être saisis dans l'objet Table3.
Y	Montant final de l'article	10 caractères décimaux	pam05	<p>Indique la valeur de l'article individuel qui est normalement calculée en multipliant le prix par la quantité</p> <p>Doit comporter 2 décimales</p> <p>Valeur minimale de 0,00 et maximale de 99 999,99</p>
Y	Description de la ligne d'article	80 caractères alphanumériques	pid05	<p>Description de la ligne d'article</p> <p>Décrit l'article individuel acheté</p> <p>Ce champ concerne chaque ligne de la transaction.</p>

\* Y = Obligatoire, N = Facultatif, C = Conditionnel

**Objet Table2 – Définition de l'objet AxIt106s**

```
AxIt106s[] it106s = {new AxIt106s(), new AxIt106s(), new AxIt106s(), new AxIt106s(), new AxIt106s()};

string[] it10618 = {"MG", "MG", "MG", "MG", "MG"}; //Product/Service ID
qualifier

string[] it10719 = {"DJFR4", "JFJ49", "FEF33", "FEE43", "DISCOUNT"};
//Product/Service ID (corresponds to it10618)
```

**Objet AxLevel23 Table1 – Objet Table2 – Champs de l'objet AxIt106s**

Requis *	Valeur	Limites	Méthode Set	Description
N	Élément d'identification du produit ou du service	2 caractères alphanumériques	it106s[0].setIt10618(it10618[0]);  it106s[1].setIt10618(it10618[1]);	Valeurs acceptées :  MG = Numéro de pièce du fabricant  VC = Numéro de catalogue du fournisseur  SK = Numéro de référence du fournisseur  UP = Code universel du produit  VP = Numéro de pièce du fournisseur  PO = Numéro du bon de commande  AN = Code du bien défini par le client
N	Numéro de produit ou de service	<b>it1061</b> Taille ou 8    type <b>it10719</b> VC              20 caractères alphanumériques	it106s[0].setIt10719(it10719[0]);  it106s[1].setIt10719(it10719[1]);	Numéro du produit ou du service cqui orrespond au qualificate

		PO	22 caractères alphanumériques		ur précédent défini par it10618
		Autre	30 caractères alphanumériques		La longueur maximale dépend du qualificateur défini dans la variable it10618.

\* Y = Obligatoire, N = Facultatif, C = Conditionnel

#### Objet Table2 – Configuration de l'objet AxTxI

##### Définition de l'objet Table2 AxTxI

```
//Create Table 2 with details

String[] txi01_GST = {"GS", "GS", "GS", "GS", "GS"}; //Tax type code
String[] txi02_GST = {"0.70", "1.75", "1.00", "0.80","0.00"}; //Monetary amount
String[] txi03_GST = {"5.0", "5.0", "5.0", "5.0","5.0"}; //Percent
String[] txi06_GST = {"2", "2", "2", "2","2"}; //Tax exempt code

String[] txi01_PST = {"PG", "PG", "PG","PG","PG"}; //Tax type code
String[] txi02_PST = {"0.80", "2.00", "1.00", "0.80","0.00"}; //Monetary amount
String[] txi03_PST = {"7.0", "7.0", "7.0", "7.0","7.0"}; //Percent
String[] txi06_PST = {"2", "2", "2", "2","2"}; //Tax exempt code

AxTxI[] txi = {new AxTxI(), new AxTxI(), new AxTxI(), new AxTxI(), new AxTxI()};
txi[0].setTxI(txi01_GST[0], txi02_GST[0], txi03_GST[0], txi06_GST[0]);
txi[0].setTxI(txi01_PST[0], txi02_PST[0], txi03_PST[0], txi06_PST[0]);
txi[1].setTxI(txi01_GST[1], txi02_GST[1], txi03_GST[1], txi06_GST[1]);
txi[1].setTxI(txi01_PST[1], txi02_PST[1], txi03_PST[1], txi06_PST[1]);
txi[2].setTxI(txi01_GST[2], txi02_GST[2], txi03_GST[2], txi06_GST[2]);
txi[2].setTxI(txi01_PST[2], txi02_PST[2], txi03_PST[2], txi06_PST[2]);
txi[3].setTxI(txi01_GST[3], txi02_GST[3], txi03_GST[3], txi06_GST[3]);
txi[3].setTxI(txi01_PST[3], txi02_PST[3], txi03_PST[3], txi06_PST[3]);
txi[4].setTxI(txi01_GST[4], txi02_GST[4], txi03_GST[4], txi06_GST[4]);
txi[4].setTxI(txi01_PST[4], txi02_PST[4], txi03_PST[4], txi06_PST[4]);
```

**Objet AxLevel23 Table1 – Objet Table2 – Champs de l'objet AxiTxi**

Requis*	Valeur	Limites	Variable	Description
C	Code du type de taxe	txi01	2 caractères alphanumériques	<p>Code du type de taxe applicable au Canada et aux États-Unis uniquement</p> <p>Pour le Canada, ce champ doit contenir un code qui précise le type de taxe.</p> <p>Si le champ txi01 est utilisé, alors le champ txi02, txi03 ou txi06 doit être rempli</p> <p>Voici certains des codes valides :</p> <ul style="list-style-type: none"> <li>CT = Taxe de comté (facultatif)</li> <li>CA = Taxe municipale (facultatif)</li> <li>EV = Taxe environnementale (facultatif)</li> <li>GS = Taxe sur les biens et services (TPS) (facultatif)</li> <li>LS = Taxe de vente d'État et locale (facultatif)</li> <li>LT = Taxe de vente locale (facultatif)</li> <li>PG = Taxe de vente provinciale (TVP)</li> </ul>

				(facultatif)  SP = Taxe d'État ou provinciale également appelée taxe de vente du Québec (TVQ) (facultatif)  ST = Taxe de vente d'État (facultatif)  TX = Toutes les taxes (obligatoire)  VA = Taxe sur la valeur ajoutée aussi appelée taxe de vente harmonisée du Canada (TVH) (facultatif)
C	Montant en numéraire	txi02	16 caractères décimaux	<p>Cet élément peut contenir le montant de la taxe en numéraire qui correspond au code de type de taxe du champ txi01.</p> <div style="background-color: #e0f2f1; padding: 10px;"> <p><b>REMARQUE :</b> Si le champ txi02 est utilisé dans l'occurrence obligatoire de txi01=TX, txi02 doit contenir le montant total de la taxe applicable à la totalité de la facture (transaction).</p> <p>Si les taxes ne s'appliquent pas à l'ensemble de la facture (transaction), txi02 doit être égal à 0,00.</p> </div> <p>La valeur maximale qui peut être entrée</p>

				<p>dans ce champ est « 9 999,99 », soit 9 999,99 \$ CA.</p> <p>Un débit est entré comme suit : En 9999,99</p> <p>Un crédit est entré comme suit : En -9999,99</p>
C	Pourcentage	txi03	10 caractères décimaux	<p>Indique le pourcentage de taxe (sous forme décimale) qui correspond au code de type de taxe défini dans le champ txi01</p> <p>Jusqu'à 2 décimales sont supportées</p>
C	Code d'exonération fiscale	txi06	1 caractère alphanumérique	<p>Cet élément peut contenir le code d'exonération fiscale qui identifie l'état d'exonération des ventes ainsi que la taxe correspondant au code de type de taxe indiqué dans le champ txi01.</p> <p>Valeurs acceptées :</p> <p>1 = Oui (exonéré d'impôt)</p> <p>2 = Non (non exonéré d'impôt)</p> <p>4 = Non exonéré ou</p>

				pour la revente A = Main d'œuvre imposable, matériel exonéré B = Matériaux taxables, main-d'œuvre exonérée C = Non imposable F = Exonéré (taxe sur les produits et services) G = Exonéré (taxe de vente provinciale) L = Service local exonéré R = Exonération périodique U = Utilisation exonérée
--	--	--	--	--

\* Y = Obligatoire, N = Facultatif, C = Conditionnel

### Objet Table3

L'objet Table3 comprend le sommaire de l'addenda de la transaction. Il contient le montant total de la facture (transaction), la taxe de vente, les frais de transport et de manutention ainsi que des renseignements sur le sommaire de la facture, y compris le nombre total d'articles, le nombre de segments dans la facture et le numéro de contrôle de l'ensemble de la transaction (le numéro de lot).

#### Définition de l'objet Table3

```
AxTable3 table3 = new AxTable3();
```

#### Objet AxLevel23 Table1 – Champs de l'objet Table3

Requis*	Valeur	Limites	Méthode Set	Description
C	AxTxI	Objet	table3.setTxI(taxTbl3);	Veuillez vous référer à la section ci-dessous pour obtenir une description et une définition des détails de

				l'objet.
				<b>REMARQUE :</b> Si les renseignements sur les taxes au niveau des lignes d'article sont remplis au format AxTxi dans l'objet Table2, les totaux des taxes pour l'ensemble de la facture (transaction) doivent être saisis dans l'objet Table3. Un maximum de 10 valeurs AxTxi peuvent être définies dans l'objet Table3.

\* Y = Obligatoire, N = Facultatif, C = Conditionnel

#### Objet Table3 – Configuration de l'objet AxTxi

L'ensemble de données obligatoire sur les taxes doit contenir le montant total de la taxe applicable à l'ensemble de la facture (transaction) qui comprend toutes les lignes d'article identifiées dans l'objet Table2. Si les taxes ne s'appliquent pas à l'ensemble de la facture (transaction), le champ txi02 doit être égal à 0,00.

Les totaux des taxes doivent être entrés dans ce segment obligatoire sur les renseignements fiscaux de l'objet Table3, même si les données sur les taxes au niveau des détails des lignes d'articles sont indiquées dans l'objet Table2.

Au moins une occurrence des champs txi02, txi03 ou txi06 est requise.

#### Définition de l'objet Table3 AxTxi

```
AxTxi taxTbl3 = new AxTxi();

taxTbl3.setTxi("GS", "4.25","",""); //sum of GST taxes
taxTbl3.setTxi("PG", "4.60","",""); //sum of PST taxes
taxTbl3.setTxi("TX", "8.85","",""); //sum of all taxes
```

#### Objet AxLevel23 Table1 – Objet Table3 – Champs de l'objet AxTxi

Requis*	Valeur	Limites	Variable	Description
C	Code du type de taxe	txi01	2 caractères alphanumériques	Code du type de taxe applicable au Canada et aux États-Unis uniquement

			<p>Pour le Canada, ce champ doit contenir un code qui précise le type de taxe.</p> <p>Si le champ txi01 est utilisé, alors le champ txi02, txi03 ou txi06 doit être rempli</p> <p>Voici certains des codes valides :</p> <p>CT = Taxe de comté (facultatif)</p> <p>CA = Taxe municipale (facultatif)</p> <p>EV = Taxe environnementale (facultatif)</p> <p>GS = Taxe sur les biens et services (TPS) (facultatif)</p> <p>LS = Taxe de vente d'État et locale (facultatif)</p> <p>LT = Taxe de vente locale (facultatif)</p> <p>PG = Taxe de vente provinciale (TVP) (facultatif)</p> <p>SP = Taxe d'État ou provinciale également appelée taxe de vente du Québec (TVQ) (facultatif)</p> <p>ST = Taxe de vente d'État (facultatif)</p> <p>TX = Toutes les taxes (obligatoire)</p> <p>VA = Taxe sur la valeur ajoutée aussi appelée taxe de vente harmonisée du Canada (TVH)</p>
--	--	--	---

				(facultatif)
C	Montant en numéraire	txi02	16 caractères décimaux	<p>Cet élément peut contenir le montant de la taxe en numéraire qui correspond au code de type de taxe du champ txi01.</p> <div style="border: 1px solid #ccc; padding: 5px; background-color: #e0f2f1;"> <p><b>REMARQUE :</b> Si le champ txi02 est utilisé dans l'occurrence obligatoire de txi01=TX, txi02 doit contenir le montant total de la taxe applicable à la totalité de la facture (transaction). Si les taxes ne s'appliquent pas à l'ensemble de la facture (transaction), txi02 doit être égal à 0,00.</p> </div> <p>La valeur maximale qui peut être entrée dans ce champ est « 9 999.99 », soit 9 999,99 \$ CA.</p> <p>Un débit est entré comme suit : En 9999,99</p> <p>Un crédit est entré comme suit : En -9999,99</p>
C	Pourcentage	txi03	10 caractères décimaux	Indique le pourcentage de taxe (sous forme

				décimale) qui correspond au code de type de taxe défini dans le champ txi01  Jusqu'à 2 décimales sont supportées
C	Code d'exonération fiscale	txi06	1 caractère alphanumérique	Cet élément peut contenir le code d'exonération fiscale qui identifie l'état d'exonération des ventes ainsi que la taxe correspondant au code de type de taxe indiqué dans le champ txi01.  Valeurs acceptées :  1 = Oui (exonéré d'impôt)  2 = Non (non exonéré d'impôt)  4 = Non exonéré ou pour la revente  A = Main d'œuvre imposable, matériel exonéré  B = Matériaux taxables, main-d'œuvre exonérée  C = Non imposable  F = Exonéré (taxe sur les produits et services)  G = Exonéré (taxe de vente provinciale)  L = Service local

				exonéré  R = Exonération périodique  U = Utilisation exonérée
--	--	--	--	---

\* Y = Obligatoire, N = Facultatif, C = Conditionnel

#### 5.4.4 Conclusion par carte Amex

Une transaction de conclusion par carte Amex est utilisée pour sécuriser les fonds bloqués par une transaction de préautorisation. Lors de l'envoi d'une demande de conclusion, vous avez besoin de deux renseignements provenant de la réponse de la préautorisation originale, soit l'ID de commande et le numéro de transaction.

##### Définition de l'objet de transaction AX Completion

```
AxCompletion axCompletion = new AxCompletion()
```

##### Objet HttpsPostRequest pour les conclusions par carte Amex

```
HttpsPostRequest mpgReq = new HttpsPostRequest();  
  
mpgReq.setTransaction(axCompletion);
```

##### Champs de demande pour les transactions de conclusion par carte Amex (obligatoires)

Variable	Type et limites	Méthode Set
No de commande	<i>Chaîne</i> 50 caractères alphanumériques	axCompletion.setOrderId(order_id);
Montant de la conclusion	<i>Chaîne</i> 10 caractères décimaux Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal	axCompletion.setCompAmount(comp_amount);
	<b>EXEMPLE : 1 234 567,89</b>	
Numéro de transaction	<i>Chaîne</i>	axCompletion.setTxnNumber(txn_number);

Variable	Type et limites	Méthode Set
	255 caractères alphanumériques	
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	<code>axCompletion.setCryptType(crypt);</code>
Données de niveaux 2 et 3	<i>Objet</i> S. O.	<code>axCompletion.setAxLevel23(level23);</code>

#### Exemple de conclusion par carte Amex

```

package Level23;
import JavaAPI.*;

public class TestAxCompletion
{
public static void main(String[] args)
{
String store_id = "moneris";
String api_token = "hurgle";
String processing_country_code = "CA";
boolean status_check = false;

String order_id="ord-210916-12:06:38";
String comp_amount="62.37";
String txn_number = "18924-0_11";
String crypt="7";

//Create Table 1 with details
String n101 = "R6"; //Entity ID Code
String n102 = "Retailing Inc. International"; //Name
String n301 = "919 Oriole Rd."; //Address Line 1
String n401 = "Toronto"; //City
String n402 = "On"; //State or Province
String n403 = "H1T6W3"; //Postal Code
String[] ref01 = {"4C", "CR"}; //Reference ID Qualifier
String[] ref02 = {"M5T3A5", "16802309004"}; //Reference ID
String big04 = "PO7758545"; //Purchase Order Number
String big05 = "RN0049858"; //Release Number
String big10 = "INV99870E"; //Invoice Number
AxRef axRef1 = new AxRef();
axRef1.setRef(ref01[0], ref02[0]);
axRef1.setRef(ref01[1], ref02[1]);
AxN1Loop n1Loop = new AxN1Loop();
n1Loop.setN1Loop(n101, n102, n301, n401, n402, n403, axRef1);
AxTable1 table1 = new AxTable1();
table1.setBig04(big04);
table1.setBig05(big05);
table1.setBig10(big10);
table1.setN1Loop(n1Loop);

//Create Table 2 with details
//the sum of the extended amount field (pam05) must equal the level 1 amount field
String[] it102 = {"1", "1", "1", "1", "1"}; //Line item quantity invoiced
String[] it103 = {"EA", "EA", "EA", "EA", "EA"}; //Line item unit or basis of measurement code
String[] it104 = {"10.00", "25.00", "8.62", "10.00", "-10.00"}; //Line item unit price
String[] it105 = {"", "", "", "", ""}; //Line item basis of unit price code

String[] it10618 = {"MG", "MG", "MG", "MG", "MG"}; //Product/Service ID qualifier
String[] it10719 = {"DJFR4", "JFJ49", "FEF33", "FEE43", "DISCOUNT"}; //Product/Service ID
//corresponds to it10618
}

```

```

String[] txi01_GST = {"GS", "GS", "GS", "GS"}; //Tax type code
String[] txi02_GST = {"0.70", "1.75", "1.00", "0.80","0.00"}; //Monetary amount
String[] txi03_GST = {"", "", "", "", ""}; //Percent
String[] txi06_GST = {"", "", "", "", ""}; //Tax exempt code

String[] txi01_PST = {"PG", "PG", "PG","PG"}; //Tax type code
String[] txi02_PST = {"0.80", "2.00", "1.00", "0.80","0.00"}; //Monetary amount
String[] txi03_PST = {"", "", "", "", ""}; //Percent
String[] txi06_PST = {"", "", "", "", ""}; //Tax exempt code
String[] pam05 = {"11.50", "28.75", "10.62", "11.50", "-10.00"}; //Extended line-item amount
String[] pid05 = {"Stapler", "Lamp", "Bottled Water", "Fountain Pen", "DISCOUNT"}; //Line item
description
AxIt106s[] it106s = {new AxIt106s(), new AxIt106s(), new AxIt106s(), new AxIt106s(), new
AxIt106s()};

it106s[0].setIt10618(it10618[0]);
it106s[0].setIt10719(it10719[0]);

it106s[1].setIt10618(it10618[1]);
it106s[1].setIt10719(it10719[1]);

it106s[2].setIt10618(it10618[2]);
it106s[2].setIt10719(it10719[2]);

it106s[3].setIt10618(it10618[3]);
it106s[3].setIt10719(it10719[3]);

it106s[4].setIt10618(it10618[4]);
it106s[4].setIt10719(it10719[4]);
AxTxi[] txi = {new AxTxi(), new AxTxi(), new AxTxi(), new AxTxi(), new AxTxi()};
txi[0].setTxi(txi01_GST[0], txi02_GST[0], txi03_GST[0], txi06_GST[0]);
txi[0].setTxi(txi01_PST[0], txi02_PST[0], txi03_PST[0], txi06_PST[0]);
txi[1].setTxi(txi01_GST[1], txi02_GST[1], txi03_GST[1], txi06_GST[1]);
txi[1].setTxi(txi01_PST[1], txi02_PST[1], txi03_PST[1], txi06_PST[1]);
txi[2].setTxi(txi01_GST[2], txi02_GST[2], txi03_GST[2], txi06_GST[2]);
txi[2].setTxi(txi01_PST[2], txi02_PST[2], txi03_PST[2], txi06_PST[2]);
txi[3].setTxi(txi01_GST[3], txi02_GST[3], txi03_GST[3], txi06_GST[3]);
txi[3].setTxi(txi01_PST[3], txi02_PST[3], txi03_PST[3], txi06_PST[3]);
txi[4].setTxi(txi01_GST[4], txi02_GST[4], txi03_GST[4], txi06_GST[4]);
txi[4].setTxi(txi01_PST[4], txi02_PST[4], txi03_PST[4], txi06_PST[4]);
AxIt1Loop it1Loop = new AxIt1Loop();
it1Loop.setIt1Loop(it102[0], it103[0], it104[0], it105[0], it106s[0], txi[0], pam05[0],
pid05[0]);
it1Loop.setIt1Loop(it102[1], it103[1], it104[1], it105[1], it106s[1], txi[1], pam05[1],
pid05[1]);
it1Loop.setIt1Loop(it102[2], it103[2], it104[2], it105[2], it106s[2], txi[2], pam05[2],
pid05[2]);
it1Loop.setIt1Loop(it102[3], it103[3], it104[3], it105[3], it106s[3], txi[3], pam05[3],
pid05[3]);
it1Loop.setIt1Loop(it102[4], it103[4], it104[4], it105[4], it106s[4], txi[4], pam05[4],
pid05[4]);
AxTable2 table2 = new AxTable2();
table2.setIt1Loop(it1Loop);
//Create Table 3 with details
AxTxi taxTbl3 = new AxTxi();
taxTbl3.setTxi("GS", "4.25","",""); //sum of GST taxes
taxTbl3.setTxi("PG", "4.60","",""); //sum of PST taxes
taxTbl3.setTxi("TX", "8.85","",""); //sum of all taxes
AxTable3 table3 = new AxTable3();
table3.setTxi(taxTbl3);

//Create and set Level23 Object
AxLevel23 level23 = new AxLevel23();
level23.setTable1(table1);
level23.setTable2(table2);
level23.setTable3(table3);
AxCompletion axCompletion = new AxCompletion();
axCompletion.setOrderId(order_id);
axCompletion.setCompAmount(comp_amount);
axCompletion.setTxnNumber(txn_number);
axCompletion.setCryptType(crypt);
axCompletion.setAxLevel23(level23);
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false for production transactions
mpgReq.setstoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(axCompletion);
mpgReq.setStatusCheck(status_check);
mpgReq.send();

```

```

try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = " + receipt.getBankTotals());
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("CavvResultCode = " + receipt.getCavvResultCode());
}
catch (Exception e)
{
System.out.println(e);
}
}
}

```

## 5.4.5 Transaction forcée par carte Amex

La transaction forcée par carte Amex est utilisée pour sécuriser les fonds bloqués par une transaction de pré-autorisation traitée par RVI ou par un terminal équivalent. Lorsque vous envoyez une demande de transaction forcée par carte Amex, vous avez besoin de l'ID de commande, du montant, du numéro de carte de crédit, de la date d'expiration, du code d'autorisation et de l'indicateur de commerce électronique.

### Définition d'objet de transaction AX Force Post

```
AxForcePost axForcePost = new AxForcePost();
```

### Objet HttpsPostRequest pour les transactions forcées par carte Amex

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
```

```
mpgReq.setTransaction(axForcePost);
```

**Champs de demande pour les transactions forcées par carte Amex (obligatoires)**

Valeur	Limites	Méthode Set
No de commande	<i>Chaîne</i>  50 caractères alphanumériques	<code>axForcePost.setOrderId(order_id);</code>
Montant	<i>Chaîne</i>  10 caractères décimaux  Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal	<code>axForcePost.setAmount(amount);</code>
	<b>EXEMPLE : 1 234 567,89</b>	
Numéro de carte de crédit	<i>Chaîne</i>  20 caractères alphanumériques	<code>axForcePost.setPan(pan);</code>
Date d'expiration	<i>Chaîne</i>  4 caractères alphanumériques (format AAMM)	<code>axForcePost.setExpDate(expiry_date);</code>
Code d'autorisation	<i>Chaîne</i>  8 caractères alphanumériques	<code>axForcePost.setAuthCode(auth_code);</code>
Indicateur de commerce électronique	<i>Chaîne</i>  1 caractère alphanumérique	<code>axForcePost.setCryptType(crypt);</code>
Données de niveaux 2 et 3	<i>Objet</i>  S. O.	<code>axForcePost.setAxLevel123(level23);</code>

## Champs de demande pour les transactions forcées par carte Amex ( facultatifs)

Variable	Type et limites	Méthode Set
ID du client	<i>Chaîne</i> 50 caractères alphanumériques	axForcePost.setCustId(cust_id);

### Exemple de transaction forcée par carte Amex

```

package Level23;
import JavaAPI.*;

public class TestAxForcePost
{
  public static void main(String[] args)
  {
String store_id = "moneris";
String api_token = "hurgle";
String processing_country_code = "CA";
boolean status_check = false;

java.util.Date createDate = new java.util.Date();
String order_id="Test"+createDate.getTime();
String cust_id="CUST13343";
String amount="62.37";
String pan="373269005095005";
String expiry_date="2012"; //YYMM
String auth_code="123456";
String crypt="7";

//Create Table 1 with details
String n101 = "R6"; //Entity ID Code
String n102 = "Retailing Inc. International"; //Name
String n301 = "919 Oriole Rd."; //Address Line 1
String n401 = "Toronto"; //City
String n402 = "On"; //State or Province
String n403 = "H1T6W3"; //Postal Code
String[] ref01 = {"4C", "CR"}; //Reference ID Qualifier
String[] ref02 = {"M5T3A5", "16802309004"}; //Reference ID
String big04 = "PO7758545"; //Purchase Order Number
String big05 = "RN0049858"; //Release Number
String big10 = "INV99870E"; //Invoice Number
AxRef axRef1 = new AxRef();
axRef1.setRef(ref01[0], ref02[0]);
axRef1.setRef(ref01[1], ref02[1]);
AxN1Loop n1Loop = new AxN1Loop();
n1Loop.setN1Loop(n101, n102, n301, n401, n402, n403, axRef1);
AxTable1 table1 = new AxTable1();
table1.setBig04(big04);
table1.setBig05(big05);
table1.setBig10(big10);
table1.setN1Loop(n1Loop);

//Create Table 2 with details
//the sum of the extended amount field (pam05) must equal the level 1 amount field

String[] it102 = {"1", "1", "1", "1", "1"}; //Line item quantity invoiced
String[] it103 = {"EA", "EA", "EA", "EA", "EA"}; //Line item unit or basis of measurement code
String[] it104 = {"10.00", "25.00", "8.62", "10.00", "-10.00"}; //Line item unit price
String[] it105 = {"", "", "", "", ""}; //Line item basis of unit price code

String[] it10618 = {"MG", "MG", "MG", "MG", "MG"}; //Product/Service ID qualifier
String[] it10719 = {"DJFR4", "JFJ49", "FEF33", "FEE43", "DISCOUNT"}; //Product/Service ID
//(corresponds to it10618)

String[] txi01_GST = {"GS", "GS", "GS", "GS", "GS"}; //Tax type code
String[] txi02_GST = {"0.70", "1.75", "1.00", "0.80", "0.00"}; //Monetary amount
String[] txi03_GST = {"", "", "", "", ""}; //Percent

```

```

String[] txi06_GST = {"", "", "", "", ""}; //Tax exempt code

String[] txi01_PST = {"PG", "PG", "PG", "PG"}; //Tax type code
String[] txi02_PST = {"0.80", "2.00", "1.00", "0.80", "0.00"}; //Monetary amount
String[] txi03_PST = {"", "", "", "", ""}; //Percent
String[] txi06_PST = {"", "", "", "", ""}; //Tax exempt code
String[] pam05 = {"11.50", "28.75", "10.62", "11.50", "-10.00"}; //Extended line-item amount
String[] pid05 = {"Stapler", "Lamp", "Bottled Water", "Fountain Pen", "DISCOUNT"}; //Line item
description
AxIt106s[] it106s = {new AxIt106s(), new AxIt106s(), new AxIt106s(), new AxIt106s(), new
AxIt106s()};

it106s[0].setIt10618(it10618[0]);
it106s[0].setIt10719(it10719[0]);

it106s[1].setIt10618(it10618[1]);
it106s[1].setIt10719(it10719[1]);

it106s[2].setIt10618(it10618[2]);
it106s[2].setIt10719(it10719[2]);

it106s[3].setIt10618(it10618[3]);
it106s[3].setIt10719(it10719[3]);

it106s[4].setIt10618(it10618[4]);
it106s[4].setIt10719(it10719[4]);
AxTxi[] txi = {new AxTxi(), new AxTxi(), new AxTxi(), new AxTxi()};
txi[0].setTxi(txi01_GST[0], txi02_GST[0], txi03_GST[0], txi06_GST[0]);
txi[0].setTxi(txi01_PST[0], txi02_PST[0], txi03_PST[0], txi06_PST[0]);
txi[1].setTxi(txi01_GST[1], txi02_GST[1], txi03_GST[1], txi06_GST[1]);
txi[1].setTxi(txi01_PST[1], txi02_PST[1], txi03_PST[1], txi06_PST[1]);
txi[2].setTxi(txi01_GST[2], txi02_GST[2], txi03_GST[2], txi06_GST[2]);
txi[2].setTxi(txi01_PST[2], txi02_PST[2], txi03_PST[2], txi06_PST[2]);
txi[3].setTxi(txi01_GST[3], txi02_GST[3], txi03_GST[3], txi06_GST[3]);
txi[3].setTxi(txi01_PST[3], txi02_PST[3], txi03_PST[3], txi06_PST[3]);
txi[4].setTxi(txi01_GST[4], txi02_GST[4], txi03_GST[4], txi06_GST[4]);
txi[4].setTxi(txi01_PST[4], txi02_PST[4], txi03_PST[4], txi06_PST[4]);
AxIt1Loop it1Loop = new AxIt1Loop();
it1Loop.setIt1Loop(it102[0], it103[0], it104[0], it105[0], it106s[0], txi[0], pam05[0],
pid05[0]);
it1Loop.setIt1Loop(it102[1], it103[1], it104[1], it105[1], it106s[1], txi[1], pam05[1],
pid05[1]);
it1Loop.setIt1Loop(it102[2], it103[2], it104[2], it105[2], it106s[2], txi[2], pam05[2],
pid05[2]);
it1Loop.setIt1Loop(it102[3], it103[3], it104[3], it105[3], it106s[3], txi[3], pam05[3],
pid05[3]);
it1Loop.setIt1Loop(it102[4], it103[4], it104[4], it105[4], it106s[4], txi[4], pam05[4],
pid05[4]);
AxTable2 table2 = new AxTable2();
table2.setIt1Loop(it1Loop);
//Create Table 3 with details
AxTxi taxTbl3 = new AxTxi();
taxTbl3.setTxi("GS", "4.25", "", ""); //sum of GST taxes
taxTbl3.setTxi("PG", "4.60", "", ""); //sum of PST taxes
taxTbl3.setTxi("TX", "8.85", "", ""); //sum of all taxes
AxTable3 table3 = new AxTable3();
table3.setTxi(taxTbl3);

AxLevel123 level123 = new AxLevel123();
level123.setTable1(table1);
level123.setTable2(table2);
level123.setTable3(table3);
AxForcePost axForcePost = new AxForcePost();
axForcePost.setOrderId(order_id);
axForcePost.setCustId(cust_id);
axForcePost.setAmount(amount);
axForcePost.setPan(pan);
axForcePost.setExpDate(expiry_date);
axForcePost.setAuthCode(auth_code);
axForcePost.setCryptType(crypt);
axForcePost.setAxLevel123(level123);
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processsing_country_code);
mpgReq.setTestMode(true); //false for production transactions
mpgReq.setstoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(axForcePost);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{

```

```

Receipt receipt = mpgReq.getReceipt();

System.out.println("CardType = " + receipt.getCardType());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = " + receipt.getBankTotals());
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("CavvResultCode = " + receipt.getCavvResultCode());
}
catch (Exception e)
{
System.out.println(e);
}
}
}

```

## 5.4.6 Correction d'achat par carte Amex

Une correction d'achat par carte Amex (annulation) est utilisée pour annuler une transaction effectuée dans le lot en cours. Aucun montant n'est requis, car la correction d'achat annule toujours la totalité du montant de la transaction initiale. Les seules transactions pouvant être annulées à l'aide de la correction d'achat par carte Amex sont les transactions de conclusion et forcées par carte Amex. Pour envoyer une transaction de correction d'achat par carte Amex, vous avez besoin de l'ID de commande et du numéro de transaction de la conclusion par carte Amex ou de la transaction forcée par carte Amex.

### Définition de l'objet de transaction AX Purchase Correction

```
AxPurchaseCorrection axPurchaseCorrection = new AxPurchaseCorrection();
```

### Objet HttpsPostRequest pour les transactions de correction d'achat par carte Amex

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
```

```
mpgReq.setTransaction(axPurchaseCorrection);
```

### Champs de demande liés aux transactions de correction d'achat par carte Amex (obligatoires)

Variable	Type et limites	Méthode Set
No de commande	<i>Chaîne</i> 50 caractères alphanumériques	axPurchaseCorrection.setOrderId(order_id);
Numéro de transaction	<i>Chaîne</i> 255 caractères alphanumériques	axPurchaseCorrection.setTxnNumber(txn_number);
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	axPurchaseCorrection.setCryptType(crypt);

**Correction d'achat par carte Amex**

```
package Level23;
import JavaAPI.*;

public class TestAxPurchaseCorrection
{
public static void main(String[] args)
{
String store_id = "moneris";
String api_token = "hurgle";
String processing_country_code = "CA";
boolean status_check = false;

String order_id="Test1485206180427";
String txn_number = "660117311902017023161620759-0_11";
String crypt="7";
AxPurchaseCorrection axPurchaseCorrection = new AxPurchaseCorrection();
axPurchaseCorrection.setOrderId(order_id);
axPurchaseCorrection.setTxnNumber(txn_number);
axPurchaseCorrection.setCryptType(crypt);
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false for production transactions
mpgReq.setstoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(axPurchaseCorrection);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = " + receipt.getBankTotals());
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("CavvResultCode = " + receipt.getCavvResultCode());
}
catch (Exception e)
{
System.out.println(e);
}
}
}
```

### 5.4.7 Remboursement par carte Amex

Une transaction de remboursement par carte Amex crédite un montant précis à la carte de crédit du titulaire de carte. Une transaction de remboursement d'une valeur allant jusqu'à la valeur totale de la transaction de conclusion ou forcée par carte Amex originale peut être envoyé. Pour effectuer un remboursement par carte Amex, vous avez besoin de l'ID de commande et du numéro de transaction de la conclusion ou forcée par carte Amex originale.

#### Définition de l'objet de transaction AX Refund

```
AxRefund axRefund = new AxRefund();
```

#### Objet HttpsPostRequest pour les transactions de remboursement par carte Amex

```
HttpsPostRequest mpgReq = new HttpsPostRequest();

mpgReq.setTransaction(axRefund);
```

#### Champs de demande liés aux transactions de remboursement par carte Amex (obligatoires)

Variable	Type et limites	Méthode Set
No de commande	<i>Chaîne</i> 50 caractères alphanumériques	axRefund.SetOrderId(order_id);
Numéro de transaction	<i>Chaîne</i> 255 caractères alphanumériques	axRefund.SetTxnNumber(txn_number);
Montant	<i>Chaîne</i> 10 caractères décimaux Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal <b>EXEMPLE : 1 234 567,89</b>	axRefund.SetAmount(amount);
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	axRefund.SetCryptType(crypt);
Données de niveaux 2 et 3	<i>Objet</i> S. O.	axRefund.SetAxLevel23(level23);

### Exemple de remboursement par carte Amex

```

package Level23;
import JavaAPI.*;
public class TestAxRefund
{
public static void main(String[] args)
{
String store_id = "moneris";
String api_token = "hurgle";
String processing_country_code = "CA";
boolean status_check = false;

String order_id="Test1485206231878";
String amount="62.37";
String txn_number = "660117311902017023161712265-0_11";
String crypt="7";

//Create Table 1 with details
String n101 = "R6"; //Entity ID Code
String n102 = "Retailing Inc. International"; //Name
String n301 = "919 Oriole Rd."; //Address Line 1
String n401 = "Toronto"; //City
String n402 = "On"; //State or Province
String n403 = "H1T6W3"; //Postal Code
String[] ref01 = {"4C", "CR"}; //Reference ID Qualifier
String[] ref02 = {"M5T3A5", "16802309004"}; //Reference ID
String big04 = "P07758545"; //Purchase Order Number
String big05 = "RN0049858"; //Release Number
String big10 = "INV99870E"; //Invoice Number
AxRef axRef1 = new AxRef();
axRef1.setRef(ref01[0], ref02[0]);
axRef1.setRef(ref01[1], ref02[1]);
AxN1Loop n1Loop = new AxN1Loop();
n1Loop.setN1Loop(n101, n102, n301, n401, n402, n403, axRef1);
AxTable1 table1 = new AxTable1();
table1.setBig04(big04);
table1.setBig05(big05);
table1.setBig10(big10);
table1.setN1Loop(n1Loop);

//Create Table 2 with details
//the sum of the extended amount field (pam05) must equal the level 1 amount field
String[] it102 = {"1", "1", "1", "1", "1"}; //Line item quantity invoiced
String[] it103 = {"EA", "EA", "EA", "EA", "EA"}; //Line item unit or basis of measurement code
String[] it104 = {"10.00", "25.00", "8.62", "10.00", "-10.00"}; //Line item unit price
String[] it105 = {"", "", "", "", ""}; //Line item basis of unit price code

String[] it10618 = {"MG", "MG", "MG", "MG", "MG"}; //Product/Service ID qualifier
String[] it10719 = {"DJFR4", "JFJ49", "FEF33", "FEE43", "DISCOUNT"}; //Product/Service ID
//(corresponds to it10618)

String[] txi01_GST = {"GS", "GS", "GS", "GS", "GS"}; //Tax type code
String[] txi02_GST = {"0.70", "1.75", "1.00", "0.80", "0.00"}; //Monetary amount
String[] txi03_GST = {"", "", "", "", ""}; //Percent
String[] txi06_GST = {"", "", "", "", ""}; //Tax exempt code

String[] txi01_PST = {"PG", "PG", "PG", "PG", "PG"}; //Tax type code
String[] txi02_PST = {"0.80", "2.00", "1.00", "0.80", "0.00"}; //Monetary amount
String[] txi03_PST = {"", "", "", "", ""}; //Percent
String[] txi06_PST = {"", "", "", "", ""}; //Tax exempt code
String[] pam05 = {"11.50", "28.75", "10.62", "11.50", "-10.00"}; //Extended line-item amount
String[] pid05 = {"Stapler", "Lamp", "Bottled Water", "Fountain Pen", "DISCOUNT"}; //Line item
description
AxIt106s[] it106s = {new AxIt106s(), new AxIt106s(), new AxIt106s(), new AxIt106s(), new
AxIt106s()};

it106s[0].setIt10618(it10618[0]);
it106s[0].setIt10719(it10719[0]);

it106s[1].setIt10618(it10618[1]);
it106s[1].setIt10719(it10719[1]);

it106s[2].setIt10618(it10618[2]);
it106s[2].setIt10719(it10719[2]);

```

```

it106s[3].setIt10618(it10618[3]);
it106s[3].setIt10719(it10719[3]);

it106s[4].setIt10618(it10618[4]);
it106s[4].setIt10719(it10719[4]);
AxTxi[] txi = {new AxTxi(), new AxTxi(), new AxTxi(), new AxTxi()};
txi[0].setTxi(txio1_GST[0], txio2_GST[0], txio3_GST[0], txio6_GST[0]);
txi[0].setTxi(txio1_PST[0], txio2_PST[0], txio3_PST[0], txio6_PST[0]);
txi[1].setTxi(txio1_GST[1], txio2_GST[1], txio3_GST[1], txio6_GST[1]);
txi[1].setTxi(txio1_PST[1], txio2_PST[1], txio3_PST[1], txio6_PST[1]);
txi[2].setTxi(txio1_GST[2], txio2_GST[2], txio3_GST[2], txio6_GST[2]);
txi[2].setTxi(txio1_PST[2], txio2_PST[2], txio3_PST[2], txio6_PST[2]);
txi[3].setTxi(txio1_GST[3], txio2_GST[3], txio3_GST[3], txio6_GST[3]);
txi[3].setTxi(txio1_PST[3], txio2_PST[3], txio3_PST[3], txio6_PST[3]);
txi[4].setTxi(txio1_GST[4], txio2_GST[4], txio3_GST[4], txio6_GST[4]);
txi[4].setTxi(txio1_PST[4], txio2_PST[4], txio3_PST[4], txio6_PST[4]);
AxIt1Loop it1Loop = new AxIt1Loop();
it1Loop.setIt1Loop(it102[0], it103[0], it104[0], it105[0], it106s[0], txi[0], pam05[0],
pid05[0]);
it1Loop.setIt1Loop(it102[1], it103[1], it104[1], it105[1], it106s[1], txi[1], pam05[1],
pid05[1]);
it1Loop.setIt1Loop(it102[2], it103[2], it104[2], it105[2], it106s[2], txi[2], pam05[2],
pid05[2]);
it1Loop.setIt1Loop(it102[3], it103[3], it104[3], it105[3], it106s[3], txi[3], pam05[3],
pid05[3]);
it1Loop.setIt1Loop(it102[4], it103[4], it104[4], it105[4], it106s[4], txi[4], pam05[4],
pid05[4]);
AxTable2 table2 = new AxTable2();
table2.setIt1Loop(it1Loop);
//Create Table 3 with details
AxTxi taxTbl3 = new AxTxi();
taxTbl3.setTxi("GS", "4.25","",""); //sum of GST taxes
taxTbl3.setTxi("PG", "4.60","",""); //sum of PST taxes
taxTbl3.setTxi("TX", "8.85","",""); //sum of all taxes
AxTable3 table3 = new AxTable3();
table3.setTxi(taxTbl3);

//Create and set Level23 Object
AxLevel23 level23 = new AxLevel23();
level23.setTable1(table1);
level23.setTable2(table2);
level23.setTable3(table3);
AxRefund axRefund = new AxRefund();
axRefund.setOrderId(order_id);
axRefund.setAmount(amount);
axRefund.setTxnNumber(txn_number);
axRefund.setCryptType(crypt);
axRefund.setAxLevel23(level23);
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false for production transactions
mpgReq.setstoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(axRefund);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = " + receipt.getBankTotals());
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("CavvResultCode = " + receipt.getCavvResultCode());
}
catch (Exception e)
{
System.out.println(e);
}

```

```
}
```

```
}
```

```
}
```

## 5.4.8 Remboursement indépendant par carte Amex

Une transaction de remboursement indépendant par carte Amex crédite un montant précis à la carte de crédit du titulaire de carte. Une transaction de remboursement indépendant ne nécessite pas qu'une commande existante soit enregistrée dans Passerelle Moneris, mais le numéro de la carte de crédit et la date d'expiration doivent être transmis.

### Définition de l'objet de transaction AX Independent Refund

```
AxIndependentRefund axIndependentRefund = new AxIndependentRefund();
```

### Objet HttpsPostRequest pour les transactions de remboursement indépendant par carte Amex

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
```

```
mpgReq.setTransaction(axIndependentRefund);
```

### Champs de demande liés aux transactions de remboursement indépendant par carte Amex (obligatoires)

Variable	Type et limites	Méthode Set
No de commande	<i>Chaîne</i> 50 caractères alphanumériques	axIndependentRefund.setOrderId(order_id);
Montant	<i>Chaîne</i> 10 caractères décimaux Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal	axIndependentRefund.setAmount(amount);
Numéro de carte de crédit	<i>Chaîne</i> 20 caractères	axIndependentRefund.setPan(pan);

**EXEMPLE : 1 234 567,89**

Variable	Type et limites	Méthode Set
	alphanumériques	
Date d'expiration	<i>Chaîne</i> 4 caractères alphanumériques (format AAMM)	axIndependentRefund.setExpDate( expiry_date);
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	axIndependentRefund.setCryptTyp e(crypt);

**Champs de demande liés aux transactions de remboursement indépendant par carte Amex (facultatifs)**

Variable	Type et limites	Méthode Set
ID du client	<i>Chaîne</i> 50 caractères alphanumériques	axIndependentRefund.setCustId( cust_id);

**Exemple d'une transaction de remboursement indépendant par carte Amex**

```

package Level23;
import JavaAPI.*;
public class TestAxIndependentRefund
{
    public static void main(String[] args)
    {
        String store_id = "moneris";
        String api_token = "hurgle";
        String processing_country_code = "CA";
        boolean status_check = false;

        java.util.Date createDate = new java.util.Date();
        String order_id="Test"+createDate.getTime();
        String cust_id="CUST13343";
        String amount="62.37";
        String pan="373269005095005";
        String expiry_date="2012"; //YYMM
        String crypt="7";

        //Create Table 1 with details
        String n101 = "R6"; //Entity ID Code
        String n102 = "Retailing Inc. International"; //Name
        String n301 = "919 Oriole Rd."; //Address Line 1
        String n401 = "Toronto"; //City
        String n402 = "On"; //State or Province
        String n403 = "H1T6W3"; //Postal Code
        String[] ref01 = {"4C", "CR"}; //Reference ID Qualifier
        String[] ref02 = {"M5T3A5", "16802309004"}; //Reference ID
        String big04 = "PO7758545"; //Purchase Order Number
        String big05 = "RN0049858"; //Release Number
        String big10 = "INV99870E"; //Invoice Number
        AxRef axRef1 = new AxRef();
        axRef1.setRef(ref01[0], ref02[0]);
        axRef1.setRef(ref01[1], ref02[1]);
        AxN1Loop n1Loop = new AxN1Loop();
        n1Loop.setN1Loop(n101, n102, n301, n401, n402, n403, axRef1);
    }
}

```

```

AxTable1 table1 = new AxTable1();
table1.setBig04(big04);
table1.setBig05(big05);
table1.setBig10(big10);
table1.setN1Loop(n1Loop);

//Create Table 2 with details
//the sum of the extended amount field (pam05) must equal the level 1 amount field
String[] it102 = {"1", "1", "1", "1", "1"}; //Line item quantity invoiced
String[] it103 = {"EA", "EA", "EA", "EA", "EA"}; //Line item unit or basis of measurement code
String[] it104 = {"10.00", "25.00", "8.62", "10.00", "-10.00"}; //Line item unit price
String[] it105 = {"", "", "", "", ""}; //Line item basis of unit price code

String[] it10618 = {"MG", "MG", "MG", "MG", "MG"}; //Product/Service ID qualifier
String[] it10719 = {"DJFR4", "JFJ49", "FEF33", "FEE43", "DISCOUNT"}; //Product/Service ID
//(corresponds to it10618)

String[] txi01_GST = {"GS", "GS", "GS", "GS"}; //Tax type code
String[] txi02_GST = {"0.70", "1.75", "1.00", "0.80", "0.00"}; //Monetary amount
String[] txi03_GST = {"", "", "", "", ""}; //Percent
String[] txi06_GST = {"", "", "", "", ""}; //Tax exempt code

String[] txi01_PST = {"PG", "PG", "PG", "PG"}; //Tax type code
String[] txi02_PST = {"0.80", "2.00", "1.00", "0.80", "0.00"}; //Monetary amount
String[] txi03_PST = {"", "", "", "", ""}; //Percent
String[] txi06_PST = {"", "", "", "", ""}; //Tax exempt code
String[] pam05 = {"11.50", "28.75", "10.62", "11.50", "-10.00"}; //Extended line-item amount
String[] pid05 = {"Stapler", "Lamp", "Bottled Water", "Fountain Pen", "DISCOUNT"}; //Line item
description
AxIt106s[] it106s = {new AxIt106s(), new AxIt106s(), new AxIt106s(), new AxIt106s(), new
AxIt106s()};

it106s[0].setIt10618(it10618[0]);
it106s[0].setIt10719(it10719[0]);

it106s[1].setIt10618(it10618[1]);
it106s[1].setIt10719(it10719[1]);
it106s[2].setIt10618(it10618[2]);
it106s[2].setIt10719(it10719[2]);

it106s[3].setIt10618(it10618[3]);
it106s[3].setIt10719(it10719[3]);

it106s[4].setIt10618(it10618[4]);
it106s[4].setIt10719(it10719[4]);
AxTxi[] txi = {new AxTxi(), new AxTxi(), new AxTxi(), new AxTxi()};
txi[0].setTxi(txi01_GST[0], txi02_GST[0], txi03_GST[0], txi06_GST[0]);
txi[0].setTxi(txi01_PST[0], txi02_PST[0], txi03_PST[0], txi06_PST[0]);
txi[1].setTxi(txi01_GST[1], txi02_GST[1], txi03_GST[1], txi06_GST[1]);
txi[1].setTxi(txi01_PST[1], txi02_PST[1], txi03_PST[1], txi06_PST[1]);
txi[2].setTxi(txi01_GST[2], txi02_GST[2], txi03_GST[2], txi06_GST[2]);
txi[2].setTxi(txi01_PST[2], txi02_PST[2], txi03_PST[2], txi06_PST[2]);
txi[3].setTxi(txi01_GST[3], txi02_GST[3], txi03_GST[3], txi06_GST[3]);
txi[3].setTxi(txi01_PST[3], txi02_PST[3], txi03_PST[3], txi06_PST[3]);
txi[4].setTxi(txi01_GST[4], txi02_GST[4], txi03_GST[4], txi06_GST[4]);
txi[4].setTxi(txi01_PST[4], txi02_PST[4], txi03_PST[4], txi06_PST[4]);
AxIt1Loop it1Loop = new AxIt1Loop();
it1Loop.setIt1Loop(it102[0], it103[0], it104[0], it105[0], it106s[0], txi[0], pam05[0],
pid05[0]);
it1Loop.setIt1Loop(it102[1], it103[1], it104[1], it105[1], it106s[1], txi[1], pam05[1],
pid05[1]);
it1Loop.setIt1Loop(it102[2], it103[2], it104[2], it105[2], it106s[2], txi[2], pam05[2],
pid05[2]);
it1Loop.setIt1Loop(it102[3], it103[3], it104[3], it105[3], it106s[3], txi[3], pam05[3],
pid05[3]);
it1Loop.setIt1Loop(it102[4], it103[4], it104[4], it105[4], it106s[4], txi[4], pam05[4],
pid05[4]);
AxTable2 table2 = new AxTable2();
table2.setIt1Loop(it1Loop);
//Create Table 3 with details
AxTxi taxTbl3 = new AxTxi();
taxTbl3.setTxi("GS", "4.25", "", ""); //sum of GST taxes
taxTbl3.setTxi("PG", "4.60", "", ""); //sum of PST taxes
taxTbl3.setTxi("TX", "8.85", "", ""); //sum of all taxes
AxTable3 table3 = new AxTable3();
table3.setTxi(taxTbl3);

//Create and set Level23 Object
AxLevel23 level23 = new AxLevel23();
level23.setTable1(table1);
level23.setTable2(table2);

```

```

level23.setTable3(table3);
AxIndependentRefund axIndependentRefund = new AxIndependentRefund();
axIndependentRefund.setOrderId(order_id);
axIndependentRefund.setCustId(cust_id);
axIndependentRefund.setAmount(amount);
axIndependentRefund.setPan(pan);
axIndependentRefund.setExpDate(expiry_date);
axIndependentRefund.setCryptType(crypt);
axIndependentRefund.setAxLevel23(level23);
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false for production transactions
mpgReq.setstoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(axIndependentRefund);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = " + receipt.getBankTotals());
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("CavvResultCode = " + receipt.getCavvResultCode());
}
catch (Exception e)
{
System.out.println(e);
}
}
}

```

## 6 3-D Secure 2.2

- 6.1 À propos de la solution 3-D Secure 2.2
- 6.2 Créer votre intégration 3-D Secure 2.2
- 6.3 Mise en œuvre de la demande de recherche de carte (CardLookup)
- 6.4 Utilisation de la méthode 3DS pour l'empreinte digitale des appareils
- 6.5 Mise en œuvre de la demande d'authentification 3DS pour les modules d'extension des commerçants
- 6.6 Traitement du flux de contestation
- 6.7 Gestion du processus d'authentification découpée
- 6.8 Effectuer l'autorisation
- 6.9 Tester votre intégration 3-D Secure 2.2
- 6.10 Passage à la phase de production avec 3-D Secure 2.2
- 6.11 Codes TransStatus de la solution 3-D Secure 2.2
- 6.12 Codes TransStatusReason de la solution 3-D Secure 2.2
- 6.13 Codes de résultat du code de vérification d'authentification du titulaire de carte (CAVV)

### 6.1 À propos de la solution 3-D Secure 2.2

3-D Secure 2.2 est un protocole d'authentification de paiement d'EMVCo conçu pour réduire la fraude avec carte absente en évaluant le risque en fonction des données de la transaction et de l'appareil, tout en prenant en charge d'autres mesures d'atténuation du risque, telles qu'une contestation du titulaire de la carte. Dans certains cas, un transfert de responsabilité a lieu pour certains débits compensatoires liés à une fraude avec carte absente, ce qui permet au commerçant d'offrir des biens et des services en toute confiance.

Passerelle Moneris peut autoriser les transactions utilisant le protocole 3-D Secure grâce au serveur 3DS de Moneris et au serveur de contrôle d'accès (ACS).

Passerelle Moneris prend en charge les mises en œuvre 3-D Secure suivantes :

- Visa Secure (Veuillez noter que Visa Secure ne prend pas en charge tous les indicateurs RI de 3D Secure 2.2. Vérifiez que l'état Soutien pour Visa Secure s'affiche dans le champ État des indicateurs RI.)
- Mastercard Identity Check
- American Express SafeKey (remarque : American Express prend uniquement en charge les demandes d'authentification pour les commerçants qui ont un compte de commerçant Amex OFI)

#### 6.1.1 Mises en place de l'outil 3-D Secure

Visa Secure, Mastercard Identity Check et American Express SafeKey sont des programmes qui reposent sur le protocole 3-D Secure pour améliorer la sécurité des transactions en ligne.

Ces programmes comprennent l'authentification du titulaire de la carte lors d'une transaction de commerce électronique en ligne.

L'authentification repose sur la méthode d'authentification choisie par l'émetteur.

Voici des exemples de méthodes d'authentification :

- Authentification basée sur le risque
- Mots de passe dynamiques
- Mots de passe fixes

Les avantages de ces programmes comprennent la réduction du risque de transactions frauduleuses et la protection contre les débits compensatoires pour certaines transactions frauduleuses.

L'API XML.NET 3DS 2.2 prend en charge deux catégories de messages et deux canaux de périphériques du protocole d'authentification 3-D Secure :

1. Catégories de messages :

- **Authentification du paiement** : L'authentification du titulaire de la carte est effectuée avant le traitement d'une transaction de commerce électronique. Après une authentification 3DS réussie, vous procédez à un achat ou à une préautorisation.
- **Authentification de non-paiement** : L'identité est vérifiée et le compte est confirmé sans transaction financière d'accompagnement. Après une authentification 3DS réussie, vous pouvez poursuivre :
  - o La transformation de la carte en jetons pour les paiements futurs
  - o L'autorisation de l'accès aux portails des clients
  - o Toute autre activité reposant sur la confirmation de l'identité ou du compte

2. Canaux de l'appareil :

- **Navigateur** : La transaction provient d'un site Web utilisé par l'entremise d'un navigateur sur l'appareil du titulaire de la carte.
- o Par exemple, une transaction de commerce électronique sur le site Web du commerçant avec un processus de paiement que le titulaire de la carte utilise par l'entremise du navigateur Web de son ordinateur personnel ou de son téléphone portable (Chrome, Edge, Safari, etc.).
- **Authentification 3DS entamée par le demandeur (3RI)** : Le compte est confirmé et le titulaire de la carte est authentifié, sans que le titulaire de la carte soit à l'origine de la transaction.
- o L'authentification 3RI peut être utilisée pour authentifier les transactions liées à des commandes postales ou téléphoniques (MOTO).
- o L'authentification 3RI peut être utilisée pour authentifier des transactions ultérieures dans le cadre d'un abonnement, comme des transactions périodiques. Le premier paiement du titulaire de la carte peut utiliser une authentification dans le navigateur, les paiements suivants utilisant une authentification 3RI liée à la précédente.
- o Dans les cas où le modèle commercial d'un commerçant permet d'attendre avant de traiter le paiement, il peut utiliser l'authentification découpée pour permettre au titulaire de la carte de s'authentifier directement auprès de l'émetteur par le biais d'une contestation qui n'est pas liée à 3DS, tel qu'une notification vers une application bancaire.

### 6.1.2 Hors de portée ou non pris en charge

- Solutions intégrées aux applications

### 6.1.3 Compatibilité des versions

Tout changement à l'API de Moneris doit être en mesure de prendre en charge l'ajout de nouveaux champs et de nouvelles conditions d'erreur dans la réponse. Sinon, tout changement affectant la rétrocompatibilité sera communiqué par Solutions Moneris avec un préavis approprié. Lors du développement de la solution, il est recommandé de valider l'état de réussite de la demande, puis de traiter les états d'erreur séparément et de s'assurer qu'une solution finale est renvoyée pour toute

erreur inattendue ou non documentée.

## 6.1.4 Passage de l'outil 3-D Secure 2.0 à 2.2

L'API 3DS 2.2 est différent de l'API 3DS 2.0. Les développeurs doivent donc suivre les étapes de la section 6.2 Créer votre intégration 3-D Secure 2.2.

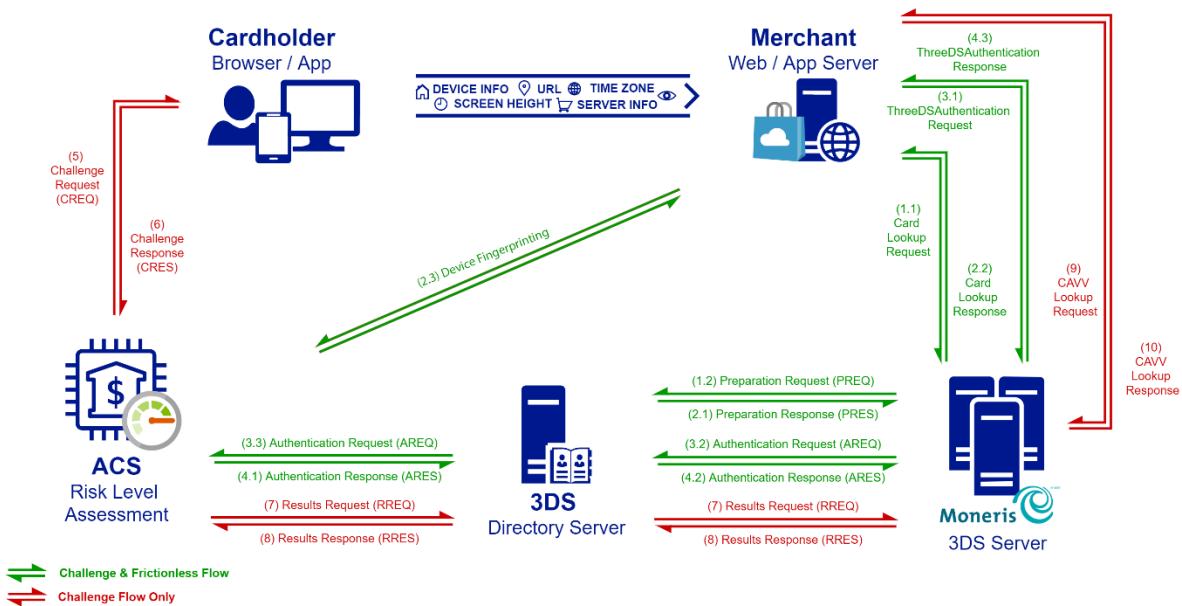
## 6.2 Créez votre intégration 3-D Secure 2.2

- 6.2.1 Activation de la fonction 3-D Secure
- 6.2.2 Flux des transactions pour 3-D Secure : canal du navigateur
- 6.2.3 Flux de transactions pour 3-D Secure : Canal 3RI

### 6.2.1 Activation de la fonction 3-D Secure

Pour activer la fonction de transaction Visa Secure, Mastercard Identity Check ou American Express SafeKey, appelez le service des ventes de Moneris en composant 1 855 465-4980 pour que Moneris vous inscrive au programme et active la fonction dans votre compte.

### 6.2.2 Flux des transactions avec la solution 3-D Secure



L'API 3DS 2.2 est utilisée lorsque le client souhaite payer un achat. Une demande facultative de recherche de carte peut être effectuée pour lancer la prise d'empreinte du navigateur du titulaire de la carte. Une fois l'empreinte terminée, ou comme première étape si l'on n'effectue pas d'empreinte, les renseignements transactionnels peuvent alors être transmis au service 3DS 2.2 afin qu'une évaluation des risques puisse être lancée.

Le flux peut se dérouler de deux façons. Les deux flux sont appelés « sans heurts (frictionless) » et « contestation (challenge) ».

Le flux « sans heurts » est transparent pour le titulaire de la carte. Si l'institution financière émettrice dispose de suffisamment de renseignements pour évaluer le risque et assumer la responsabilité, cela se présentera sous la forme d'une tentative ou d'un succès d'authentification accompagné d'une valeur

CAVV. Aucune « contestation » n'est présentée au titulaire de carte.

Dans le flux de « contestation », l'institution financière émettrice peut décider de prendre une mesure supplémentaire et de contester le titulaire de la carte. Ici, le navigateur du titulaire de la carte est redirigé vers la plateforme 3DS de l'émetteur à des fins d'authentification. Une fois l'authentification terminée, le navigateur du titulaire de la carte est à nouveau redirigé vers le site du commerçant. Le serveur du commerçant émet alors une requête entre serveurs afin d'obtenir la valeur CAVV de Moneris.

### **Étapes 1 et 2 (facultatif)**

Une demande facultative de recherche de carte peut être effectuée pour lancer la prise d'empreinte du navigateur du titulaire de la carte. Le site Web du commerçant recueille des informations sur l'appareil et les fournit à Moneris par l'entremise de la demande card\_lookup (1.1). Moneris transmet ces données au serveur de répertoires 3DS et renvoie la réponse card\_lookup contenant la version 3DS prise en charge par la carte, une URL ACS et des données de méthode 3DS représentant l'empreinte digitale (2.2). Le navigateur du commerçant envoie ensuite un message HTTP POST à l'URL ACS avec les données de la méthode. (2.3)

Une fois l'empreinte terminée, ou comme première étape si l'on n'effectue pas d'empreinte, les renseignements transactionnels peuvent alors être transmis au service 3DS 2.2 afin qu'une évaluation des risques puisse être lancée.

### **Étapes 3 et 4 (obligatoires)**

La demande d'authentification 3DS threeDSAuthentication est exécutée par le site Web du commerçant pour commencer à valider l'identité du titulaire de la carte. Moneris communique avec le répertoire 3DS et le système de l'ACS de cet émetteur afin de fournir une évaluation initiale du risque (3.2-4.2). Moneris renvoie au commerçant une réponse threeDSAuthentication avec un TransStatus, qui indique au site Web la mesure à prendre :

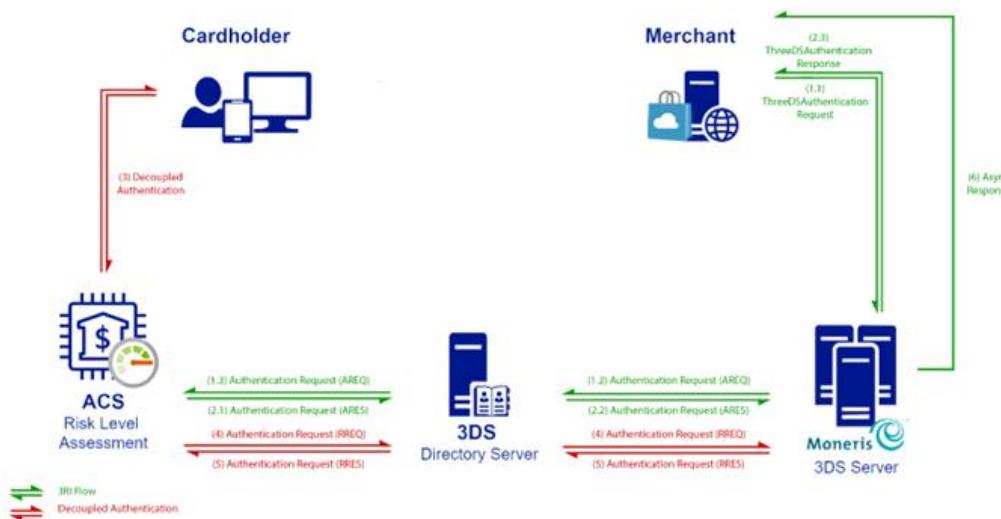
- Un TransStatus = "Y" ou "A" signifie que le site Web peut passer immédiatement à la transaction financière au moyen du code de vérification d'authentification du titulaire de carte (valeur « CAVV ») fourni. Il s'agit d'un processus de transaction sans heurts qui n'a pas de contestation.
- Un TransStatus = "C" indique que le titulaire de carte doit recevoir une contestation. Pour présenter la contestation, vous devez utiliser la méthode POST et mettre un formulaire (<form>) avec un champ « creq », qui contient les données de la contestation (ChallengeData), lié à l'URL définie dans le champ « ChallengeURL ».
- Un TransStatus = "D" indique que l'authentification découpée nécessite la contestation pour authentifier le titulaire de la carte (voir la section Authentication découpée).

### **Étapes 5 à 10 (contestation uniquement)**

Dans les cas où une contestation est nécessaire, le site Web du commerçant envoie un message HTTP POST à l'URL de contestation avec les données de contestation envoyées sous la forme d'une valeur "CREQ" (5). Le système de l'ACS présentera une contestation au titulaire de carte qui fournira tous les identifiants requis par son émetteur. Le site Web du commerçant reçoit une valeur "CRES" de l'ACS via la réponse HTTP POST (6). Pendant ce temps, l'ACS fournit les résultats au répertoire 3DS, qui les transmet ensuite à Moneris (7-8).

Le site Web du commerçant envoie ensuite une demande de recherche du CAVV à Moneris par le biais d'une requête cavv\_lookup et inclut son "CRES" (9). Moneris répond avec la réponse cavv\_lookup avec les valeurs ECI et CAVV nécessaires. Une fois l'authentification 3DS terminée, vous pouvez procéder à la transaction financière.

## 6.2.3 Flux de transactions pour 3-D Secure : Canal 3RI



Dans un processus amorcé par un demandeur 3DS, le titulaire de carte ne déclenche pas directement le processus de transaction par l'entremise de l'expérience du navigateur comme plus haut. Il se peut que la transaction soit amorcée à l'extérieur du protocole 3DS, notamment par la poste ou en appelant le commerçant (connu comme commande postale et téléphonique), ou il se peut que le commerçant traite un paiement périodique ou par versements dans le cadre de l'abonnement d'un titulaire de carte. Le commerçant peut aussi avoir besoin d'une authentification de non-paiement dans le cadre de la transformation de la carte en jetons à des fins d'utilisation ultérieure.

Les processus 3RI n'ont pas d'interaction directe avec le titulaire de carte. Le commerçant envoie sa requête <threeDSAuthentication> en suivant les étapes 3 et 4, mais il doit inclure les champs supplémentaires pour décrire son utilisation 3RI.

- S'il s'agit d'une authentification de paiement liée à une commande postale ou téléphonique, l'ACS peut déclencher une authentification découpée entre l'émetteur et le titulaire de carte (voir la section Authentification découpée).
- Si le paiement de suivi provient d'une transaction 3DS authentifiée précédemment, vous pouvez inclure la variable `prior_request_auth_info` pour faire le lien avec l'authentification précédente et améliorer les chances d'obtenir un résultat réussi.

Votre serveur peut utiliser les champs `device_channel`, `ri_indicator` et `messageCategory` pour informer Moneris si le serveur de votre commerçant tente d'utiliser le processus amorcé par le demandeur 3DS.

### 6.2.3.1 Authentification découpée

Pour les scénarios qui ont une authentification 3RI qui nécessite une contestation, au lieu d'avoir une requête et une réponse de contestation standard, l'ACS authentifie le titulaire de carte à l'extérieur du protocole 3-D Secure, notamment dans une application bancaire ou par l'entremise d'un message texte envoyé au titulaire de carte. Le serveur 3DS de Moneris attend que l'ACS authentifie le titulaire de carte. Cette authentification peut prendre jusqu'à sept jours. Puisque ce processus est basé sur une action du titulaire de carte à l'extérieur du processus 3DS, il a lieu de façon asynchrone par rapport au traitement des transactions.

Votre serveur peut utiliser les champs `decoupled_request_indicator` et `decoupled_request_async_url` pour informer Moneris que vous acceptez une tentative d'authentification découpée et où vous voulez que Moneris POST les résultats de manière asynchrone.

## 6.3 Mise en œuvre de la demande de recherche de carte (CardLookup)

La demande de recherche de carte (CardLookup) vérifie si la carte est compatible avec la norme 3DS et renvoie l'URL de la méthode 3DS. Celle-ci est utilisée pour prendre l'empreinte de l'appareil. Cette demande est facultative, mais elle peut augmenter les chances d'un flux sans heurts.

Les données threeDSMethodURL et threeDSMethodData sont renvoyées au serveur du commerçant dans la réponse CardLookup, si la fonction est prise en charge.

- Si vous recevez le threeDSMethodURL, Les données threeDSMethodData peuvent être transmises à threeDSMethodURL en utilisant la méthode POST du navigateur afin de compléter la demande d'authentification avec des données relatives au navigateur du titulaire de la carte.
- Si vous ne recevez pas le threeDSMethodURL, vous pouvez toujours procéder à l'authentification 3DS. Les données threeDSMethodData doivent être envoyées grâce à HTTP POST à l'URL threeDSMethodURL dans un cadre en ligne caché.

Dans votre mise en œuvre, utilisez les URL suivantes comme serveur, en fonction du stade de développement :

Test :

esqa.moneris.com

Production :

www3.moneris.com

### 6.3.1 Demande de recherche de carte (Card Lookup)

#### Définition de l'objet de transaction Card Lookup Request

```
MpiCardLookup mpiCardLookup = new MpiCardLookup();
```

#### Objet HttpsPostRequest pour les transactions de recherche de carte (Card Lookup)

```
HttpsPostRequest mpgReq = new HttpsPostRequest();  
mpgReq.SetTransaction(mpiCardLookup);
```

#### Principaux champs d'objet de connexion (toutes les transactions API)

Variable	Type et limites	
ID de commerce	<i>Chaîne</i> S. O.	mpgReq.setstoreId(store_id);
Jeton API	<i>Chaîne</i> S. O.	mpgReq.setApiToken(api_token);

## Champs de demande pour les transactions de recherche de carte (obligatoires)

**REMARQUE :** Soit un numéro de carte de crédit (pan) ou soit une clé de données (data\_key) doit être envoyé dans la demande.

Variable	Type et limites	
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques a-z A-Z 0-9 _ - : . @ espaces	cardLookup.setOrderId(order_id);
Numéro de carte de crédit	<i>Chaîne</i> Maximum 20 caractères alphanumériques	cardLookup.setPan(pan);
Clé de données	<i>Chaîne</i> 25 caractères alphanumériques	cardLookup.setData(data_key);
URL de notification	<i>Chaîne</i> 256 caractères alphanumériques	cardLookup.setNotificationURL("https://yournotificationurl.com");

### Exemple de demande de recherche de carte (Card Lookup)

```

package Canada;
import JavaAPI.*;
public class TestCanadaMpiCardLookup
{
  public static void main(String[] args)
  {
    String store_id = "moneris";
    String api_token = "hurgle";
    java.util.Date createDate = new java.util.Date();
    String order_id = "Test"+createDate.getTime();
    String pan = "347668693641199";

    String processing_country_code = "CA";
    MpiCardLookup mpiCardLookup = new MpiCardLookup();
    mpiCardLookup.setOrderId(order_id);
    mpiCardLookup.setPan(pan);
    //mpiCardLookup.setDataKey("800XGiwxgvfbZngigVFeld9d2"); //Optional - For Moneris Vault and
    Hosted Tokenization tokens in place of setPan
    mpiCardLookup.setNotificationUrl("https://yournotificationurl.com"); //(Website URL that will
    receive 3DS Method Completion response from ACS)
    //*****OPTIONAL VARIABLES*****
    HttpsPostRequest mpgReq = new HttpsPostRequest();
    mpgReq.setProcCountryCode(processing_country_code);
    mpgReq.setTestMode(true); //false or comment out this line for production transactions
    mpgReq.setStoreId(store_id);
    mpgReq.setApiToken(api_token);
    mpgReq.setTransaction(mpiCardLookup);
    mpgReq.send();
    //***** REQUEST *****/
  }
}

```

```

try
{
    Receipt receipt = mpgReq.getReceipt();
    System.out.println("ResponseCode = " + receipt.getResponseCode());
    System.out.println("ReceiptId = " + receipt.getReceiptId());
    System.out.println("Message = " + receipt.getMessage());
    System.out.println("MessageType = " + receipt.getMpiMessageType());
    System.out.println("ThreeDSVersion = " + receipt.getThreeDSVersion());
    System.out.println("ThreeDSMethodURL = " + receipt.getMpiThreeDSMethodURL());
    System.out.println("ThreeDSMethodData = " + receipt.getMpiThreeDSMethodData());
    System.out.println("ThreeDSServerTransId = " + receipt.getMpiThreeDSServerTransId());
}
catch (Exception e)
{
    e.printStackTrace();
}
}
} // end TestResMpiTxn

```

## 6.4 Gestion de la méthode 3DS pour l'empreinte digitale des appareils

Vous pouvez utiliser les **threeDSMethodURL** et **threeDSMethodData** renvoyés par une réponse Card Lookup pour augmenter la probabilité d'un flux 3DS sans friction pour le titulaire de la carte. La transmission des **threeDSMethodData** au **threeDSMethodURL** par l'entremise d'un navigateur HTTP POST permet à l'émetteur d'utiliser un cadre en ligne caché sur le site Web du commerçant pour obtenir des détails sur l'appareil du client.

Les résultats de la méthode 3DS sont renvoyés au **notificationURL** du commerçant fourni dans la recherche de carte précédente.

Vous trouverez ci-dessous un exemple de formulaire statique de base pour vous aider à visualiser les données et les champs qui doivent être soumis.

**Formulaire de demande de prise d'empreintes digitales d'un appareil (navigateur de commerçant vers ACS) :** <form name="frm" method="POST" action="Rendering URL">

```

<input type="hidden" name="threeDSMethodData"
value="eyJ0aHJlZURTU2VydmyHJhbnNRCI6IjNhYzdjYWE3LWFhDITMjY2My03OTFiLTJhYzA1
YTU0MmM0SISInRocmV1RFNZRob2ROb3RpZmljYXRpb25VUkwiOiJ0aHJlZURTTWV0aG9kTm90aWZp
Y2F0aW9uVJMIIn0">
</form>

```

**threeDSMethodData décodé :**

```
{"threeDSServerTransID": "3ac7caa7-aa42-2663-791b-2ac05a542c4a", "threeDSMethodNotificationURL": "threeDSMethodNotificationURL"}
```

**Formulaire de réponse à la prise d'empreinte de l'appareil (ACS à Merchant notificationURL) :** <form name="frm" method="POST" action="threeDSMethodNotificationURL">

```

<input type="hidden" name="threeDSMethodData"
value="eyJ0aHJlZURTU2VydmyVHJhbnNRCI6IjNhYzdjYWE3LWFhNDITMjY2My03OTFiLTJhYzA1
YTU0Mm0YSJ9">
</form>

```

**threeDSMethodData décodé :** {"threeDSServerTransID": "3ac7caa7-aa42-2663-791b-2ac05a542c4a"}

## 6.5 Mise en œuvre de la demande d'authentification 3DS

La demande d'authentification 3DS pour les modules d'extension des commerçants est utilisée pour lancer le processus de validation de la carte. Le résultat de cette demande détermine si le système 3DS 2.2 est pris en charge par la carte et quel type d'authentification est requis.

Dans votre mise en œuvre, utilisez les URL suivantes comme serveur, en fonction du stade de développement :

Test :

<https://mpg1t.monteris.io/mpi2/servlet/MpiServlet>

Production :

<https://mpg1.monteris.io/mpi2/servlet/MpiServlet>

Nous détaillons ci-dessous trois scénarios différents d'utilisation de l'authentification MPI 3DS de Moneris. Chaque scénario comporte des conditions pour lesquelles les champs sont obligatoires ou facultatifs pour le point de terminaison.

### 6.5.1 Mise en œuvre de la demande d'authentification 3DS pour les modules d'extension des commerçants

La demande d'authentification est utilisée pour lancer le processus de validation de la carte.

Le résultat de cette demande détermine si le système 3DS 2.0 est pris en charge par la carte et quel type d'authentification est requis.

Dans votre mise en œuvre, utilisez les URL suivantes comme serveur, en fonction du stade de développement :

Test : [mpg1t.monteris.io](http://mpg1t.monteris.io)

Production : [mpg1.monteris.io](http://mpg1.monteris.io)

**REMARQUE :** Les champs de demande liés à la facturation doivent être envoyés pour cette transaction, sinon le processus d'authentification peut échouer.

#### Définition de l'objet de transaction MPI 3DS Authentication Request

```
MpiThreeDSAuthentication mpiThreeDSAuthentication = new  
MpiThreeDSAuthentication();
```

#### Objet HttpsPostRequest pour les transactions de demande d'authentification 3DS des modules d'extension pour les commerçants

```
HttpsPostRequest mpgReq = new HttpsPostRequest();  
  
mpgReq.setTransaction(mpiThreeDSAuthentication);
```

**AVERTISSEMENT :** N'envoyez pas de champs relatifs à l'authentification 3RI pour des authentifications dans le navigateur.

### Principaux champs d'objet de connexion (toutes les transactions API)

Variable	Type et limites	
ID de commerce <store_id>	<i>Chaîne</i> S. O.	mpgReq.setstoreId(store_id);  Identifiant unique fourni par Moneris lors de la configuration du compte de commerçant
Jeton API <api_token>	<i>Chaîne</i> S. O.	mpgReq.setApiToken(api_token);  Chaîne de caractères alphanumériques unique créée par Moneris lors de l'activation du compte de commerçant  Pour obtenir votre jeton API, accédez aux paramètres Admin de l'environnement de test ou de production de votre magasin dans le Centre de ressources pour commerçants, qui se trouve aux liens URL qui suivent :  Test : <a href="https://esqa.moneris.com/mpg/?chlang=fr">https://esqa.moneris.com/mpg/?chlang=fr</a> Production : <a href="https://www3.moneris.com/mpg/?chlang=fr">https://www3.moneris.com/mpg/?chlang=fr</a>

**Champs de demande liés à la transaction de demande d'authentification 3DS des modules d'extension pour les commerçants (obligatoires)**

Variable	Type et limites	
Catégorie de message  <messageCategory>	<i>Chaîne</i>  2 caractères numériques	<pre>mpiThreeDSAuthentication.setMessageCategory("02");</pre> <p>Si la demande d'authentification concerne une utilisation avec ou sans paiement :</p> <p>01 = Authentification de paiement 02 = Authentification de non-paiement</p>
Canal de l'appareil  <device_channel>	<i>Chaîne</i>  2 caractères numériques	<pre>mpiThreeDSAuthentication.setDeviceChannel("02");</pre> <p>L'interface utilisée pour commencer l'authentification :</p> <p>02 = Navigateur 03 = Authentification 3RI</p>
Type de demande  <requestType>	<i>Chaîne</i>  2 caractères alphanumériques	<pre>mpiThreeDSAuthentication.setRequestType("REQUEST_TYPE_VALUE");</pre> <p>Le type de demande d'authentification dans le navigateur :</p> <p>01 = Paiement entamé par le titulaire de la carte 02 = Transaction périodique 03 = transaction effectuée par versements 04 = ajout d'une carte 05 = gestion de la carte 06 = vérification du titulaire de la carte dans le cadre de l'identification et de la vérification du jeton EMV  Conditionnel : Requis si device_channel = 02</p>
ID de commande  <order_id>	<i>Chaîne</i>  50 caractères alphanumériques a-z A-Z 0-9 _ - : . @ espaces	<pre>mpiThreeDSAuthentication.setOrderId(order_id);</pre> <p>Identifiant de transaction déterminé par le commerçant et unique pour chaque transaction d'achat, de préautorisation et de remboursement indépendant Il ne</p>

		<p>peut y avoir deux de ces types transactions avec le même ID de commande.</p> <p>Pour les transactions de remboursement, de conclusion ou de correction d'achat, l'ID de la commande doit correspondre à celui de la transaction originale.</p>
Clé de donnée <data_key> OU Numéro de carte de crédit <pan>	<i>Chaîne</i> Limites de la clé de données : 25 caractères alphanumériques Numéro de carte de crédit (PAN) Maximum de 20 caractères alphanumériques	<pre>mpiThreeDSAuthentication.setDat a(data_key);</pre> <p><b>Description de la clé de données :</b></p> <p>Il s'agit d'un identifiant unique pour un profil de chambre forte, qui est utilisé dans les futures transactions financières de la chambre forte pour associer une transaction à ce profil.</p> <p>La clé de donnée est générée par Moneris et vous est retournée dans l'objet Reçu lors du premier enregistrement du profil.</p> <pre>mpiThreeDSAuthentication.SetPan (pan);</pre> <p><b>Numéro de carte de crédit</b></p> <p>Numéro de carte de crédit, qui comporte généralement 16 chiffres – le champ peut comporter un maximum de 20 chiffres en vue de l'expansion future des numéros de carte</p> <p>Contient le jeton requis pour les transactions de transformation en jetons</p>
Date d'expiration <expdate>	<i>Chaîne</i> 4 caractères alphanumériques AAMM	<pre>mpiThreeDSAuthentication.setExp Date(expiry_date);</pre> <p>Date d'expiration de la carte de crédit, au format AAMM</p> <p><b>REMARQUE :</b> Il s'agit de l'inverse du format de date MMAA qui est affichée sur la carte.</p>
Montant <amount>	<i>Chaîne</i> 10 caractères décimaux Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal	<pre>mpiThreeDSAuthentication.setAmo unt(amount);</pre> <p><b>Montant de la transaction en dollars</b></p> <p>Doit comporter au moins trois chiffres, dont deux décimales</p> <p>Valeur minimale acceptée : 0,01 \$, valeur maximale acceptée : 9 999 999,99 \$</p> <p><b>EXEMPLE : 1 234 567,89</b></p>

Nom du titulaire de carte  <cardholderName>	<i>Chaîne</i>  45 caractères alphanumériques	mpiThreeDSAuthentication.setCardholderName ("CARDHOLDER_NAME_VALUE");  Nom du titulaire de carte
	<b>REMARQUE :</b> Les caractères accentués ne sont pas autorisés.	
Indicateur de conclusion 3DS  <threeDSCompletionInd>	<i>Chaîne</i>  1 caractère alphanumérique	mpiThreeDSAuthentication.setThreeDSCompletionInd (THREEDSCOMPLETION_VALUE);  Indique si le processus de MpICardLookup de la méthode 3ds s'est déroulée avec succès  Valeurs acceptées :  Y = Conclu avec succès  N = N'a pas été conclu avec succès  U = Non disponible  Conditionnel : Requis si card_lookup est utilisé
Adresse de facturation  <billAddress1>	<i>Chaîne</i>  50 caractères alphanumériques	mpiThreeDSAuthentication.setBillAddress1 ("BILL_STREET_ADDRESS_VALUE");  Adresse de facturation du titulaire de la carte
Province de facturation  <billProvince>	<i>Chaîne</i>  3 caractères alphanumériques	mpiThreeDSAuthentication.setBillProvince ("BILL_PROV_VALUE");  Province ou état de facturation du titulaire de la carte  Défini dans la sous-division du pays de la norme ISO 3166-2
Ville de facturation  <billCity>	<i>Chaîne</i>  50 caractères alphanumériques	mpiThreeDSAuthentication.setBillCity ("BILL_CITY_VALUE");  Ville de facturation du titulaire de la carte
Code postal de facturation  <billPostalCode>	<i>Chaîne</i>  16 caractères alphanumériques	mpiThreeDSAuthentication.setBillPostalCode ("BILL_POSTAL_CODE_VALUE");  Code postal de facturation du titulaire de la carte

Pays de facturation <code>&lt;billCountry&gt;</code>	<i>Chaîne</i> 3 caractères alphanumériques	<code>mpiThreeDSAuthentication.setBillCountry ("BILL_COUNTRY_VALUE");</code> <b>Pays de facturation du titulaire de la carte</b> Correspond à un code de pays de 3 chiffres de la norme ISO 3166-1
Adresse d'expédition <code>&lt;shipAddress1&gt;</code>	<i>Chaîne</i> 50 caractères alphanumériques	<code>mpiThreeDSAuthentication.setShipAddress1 ("SHIP_STREET_ADDRESS_VALUE");</code> <b>Rue de l'adresse d'expédition</b>
Province d'expédition <code>&lt;shipProvince&gt;</code>	<i>Chaîne</i> 3 caractères alphanumériques	<code>mpiThreeDSAuthentication.setShipProvince ("SHIP_PROV_VALUE");</code> <b>Province ou état de l'adresse d'expédition</b> Défini dans la sous-division du pays de la norme ISO 3166-2
Ville d'expédition <code>&lt;shipCity&gt;</code>	<i>Chaîne</i> 50 caractères alphanumériques	<code>mpiThreeDSAuthentication.setShipCity ("SHIP_CITY_VALUE");</code> <b>Ville de l'adresse d'expédition</b>
Code postal d'expédition <code>&lt;shipPostalCode&gt;</code>	<i>Chaîne</i> 16 caractères alphanumériques	<code>mpiThreeDSAuthentication.setShipPostalCode ("SHIP_POSTAL_CODE_VALUE");</code> <b>Code postal ou ZIP de l'adresse d'expédition</b>
Pays d'expédition <code>&lt;shipCountry&gt;</code>	<i>Chaîne</i> 3 caractères alphanumériques	<code>mpiThreeDSAuthentication.setShipCountry ("SHIP_COUNTRY_VALUE");</code> <b>Pays de l'adresse d'expédition</b> Correspond à un code de pays de 3 chiffres de la norme ISO 3166-1
URL de notification <code>&lt;notificationUrl&gt;</code>	<i>Chaîne</i> 256 caractères alphanumériques	<code>mpiThreeDSAuthentication.setNotificationURL ("HTTPS://YOURURL.COM");</code> <b>URL de notification pour la réception de la réponse POST de la méthode 3DS de l'ACS de l'émetteur</b> Conditionnel : Requis si device_channel = 02
Taille de la fenêtre <code>&lt;challengeWindowsize&gt;</code>	<i>Chaîne</i> 2 caractères alphanumériques	<code>mpiThreeDSAuthentication.setChallengeWindowSize ("CWS_VALUE");</code> <b>Concerne le rendu de la contestation du serveur de contrôle d'accès (ACS) dans le navigateur</b> Valeurs acceptées : 01 = 250 x 400 02 = 390 x 400

		<p>03 = 500 x 600</p> <p>04 = 600 x 400</p> <p>05 = Écran complet</p> <p>Conditionnel : Requis si device_channel = 02</p>
--	--	---

<p>browser IP Address</p> <p>&lt;browser_ip&gt;</p>	<p><i>Chaîne</i></p> <p>Permet l'utilisation des caractères « . » et « : ». 45 caractères alphanumériques</p>	<p>mpiThreeDSAuthentication.setBrowserIP ("10.10.10.10") or ("011:0db8:85a3:0101:0101:8a2e:0370:7334"); // (IPv4 or IPv6)</p> <p>Adresse IP du navigateur telle qu'elle est renvoyée par les en-têtes HTTP au demandeur 3DS.</p> <p>{b}REMARQUE : {/b}ce champ n'est pas obligatoire, mais vous devez tout de même le définir. Il est fortement recommandé de fournir cette information. Le fait de ne pas fournir la valeur de ce champ peut augmenter le risque de refus.</p>
---	---	---

<p>Agent utilisateur du navigateur</p> <p>&lt;browserUserAgent&gt;</p>	<p><i>Chaîne</i></p> <p>2048 caractères alphanumériques</p>	<p>mpiThreeDSAuthentication.setBrowserUserAgent ("BROWSER_USER_AGENT_VALUE");</p> <p>Agent utilisateur du navigateur</p> <p>Conditionnel : Requis si device_channel = 02</p>
<p>Java activé dans le navigateur</p> <p>&lt;browserJavaEnabled&gt;</p>	<p><i>Chaîne</i></p> <p>1 caractère alphabétique</p> <p>T ou F</p>	<p>mpiThreeDSAuthentication.setBrowserJavaEnabled(BROWSER_JAVA_VALUE);</p> <p>Indique si Java est activé dans le navigateur</p> <p>Valeurs acceptées :</p> <p>T = Vrai</p> <p>F = Faux</p> <p>Conditionnel : Requis si device_channel = 02</p>
<p>Hauteur de la fenêtre du navigateur</p> <p>&lt;browserScreenHeight&gt;</p>	<p><i>Chaîne</i></p> <p>6 caractère numérique</p>	<p>mpiThreeDSAuthentication.setBrowserScreenHeight(BROWSER_SCREEN_HEIGHT_VALUE);</p> <p>Hauteur en pixels de l'écran du titulaire de carte</p> <p>Conditionnel : Requis si device_channel = 02</p>
<p>Largeur de la fenêtre du navigateur</p> <p>&lt;browserScreenWidth&gt;</p>	<p><i>Chaîne</i></p> <p>6 caractère numérique</p>	<p>mpiThreeDSAuthentication.setBrowserScreenWidth(BROWSER_SCREEN_WIDTH_VALUE);</p> <p>Largeur en pixels de l'écran du titulaire</p>

		de carte  Conditionnel : Requis si device_channel = 02
Langue du navigateur <browserLanguage>	<i>Chaîne</i> 8 caractères alphanumériques	mpiThreeDSAuthentication.setBrowserLanguage(BROWSER_LANGUAGE_VALUE);  Comme défini dans IETF BCP47  Conditionnel : Requis si device_channel = 02
Courriel <email>	<i>Chaîne</i> 254 caractères alphanumériques	mpiThreeDSAuthentication.setEmail("EMAIL_VALUE");  Adresse courriel du titulaire de la carte   <b>REMARQUE :</b> Ce champ n'est pas obligatoire, mais il est requis. Il est fortement recommandé de fournir l'adresse courriel du titulaire de la carte. Le risque de rejet peut augmenter si l'adresse courriel du titulaire de la carte n'est pas fournie.

cardholder work phone number  <work_phone>	<i>Objet</i> S.O.	mpiThreeDSAuthentication.setWorkPhone(workPhone);  Numéro de téléphone au travail du titulaire de la carte   <b>REMARQUE :</b> ce champ n'est pas obligatoire, mais vous devez tout de même le définir. Il est fortement recommandé de fournir au moins un des numéros de téléphone du titulaire de la carte. Le fait de ne pas fournir au moins un des numéros de téléphone du titulaire de la carte peut augmenter le risque de refus.  <b>REMARQUE :</b> il s'agit d'un objet imbriqué dans la transaction. Pour plus d'information sur les champs de l'objet Cardholder Phone Number Info, consultez la section Objet Cardholder Phone Number Info et variables.
cardholder home phone number  <home_phone>	<i>Objet</i> S.O.	mpiThreeDSAuthentication.setHomePhone(homePhone);  Numéro de téléphone à domicile du titulaire de la carte   <b>REMARQUE :</b> ce champ n'est pas obligatoire, mais vous devez tout de même le définir. Il est fortement

		<p>recommandé de fournir au moins un des numéros de téléphone du titulaire de la carte. Le fait de ne pas fournir au moins un des numéros de téléphone du titulaire de la carte peut augmenter le risque de refus.</p> <p>{b}REMARQUE : {/b}il s'agit d'un objet imbriqué dans la transaction. Pour plus d'information sur les champs de l'objet Cardholder Phone Number Info, consultez la section Objet Cardholder Phone Number Info et variables.</p>
cardholder mobile phone number  <mobile_phone>	<i>Objet</i>  S.O.	<pre>mpiThreeDSAuthentication.setMobilePhone(mobilePhone);</pre> <p>Numéro de téléphone cellulaire du titulaire de la carte</p> <p>{b}REMARQUE : {/b}ce champ n'est pas obligatoire, mais vous devez tout de même le définir. Il est fortement recommandé de fournir au moins un des numéros de téléphone du titulaire de la carte. Le fait de ne pas fournir au moins un des numéros de téléphone du titulaire de la carte peut augmenter le risque de refus.</p> <p>{b}REMARQUE : {/b}il s'agit d'un objet imbriqué dans la transaction. Pour plus d'information sur les champs de l'objet Cardholder Phone Number Info, consultez la section Objet Cardholder Phone Number Info et variables.</p>

#### Numéro de téléphone du titulaire de la carte 3DS pour les modules d'extension des commerçants

Variable	Type et limites	Description
country code  <country_code>	<i>Chaîne</i>  3 caractères numériques	Code de pays du numéro de téléphone fourni par le titulaire de la carte.
phone number  <phone_number>	<i>Chaîne</i>  15 caractères numériques	Le numéro de téléphone fourni par le titulaire de la carte.

**Champs de demande liés à la transaction de demande d'authentification 3DS des modules d'extension pour les commerçants ( facultatifs )**

Variable	Type et limites	Méthode Set
Devise <currency>  <b>REMARQUE :</b> Ce champ ne devrait pas être envoyé, à moins que la tarification multidevise soit activée dans votre compte de commerçant.	<i>Chaîne</i> 3 caractères numériques	<pre>mpiThreeDSAuthentication.setCurrency ("CURRENCY_VALUE");</pre> <p>Code de devise à 3 chiffres ISO 4217</p> <p>CAD = 124</p> <p>USD = 840</p>
Demande de contestation <requestChallenge>	<i>Chaîne</i> 2 caractères numériques	<pre>setRequestChallenge ("CHALLENGE_VALUE");</pre> <p>Indique si une contestation dans un navigateur est demandée pour cette transaction ("01" est la norme.)</p> <p>01 = Pas de préférence</p> <p>02 = Pas de contestation demandée</p> <p>03 = Contestation demandée : Préférence du demandeur 3DS</p> <p>04 = Contestation demandée : Mandat Conditionnel : Requis si device_channel = 02</p>

## Exemple de la demande d'authentification 3DS pour les modules d'extension des Commerçants

```

package Canada;
import JavaAPI.*;
public class TestCanadaMpiThreeDSAuthentication
{
    public static void main(String[] args)
    {
        String store_id = "moneris";
        String api_token = "hurgle";
        String prior_request_ref="d7c1ee99-9478-44a6-b1f2-391e29c6b340";
        String prior_request_auth_data="abc";
        String prior_request_auth_method = "01";
        String prior_request_auth_timestamp = "201710282113";
        String processing_country_code = "CA";

        Hashtable<String, String> workPhoneParams = new Hashtable<>();
        workPhoneParams.put("cc", "1");
        workPhoneParams.put("subscriber", "1111111111");

        Hashtable<String, String> homePhoneParams = new Hashtable<>();
        homePhoneParams.put("cc", "3");
        homePhoneParams.put("subscriber", "3333333333");

        Hashtable<String, String> mobilePhoneParams = new Hashtable<>();
        mobilePhoneParams.put("cc", "2");
        mobilePhoneParams.put("subscriber", "2222222222");

        MpiThreeDSAuthentication mpiThreeDSAuthentication = new MpiThreeDSAuthentication();
        mpiThreeDSAuthentication.setOrderId("Test159787ssa3215193"); //must be the same one that was used in MpiCardLookup call
        mpiThreeDSAuthentication.setCardholderName("Moneris Test");
        mpiThreeDSAuthentication.setPan("347668693641199");
        // mpiThreeDSAuthentication.setDataKey("xR1904FgrZUYdGkmqHTqiEw97"); //Optional - For Moneris Vault and Hosted Tokenization tokens in place of setPan
        mpiThreeDSAuthentication.setExpdate("2310");
        mpiThreeDSAuthentication.setAmount("1.00");
        mpiThreeDSAuthentication.setThreeDSCompletionInd("Y"); // (Y|N|U) indicates whether 3ds method MpiCardLookup was successfully completed
        mpiThreeDSAuthentication.setRequestType("01"); // (01=payment | 02=recur)
        mpiThreeDSAuthentication.setPurchaseDate("20200819035249"); // (YYYYMMDDHHMMSS)
        mpiThreeDSAuthentication.setNotificationURL("https://yournotificationurl.com"); // (Website where response from RRes or CRes after challenge will go)
        mpiThreeDSAuthentication.setChallengeWindowSize("03"); // (01 = 250 x 400, 02 = 390 x 400, 03 = 500 x 600, 04 = 600 x 400, 05 = Full screen)
        mpiThreeDSAuthentication.setEmail("test@email.com");

        mpiThreeDSAuthentication.setBrowserIP ("10.10.10.10") or
        ("011:0db8:85a3:0101:0101:8a2e:0370:7334"); // (IPv4 or IPv6)

        mpiThreeDSAuthentication.setBrowserUserAgent("Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.132 Safari/537.36\\");
        mpiThreeDSAuthentication.setBrowserJavaEnabled("true"); // (true|false)
        mpiThreeDSAuthentication.setBrowserScreenHeight("1000"); // (pixel height of cardholder screen)
        mpiThreeDSAuthentication.setBrowserScreenWidth("1920"); // (pixel width of cardholder screen)
        mpiThreeDSAuthentication.setBrowserLanguage("en-GB"); // (defined by IETF BCP47)

        WorkPhone workPhone = new WorkPhone(workPhoneParams);
        mpiThreeDSAuthentication.setWorkPhone(workPhone);

        HomePhone homePhone = new HomePhone(homePhoneParams);
        mpiThreeDSAuthentication.setHomePhone(homePhone);

        MobilePhone mobilePhone = new MobilePhone(mobilePhoneParams);
        mpiThreeDSAuthentication.setMobilePhone(mobilePhone);

        //Optional Methods
        mpiThreeDSAuthentication.setBillAddress1("3300 Bloor St W");
        mpiThreeDSAuthentication.setBillProvince("ON");
        mpiThreeDSAuthentication.setBillCity("Toronto");
        mpiThreeDSAuthentication.setBillPostalCode("M8X 2X2");
        mpiThreeDSAuthentication.setBillCountry("124");
    }
}

```

```

mpiThreeDSAuthentication.setShipAddress1("3300 Bloor St W");
mpiThreeDSAuthentication.setShipProvince("ON");
mpiThreeDSAuthentication.setShipCity("Toronto");
mpiThreeDSAuthentication.setShipPostalCode("M8X 2X2");
mpiThreeDSAuthentication.setShipCountry("124");

mpiThreeDSAuthentication.setRequestChallenge("Y"); // (Y|N Requesting challenge regardless of outcome)
//*****3DS2.2*****
mpiThreeDSAuthentication.setMessageCategory("01");
mpiThreeDSAuthentication.setDeviceChannel("02");
// mpiThreeDSAuthentication.setDecoupledRequestIndicator("Y");
// mpiThreeDSAuthentication.setDecoupledRequestMaxTime("000010");
// mpiThreeDSAuthentication.setDecoupledRequestAsyncUrl("localhost:8080/googlepay");

// mpiThreeDSAuthentication.setRiIndicator("01");
// PriorAuthenticationInfo priorAuthenticationInfo= new PriorAuthenticationInfo();
// priorAuthenticationInfo.setPriorParams(prior_request_auth_data,prior_request_ref,
// prior_request_auth_method,prior_request_auth_timestamp);
// mpiThreeDSAuthentication.setRecurringExpiry("20221230");
// mpiThreeDSAuthentication.setRecurringFrequency("031");
//*****OPTIONAL VARIABLES*****
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false or comment out this line for production transactions
mpgReq.setstoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(mpiThreeDSAuthentication);
System.out.println(mpiThreeDSAuthentication.toXML());
mpgReq.send();
/************* REQUEST *****/
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("Message = " + receipt.getMessage());
System.out.println("MessageType = " + receipt.getMpiMessageType());
System.out.println("TransStatus = " + receipt.getMpiTransStatus());
System.out.println("TransStatusReason = " + receipt.getMpiTransStatusReason());
System.out.println("ChallengeURL = " + receipt.getMpiChallengeURL());
System.out.println("ChallengeData = " + receipt.getMpiChallengeData());
System.out.println("ThreeDSServerTransId = " + receipt.getThreeDServerTransId());
System.out.println("ThreeDSVersion = " + receipt.getThreeDSVersion());
System.out.println("ThreeDSAcstransID = " + receipt.getMpiThreeDSAcstransID());
System.out.println("DSTransID = " + receipt.getMpiDSTransId());
System.out.println("ThreeDSAuthTimeStamp = " + receipt.getMpiThreeDSAuthTimeStamp());
System.out.println("AuthenticationType = " + receipt.getMpiAuthenticationType());
//In Frictionless flow, you may receive TransStatus as "Y",
//in which case you can then proceed directly to Cavv Purchase/Preauth with values below
if(receipt.getMpiTransStatus().equals("Y"))
{
System.out.println("Cavv = " + receipt.getMpiCavv());
System.out.println("ECI = " + receipt.getMpiEci());
}
}
catch (Exception e)
{
e.printStackTrace();
}
}
} // end TestResMpiTxn

```

## 6.5.2 Demande d'authentification MPI 3DS - Authentification 3RI avec un processus d'authentification périodique

**REMARQUE :** Les champs de demande liés à l'adresse de facturation doivent être envoyés pour cette transaction, sinon le processus d'authentification peut échouer.

## **Définition de l'objet de transaction MPI 3DS Authentication Request**

```
MpiThreeDSAuthentication mpiThreeDSAuthentication = new  
MpiThreeDSAuthentication();
```

## **Objet HttpsPostRequest pour les transactions de demande d'authentification 3DS des modules d'extension pour les commerçants**

```
HttpsPostRequest mpgReq = new HttpsPostRequest();  
  
mpgReq.setTransaction(mpiThreeDSAuthentication);
```

## Principaux champs d'objet de connexion (toutes les transactions API)

Variable	Type et limites	
ID de commerce <store_id>	<i>Chaîne</i> S. O.	<pre>mpgReq.setstoreId(store_id);</pre> <p>Identifiant unique fourni par Moneris lors de la configuration du compte de commerçant</p>
Jeton API <api_token>	<i>Chaîne</i> S. O.	<pre>mpgReq.setApiToken(api_token);</pre> <p>Chaîne de caractères alphanumériques unique créée par Moneris lors de l'activation du compte de commerçant</p> <p>Pour obtenir votre jeton API, accédez aux paramètres Admin de l'environnement de test ou de production de votre magasin dans le Centre de ressources pour commerçants, qui se trouve aux liens URL qui suivent :</p> <p>Test : <a href="https://esqa.moneris.com/mpg/?chlang=fr">https://esqa.moneris.com/mpg/?chlang=fr</a></p> <p>Production : <a href="https://www3.moneris.com/mpg/?chlang=fr">https://www3.moneris.com/mpg/?chlang=fr</a></p>

## Champs de demande liés à la transaction de demande d'authentification 3DS des modules d'extension pour les commerçants (obligatoires)

Variable	Type et limites	
Catégorie de message <messageCategory>	<i>Chaîne</i> 2 caractères numériques	<pre>mpiThreeDSAuthentication.setMessageCategory("02");</pre> <p>Si la demande d'authentification concerne une utilisation avec ou sans paiement :</p> <p>01 = Authentification de paiement</p> <p>02 = Authentification de non-paiement</p>
Canal de l'appareil <device_channel>	<i>Chaîne</i> 2 caractères numériques	<pre>mpiThreeDSAuthentication.setDeviceChannel("02");</pre> <p>L'interface utilisée pour commencer l'authentification :</p> <p>02 = Navigateur</p> <p>03 = Authentification 3RI</p>

Fréquence périodique  <recurring_frequency>	<i>Chaîne</i>  <i>4 caractères numériques</i>	mpiThreeDSAuthentication.SetRecurringFrequency("031") ;  Le nombre minimum de jours entre les transactions périodiques, des valeurs numériques comprises entre 1 et 9999 (les zéros initiaux étant acceptés)  Conditionnel : Obligatoire si l'indicateur ri = 01
Expiration périodique  <recurring_expiry>	<i>Chaîne</i>  <i>8 caractères numériques</i>	mpiThreeDSAuthentication.setRecurringExpiry("20221230");  Date limite après laquelle plus aucune transaction périodique ne sera effectuée (Le format est AAAAMMMJJ.)  Conditionnel : Obligatoire si l'indicateur ri = 01
Indicateur ri  <ri_indicator>  <b>REMARQUE :</b> Visa Secure ne prend en charge que ri_Indicator = 01, 02, 06, 07 ou 11 pour les transactions de paiement et ri Indicator = 03, 04, 05 et 10 pour les transactions de non-paiement.	<i>Chaîne</i>  <i>2 caractères numériques</i>	mpiThreeDSAuthentication.setRiIndicator("03");  Le type de demande 3DS entamée par le demandeur (3RI) :  01 = Périodique 02 = Versement 03 = Ajouter une carte 04 = Gestion des informations relatives à la carte 05 = Vérification du compte 06 = Expédition fractionnée ou retardée 07 = Complément d'information 08 = Commande postale 09 = Commande par téléphone 10 = Liste blanche 11 = Autre paiement  Conditionnel : Requis si device_channel = 03
ID de commande  <order_id>	<i>Chaîne</i>  50 caractères alphanumériques a-z A-Z 0-9 _ - : . @ espaces	mpiThreeDSAuthentication.setOrderId(order_id);  Identifiant de transaction déterminé par le commerçant et unique pour chaque transaction d'achat, de préautorisation et de remboursement indépendant Il ne peut y avoir deux de ces types

		<p>transactions avec le même ID de commande.</p> <p>Pour les transactions de remboursement, de conclusion ou de correction d'achat, l'ID de la commande doit correspondre à celui de la transaction originale.</p>
Clé de donnée <data_key>  OU  Numéro de carte de crédit <pan>	<p><i>Chaîne</i></p> <p>Limites de la clé de données : 25 caractères alphanumériques</p> <p>Numéro de carte de crédit (PAN)</p> <p>Maximum de 20 caractères alphanumériques</p>	<pre>mpiThreeDSAuthentication.SetData(data_key);</pre> <p><b>Description de la clé de données :</b> Il s'agit d'un identifiant unique pour un profil de chambre forte, qui est utilisé dans les futures transactions financières de la chambre forte pour associer une transaction à ce profil.</p> <p>La clé de donnée est générée par Moneris et vous est retournée dans l'objet Reçu lors du premier enregistrement du profil.</p> <pre>mpiThreeDSAuthentication.setPan(pan);</pre> <p><b>Numéro de carte de crédit</b> Numéro de carte de crédit, qui comporte généralement 16 chiffres – le champ peut comporter un maximum de 20 chiffres en vue de l'expansion future des numéros de carte</p> <p>Contient le jeton requis pour les transactions de transformation en jetons</p>
Date d'expiration <expdate>	<p><i>Chaîne</i></p> <p>4 caractères alphanumériques</p> <p>AAMM</p>	<pre>mpiThreeDSAuthentication.setExpDate(expiry_date);</pre> <p>Date d'expiration de la carte de crédit, au format AAMM</p> <p><b>REMARQUE :</b> Il s'agit de l'inverse du format de date MMAA qui est affichée sur la carte.</p>
Montant <amount>	<p><i>Chaîne</i></p> <p>10 caractères décimaux</p> <p>Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal</p>	<pre>mpiThreeDSAuthentication.setAmount(amount);</pre> <p>Montant de la transaction en dollars</p> <p>Doit comporter au moins trois chiffres, dont deux décimales</p> <p><b>EXEMPLE :</b> 1 234 567,89</p>

Nom du titulaire de carte  <cardholderName>	<i>Chaîne</i>  45 caractères alphanumériques	mpiThreeDSAuthentication.setCardholderName ("CARDHOLDER_NAME_VALUE");  Nom du titulaire de carte
	<b>REMARQUE :</b> Les caractères accentués ne sont pas autorisés.	
Adresse de facturation  <billAddress1>	<i>Chaîne</i>  50 caractères alphanumériques	mpiThreeDSAuthentication.setBillAddress1 ("BILL_STREET_ADDRESS_VALUE");  Adresse de facturation du titulaire de la carte
Province de facturation  <billProvince>	<i>Chaîne</i>  3 caractères alphanumériques	mpiThreeDSAuthentication.setBillProvince ("BILL_PROV_VALUE"); Province ou état de facturation du titulaire de la carte  Défini dans la sous-division du pays de la norme ISO 3166-2
Ville de facturation  <billCity>	<i>Chaîne</i>  50 caractères alphanumériques	mpiThreeDSAuthentication.setBillCity ("BILL_CITY_VALUE"); Ville de facturation du titulaire de la carte
Code postal de facturation  <billPostalCode>	<i>Chaîne</i>  16 caractères alphanumériques	mpiThreeDSAuthentication.setBillPostalCode ("BILL_POSTAL_CODE_VALUE"); Code postal de facturation du titulaire de la carte
Pays de facturation  <billCountry>	<i>Chaîne</i>  3 caractères alphanumériques	mpiThreeDSAuthentication.setBillCountry ("BILL_COUNTRY_VALUE"); Pays de facturation du titulaire de la carte  Correspond à un code de pays de 3 chiffres de la norme ISO 3166-1
Adresse d'expédition  <shipAddress1>	<i>Chaîne</i>  50 caractères alphanumériques	mpiThreeDSAuthentication.setShipAddress1 ("SHIP_STREET_ADDRESS_VALUE"); Rue de l'adresse d'expédition
Province d'expédition  <shipProvince>	<i>Chaîne</i>  3 caractères alphanumériques	mpiThreeDSAuthentication.setShipProvince ("SHIP_PROV_VALUE"); Province ou état de l'adresse d'expédition  Défini dans la sous-division du pays de la norme ISO 3166-2

Ville d'expédition <code>&lt;shipCity&gt;</code>	<i>Chaîne</i> 50 caractères alphanumériques	<code>mpiThreeDSAuthentication.setShipCity ("SHIP_CITY_VALUE");</code> Ville de l'adresse d'expédition
Code postal d'expédition <code>&lt;shipPostalCode&gt;</code>	<i>Chaîne</i> 16 caractères alphanumériques	<code>mpiThreeDSAuthentication.setShipPostalCode ("SHIP_POSTAL_CODE_VALUE");</code> Code postal ou ZIP de l'adresse d'expédition
Pays d'expédition <code>&lt;shipCountry&gt;</code>	<i>Chaîne</i> 3 caractères alphanumériques	<code>mpiThreeDSAuthentication.setShipCountry ("SHIP_COUNTRY_VALUE");</code> Pays de l'adresse d'expédition Correspond à un code de pays de 3 chiffres de la norme ISO 3166-1
Courriel <code>&lt;email&gt;</code>	<i>Chaîne</i> 254 caractères alphanumériques	<code>mpiThreeDSAuthentication.setEmail ("EMAIL_VALUE");</code> Adresse courriel du titulaire de la carte  <b>REMARQUE :</b> Ce champ n'est pas obligatoire, mais il est requis. Il est fortement recommandé de fournir l'adresse courriel du titulaire de la carte. Le risque de rejet peut augmenter si l'adresse courriel du titulaire de la carte n'est pas fournie.

#### Champs de demande liés à la transaction de demande d'authentification 3DS des modules d'extension pour les commerçants (facultatifs)

Variable	Type et limites	Méthode Set
Devise <code>&lt;currency&gt;</code>	<i>Chaîne</i> 3 caractères numériques	<code>mpiThreeDSAuthentication.setCurrency ("CURRENCY_VALUE");</code> Code de devise à 3 chiffres ISO 4217  <b>REMARQUE :</b> Ce champ ne devrait pas être envoyé, à moins que la tarification multidevise soit activée dans votre compte de commerçant.
Indicateur de demande découpée <code>&lt;decoupled_request_indicator&gt;</code>	<i>Chaîne</i> 1 caractère alphabétique	<code>mpiThreeDSAuthentication.setDecoupledRequestIndicator ("Y");</code> Si la demande utilise l'authentification découpée ou non, l'ACS confirme son utilisation.  Y = L'authentification découpée est prise en charge et préférée si une contestation est nécessaire.  N = Ne pas utiliser l'authentification découpée (par défaut)

		La valeur par défaut est N si l'authentification découpée n'est pas utilisée.
Temps maximum pour une demande découpée  <decoupled_request_max_time>	<i>Chaîne</i>  5 caractères numériques	<pre>mpiThreeDSAuthentication.setDecoupledRequestMaxTime ("00010");</pre> <p>Le nombre maximum de minutes pendant lesquelles Moneris attend qu'un ACS fournit des résultats.</p> <p>Valeurs numériques comprises entre 1 et 10080 (La période maximale est de 7 jours.)</p> <p>Conditionnel : Requis si device_channel = 03 et decoupled_request_indicator = Y</p>
demande découpée asynchrone URL  <decoupled_request_async_url>	<i>Chaîne</i>  256 caractères alphanumériques	<pre>mpiThreeDSAuthentication.setDecoupledRequestAsyncUrl ("https://yourasyncnotificationurl.com");</pre> <p>Votre URL où Moneris va POSTer la réponse de l'ACS. Moneris tente à nouveau trois fois de POST la réponse.</p> <p>Conditionnel : Uniquement envoyé si decoupled_request_indicator = Y</p>
Demande préalable d'informations d'authentification  <prior_request_auth_info>	<i>Objet</i>  S.O.	<pre>mpiThreeDSAuthentication.setPriorRequestAuthInfo (pai);</pre> <p>Il s'agit d'un objet contenant les détails d'une authentification 3DS précédente pour cette série de transactions. L'objet est imbriqué dans la transaction d'authentification et est nécessaire pour stocker ou utiliser les informations relatives à l'authentification précédente pour cette carte. Pour plus d'informations sur les champs de l'objet Prior Authentication Info, voir la section Informations d'authentification MPI 3DS préalables.</p>

### Informations d'authentification MPI 3DS préalables

Variable	Type et limites	Description
Demande préalable de données d'authentification  <prior_request_auth_data>	<i>Chaîne</i>  36 caractères alphanumériques	<pre>prior_request_auth_data="abc";</pre> <p>Fait référence au DSTransID dans la réponse de l'authentification 3DS précédente</p>

Variable	Type et limites	Description
Référence à une demande préalable <prior_request_auth_ref>	<i>Chaîne</i> 36 caractères alphanumériques	<pre>priorRequestParams.Add("prior_request_ref", "2c581b9d-7f77-4772-a92b-e11df34bce61");</pre> <p>Fait référence à l'ID de transaction de l'ACS 3DS dans la réponse de l'authentification 3DS précédente</p>
Méthode d'authentification de la demande préalable <prior_request_auth_method>	<i>Chaîne</i> 2 caractères numériques	<pre>prior_request_auth_method = "01";</pre> <p>Il s'agit d'un mécanisme utilisé par le titulaire de la carte pour s'authentifier lors de l'authentification 3DS précédente :</p> <ul style="list-style-type: none"> <li>01 = Authentification sans friction</li> <li>02 = Authentification par contestation</li> <li>03 = Vérification par le SVA</li> <li>04 = Autres méthodes de l'émetteur</li> </ul>
Horodatage de la demande préalable d'autorisation <prior_request_auth_timestamp>	<i>Chaîne</i> 12 caractères numériques	<pre>prior_request_auth_timestamp = "201710282113";</pre> <p>Date et heure au fuseau horaire UTC auxquelles a eu lieu l'authentification 3DS du titulaire de carte, qui est trouvé dans la réponse d'authentification 3DS précédente en tant que 3DS Auth TimeStamp (horodatage de l'authentification 3DS)</p> <p>Le format est AAAAMMMJJHHMM.</p>

## Exemple d'une demande d'authentification MPI 3DS – Processus d'authentification périodique 3RI

```
package Canada;
import JavaAPI.*;
public class TestCanadaMpiThreeDSAuthentication
{
public static void main(String[] args)
{
String store_id = "moneris";
String api_token = "hurgle";
String prior_request_ref="d7c1ee99-9478-44a6-b1f2-391e29c6b340";
String prior_request_auth_data="abc";
String prior_request_auth_method = "01";
String prior_request_auth_timestamp = "201710282113";
String processing_country_code = "CA";
MpiThreeDSAuthentication mpiThreeDSAuthentication = new MpiThreeDSAuthentication();
mpiThreeDSAuthentication.setOrderId("Test159787ssa3215193"); //must be the same one
that was used in MpiCardLookup call
mpiThreeDSAuthentication.setCardholderName("Moneris Test");
mpiThreeDSAuthentication.setPan("343427006265962");
// mpiThreeDSAuthentication.setDataKey("xR1904FgrZUYdGkmqHTqiEw97"); //Optional - For
Moneris Vault and Hosted Tokenization tokens in place of setPan
mpiThreeDSAuthentication.setExpdate("2310");
mpiThreeDSAuthentication.setAmount("1.00");
mpiThreeDSAuthentication.setThreeDSCompletionInd("Y"); //(Y|N|U) indicates whether 3ds
method MpiCardLookup was successfully completed
mpiThreeDSAuthentication.setRequestType("02"); //(01=payment|02=recur)
mpiThreeDSAuthentication.setPurchaseDate("20200819035249"); //(YYYYMMDDHHMMSS)
mpiThreeDSAuthentication.setNotificationURL("https://yournotificationurl.com");
//(Website where response from RRes or CRes after challenge will go)
mpiThreeDSAuthentication.setChallengeWindowSize("03"); //(01 = 250 x 400, 02 = 390 x
400, 03 = 500 x 600, 04 = 600 x 400, 05 = Full screen)
mpiThreeDSAuthentication.setEmail("test@email.com");
mpiThreeDSAuthentication.setBrowserUserAgent("Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.132 Safari/537.36\\");
mpiThreeDSAuthentication.setBrowserJavaEnabled("true"); //(true|false)
mpiThreeDSAuthentication.setBrowserScreenHeight("1000"); //(pixel height of cardholder
screen)
mpiThreeDSAuthentication.setBrowserScreenWidth("1920"); //(pixel width of cardholder
screen)
mpiThreeDSAuthentication.setBrowserLanguage("en-GB"); //(defined by IETF BCP47)
//Optional Methods
mpiThreeDSAuthentication.setBillAddress1("3300 Bloor St W");
mpiThreeDSAuthentication.setBillProvince("ON");
mpiThreeDSAuthentication.setBillCity("Toronto");
mpiThreeDSAuthentication.setBillPostalCode("M8X 2X2");
mpiThreeDSAuthentication.setBillCountry("124");
mpiThreeDSAuthentication.setShipAddress1("3300 Bloor St W");
mpiThreeDSAuthentication.setShipProvince("ON");
mpiThreeDSAuthentication.setShipCity("Toronto");
mpiThreeDSAuthentication.setShipPostalCode("M8X 2X2");
mpiThreeDSAuthentication.setShipCountry("124");
mpiThreeDSAuthentication.setRequestChallenge("Y"); //(Y|N Requesting challenge
regardless of outcome)
*****3DS2.2*****
mpiThreeDSAuthentication.setMessageCategory("01");
mpiThreeDSAuthentication.setDeviceChannel("03");
// mpiThreeDSAuthentication.setDecoupledRequestIndicator("Y");
// mpiThreeDSAuthentication.setDecoupledRequestMaxTime("000010");
// mpiThreeDSAuthentication.setDecoupledRequestAsyncUrl("localhost:8080/googlepay");
mpiThreeDSAuthentication.setRiIndicator("01");
PriorAuthenticationInfo priorAuthenticationInfo= new PriorAuthenticationInfo();
//
priorAuthenticationInfo.setPriorParams(prior_request_auth_data,prior_request_ref,prior
_request_auth_method,prior_request_auth_timestamp);
mpiThreeDSAuthentication.setRecurringExpiry("20221230");
mpiThreeDSAuthentication.setRecurringFrequency("031");
*****OPTIONAL VARIABLES*****
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false or comment out this line for production transactions
mpgReq.setstoreId(store_id);
mpgReq.setApiToken(api_token);
```

```

mpgReq.setTransaction(mpiThreeDSAuthentication);
System.out.println(mpiThreeDSAuthentication.toXML());
mpgReq.send();
/********************* REQUEST *****/
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("Message = " + receipt.getMessage());
System.out.println("MessageType = " + receipt.getMpiMessageType());
System.out.println("TransStatus = " + receipt.getMpiTransStatus());
System.out.println("TransStatusReason = " + receipt.getMpiTransStatusReason());
System.out.println("ChallengeURL = " + receipt.getMpiChallengeURL());
System.out.println("ChallengeData = " + receipt.getMpiChallengeData());
System.out.println("ThreeDSserverTransId = " + receipt.getMpiThreeDSserverTransId());
System.out.println("ThreeDSVersion = " + receipt.getThreeDSVersion());
System.out.println("ThreeDSacsTransID = " + receipt.getMpiThreeDSacsTransID());
System.out.println("DSTransID = " + receipt.getMpiDSTransId());
System.out.println("ThreeDSAuthTimeStamp = " + receipt.getMpiThreeDSAuthTimeStamp());
System.out.println("AuthenticationType = " + receipt.getMpiAuthenticationType());
System.out.println("CardHolderInfo = " + receipt.getMpiCardholderInfo());
//In Frictionless flow, you may receive TransStatus as "Y",
//in which case you can then proceed directly to Cavv Purchase/Preauth with values
below
if(receipt.getMpiTransStatus().equals("Y"))
{
System.out.println("Cavv = " + receipt.getMpiCavv());
System.out.println("ECI = " + receipt.getMpiEci());
}
}
catch (Exception e)
{
e.printStackTrace();
}
}
} // end TestResMpiTxn

```

### 6.5.3 Demande d'authentification MPI 3DS – Authentification 3RI sans processus périodique

**REMARQUE :** Les champs de demande liés à l'adresse de facturation doivent être envoyés pour cette transaction, sinon le processus d'authentification peut échouer.

#### MPI 3DS Authentication Request transaction object definition

```

MpiThreeDSAuthentication mpiThreeDSAuthentication = new
MpiThreeDSAuthentication();

```

#### Objet HttpsPostRequest pour les transactions de demande d'authentification 3DS MPI

```

HttpsPostRequest mpgReq = new HttpsPostRequest();

mpgReq.setTransaction(mpiThreeDSAuthentication);

```

**AVERTISSEMENT :** N'envoyez pas de champs relatifs à l'authentification 3RI sur les authentifications dans le navigateur.

## Principaux champs d'objet de connexion (toutes les transactions API)

Variable	Type et limites	
ID de commerce <store_id>	Chaîne S. O.	<pre>mpgReq.setstoreId(store_id);</pre> <p>Identifiant unique fourni par Moneris lors de la configuration du compte de commerçant</p>
Jeton API <api_token>	Chaîne S. O.	<pre>mpgReq.setApiToken(api_token);</pre> <p>Chaîne de caractères alphanumériques unique créée par Moneris lors de l'activation du compte de commerçant</p> <p>Pour obtenir votre jeton API, accédez aux paramètres Admin de l'environnement de test ou de production de votre magasin dans le Centre de ressources pour commerçants, qui se trouve aux liens URL qui suivent :</p> <p>Test : <a href="https://esqa.moneris.com/mpg/?chlang=fr">https://esqa.moneris.com/mpg/?chlang=fr</a></p> <p>Production : <a href="https://www3.moneris.com/mpg/?chlang=fr">https://www3.moneris.com/mpg/?chlang=fr</a></p>

## Champs de demande liés à la transaction de demande d'authentification 3DS des modules d'extension pour les commerçants (obligatoires)

Variable	Type et limites	
Catégorie de message <messageCategory>	Chaîne 2 caractères numériques	<pre>mpiThreeDSAuthentication.setMessageCategory("02");</pre> <p>Si la demande d'authentification concerne une utilisation avec ou sans paiement :</p> <p>01 = Authentification de paiement</p> <p>02 = Authentification de non-paiement</p>
Canal de l'appareil <device_channel>	Chaîne 2 caractères numériques	<pre>mpiThreeDSAuthentication.setDeviceChannel("02");</pre> <p>L'interface utilisée pour commencer l'authentification :</p> <p>02 = Navigateur</p>

		03 = Authentification 3RI
Indicateur ri  <ri_indicator>  <b>REMARQUE :</b> Visa Secure ne prend en charge que ri_Indicator = 6 ou 11 pour les transactions de paiement et ri Indicator = 3, 4, 5 et 10 pour les transactions de non-paiement.	<i>Chaîne</i>  2 caractères numériques	<pre>mpiThreeDSAuthentication.setRiIndicator("03");</pre> <p>Le type de demande 3DS entamée par le demandeur (3RI) :</p> <ul style="list-style-type: none"> <li>01 = Périodique</li> <li>02 = Versement</li> <li>03 = Ajouter une carte</li> <li>04 = Gestion des informations relatives à la carte</li> <li>05 = Vérification du compte</li> <li>06 = Expédition fractionnée ou retardée</li> <li>07 = Complément d'information</li> <li>08 = Commande postale</li> <li>09 = Commande par téléphone</li> <li>10 = Liste blanche</li> <li>11 = Autre paiement</li> </ul> <p>Conditionnel : Requis si device_channel = 03</p>
ID de commande  <order_id>	<i>Chaîne</i>  50 caractères alphanumériques a-z A-Z 0-9 _ - : . @ espaces	<pre>mpiThreeDSAuthentication.setOrderId(order_id);</pre> <p>Identifiant de transaction déterminé par le commerçant et unique pour chaque transaction d'achat, de préautorisation et de remboursement indépendant Il ne peut y avoir deux de ces types transactions avec le même ID de commande.</p> <p>Pour les transactions de remboursement, de conclusion ou de correction d'achat, l'ID de la commande doit correspondre à celui de la transaction originale.</p>

Clé de donnée <data_key> OU Numéro de carte de crédit <pan>	<i>Chaîne</i> Limites de la clé de données : 25 caractères alphanumériques Numéro de carte de crédit (PAN) Maximum de 20 caractères alphanumériques	<pre>mpiThreeDSAuthentication.setDat a(data_key);</pre> <p><b>Description de la clé de données :</b></p> <p>Il s'agit d'un identifiant unique pour un profil de chambre forte, qui est utilisé dans les futures transactions financières de la chambre forte pour associer une transaction à ce profil.</p> <p>La clé de donnée est générée par Moneris et vous est retournée dans l'objet Reçu lors du premier enregistrement du profil.</p> <pre>mpiThreeDSAuthentication.SetPan (pan);</pre> <p><b>Numéro de carte de crédit</b></p> <p>Numéro de carte de crédit, qui comporte généralement 16 chiffres – le champ peut comporter un maximum de 20 chiffres en vue de l'expansion future des numéros de carte</p> <p>Contient le jeton requis pour les transactions de transformation en jetons</p>
Date d'expiration <expdate>	<i>Chaîne</i> 4 caractères alphanumériques AAMM	<pre>mpiThreeDSAuthentication.setExp Date(expiry_date);</pre> <p>Date d'expiration de la carte de crédit, au format AAMM</p>
Montant <amount>	<i>Chaîne</i> 10 caractères décimaux Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal	<pre>mpiThreeDSAuthentication.setAmo unt(amount);</pre> <p>Montant de la transaction en dollars</p> <p>Doit comporter au moins trois chiffres, dont deux décimales</p> <p>Valeur minimale acceptée : 0,01 \$, valeur maximale acceptée : 9 999 999,99 \$</p>
Nom du titulaire de carte <cardholderName>	<i>Chaîne</i> 45 caractères alphanumériques	<pre>mpiThreeDSAuthentication.setCar dholderName("CARDHOLDER_NAME_VA LUE");</pre> <p>Nom du titulaire de carte</p> <p><b>REMARQUE :</b> Les caractères accentués ne sont pas autorisés.</p>

Adresse de facturation <billAddress1>	<i>Chaîne</i> 50 caractères alphanumériques	mpiThreeDSAuthentication.setBillAddress1("BILL_STREET_ADDRESS_VALUE");  Adresse de facturation du titulaire de la carte
Province de facturation <billProvince>	<i>Chaîne</i> 3 caractères alphanumériques	mpiThreeDSAuthentication.setBillProvince ("BILL_PROV_VALUE"); Province ou état de facturation du titulaire de la carte  Défini dans la sous-division du pays de la norme ISO 3166-2
Ville de facturation <billCity>	<i>Chaîne</i> 50 caractères alphanumériques	mpiThreeDSAuthentication.SetBillCity ("BILL_CITY_VALUE"); Ville de facturation du titulaire de la carte
Code postal de facturation <billPostalCode>	<i>Chaîne</i> 16 caractères alphanumériques	mpiThreeDSAuthentication.setBillPostalCode ("BILL_POSTAL_CODE_VALUE"); Code postal de facturation du titulaire de la carte
Pays de facturation <billCountry>	<i>Chaîne</i> 3 caractères alphanumériques	mpiThreeDSAuthentication.setBillCountry ("BILL_COUNTRY_VALUE"); Pays de facturation du titulaire de la carte  Correspond à un code de pays de 3 chiffres de la norme ISO 3166-1
Adresse d'expédition <shipAddress1>	<i>Chaîne</i> 50 caractères alphanumériques	mpiThreeDSAuthentication.setShipAddress1 ("SHIP_STREET_ADDRESS_VALUE"); Rue de l'adresse d'expédition
Province d'expédition <shipProvince>	<i>Chaîne</i> 3 caractères alphanumériques	mpiThreeDSAuthentication.setShipProvince ("SHIP_PROV_VALUE"); Province ou état de l'adresse d'expédition  Défini dans la sous-division du pays de la norme ISO 3166-2
Ville d'expédition <shipCity>	<i>Chaîne</i> 50 caractères alphanumériques	mpiThreeDSAuthentication.setShipCity ("SHIP_CITY_VALUE"); Ville de l'adresse d'expédition
Code postal d'expédition <shipPostalCode>	<i>Chaîne</i> 16 caractères alphanumériques	mpiThreeDSAuthentication.setShipPostalCode ("SHIP_POSTAL_CODE_VALUE"); Code postal ou ZIP de l'adresse d'expédition

Pays d'expédition  <shipCountry>	<i>Chaîne</i>  3 caractères alphanumériques	mpiThreeDSAuthentication.setShipCountry ("SHIP_COUNTRY_VALUE");  Pays de l'adresse d'expédition  Correspond à un code de pays de 3 chiffres de la norme ISO 3166-1
Courriel  <email>	<i>Chaîne</i>  254 caractères alphanumériques	mpiThreeDSAuthentication.setEmail ("EMAIL_VALUE");  Adresse courriel du titulaire de la carte  <b>REMARQUE :</b> Ce champ n'est pas obligatoire, mais il est requis. Il est fortement recommandé de fournir l'adresse courriel du titulaire de la carte. Le risque de rejet peut augmenter si l'adresse courriel du titulaire de la carte n'est pas fournie.

**Champs de demande liés à la transaction de demande d'authentification 3DS des modules d'extension pour les commerçants (facultatifs)**

Variable	Type et limites	Méthode Set
Devise  <currency>	<i>Chaîne</i>  3 caractères numériques	mpiThreeDSAuthentication.setCurrency ("CURRENCY_VALUE");  Code de devise à 3 chiffres ISO 4217  CAD = 124  USD = 840
Indicateur de demande découpée  <decoupled_request_indicator>	<i>Chaîne</i>  1 caractère alphabétique	mpiThreeDSAuthentication.setDecoupledRequestIndicator ("Y");  Si la demande utilise l'authentification découpée ou non, l'ACS confirme son utilisation.  Y = L'authentification découpée est prise en charge et préférée si une contestation est nécessaire.  N = Ne pas utiliser l'authentification découpée (par défaut)  La valeur par défaut est N si l'authentification découpée n'est pas utilisée.
Temps maximum pour une demande découpée  <decoupled_request_max_time>	<i>Chaîne</i>  5 caractères numériques	mpiThreeDSAuthentication.setDecoupledRequestMaxTime ("00010");  Le nombre maximum de minutes

		<p>pendant lesquelles Moneris attend qu'un ACS fournit des résultats.</p> <p>Valeurs numériques comprises entre 1 et 10080 (La période maximale est de 7 jours.)</p> <p>Conditionnel : Requis si device_channel = 03 et decoupled_request_indicator = Y</p>
demande découpée asynchrone URL <decoupled_request_async_url>	<i>Chaîne</i> 256 caractères alphanumériques	<pre>mpiThreeDSAuthentication.setDecoupledRequestAsyncUrl("https://yourasyncnotificationurl.com");</pre> <p>Votre URL où Moneris va POSTer la réponse de l'ACS. Moneris tente à nouveau trois fois de POST la réponse.</p> <p>Conditionnel : Uniquement envoyé si decoupled-request-indicator = Y</p>
Demande préalable d'informations d'authentification <prior_request_auth_info>	<i>Objet</i> S.O.	<pre>mpiThreeDSAuthentication.setPriorRequestAuthInfo(pai);</pre> <p>Il s'agit d'un objet contenant les détails d'une authentification 3DS précédente pour cette série de transactions. L'objet est imbriqué dans la transaction d'authentification et est nécessaire pour stocker ou utiliser les informations relatives à l'authentification précédente pour cette carte. Pour plus d'informations sur les champs de l'objet Prior Authentication Info, voir la section Informations d'authentification MPI 3DS préalables.</p>

## Informations d'authentification MPI 3DS préalables

Variable	Type et limites	Description
Demande préalable de données d'authentification <prior_request_auth_data>	<i>Chaîne</i> 36 caractères alphanumériques	<pre>prior_request_auth_data="abc";</pre> <p>Fait référence au DSTransID dans la réponse de l'authentification 3DS précédente</p>
Référence à une demande préalable <prior_request_auth_ref>	<i>Chaîne</i> 36 caractères alphanumériques	<pre>prior_request_ref="d7c1ee99-9478-44a6-b1f2-391e29c6b340";</pre> <p>Fait référence à l'ID de transaction de l'ACS 3DS dans la réponse de l'authentification 3DS précédente</p>

Variable	Type et limites	Description
Méthode d'authentification de la demande préalable <prior_request_auth_method>	<i>Chaîne</i> 2 caractères numériques	prior_request_auth_method = "01";  Il s'agit d'un mécanisme utilisé par le titulaire de la carte pour s'authentifier lors de l'authentification 3DS précédente :  01 = Authentification sans friction  02 = Authentification par contestation  03 = Vérification par le SVA  04 = Autres méthodes de l'émetteur
Horodatage de la demande préalable d'autorisation <prior_request_auth_timestamp>	<i>Chaîne</i> 12 caractères numériques	prior_request_auth_timestamp = "201710282113";  Date et heure au fuseau horaire UTC auxquelles a eu lieu l'authentification 3DS du titulaire de carte, qui est trouvé dans la réponse d'authentification 3DS précédente en tant que 3DS Auth TimeStamp (horodatage de l'authentification 3DS)  Le format est AAAAMMMJJHHMM.

### Exemple d'une demande d'authentification MPI 3DS – Authentification 3RI sans processus périodique

```

package Canada;
import JavaAPI.*;
public class TestCanadaMpithreeDSAuthentication
{
    public static void main(String[] args)
    {
        String store_id = "moneris";
        String api_token = "hurgle";
        String prior_request_ref="d7c1ee99-9478-44a6-b1f2-391e29c6b340";
        String prior_request_auth_data="abc";
        String prior_request_auth_method = "01";
        String prior_request_auth_timestamp = "201710282113";
        String processing_country_code = "CA";
        MpithreeDSAuthentication mpiThreeDSAuthentication = new MpithreeDSAuthentication();
        mpiThreeDSAuthentication.setOrderId("Test159787ssa3215193"); //must be the same one that was used in MpicardLookup call
        mpiThreeDSAuthentication.setCardholderName("Moneris Test");
        mpiThreeDSAuthentication.setPan("347668693641199");
        // mpiThreeDSAuthentication.setDataKey("xRl904FgrZUYdGkmqHTqiEw97"); //Optional - For Moneris Vault and Hosted Tokenization tokens in place of setPan
        mpiThreeDSAuthentication.setExpdate("2310");
        mpiThreeDSAuthentication.setAmount("1.00");
        mpiThreeDSAuthentication.setThreeDSCompletionInd("Y"); // (Y|N|U) indicates whether 3ds method MpicardLookup was successfully completed
        mpiThreeDSAuthentication.setRequestType("01"); // (01=payment|02=recur)
        mpiThreeDSAuthentication.setPurchaseDate("20200819035249"); // (YYYYMMDDHHMMSS)
        mpiThreeDSAuthentication.setNotificationURL("https://yournotificationurl.com"); // (Website where response from RRes or CRes after challenge will go)
    }
}

```

```

mpiThreeDSAuthentication.setChallengeWindowSize("03"); // (01 = 250 x 400, 02 = 390 x 400, 03
= 500 x 600, 04 = 600 x 400, 05 = Full screen)
mpiThreeDSAuthentication.setEmail("test@email.com");
mpiThreeDSAuthentication.setBrowserUserAgent("Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.132 Safari/537.36\");
mpiThreeDSAuthentication.setBrowserJavaEnabled("true"); //(true|false)
mpiThreeDSAuthentication.setBrowserScreenHeight("1000"); //(pixel height of cardholder
screen)
mpiThreeDSAuthentication.setBrowserScreenWidth("1920"); //(pixel width of cardholder screen)
mpiThreeDSAuthentication.setBrowserLanguage("en-GB"); //(defined by IETF BCP47)

//Optional Methods
mpiThreeDSAuthentication.setBillAddress1("3300 Bloor St W");
mpiThreeDSAuthentication.setBillProvince("ON");
mpiThreeDSAuthentication.setBillCity("Toronto");
mpiThreeDSAuthentication.setBillPostalCode("M8X 2X2");
mpiThreeDSAuthentication.setBillCountry("124");

mpiThreeDSAuthentication.setShipAddress1("3300 Bloor St W");
mpiThreeDSAuthentication.setShipProvince("ON");
mpiThreeDSAuthentication.setShipCity("Toronto");
mpiThreeDSAuthentication.setShipPostalCode("M8X 2X2");
mpiThreeDSAuthentication.setShipCountry("124");

mpiThreeDSAuthentication.setRequestChallenge("Y"); //(Y|N Requesting challenge regardless of
outcome)
//*****3DS2.2*****
mpiThreeDSAuthentication.setMessageCategory("01");
mpiThreeDSAuthentication.setDeviceChannel("03");
// mpiThreeDSAuthentication.setDecoupledRequestIndicator("Y");
// mpiThreeDSAuthentication.setDecoupledRequestMaxTime("000010");
// mpiThreeDSAuthentication.setDecoupledRequestAsyncUrl("localhost:8080/googlepay");
mpiThreeDSAuthentication.setRiIndicator("01");
PriorAuthenticationInfo priorAuthenticationInfo= new PriorAuthenticationInfo();
priorAuthenticationInfo.setPriorParams(prior_request_auth_data,prior_request_ref,
prior_request_auth_method,prior_request_auth_timestamp);
// mpiThreeDSAuthentication.setRecurringExpiry("20221230");
// mpiThreeDSAuthentication.setRecurringFrequency("031");
//*****OPTIONAL VARIABLES*****
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false or comment out this line for production transactions
mpgReq.setstoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(mpiThreeDSAuthentication);
System.out.println(mpiThreeDSAuthentication.toXML());
mpgReq.send();
/************* REQUEST *****/
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("Message = " + receipt.getMessage());
System.out.println("MessageType = " + receipt.getMpiMessageType());
System.out.println("TransStatus = " + receipt.getMpiTransStatus());
System.out.println("TransStatusReason = " + receipt.getMpiTransStatusReason());
System.out.println("ChallengeURL = " + receipt.getMpiChallengeURL());
System.out.println("ChallengeData = " + receipt.getMpiChallengeData());
System.out.println("ThreeDSServerTransId = " + receipt.getMpiThreeDSserverTransId());
System.out.println("ThreeDSVersion = " + receipt.getMpiThreeDSVersion());
System.out.println("ThreeDSAcctsTransID = " + receipt.getMpiThreeDSAcctsTransID());
System.out.println("DSTransID = " + receipt.getMpiDSTransId());
System.out.println("ThreeDSAuthTimeStamp = " + receipt.getMpiThreeDSAuthTimeStamp());
System.out.println("AuthenticationType = " + receipt.getMpiAuthenticationType());
//In Frictionless flow, you may receive TransStatus as "Y",
//in which case you can then proceed directly to Cavv Purchase/Preatuh with values below
if(receipt.getMpiTransStatus().equals("Y"))
{
System.out.println("Cavv = " + receipt.getMpiCavv());
System.out.println("ECI = " + receipt.getMpiEci());
}
}
catch (Exception e)
{
e.printStackTrace();
}
}
} // end TestResMpiTxn

```

## 6.6 Traitement du flux de contestation

Si vous recevez une valeur TransStatus = « C » dans la réponse de threeDSAuthentication, un formulaire doit être créé et envoyé à l'URL fournie.

Le formulaire peut être produit dynamiquement avant d'être ajouté au DOM et soumis ou il peut être créé et soumis d'une manière adaptée à votre environnement. Il peut être construit comme une redirection de page complète ou présenté comme iFrame ou comme un lightbox.

Si vous souhaitez qu'il soit chargé dans un espace défini, il doit être conforme à la taille précisée dans la variable challengeWindowSize (taille de la fenêtre de contestation) de la requête. L'« action » est récupérée dans la variable ChallengeURL (URL de contestation) et le champ « **creq** » est récupéré dans la variable ChallengeData (données de contestation).

Vous trouverez ci-dessous un exemple de formulaire statique de base pour vous aider à visualiser les données et les champs qui doivent être soumis.

```
<form method="POST" action="https://3dsurl.example.com/do3DS">  
  
<input name="creq" value="thisissamplechallengedata1234567890">  
  
</form>
```

### 6.6.1 Demande de recherche de code de vérification d'authentification du titulaire de carte (CAVV) - mpiCavvLookup

(Flux de contestation uniquement)

Dans le flux de contestation, le serveur 3DS renvoie une valeur **cres** à l'URL de notification (notificationURL) fournie dans la demande d'authentification 3DS (threeDSAuthentication) une fois que le titulaire de la carte a terminé la contestation. La valeur « **cres** » est ensuite envoyée au serveur 3DS de Moneris dans la demande de recherche de code de vérification d'authentification du titulaire de carte (CavvLookup). La réponse à cette demande comprendra le résultat de la contestation, qui inclura l'indicateur CE (eic) et le code de vérification d'authentification du titulaire de carte (cavv) si la contestation est réussie.

#### Définition de l'objet de transaction Cavv Lookup Request

```
MpiCavvLookup mpiCavvLookup = new MpiCavvLookup();
```

#### Objet HttpsPostRequest pour les transactions de recherche de code de vérification d'authentification du titulaire de carte (CAVV)

```
HttpsPostRequest mpgReq = new HttpsPostRequest();  
  
mpgReq.setTransaction(mpiCavvLookup);
```

#### Principaux champs d'objet de connexion (toutes les transactions API)

Variable	Type et limites	
ID de commerce	<i>Chaîne</i> S. O.	mpgReq.setstoreId(store_id);

Jeton API	<i>Chaîne</i>  S. O.	mpgReq.setApiToken(api_token);
-----------	----------------------------	--------------------------------

### Champs de demande pour les transactions de demande recherche Cavv (obligatoires)

Variable	Type et limites	
cres	<i>Chaîne</i>  200 caractères alphanumériques	mpiCavvLookup.setCRes(cres);

### Exemple de demande de recherche de code de vérification d'authentification du titulaire de carte (CAVV)

```

package Canada;
import JavaAPI.*;
public class TestCanadaMpiCavvLookup
{
    public static void main(String[] args)
    {
        String store_id = "moneris";
        String api_token = "hurgle";
        String processing_country_code = "CA";

        //BASE64 Encoded CRes value returned from response at completion of challenge flow.
        String cres =
"eyJhY3NUcmFuc0lEIjoiNzQ0ZDI2NjUtNjU2Yy00ZGNiLTg3MWUtYTBkYmMwODA0OTYzIiwbWVzc2FnZVR5cGUiOiJD
UmVzIiwiY2hhbGxlbfdlQ29tcGxldGlvbkluZCI6IlkiLCJtZXNzYWdlVmVyc2lvbiI6IjIuMS4wIiwidHJhbnNTdGF0d
XMiOiJZIiwidGhyZWVEU1NlcnZlclRyYW5zSUQiOjLMTFkNDk4NS04ZDI1LTQwZWQtOTlkNl1jMzgwM2ZlNWU2OGYifQ
==";

        MpiCavvLookup mpiCavvLookup = new MpiCavvLookup();
        mpiCavvLookup.setCRes(cres);
        //*****OPTIONAL VARIABLES*****
        HttpsPostRequest mpgReq = new HttpsPostRequest();
        mpgReq.setProcCountryCode(processing_country_code);
        mpgReq.setTestMode(true); //false or comment out this line for production transactions
        mpgReq.setStoreId(store_id);
        mpgReq.setApiToken(api_token);
        mpgReq.setTransaction(mpiCavvLookup);
        mpgReq.send();
        //***** REQUEST *****/
        try
        {
            Receipt receipt = mpgReq.getReceipt();
            System.out.println("ResponseCode = " + receipt.getResponseCode());
            System.out.println("ReceiptId = " + receipt.getReceiptId());
            System.out.println("Message = " + receipt.getMessage());
            System.out.println("MessageType = " + receipt.getMpiMessageType());
            System.out.println("ThreeDSServerTransId = " + receipt.getMpiThreeDSServerTransId());
            System.out.println("TransStatus = " + receipt.getMpiTransStatus());
            System.out.println("ChallengeCompletionIndicator = " +
receipt.getMpiChallengeCompletionIndicator());
            System.out.println("Cavv = " + receipt.getMpiCavv());
            System.out.println("ECI = " + receipt.getMpiEci());
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
} // end TestResMpiTxn

```

## 6.7 Gestion du processus d'authentification découpée

Si vous obtenez un TransStatus = "D" dans votre réponse d'authentification threeDSAuthentication, votre serveur doit être prêt à accepter une deuxième réponse asynchrone de Moneris.

Le titulaire de la carte sera demandé par son émetteur d'authentifier le titulaire de la carte en dehors du protocole 3DS. L'authentification peut exiger l'utilisation d'autres applications d'authentification ou l'envoi d'un message SMS invitant le titulaire de la carte à confirmer son identité.

Le titulaire de la carte a jusqu'à 7 jours pour relever cette contestation découpée. Une fois l'opération terminée, l'émetteur communique avec Moneris et notre système MPI envoie une deuxième réponse d'authentification 3DS à l'adresse que vous avez définie dans le champ  
<decoupled\_request\_async\_url>.

### Exemple d'une demande d'authentification - 3RI avec l'authentification découpée pour les commandes postales ou téléphoniques

```
package Canada;
import JavaAPI.*;
public class TestCanadaMpithreeDSAuthentication
{
    public static void main(String[] args)
    {
        String store_id = "moneris";
        String api_token = "hurgle";
        String prior_request_ref="d7c1ee99-9478-44a6-b1f2-391e29c6b340";
        String prior_request_auth_data="abc";
        String prior_request_auth_method = "01";
        String prior_request_auth_timestamp = "201710282113";
        String processing_country_code = "CA";
        MpithreeDSAuthentication mpiThreeDSAuthentication = new MpithreeDSAuthentication();
        mpiThreeDSAuthentication.setOrderId("Test159787ssa3215193"); //must be the same one that
        was used in MpicardLookup call
        mpiThreeDSAuthentication.setCardholderName("Moneris Test");
        mpiThreeDSAuthentication.setPan("347668693641199");
        // mpiThreeDSAuthentication.setDataKey("xR1904FgrZUYdGkmqHTqiEw97"); //Optional - For
        Moneris Vault and Hosted Tokenization tokens in place of setPan
        mpiThreeDSAuthentication.setExpdate("2310");
        mpiThreeDSAuthentication.setAmount("1.00");
        mpiThreeDSAuthentication.setThreeDSCompletionInd("Y"); // (Y|N|U) indicates whether 3ds
        method MpicardLookup was successfully completed
        mpiThreeDSAuthentication.setRequestType("01"); //(01=payment|02=recur)
        mpiThreeDSAuthentication.setPurchaseDate("20200819035249"); //(YYYYMMDDHHMMSS)
        mpiThreeDSAuthentication.setNotificationURL("https://yournotificationurl.com"); // (Website
        where response from RRes or CRes after challenge will go)
        mpiThreeDSAuthentication.setChallengeWindowSize("03"); //(01 = 250 x 400, 02 = 390 x 400,
        03 = 500 x 600, 04 = 600 x 400, 05 = Full screen)
        mpiThreeDSAuthentication.setEmail("test@email.com");
        mpiThreeDSAuthentication.setBrowserUserAgent("Mozilla/5.0 (Windows NT 10.0; Win64; x64)
        AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.132 Safari/537.36\\");
        mpiThreeDSAuthentication.setBrowserJavaEnabled("true"); //(true|false)
        mpiThreeDSAuthentication.setBrowserScreenHeight("1000"); //(pixel height of cardholder
        screen)
        mpiThreeDSAuthentication.setBrowserScreenWidth("1920"); //(pixel width of cardholder
        screen)
        mpiThreeDSAuthentication.setBrowserLanguage("en-GB"); //(defined by IETF BCP47)

        //Optional Methods
        mpiThreeDSAuthentication.setBillAddress1("3300 Bloor St W");
        mpiThreeDSAuthentication.setBillProvince("ON");
        mpiThreeDSAuthentication.setBillCity("Toronto");
        mpiThreeDSAuthentication.setBillPostalCode("M8X 2X2");
        mpiThreeDSAuthentication.setBillCountry("124");

        mpiThreeDSAuthentication.setShipAddress1("3300 Bloor St W");
        mpiThreeDSAuthentication.setShipProvince("ON");
        mpiThreeDSAuthentication.setShipCity("Toronto");
        mpiThreeDSAuthentication.setShipPostalCode("M8X 2X2");
        mpiThreeDSAuthentication.setShipCountry("124");

        mpiThreeDSAuthentication.setRequestChallenge("Y"); // (Y|N Requesting challenge regardless
        of outcome)
        //*****3DS2.2*****
```

```

mpiThreeDSAuthentication.setMessageCategory("01");
mpiThreeDSAuthentication.setDeviceChannel("03");
mpiThreeDSAuthentication.setDecoupledRequestIndicator("Y");
mpiThreeDSAuthentication.setDecoupledRequestMaxTime("000010");
mpiThreeDSAuthentication.setDecoupledRequestAsyncUrl("localhost:8080/googlepay");
mpiThreeDSAuthentication.setRiIndicator("08");
// PriorAuthenticationInfo priorAuthenticationInfo= new PriorAuthenticationInfo();
// priorAuthenticationInfo.setPriorParams(prior_request_auth_data,prior_request_ref,
// prior_request_auth_method,prior_request_auth_timestamp);
// mpiThreeDSAuthentication.setRecurringExpiry("20221230");
// mpiThreeDSAuthentication.setRecurringFrequency("031");
//*****OPTIONAL VARIABLES*****
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false or comment out this line for production transactions
mpgReq.setStoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(mpiThreeDSAuthentication);
System.out.println(mpiThreeDSAuthentication.toXML());
mpgReq.send();
/************* REQUEST *****/
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("Message = " + receipt.getMessage());
System.out.println("MessageType = " + receipt.getMpiMessageType());
System.out.println("TransStatus = " + receipt.getMpiTransStatus());
System.out.println("TransStatusReason = " + receipt.getMpiTransStatusReason());
System.out.println("ChallengeURL = " + receipt.getMpiChallengeURL());
System.out.println("ChallengeData = " + receipt.getMpiChallengeData());
System.out.println("ThreeDServerTransId = " + receipt.getMpiThreeDServerTransId());
System.out.println("ThreeDSVersion = " + receipt.getThreeDSVersion());
System.out.println("ThreeDSAcctsTransID = " + receipt.getMpiThreeDSAcctsTransID());
System.out.println("DSTransID = " + receipt.getMpiDSTransId());
System.out.println("ThreeDSAuthTimeStamp = " + receipt.getMpiThreeDSAuthTimeStamp());
System.out.println("AuthenticationType = " + receipt.getMpiAuthenticationType());
//In Frictionless flow, you may receive TransStatus as "Y",
//in which case you can then proceed directly to Cavv Purchase/Preauth with values below
if(receipt.getMpiTransStatus().equals("Y"))
{
System.out.println("Cavv = " + receipt.getMpiCavv());
System.out.println("ECI = " + receipt.getMpiEci());
}
}
catch (Exception e)
{
e.printStackTrace();
}
}
} // end TestResMpiTxn

```

## Exemple d'une demande d'authentification - 3RI avec l'authentification préalable et l'authentification découpée

```

package Canada;
import JavaAPI.*;
public class TestCanadaMpiThreeDSAuthentication
{
public static void main(String[] args)
{
String store_id = "moneris";
String api_token = "hurgle";
String prior_request_ref="d7c1ee99-9478-44a6-b1f2-391e29c6b340";
String prior_request_auth_data="abc";
String prior_request_auth_method = "01";
String prior_request_auth_timestamp = "201710282113";
String processing_country_code = "CA";
MpithreeDSAuthentication mpiThreeDSAuthentication = new
MpithreeDSAuthentication();
mpiThreeDSAuthentication.setOrderId("Test159787ssa3215193"); //must be the same
one that was used in MpiCardLookup call
mpiThreeDSAuthentication.setCardholderName("Moneris Test");
mpiThreeDSAuthentication.setPan("347668693641199");
// mpiThreeDSAuthentication.setDataKey("xRl904FgrZUYdGkmqHTqiEw97"); //Optional -
For Moneris Vault and Hosted Tokenization tokens in place of setPan
mpiThreeDSAuthentication.setExppdate("2310");
mpiThreeDSAuthentication.setAmount("1.00");

```

```

mpiThreeDSAuthentication.setThreeDSCompletionInd("Y"); // (Y|N|U) indicates
whether 3ds method MpiCardLookup was successfully completed
mpiThreeDSAuthentication.setRequestType("01"); // (01=payment|02=recur)
mpiThreeDSAuthentication.setPurchaseDate("20200819035249"); //(YYYYMMDDHHMMSS)
mpiThreeDSAuthentication.setNotificationURL("https://yournotificationurl.com");
//(Website where response from RRes or CRes after challenge will go)
mpiThreeDSAuthentication.setChallengeWindowSize("03"); //(01 = 250 x 400, 02 =
390 x 400, 03 = 500 x 600, 04 = 600 x 400, 05 = Full screen)
mpiThreeDSAuthentication.setEmail("test@email.com");
mpiThreeDSAuthentication.setBrowserUserAgent("Mozilla/5.0 (Windows NT 10.0;
Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.132
Safari/537.36\\");
mpiThreeDSAuthentication.setBrowserJavaEnabled("true"); //(true|false)
mpiThreeDSAuthentication.setBrowserScreenHeight("1000"); //(pixel height of
cardholder screen)
mpiThreeDSAuthentication.setBrowserScreenWidth("1920"); //(pixel width of
cardholder screen)
mpiThreeDSAuthentication.setBrowserLanguage("en-GB"); //(defined by IETF BCP47)

//Optional Methods
mpiThreeDSAuthentication.setBillAddress1("3300 Bloor St W");
mpiThreeDSAuthentication.setBillProvince("ON");
mpiThreeDSAuthentication.setBillCity("Toronto");
mpiThreeDSAuthentication.setBillPostalCode("M8X 2X2");
mpiThreeDSAuthentication.setBillCountry("124");

mpiThreeDSAuthentication.setShipAddress1("3300 Bloor St W");
mpiThreeDSAuthentication.setShipProvince("ON");
mpiThreeDSAuthentication.setShipCity("Toronto");
mpiThreeDSAuthentication.setShipPostalCode("M8X 2X2");
mpiThreeDSAuthentication.setShipCountry("124");

mpiThreeDSAuthentication.setRequestChallenge("Y"); //(Y|N Requesting challenge
regardless of outcome)
//*****3DS2.2*****
mpiThreeDSAuthentication.setMessageCategory("01");
mpiThreeDSAuthentication.setDeviceChannel("03");
mpiThreeDSAuthentication.setDecoupledRequestIndicator("Y");
mpiThreeDSAuthentication.setDecoupledRequestMaxTime("000010");
mpiThreeDSAuthentication.setDecoupledRequestAsyncUrl("localhost:8080/googlepay");
mpiThreeDSAuthentication.setRiIndicator("08");
PriorAuthenticationInfo priorAuthenticationInfo= new PriorAuthenticationInfo();
priorAuthenticationInfo.setPriorParams(prior_request_auth_data,prior_request_ref,
prior_request_auth_method,prior_request_auth_timestamp);
// mpiThreeDSAuthentication.setRecurringExpiry("20221230");
// mpiThreeDSAuthentication.setRecurringFrequency("031");
//*****OPTIONAL VARIABLES*****
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false or comment out this line for production
transactions
mpgReq.setstoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(mpiThreeDSAuthentication);
System.out.println(mpiThreeDSAuthentication.toXML());
mpgReq.send();
***** REQUEST *****/
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("Message = " + receipt.getMessage());
System.out.println("MessageType = " + receipt.getMpiMessageType());
System.out.println("TransStatus = " + receipt.getMpiTransStatus());
System.out.println("TransStatusReason = " + receipt.getMpiTransStatusReason());
System.out.println("ChallengeURL = " + receipt.getMpiChallengeURL());
System.out.println("ChallengeData = " + receipt.getMpiChallengeData());
System.out.println("ThreeDServerTransId = " +
receipt.getMpiThreeDServerTransId());
System.out.println("ThreeDSVersion = " + receipt.getThreeDSVersion());
System.out.println("ThreeDSAcctsTransID = " + receipt.getMpiThreeDSAcctsTransID());
System.out.println("DSTransID = " + receipt.getMpiDSTransId());
System.out.println("ThreeDSAAuthTimeStamp = " +
receipt.getMpiThreeDSAAuthTimeStamp());
System.out.println("AuthenticationType = " + receipt.getMpiAuthenticationType());
//In Frictionless flow, you may receive TransStatus as "Y",
//in which case you can then proceed directly to Cavv Purchase/Preauth with values below
if(receipt.getMpiTransStatus().equals("Y"))
{
System.out.println("Cavv = " + receipt.getMpiCavv());
}
}

```

```
System.out.println("ECI = " + receipt.getMpiEci());
}
}
catch (Exception e)
{
e.printStackTrace();
}
}
} // end TestResMpiTxn
```

## 6.8 Effectuer l'autorisation

Une fois l'authentification terminée et les valeurs CAVV et ECI récupérées, ces valeurs peuvent être envoyées à Moneris à laide des transactions suivantes : Purchase with 3-D Secure – cavv\_Purchase ou Pre-Authorization with 3-D Secure – cavv\_Preauth.

### 6.8.1 Achat avec la solution 3-D Secure (cavv\_Purchase)

Une transaction d'achat avec 3-D Secure est effectuée après une authentification 3-D Secure des modules d'extension pour les commerçants. Après avoir reçu la confirmation de la transaction ACS des modules d'extension pour les commerçants, cet achat vérifie que les fonds requis sont présents sur la carte du client, retire ces fonds de la carte et les prépare à être déposés dans le compte du commerçant.

Pour effectuer l'authentification 3-D Secure, le module d'extension de Moneris pour les commerçants ou tout autre module d'extension tiers peut être utilisé.

En plus des transactions 3-D Secure, cette transaction peut également être utilisée pour traiter les transactions Apple Pay et Google Pay<sup>MC</sup>. Cette transaction s'applique uniquement si vous choisissez de l'intégrer directement à Apple Wallet ou à Google Wallet (si vous n'utilisez pas les trousses SDK Apple Pay ou Google Pay<sup>MC</sup> de Moneris).

Référez-vous aux portails pour développeurs d'Apple ou de Google pour en savoir plus sur l'intégration directe à leurs portefeuilles et recueillir les données utiles.

**AVERTISSEMENT :** Moneris déconseille fortement l'utilisation de cadres pour l'intégration de la solution 3-D Secure et ne peut garantir leur fiabilité lors du traitement des transactions dans l'environnement de production.

```
CavvPurchase cavv_purchase = new CavvPurchase();
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setTransaction(cavv_purchase);
```

## Principaux champs d'objet de connexion (toutes les transactions API)

Variable	Type et limites	
ID de commerce	<i>Chaîne</i> S. O.	mpgReq.setstoreId(store_id);
Jeton API	<i>Chaîne</i> S. O.	mpgReq.setApiToken(api_token);

## Champs de demande liés aux transactions de d'achat avec la solution 3-D Secure (obligatoires)

Variable	Type et limites	
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques a-Z A-Z 0-9 _ - : . @ espaces	cavv_purchase.setOrderId(order_id);
Montant	<i>Chaîne</i> 10 caractères décimaux Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal  <b>EXAMPLE : 1 234 567,89</b>	cavv_purchase.setAmount(amount);
Numéro de carte de crédit	<i>Chaîne</i> Maximum 20 caractères alphanumériques	cavvPurchase.setPan(pan);
Date d'expiration	<i>Chaîne</i> 4 caractères alphanumériques AAMM	cavvPurchase.setExpDate(expiry_date);
Code de vérification d'authentification du titulaire de carte	<i>Chaîne</i> 50 caractères alphanumériques	cavvPurchase.setCavv(cavv);

<b>REMARQUE :</b> Pour les transactions d'achat et de préautorisation Apple Pay et Google Pay <sup>MC</sup> utilisant le code de vérification d'authentification du titulaire de carte, le champ CAVV contient le cryptogramme déchiffré. Pour plus de renseignements, consultez l'annexe A Définition des champs de demande.		
Indicateur de commerce électronique  <b>REMARQUE :</b> Pour les transactions d'achat et de préautorisation Apple Pay et Google Pay <sup>MC</sup> utilisant le code de vérification d'authentification du titulaire de carte, l'indicateur de commerce électronique est un champ obligatoire contenant la valeur reçue des données déchiffrées ou une valeur par défaut de 5. Si vous obtenez une valeur à deux caractères (par exemple, 05 ou 07) à partir des données, supprimez le 0 initial et envoyez-nous simplement le deuxième caractère. Pour plus de renseignements, consultez l'annexe A Définition des champs de demande.	<p><i>Chaîne</i> 1 caractère alphanumérique</p>	cavvPurchase.setCryptType(crypt);

### 3-D Secure 2.2 – Champs particuliers (obligatoires)

Variable	Type et limites	
Version de 3DS  <b>REMARQUE :</b> Obligatoire pour les transactions utilisant les services de 3-D Secure d'un tiers	<p><i>Chaîne</i> 10 caractères numériques</p>	<pre>cavv_purchase.setThreeDSVersion("ThreeDSVersion");</pre> <p>Valeurs acceptées :</p> <ul style="list-style-type: none"> <li>2.0.0 = protocole 3DS 2.0.0</li> <li>2.1.0 = protocole 3DS 2.1.0</li> <li>2.2.0 = protocole 3DS 2.2.0</li> <li>2.3.0 = protocole 3DS 2.3.0</li> </ul>
ID de transaction du serveur 3DS	<p><i>Chaîne</i> 36 caractère numérique</p>	<pre>cavv_purchase.setThreeDSServerTransId("ThreeDSServerTransId");</pre>

**REMARQUE :** Obligatoire pour les transactions utilisant les services de 3-D Secure d'un tiers, obtenus à partir de la demande de recherche de code de vérification d'authentification du titulaire de carte ou de la demande d'authentification MPI 3DS

Les champs suivants sont requis pour Apple Pay et Google Pay uniquement :

Variable	Type et limites	
Réseau	<i>Chaîne</i> Caractère alphabétique	<code>cavv_purchase.setNetwork(network);</code>
Type de données	<i>Chaîne</i> 3 caractères alphanumériques	<code>cavv_purchase.setDataType(data_type);</code>

#### Champs de demande liés aux transactions d'achat avec 3-D Secure (facultatifs)

Variable	Type et limites	
ID de client	<i>Chaîne</i> 50 caractères alphanumériques <div style="background-color: #e0f2e0; padding: 5px;"> <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : &lt; &gt; \$ % = ? ^ { } [ ] \          </div>	<code>cavv_purchase.setCustId(cust_id);</code>
Descripteur dynamique	<i>Chaîne</i> 20 caractères alphanumériques Total de 22 caractères incluant votre nom de commerçant et un séparateur <div style="background-color: #e0f2e0; padding: 5px;"> <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : &lt; &gt; \$ % = ? ^ { } [ ] \         </div>	<code>cavv_purchase.setDynamicDescriptor(dynamic_descriptor);</code>
Indicateur étranger	<i>Boolean</i> true or false	<code>cavvPurchase.setForeignIndicator(foreign_indicator);</code>
ID de correspondance de carte	<i>Chaîne</i> 50 caractères	<code>cavv_purchase.setCmId(transaction_id);</code>

<b>REMARQUE :</b> Applicables à Offlinx <sup>MC</sup> seulement, chaque transaction doit avoir une valeur unique	alphanumériques	
Renseignements du client	<i>Objet</i> S. O.	cavv_purchase.setCustInfo(custome r);
Renseignements du SVA	<i>Objet</i> S. O.	cavv_purchase.setAvsInfo(avscCheck) ;
Renseignements du NVC	<i>Objet</i> S. O.	cavv_purchase.setCvdInfo(cvdCheck) ;
<b>REMARQUE :</b> Lorsque les renseignements d'identification sont stockés lors de la première transaction, l'objet CVD (numéro de vérification de carte, ou NVC) doit être envoyé. Pour les transactions subséquentes utilisant les renseignements d'identification stockés, le NVC peut être envoyé avec les transactions entamées par le titulaire de carte seulement. <b>Les commerçants ne doivent pas enregistrer le NVC.</b>		
Information sur les frais de commodité	<i>Objet</i> S. O.	ConvFeeInfo convFeeInfo = new ConvFeeInfo();  cavv_purchase.setConvenienceFee(co nvFeeInfo);
<b>REMARQUE :</b> Ne s'applique pas lors du traitement de transactions Apple Pay ou Google Pay.		
Facturation périodique  recur	<i>Objet</i> S. O.	cavv_purchase.setRecurInfo(recurInfo);
<b>REMARQUE :</b> Pour un exemple de code concernant un achat avec la solution 3-D Secure incluant l'objet Recurring Billing Info, consultez la section 6.8.5 Achat avec la solution 3-D Secure et facturation périodique		
Indicateur de portefeuille électronique	<i>Chaîne</i> 3 caractères alphanumériques	cavv_purchase.setWalletIndicator(wallet_indicator);

<b>REMARQUE :</b> Pour les achats et la préautorisation utilisant le code de vérification d'authentification du titulaire de carte, l'indicateur de portefeuille s'applique uniquement à Apple Pay ou Google PayMC. Pour plus d'information, consultez l'annexe A Définition des champs de demande.		
Renseignements d'identification au dossier  cof	<i>Objet</i>  S. O.	cavv_purchase.setCofInfo(cof);

### Champs de demande pour l'objet Credential on File Info

Variable	Type et limites	
ID de l'émetteur	<i>Chaîne</i>  15 caractères alphanumériques  Longueur variable	cof.setIssuerId("VALUE_FOR_IS SUER_ID");
<b>REMARQUE :</b> Cette variable est requise pour chaque transaction entamée par le commerçant à la suite de la première transaction. Après l'envoi de la première transaction, l'ID d'émetteur (issuer ID) est reçu dans la réponse de transaction, puis utilisé dans les demandes de transaction subséquentes.		<b>REMARQUE :</b> Pour obtenir une liste et une explication des valeurs possibles à envoyer pour cette variable, consultez la section Définition des champs de demande – Renseignements d'identification au dossier (cof)
Indicateur de paiement	<i>Chaîne</i>  1 caractère alphabétique	cof.setPaymentIndicator("PAYM ENT_INDICATOR_VALUE");
<b>REMARQUE :</b> Lors d'une transaction utilisant les renseignements d'identification au dossier pour laquelle le champ de demande e-commerce indicator est également inclus, les valeurs		<b>REMARQUE :</b> Pour obtenir une liste et une explication des valeurs possibles à envoyer pour cette variable, consultez la section Définition des champs de demande – Renseignements d'identification au dossier (cof)

acceptables de ce champ dépendent de la valeur envoyée pour l'indicateur de paiement. Pour plus d'information, consultez les définitions des champs de réponse.		
Information de paiement	<p><i>Chaîne</i></p> <p>1 caractère numérique</p>	<pre>cof.setPaymentInformation("PAYMENT_INFO_VALUE");</pre> <p><b>REMARQUE :</b> Pour obtenir une liste et une explication des valeurs possibles à envoyer pour cette variable, consultez la section Définition des champs de demande – Renseignements d'identification au dossier (cof)</p>

### 3-D Secure 2.2 – Champs particuliers (facultatifs)

Variable	Type et limites	
ID de transaction DS  <b>REMARQUE :</b> Uniquement utilisé dans les transactions financières utilisant les services 3-D Secure d'un tiers	<p><i>Chaîne</i></p> <p>36 caractères alphanumériques</p>	cavvPurchase.setDsTransId("DsTransId");

### Champs de demande pour l'objet Recurring Billing Info

Variable	Type et limites	Description
Nombre d'occurrences	<p><i>Chaîne</i></p> <p>Valeur numérique</p> <p>1 ou 999</p>	Il s'agit du nombre d'occurrences de la transaction.
Période	<p><i>Chaîne</i></p> <p>Valeur numérique</p> <p>1 ou 999</p>	Il s'agit du nombre d'intervalles de la variable recur unit qui doivent s'écouler entre chaque facturation périodique.
Date de début	<p><i>Chaîne</i></p> <p>Format AAAAMMMJJ</p>	<p>Il s'agit de la date de la première transaction périodique future (la date doit être future).</p> <p>Si un montant additionnel est facturé immédiatement, la variable start now doit être réglée à true.</p>

Commencer maintenant	<i>Chaîne</i>	Réglez cette variable à true si un montant est porté immédiatement à la carte. Autrement, réglez la variable à false.
		Lorsque la variable est réglée à false, effectuez une transaction de vérification de carte avant d'envoyer l'achat avec les objets Recurring Billing et Credential on File.
		<b>REMARQUE :</b> Le montant à facturer immédiatement peut différer des montants subséquents.
Montant récurrent	<i>Chaîne</i> 10 caractères décimaux, minimum de 3 chiffres Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal <b>EXEMPLE : 1 234 567,89</b>	Il s'agit du montant en dollars de la transaction périodique.  Il s'agit du montant facturé à la date de départ (start_date) et qui sera ensuite facturé à répétition en fonction de l'intervalle défini par les valeurs periodet recur unit.
Unité répétée	<i>Chaîne</i> Jour, semaine, mois ou fin du mois	Il s'agit de l'unité utilisée comme base pour l'intervalle.  Elle fonctionne avec la variable period pour déterminer la fréquence de facturation.

### Exemple d'achat avec la solution 3-D Secure – cavvPurchase

```

package Canada;
import JavaAPI.*;
public class TestCanadaCavvPurchase
{
  public static void main(String[] args)
  {
    String store_id = "store5";
    String api_token = "yesguy";
    java.util.Date createDate = new java.util.Date();
    String order_id = "Test"+createDate.getTime();
    String cust_id = "CUS887H67";
    String amount = "10.42";
    String pan = "4740611374762707";
    String expdate = "1901"; //YYMM
    String cavv = "BwABA�FSYyd412eQQFJjAAAAAAA=";
    String dynamic_descriptor = "123456";
    String processing_country_code = "CA";
    String crypt_type = "5";
    boolean status_check = false;
    boolean foreign_indicator= true; //New Foreign Indicator field

    CavvPurchase cavvPurchase = new CavvPurchase();
    cavvPurchase.setOrderId(order_id);
    cavvPurchase.setCustomerId(cust_id);
    cavvPurchase.setAmount(amount);
    cavvPurchase.setPan(pan);
    cavvPurchase.setExpdate(expdate);
    cavvPurchase.setCavv(cavv);
    cavvPurchase.setCryptType(crypt_type); //Mandatory for AMEX only
  }
}

```

```

cavvPurchase.setDynamicDescriptor(dynamic_descriptor);
//cavvPurchase.setWalletIndicator("APP"); //set only for wallet transactions. e.g APPLE PAY
//cavvPurchase.setNetwork("Interac"); //set only for Interac e-commerce
//cavvPurchase.setDataType("3DSecure"); //set only for Interac e-commerce
//cavvPurchase.setCmId("8nAK8712sGaAkls56"); //set only for usage with Offlinx - Unique max
50 alphanumeric characters transaction id generated by merchant

cavvPurchase.setThreeDSVersion("2.2.0"); //Mandatory for financial transactions using 3rd
Party 3-D Secure services.
cavvPurchase.setThreeDServerTransId("e11d4985-8d25-40ed-99d6-c3803fe5e68f"); //Mandatory for
financial transactions using 3rd Party 3-D Secure services - obtained from MpICavvLookup or
MpThreeDSAuthentication
//cavvPurchase.setDsTransId("12345");//Optional - to be used only if you are using 3rd party
3ds service
cavvPurchase.setForeignIndicator(foreign_indicator);

// TrId and TokenCryptogram are optional, refer documentation for more details.
cavvPurchase.setTrId("50189815682");
cavvPurchase.setTokenCryptogram("APmbM/411e0uAAH+s6xMAAADFA==");

//optional - Credential on File details
CofInfo cof = new CofInfo();
cof.setPaymentIndicator("U");
cof.setPaymentInformation("2");
cof.setIssuerId("139X3130ASCXAS9");

cavvPurchase.setCofInfo(cof);
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false for production transactions
mpgReq.setStoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(cavvPurchase);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = " + receipt.getBankTotals());
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("CavvResultCode = " + receipt.getCavvResultCode());
System.out.println("IssuerId = " + receipt.getIssuerId());
System.out.println("ThreeDSVersion = " + receipt.getThreeDSVersion());
System.out.println("SourcePanLast4 = " + receipt.getSourcePanLast4());
System.out.println("Advice Code= "+ receipt.getAdviceCode());
}
catch (Exception e)
{
e.printStackTrace();
}
}
}
}

```

## 6.8.2 Préautorisation avec la solution 3-D Secure (cavv\_Preach)

La transaction de préautorisation avec la solution 3-D Secure est effectuée après une authentification 3-D Secure des modules d'extension pour les commerçants. Après avoir reçu la confirmation de la transaction de demande ACS des modules d'extension pour les commerçants, cette préautorisation vérifie que les fonds requis sont présents sur la carte du client, retire ces fonds de la carte et les prépare à être déposés dans le compte du commerçant.

Pour effectuer l'authentification 3-D Secure, le module d'extension de Moneris pour les commerçants ou tout autre module d'extension tiers peut être utilisé.

En plus des transactions 3-D Secure, cette transaction peut également être utilisée pour traiter les transactions Apple Pay et Google Pay<sup>MC</sup>. Cette transaction s'applique uniquement si vous choisissez de l'intégrer directement à Apple Wallet ou à Google Wallet (si vous n'utilisez pas les trousseuses SDK Apple Pay ou Google Pay<sup>MC</sup> de Moneris).

Référez-vous aux portails pour développeurs d'Apple ou de Google pour en savoir plus sur l'intégration directe à leurs portefeuilles et recueillir les données utiles.

API de Passerelle Moneris – Guide d'intégration

**AVERTISSEMENT :** Moneris déconseille fortement l'utilisation de cadres pour l'intégration de la solution 3-D Secure et ne peut garantir leur fiabilité lors du traitement des transactions dans l'environnement de production.

```
CavvPreAuth cavv_preatuth = new CavvPreAuth();  
HttpsPostRequest mpgReq = new HttpsPostRequest();  
mpgReq.setTransaction(cavv_preatuth);
```

#### Principaux champs d'objet de connexion (toutes les transactions API)

Variable	Type et limites	
ID de commerce	<i>Chaîne</i> S. O.	mpgReq.setstoreId(store_id);
Jeton API	<i>Chaîne</i> S. O.	mpgReq.setApiToken(api_token);

## Champs de demande pour les transactions de préautorisation avec la solution 3-D Secure (obligatoires)

Variable	Type et limites	
ID de commande	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p> <p>a-Z A-Z 0-9 _ - : . @ espaces</p>	cavv_preatuh.setOrderId(order_id);
Montant	<p><i>Chaîne</i></p> <p>10 caractères décimaux</p> <p>Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal</p>	cavv_preatuh.setAmount(amount);
	<b>EXEMPLE : 1 234 567,89</b>	
Numéro de carte de crédit	<p><i>Chaîne</i></p> <p>Maximum 20 caractères alphanumériques</p>	cavv_preatuh.setPan(pan);
Code de vérification d'authentification du titulaire de carte	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p>	cavv_preatuh.setCavv(cavv);
<b>REMARQUE :</b> Pour les transactions d'achat et de préautorisation Apple Pay et Google Pay <sup>MC</sup> utilisant le code de vérification d'authentification du titulaire de carte, le champ CAVV contient le cryptogramme déchiffré. Pour plus de renseignements, consultez l'annexe A Définition des champs de demande.		
Date d'expiration	<p><i>Chaîne</i></p> <p>4 caractères alphanumériques</p> <p>AAMM</p>	cavv_preatuh.setExpDate(expiry_date);
Indicateur de commerce électronique	<p><i>Chaîne</i></p> <p>1 caractère alphanumérique</p>	cavv_preatuh.setCryptType(crypt);

**REMARQUE :** Pour les transactions d'achat et de préautorisation Apple Pay et Google Pay<sup>MC</sup> utilisant le code de vérification d'authentification du titulaire de carte, l'indicateur de commerce électronique est un champ obligatoire contenant la valeur reçue des données déchiffrées ou une valeur par défaut de 5. Si vous obtenez une valeur à deux caractères (par exemple, 05 ou 07) à partir des données, supprimez le 0 initial et envoyez-nous simplement le deuxième caractère. Pour plus de renseignements, consultez l'annexe A Définition des champs de demande.

### 3-D Secure 2.2 – Champs particuliers (obligatoires)

<b>Version de 3DS</b>  <b>REMARQUE :</b> Obligatoire pour les transactions utilisant les services de 3-D Secure d'un tiers	<i>Chaîne</i> 10 caractères numériques	<pre>cavv_preauth.setThreeDSVersion("ThreeDSVersion");</pre> <p>Valeurs acceptées :</p> <ul style="list-style-type: none"> <li>2.0.0 = protocole 3DS 2.0.0</li> <li>2.1.0 = protocole 3DS 2.1.0</li> <li>2.2.0 = protocole 3DS 2.2.0</li> <li>2.3.0 = protocole 3DS 2.3.0</li> </ul>
<b>ID de transaction du serveur 3DS</b>  <b>REMARQUE :</b> Obligatoire pour les transactions utilisant les services de 3-D Secure d'un tiers, obtenue à partir de la demande de recherche code de vérification d'authentification du titulaire de carte ou de la demande d'authentification 3DS des modules d'extension pour les commerçants.	<i>Chaîne</i> 36 caractère numérique	<pre>cavv_preauth.setThreeDSServerTransId("ThreeDSServerTransId");</pre>

## Champs de demande pour les transactions de préautorisation avec la solution 3-D Secure ( facultatifs )

Variable	Type et limites	
Vérification d'état	<i>Valeur booléenne</i> true/false	mpgReq.setStatusCheck(status_check);
ID de client	<i>Chaîne</i> 50 caractères alphanumériques  <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\	cavv_preatuh.setCustId(cust_id);
Descripteur dynamique	<i>Chaîne</i> 20 caractères alphanumériques Total de 22 caractères incluant votre nom de commerçant et un séparateur  <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\	cavv_preatuh.setDynamicDescriptor(dynamic_descriptor);
Pour les transactions de préautorisation <b>REMARQUE :</b> La valeur du champ Dynamic descriptor est uniquement transférée à une conclusion de préautorisation lorsque cette dernière est effectuée par l'entremise du centre de ressources pour commerçants.  Autrement, la valeur du champ doit être renvoyée durant la conclusion de préautorisation.		
is incremental  is_incremental	<i>Valeur booléenne</i> true/false	cavv_preatuh.setIsIncremental(is_incremental);  Indique si cette préautorisation utilise un montant estimé. Les estimations permettent d'incrémenter le montant retenu au moyen de demandes subséquentes d'autorisation incrémentale. La valeur par défaut est « false ».  <b>REMARQUE :</b> veuillez noter que si ce champ contient la valeur « true », la préautorisation n'est valable que pour une seule conclusion de préautorisation. Toute conclusion soumise à titre de conclusion partielle est traitée comme une conclusion complète (ship_indicator= P est traité comme ship_indicator= F lorsque, dans la

		préautorisation originale, le paramètre is_incremental= true).
Indicateur étranger	<i>Boolean</i> true or false	cavvPreauth.setForeignIndicator(foreign_indicator);
ID de correspondance de carte	<i>Chaîne</i> 50 caractères alphanumériques	cavv_preatuh.setCmId(transaction_id);
<b>REMARQUE :</b> Applicables à Offlinx <sup>MC</sup> seulement, chaque transaction doit avoir une valeur unique		
Renseignements du SVA	<i>Objet</i> S. O.	cavv_preatuh.setAvsInfo(avscCheck);
Renseignements du NVC	<i>Objet</i> S. O.	cavv_preatuh.setCvdInfo(cvdCheck);
<b>REMARQUE :</b> Lorsque les renseignements d'identification sont stockés lors de la première transaction, l'objet CVD (numéro de vérification de carte, ou NVC) doit être envoyé. Pour les transactions subséquentes utilisant les renseignements d'identification stockés, le NVC peut être envoyé avec les transactions entamées par le titulaire de carte seulement. <b>Les commerçants ne doivent pas enregistrer le NVC.</b>		
Indicateur de portefeuille électronique	<i>Chaîne</i> 3 caractères alphanumériques	cavv_preatuh.setWalletIndicator(wallet_indicator);
Autorisation finale	<i>Chaîne</i> true/false	cavv_preatuh.setFinalAuth("true");
<b>REMARQUE :</b> Applicable uniquement aux transactions par carte Mastercard		
Renseignements d'identification au dossier cof	<i>Objet</i> S. O.	cavv_preatuh.setCofInfo(cof);
<b>REMARQUE :</b> Cette variable est un objet imbriqué au sein d'une transaction et elle est requise lors du stockage des renseignements d'identification d'un client ou lors de l'utilisation de tels renseignements stockés. Pour plus de renseignements sur les champs de l'objet Credential on File Info, consultez la section Objet Credential		

### Champs de demande pour l'objet Credential on File Info

Variable	Type et limites	
ID de l'émetteur	<p><i>Chaîne</i> 15 caractères</p> <p><b>REMARQUE :</b> Cette variable est requise pour chaque transaction entamée par le commerçant à la suite de la première transaction. Après l'envoi de la première transaction, l'ID d'émetteur (issuer ID) est reçu dans la réponse de transaction, puis utilisé dans les demandes de transaction subséquentes.</p>	<pre>cof.setIssuerId("VALUE_FOR_ISSUER_ID");</pre> <p><b>REMARQUE :</b> Pour obtenir une liste et une explication des valeurs possibles à envoyer pour cette variable, consultez la section Définition des champs de demande – Renseignements d'identification au dossier (cof)</p>
Indicateur de paiement	<p><i>Chaîne</i> 1 caractère alphabétique</p> <p><b>REMARQUE :</b> Lors d'une transaction utilisant les renseignements d'identification au dossier pour laquelle le champ de demande e-commerce indicator est également inclus, les valeurs acceptables de ce champ dépendent de la valeur envoyée pour l'indicateur de paiement. Pour plus d'information, consultez les définitions des champs de réponse.</p>	<pre>cof.setPaymentIndicator("PAYMENT_INDICATOR_VALUE");</pre> <p><b>REMARQUE :</b> Pour obtenir une liste et une explication des valeurs possibles à envoyer pour cette variable, consultez la section Définition des champs de demande – Renseignements d'identification au dossier (cof)</p>
Information de paiement	<p><i>Chaîne</i> 1 caractère numérique</p>	<pre>cof.setPaymentInformation("PAYMENT_INFO_VALUE");</pre> <p><b>REMARQUE :</b> Pour obtenir une liste et une explication des valeurs possibles à envoyer pour cette variable, consultez la section Définition des champs de demande – Renseignements d'identification au dossier (cof)</p>

### 3-D Secure 2.2 – Champs particuliers (facultatifs)

Variable	Type et limites	
ID de transaction DS	<p><i>Chaîne</i> 36 caractères alphanumériques</p> <p><b>REMARQUE :</b> Uniquement utilisé dans les transactions financières utilisant les services 3-D Secure</p>	<pre>cavv_preatuh.setDsTransId("DsTransId");</pre>

## Exemple de préautorisation avec la solution 3-D Secure – cavv\_Preach

```

package Canada;
import JavaAPI.*;
public class TestCanadaCavvPreauth
{
    public static void main(String[] args)
    {
        String store_id = "monca03650";
        String api_token = "7Yw0MPTlhbRcZiE6837";
        java.util.Date createDate = new java.util.Date();
        String order_id = "Test"+createDate.getTime();
        String cust_id = "CUS887H67";
        String amount = "20.00";
        String pan = "5454545454545454";
        String expdate = "2301"; //YYMM
        String cavv = "AAABBjg0Vhi0VniQEjRWAAAAAAA=";
        String dynamic_descriptor = "123456";
        String processing_country_code = "CA";
        String crypt_type = "5";
        String ds_trans_id = "12345";
        boolean status_check = false;
        boolean foreign_indicator= true; //New Foreign Indicator field
        boolean is_incremental = true;

        CavvPreAuth cavvPreauth = new CavvPreAuth();
        cavvPreauth.setOrderId(order_id);
        cavvPreauth.setCustomerId(cust_id);
        cavvPreauth.setAmount(amount);
        cavvPreauth.setPan(pan);
        cavvPreauth.setExpdate(expdate);
        cavvPreauth.setCavv(cavv);
        cavvPreauth.setCryptType(crypt_type); //Mandatory for AMEX only
        cavvPreauth.setDynamicDescriptor(dynamic_descriptor);
        cavvPreauth.setIs_incremental(is_incremental);
        //cavvPreauth.setWalletIndicator("APP"); //set only for wallet transactions. e.g APPLE PAY
        //cavvPreauth.setCmid("8nAK8712sGaAkls56"); //set only for usage with Offlinx - Unique max
        50 alphanumeric characters transaction id generated by merchant
        // cavvPreauth.setThreeDSVersion("2"); //Mandatory for 3DS Version 2.0+
        // cavvPreauth.setThreeDSServerTransId("e11d4985-8d25-40ed-99d6-c3803fe5e68f"); //Mandatory
        for 3DS Version 2.0+ - obtained from MpiCavvLookup or MpiThreeDSAuthentication
        // cavvPreauth.setDsTransId(ds_trans_id); //Optional - to be used only if you are using 3rd
        party 3ds 2.0 service
        //
        // TrId and TokenCryptogram are optional, refer documentation for more details.
        cavvPreauth.setTrId("50189815682");
        cavvPreauth.setTokenCryptogram("APmbM/411e0uAAH+s6xMAAADFA==");
        //optional - Credential on File details
        CofInfo cof = new CofInfo();
        cof.setPaymentIndicator("U");
        cof.setPaymentInformation("2");
        cof.setIssuerId("139X3130ASCXAS9");

        cavvPreauth.setCofInfo(cof);
        cavvPreauth.setForeignIndicator(foreign_indicator);

        HttpsPostRequest mpgReq = new HttpsPostRequest();
        mpgReq.setProcCountryCode(processing_country_code);
        mpgReq.setTestMode(true); //false or comment out this line for production transactions
        mpgReq.setstoreId(store_id);
        mpgReq.setApiToken(api_token);
        mpgReq.setTransaction(cavvPreauth);
        mpgReq.setStatusCheck(status_check);
        mpgReq.send();
        try
        {
            Receipt receipt = mpgReq.getReceipt();
            System.out.println("CardType = " + receipt.getCardType());
            System.out.println("TransAmount = " + receipt.getTransAmount());
            System.out.println("TxnNumber = " + receipt.getTxnNumber());
            System.out.println("ReceiptId = " + receipt.getReceiptId());
            System.out.println("TransType = " + receipt.getTransType());
            System.out.println("ReferenceNum = " + receipt.getReferenceNum());
        }
    }
}

```

```

System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = " + receipt.getBankTotals());
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("CavvResultCode = " + receipt.getCavvResultCode());
System.out.println("IssuerId = " + receipt.getIssuerId());
System.out.println("ThreeDSVersion = " + receipt.getThreeDSVersion());
System.out.println("SourcePanLast4 = " + receipt.getSourcePanLast4());
System.out.println("Advice Code= "+ receipt.getAdviceCode());
}
catch (Exception e)
{
e.printStackTrace();
}
}
}

import JavaAPI.*;
public class TestCanadaCavvPreauth
{
public static void main(String[] args)
{
String store_id = "store5";
String api_token = "yesguy";
java.util.Date createDate = new java.util.Date();
String order_id = "Test"+createDate.getTime();
String cust_id = "CUS887H67";
String amount = "10.42";
String pan = "4242424242424242";
String expdate = "1911"; //YYMM format
String cavv = "AAABBJg0VhI0VniQEjRWAAAAAAA=";
String dynamic_descriptor = "123456";
String processing_country_code = "CA";
String crypt_type = "5";
String ds_trans_id = "12345";
boolean status_check = false;
boolean foreign_indicator= true; //New Foreign Indicator field

CavvPreAuth cavvPreauth = new CavvPreAuth();
cavvPreauth.setOrderId(order_id);
cavvPreauth.setCustomerId(cust_id);
cavvPreauth.setAmount(amount);
cavvPreauth.setPan(pan);
cavvPreauth.setExpdate(expdate);
cavvPreauth.setCavv(cavv);
cavvPreauth.setCryptType(crypt_type); //Mandatory for AMEX only
cavvPreauth.setDynamicDescriptor(dynamic_descriptor);
//cavvPreauth.setWalletIndicator("APP"); //set only for wallet transactions. e.g APPLE PAY
//cavvPreauth.setCmid("8nAK8712sGaAkls56"); //set only for usage with Offlinx - Unique max
50 alphanumeric characters transaction id generated by merchant
cavvPreauth.setThreeDSVersion("2.2.0"); //Mandatory for financial transactions using 3rd
Party 3-D Secure services
cavvPreauth.setThreeDSServerTransId("e11d4985-8d25-40ed-99d6-c3803fe5e68f"); //Mandatory for
financial transactions using 3rd Party 3-D Secure services" - obtained from MpiCavvLookup or
MpiThreeDSAuthentication
//cavvPreauth.setDsTransId("12345");//Optional - to be used only if you are using 3rd party
3ds service

// TrId and TokenCryptogram are optional, refer documentation for more details.
cavvPreauth.setTrId("50189815682");
cavvPreauth.setTokenCryptogram("APmbM/411e0uAAH+s6xMAAADFA==");

//optional - Credential on File details
CofInfo cof = new CofInfo();
cof.setPaymentIndicator("U");
cof.setPaymentInformation("2");
cof.setIssuerId("139X3130ASCXAS9");

cavvPreauth.setCofInfo(cof);
cavvPreauth.setForeignIndicator(foreign_indicator);

HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false for production transactions
mpgReq.setstoreId(store_id);
mpgReq.setApiToken(api_token);

```

```
mpgReq.setTransaction(cavvPreauth);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = " + receipt.getBankTotals());
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("CavvResultCode = " + receipt.getCavvResultCode());
System.out.println("IssuerId = " + receipt.getIssuerId());
System.out.println("ThreeDSVersion = " + receipt.getThreeDSVersion());
System.out.println("SourcePanLast4 = " + receipt.getSourcePanLast4());
System.out.println("Advice Code= "+ receipt.getAdviceCode());
}
catch (Exception e)
{
e.printStackTrace();
}
}
```

### 6.8.3 Achat avec la chambre forte et la solution 3-D Secure

**REMARQUE :** Cette transaction prend en charge les jetons temporaires et permanents.

**AVERTISSEMENT :** Moneris déconseille fortement l'utilisation de cadres pour l'intégration de la solution 3-D Secure et ne peut garantir leur fiabilité lors du traitement des transactions dans l'environnement de production.

```
ResCavvPurchaseCC resCavvPurchaseCC = new ResCavvPurchaseCC();  
  
HttpsPostRequest mpgReq = new HttpsPostRequest();  
  
mpgReq.setTransaction(resCavvPurchaseCC);
```

#### Principaux champs d'objet de connexion (toutes les transactions API)

Variable	Type et limites	
ID de commerce	<i>Chaîne</i> S. O.	mpgReq.setstoreId(store_id);
Jeton API	<i>Chaîne</i> S. O.	mpgReq.setApiToken(api_token);

#### Champs de demande de transaction d'achat avec la chambre forte et la solution 3-D Secure

Variable	Type et limites	
Clé de données	<i>Chaîne</i> 25 caractères alphanumériques	resCavvPurchaseCC.setData(data_key);
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques a-Z A-Z 0-9 _ - : . @ espaces	resCavvPurchaseCC.setorderId(order_id);

Montant	<i>Chaîne</i>	resCavvPurchaseCC.setAmount (amount);
	10 caractères décimaux Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal	
	<b>EXEMPLE : 1 234 567,89</b>	
Code de vérification d'authentification du titulaire de carte (CAVV)	<i>Chaîne</i> 50 caractères alphanumériques	resCavvPurchaseCC.setCavv (cavv);
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	resCavvPurchaseCC.setCryptType (crypt);

### 3-D Secure 2.2 – Champs particuliers (obligatoires)

Variable	Type et limites	
Version de 3DS  <b>REMARQUE :</b> Obligatoire pour les transactions utilisant les services de 3-D Secure d'un tiers	<i>Chaîne</i> 10 caractères numériques	resCavvPurchaseCC.setThreeDSVersion ("ThreeDSVersion");  Valeurs acceptées :  2.0.0 = protocole 3DS 2.0.0 2.1.0 = protocole 3DS 2.1.0 2.2.0 = protocole 3DS 2.2.0 2.3.0 = protocole 3DS 2.3.0
ID de transaction du serveur 3DS  <b>REMARQUE :</b> Obligatoire pour les transactions utilisant les services de 3-D Secure d'un tiers, obtenue à partir de la demande de recherche code de vérification d'authentification du titulaire de carte ou de la demande d'authentification 3DS des modules d'extension pour les commerçants.	<i>Chaîne</i> 36 caractère numérique	resCavvPurchaseCC.setThreeDSServerTransId ("ThreeDSServerTransId");

## Champs de demande liés aux transactions de d'achat avec la chambre forte et la solution 3-D Secure (facultatifs)

Variable	Type et limites	
ID de client	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p> <div style="background-color: #e0f2e0; padding: 5px;"> <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : &lt;&gt;\$%=?^{}[]\</div>	resCavvPurchaseCC.setCustId(cust_id);
Date d'expiration	<p><i>Chaîne</i></p> <p>4 caractères alphanumériques</p> <p>AAMM</p>	resCavvPurchaseCC.setExpDate(expiry_date);
Autorisation finale	<p><i>Chaîne</i></p> <p>true/false</p> <div style="background-color: #e0f2e0; padding: 5px;"> <b>REMARQUE :</b> Applicable uniquement aux transactions par carte Mastercard       </div>	resCavvPurchaseCC.setFinalAuth("true");

### 3-D Secure 2.2 – Champs particuliers (facultatifs)

Variable	Type et limites
ID de transaction DS	<p><i>Chaîne</i></p> <p>36 caractères alphanumériques</p> <div style="background-color: #e0f2e0; padding: 5px;"> <b>REMARQUE :</b> Uniquement utilisé dans les transactions financières utilisant les services 3-D Secure d'un tiers       </div>

### Exemple d'achat avec la chambre forte et la solution 3-D Secure

```

public class TestCanadaResCavvPurchaseCC
{
  public static void main(String[] args)
  {
    String store_id = "store1";
    String api_token = "yesguyl";
    String data_key = "4INQR1A8ocxD0oafSz50LADXY";
    java.util.Date createDate = new java.util.Date();
    String order_id = "Test"+createDate.getTime();
    String amount = "1.00";
    String cust_id = "customer1"; //if sent will be submitted, otherwise cust_id from profile
    will be used
    String cavv = "AAABBJg0VhI0VniQEjRWAAAAAAA";
    String processing_country_code = "CA";
    String exp_date = "1901";
    String ds_trans_id = "12345";
    boolean status_check = false;

    ResCavvPurchaseCC resCavvPurchaseCC = new ResCavvPurchaseCC();
    resCavvPurchaseCC.setOrderId(order_id);
    resCavvPurchaseCC.setDataKey(data_key);
  }
}

```

```

resCavvPurchaseCC.setCustId(cust_id);
resCavvPurchaseCC.setAmount(amount);
resCavvPurchaseCC.setCavv(cavv);
resCavvPurchaseCC.setExpDate(exp_date);

//NT Response Option
boolean get_nt_response = true;
resCavvPurchaseCC.setGetNtResponse(get_nt_response);

//optional - Installment Info
// InstallmentInfo installmentInfo = new InstallmentInfo();
// installmentInfo.setPlanId("ae859ef1-eb91-b708-8b80-1dd481746401");
// installmentInfo.setPlanIdRef("0000000065");
// installmentInfo.setTacVersion("2");
// resCavvPurchaseCC.setInstallmentInfo(installmentInfo);

resCavvPurchaseCC.setThreeDSVersion("2.2.0"); //Mandatory for financial transactions using
3rd Party 3-D Secure services
resCavvPurchaseCC.setThreeDSServerTransId("e11d4985-8d25-40ed-99d6-c3803fe5e68f");
//Mandatory for financial transactions using 3rd Party 3-D Secure services - obtained from
MpICavvLookup or MpIThreeDSAuthentication
//resCavvPurchaseCC.setDsTransId("12345");//Optional - to be used only if you are using 3rd
party 3ds 2.0 service

//Mandatory - Credential on File details
CofInfo cof = new CofInfo();
cof.setPaymentIndicator("U");
cof.setPaymentInformation("2");
cof.setIssuerId("139X3130ASCXAS9");

resCavvPurchaseCC.setCofInfo(cof);

HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false for production transactions
mpgReq.setstoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(resCavvPurchaseCC);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("DataKey = " + receipt.getDataKey());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Message = " + receipt.getMessage());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("ResSuccess = " + receipt.getResSuccess());
System.out.println("PaymentType = " + receipt.getPaymentType());
System.out.println("CavvResultCode = " + receipt.getCavvResultCode());
System.out.println("IssuerId = " + receipt.getIssuerId());
System.out.println("ThreeDSVersion = " + receipt.getThreeDSVersion());
//ResolveData
System.out.println("Cust ID = " + receipt.getResCustId());
System.out.println("Phone = " + receipt.getResPhone());
System.out.println("Email = " + receipt.getResEmail());
System.out.println("Note = " + receipt.getResNote());
System.out.println("Masked Pan = " + receipt.getResMaskedPan());
System.out.println("Exp Date = " + receipt.getResExpdate());
System.out.println("Crypt Type = " + receipt.getResCryptType());
System.out.println("Avs Street Number = " + receipt.getResAvsStreetNumber());
System.out.println("Avs Street Name = " + receipt.getResAvsStreetName());
System.out.println("Avs Zipcode = " + receipt.getResAvsZipcode());
System.out.println("SourcePanLast4 = " + receipt.getSourcePanLast4());

if(get_nt_response) {
System.out.println("\nNTResponseCode = " + receipt.getNTResponseCode());
System.out.println("NTMessage = " + receipt.getNTMessage());
System.out.println("NTUsed = " + receipt.getNTUsed());
System.out.println("NTTokenBin = " + receipt.getNTTokenBin());
System.out.println("NTTokenLast4 = " + receipt.getNTTokenLast4());
}
}

```

```
System.out.println("NTTokenExpDate = " + receipt.getNTTokenExpDate());
}
// InstallmentResults installmentResults = receipt.getInstallmentResults();
// System.out.println("\nPlanId = " + installmentResults.getPlanId() +"\n");
// System.out.println("PlanIDRef = " + installmentResults.getPlanIDRef());
// System.out.println("TacVersion = " + installmentResults.getTacVersion());
// System.out.println("PlanAcceptanceId = " + installmentResults.getPlanAcceptanceId());
// System.out.println("PlanStatus = " + installmentResults.getPlanStatus());
// System.out.println("PlanResponse = " + installmentResults.getPlanResponse());
}
catch (Exception e)
{
e.printStackTrace();
}
}
```

## 6.8.4 Préautorisation avec la chambre forte et la solution 3-D Secure

**REMARQUE :** Cette transaction prend en charge les jetons temporaires et permanents.

**AVERTISSEMENT :** Moneris déconseille fortement l'utilisation de cadres pour l'intégration de la solution 3-D Secure et ne peut garantir leur fiabilité lors du traitement des transactions dans l'environnement de production.

### Définition de l'objet de transaction Pre-Authorization with Vault & 3-D Secure

```
ResCavvPreAuthCC resCavvPreauthCC = new ResCavvPreAuthCC();
```

### Objet HttpsPostRequest pour les transactions de préautorisation avec la chambre forte et 3- D Secure

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
```

```
mpgReq.setTransaction(resCavvPreauthCC);
```

### Principaux champs d'objet de connexion (toutes les transactions API)

Variable	Type et limites	
ID de commerce	<i>Chaîne</i> S. O.	mpgReq.setstoreId(store_id);
Jeton API	<i>Chaîne</i> S. O.	mpgReq.setApiToken(api_token);

### Champs de demande pour les transactions de préautorisation avec la chambre forte et la solution 3-D Secure (obligatoires)

Variable	Type et limites	Méthode Set
Clé de données	<i>Chaîne</i> 25 caractères alphanumériques	resCavvPreauthCC.setData(data_key);
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques a-Z A-Z 0-9 _ - : . @ espaces	resCavvPreauthCC.setorderId(order_id);

Montant	<i>Chaîne</i>	resCavvPreauthCC.setAmount(amount);
	10 caractères décimaux  Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal	
	<b>EXEMPLE : 1 234 567,89</b>	
Code de vérification d'authentification du titulaire de carte (CAVV)	<i>Chaîne</i>  50 caractères alphanumériques	resCavvPreauthCC.setCavv(cavv);
Indicateur de commerce électronique	<i>Chaîne</i>  1 caractère alphanumérique	resCavvPreauthCC.setCryptType(crypt);

**Champs de demande pour les transactions de préautorisation avec la chambre forte et la solution 3-D Secure ( facultatifs )**

Variable	Type et limites	Méthode Set
ID de client	<i>Chaîne</i>  50 caractères alphanumériques  <div style="background-color: #e0f2e0; padding: 5px;"> <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : &lt;&gt;\$%=?^{}[]\ </div>	resCavvPreauthCC.setCustomerId(cust_id);
Descripteur dynamique	<i>Chaîne</i>  20 caractères alphanumériques  Total de 22 caractères incluant votre nom de commerçant et un séparateur  <div style="background-color: #e0f2e0; padding: 5px;"> <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : &lt;&gt;\$%=?^{}[]\ </div>	resCavvPreauthCC.setDynamicDescriptor(dynamic_descriptor);

Date d'expiration	<i>Chaîne</i> 4 caractères alphanumériques AAMM	resCavvPreauthCC.setExpDate(expiry_date);
ID de transaction DS	<i>Chaîne</i> 36 caractères alphanumériques	resCavvPreauthCC.setDsTransId("DsTransId");
<b>REMARQUE :</b> Uniquement utilisé dans les transactions financières utilisant les services 3-D Secure d'un tiers		
Autorisation finale	<i>Chaîne</i> true/false	resCavvPreauthCC.setFinalAuth("true");
<b>REMARQUE :</b> Applicable uniquement aux transactions par carte Mastercard		

### Exemple de transaction de préautorisation avec la chambre forte et la solution 3-D Secure

```

package Canada;
import JavaAPI.*;
public class TestCanadaResCavvPreauthCC
{
    public static void main(String[] args)
    {
        String store_id = "store1";
        String api_token = "yesguy1";
        String data_key = "4INQR1A8ocxD0oafSz50LADXY";
        java.util.Date createDate = new java.util.Date();
        String order_id = "Test"+createDate.getTime();
        String amount = "1.00";
        String cust_id = "customer1"; //if sent will be submitted, otherwise cust_id from profile
        will be used
        String cavv = "AAABBJg0VhI0VniQEjRWAAAAAA";
        String processing_country_code = "CA";
        String expdate = "1901";
        boolean status_check = false;

        ResCavvPreauthCC resCavvPreauthCC = new ResCavvPreauthCC();
        resCavvPreauthCC.setOrderId(order_id);
        resCavvPreauthCC.setDataKey(data_key);
        resCavvPreauthCC.setCustomerId(cust_id);
        resCavvPreauthCC.setAmount(amount);
        resCavvPreauthCC.setCavv(cavv);
        resCavvPreauthCC.setExpDate(expdate);

        //NT Response Option
        boolean get_nt_response = true;
        resCavvPreauthCC.setGetNtResponse(get_nt_response);

        //optional - Installment Info
        // InstallmentInfo installmentInfo = new InstallmentInfo();
        // installmentInfo.setPlanId("ae859ef1-eb91-b708-8b80-1dd481746401");
        // installmentInfo.setPlanIdRef("0000000065");
        // installmentInfo.setTacVersion("2");
        // resCavvPreauthCC.setInstallmentInfo(installmentInfo);

        resCavvPreauthCC.setThreeDSVersion("2.2.0"); //Mandatory for financial transactions using 3rd
        Party 3-D Secure services.
        resCavvPreauthCC.setThreeDSServerTransId("e11d4985-8d25-40ed-99d6-c3803fe5e68f"); //Mandatory
        for financial transactions using 3rd Party 3-D Secure services - obtained from MpiCavvLookup
        or MpiThreeDSAuthentication
        //resCavvPreauthCC.setDsTransId("12345");//Optional - to be used only if you are using 3rd
        party 3ds 2.0 service

        //Mandatory - Credential on File details
    }
}

```

```

CofInfo cof = new CofInfo();
cof.setPaymentIndicator("U");
cof.setPaymentInformation("2");
cof.setIssuerId("139X3130ASCXAS9");

resCavvPreauthCC.setCofInfo(cof);

HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false for production transactions
mpgReq.setstoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(resCavvPreauthCC);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("DataKey = " + receipt.getDataKey());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Message = " + receipt.getMessage());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("ResSuccess = " + receipt.getResSuccess());
System.out.println("PaymentType = " + receipt.getPaymentType());
System.out.println("CavvResultCode = " + receipt.getCavvResultCode());
System.out.println("IssuerId = " + receipt.getIssuerId());
System.out.println("ThreeDSVersion = " + receipt.getThreeDSVersion());

//ResolveData
System.out.println("Cust ID = " + receipt.getResCustomerId());
System.out.println("Phone = " + receipt.getResPhone());
System.out.println("Email = " + receipt.getResEmail());
System.out.println("Note = " + receipt.getResNote());
System.out.println("Masked Pan = " + receipt.getResMaskedPan());
System.out.println("Exp Date = " + receipt.getResExpdate());
System.out.println("Crypt Type = " + receipt.getResCryptType());
System.out.println("Avs Street Number = " + receipt.getResAvsStreetNumber());
System.out.println("Avs Street Name = " + receipt.getResAvsStreetName());
System.out.println("Avs Zipcode = " + receipt.getResAvsZipcode());
System.out.println("SourcePanLast4 = " + receipt.getSourcePanLast4());

if(get_nt_response) {
System.out.println("\nNTResponseCode = " + receipt.getNTResponseCode());
System.out.println("NTMessage = " + receipt.getNTMessage());
System.out.println("NTUsed = " + receipt.getNTUsed());
System.out.println("NTTokenBin = " + receipt.getNTTokenBin());
System.out.println("NTTokenLast4 = " + receipt.getNTTokenLast4());
System.out.println("NTTokenExpDate = " + receipt.getNTTokenExpDate());
}
// InstallmentResults installmentResults = receipt.getInstallmentResults();
// System.out.println("\nPlanId = " + installmentResults.getPlanId() +"\n");
// System.out.println("PlanIDRef = " + installmentResults.getPlanIDRef());
// System.out.println("TacVersion = " + installmentResults.getTacVersion());
// System.out.println("PlanAcceptanceId = " + installmentResults.getPlanAcceptanceId());
// System.out.println("PlanStatus = " + installmentResults.getPlanStatus());
// System.out.println("PlanResponse = " + installmentResults.getPlanResponse());
}
catch (Exception e)
{
e.printStackTrace();
}
}

```

## 6.8.5 Achat avec la solution 3-D Secure et facturation périodique

**AVERTISSEMENT :** Moneris déconseille fortement l'utilisation de cadres pour l'intégration de la solution 3-D Secure et ne peut garantir leur fiabilité lors du traitement des transactions dans l'environnement de production.

L'exemple ci-dessous illustre un achat avec la solution 3-D Secure lorsque l'objet Recurring Billing Info est également envoyé dans la transaction.

### Achat avec 3-D Secure et facturation périodique

```
package Canada;
import JavaAPI.*;
public class TestCanadaCavvPurchaseRecur
{
    public static void main(String[] args)
    {
        String store_id = "store5";
        String api_token = "yesguy";
        java.util.Date createDate = new java.util.Date();
        String order_id = "Test"+createDate.getTime();
        String cust_id = "CUS887H67";
        String amount = "10.42";
        String pan = "4242424242424242";
        String expdate = "1901"; //YYMM
        String cavv = "AAABBJg0VhI0VniQEjRWAAAAAAA=";
        String dynamic_descriptor = "123456";
        String processing_country_code = "CA";
        String crypt_type = "5";
        boolean status_check = false;

        /***** Recur Variables *****/
        String recur_unit = "month"; //eom = end of month
        String start_now = "true";
        String start_date = "2022/02/09";
        String num_recurr = "12";
        String period = "1";
        String recur_amount = "5.00";
        /***** Recur Object Option1 *****/
        Recur recurring_cycle = new Recur(recur_unit, start_now, start_date,
        num_recurr, period, recur_amount);
        CavvPurchase cavvPurchase = new CavvPurchase();
        cavvPurchase.setOrderId(order_id);
        cavvPurchase.setCustomerId(cust_id);
        cavvPurchase.setAmount(amount);
        cavvPurchase.setPan(pan);
        cavvPurchase.setExppdate(expdate);
        cavvPurchase.setCavv(cavv);
        cavvPurchase.setCryptType(crypt_type); //Mandatory for AMEX only
        cavvPurchase.setDynamicDescriptor(dynamic_descriptor);
        cavvPurchase.setRecur(recurring_cycle);
        //cavvPurchase.setWalletIndicator("APP"); //set only for wallet transactions. e.g APPLE PAY
        //cavvPurchase.setNetwork("Interac"); //set only for Interac e-commerce
        //cavvPurchase.setDataType("3DSecure"); //set only for Interac e-commerce
        cavvPurchase.setThreeDSVersion("2.2.0"); //Mandatory for financial transactions using 3rd
        Party 3-D Secure services.
        cavvPurchase.setThreeDSServerTransId("e11d4985-8d25-40ed-99d6-c3803fe5e68f"); Mandatory
        for financial transactions using 3rd Party 3-D Secure services - obtained from MpiCavvLookup
        or MpiThreeDSAuthentication
        cavvPurchase.setDsTransId("12345");

        //Mandatory on Recurs - Credential on File details
        CofInfo cof = new CofInfo();
        cof.setPaymentIndicator("R");
        cof.setPaymentInformation("2");
        cof.setIssuerId("139X3130ASCXAS9");

        cavvPurchase.setCofInfo(cof);

        HttpsPostRequest mpgReq = new HttpsPostRequest();
        mpgReq.setProcCountryCode(processing_country_code);
        mpgReq.setTestMode(true); //false for production transactions
```

```

mpgReq.setstoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(cavvPurchase);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = " + receipt.getBankTotals());
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("CavvResultCode = " + receipt.getCavvResultCode());
System.out.println("IssuerId = " + receipt.getIssuerId());
}
catch (Exception e)
{
e.printStackTrace();
}
}
}

```

## 6.9 Tester votre intégration 3-D Secure 2.2

Dans la phase de test du développement :

1. Utilisez l'URL de test comme serveur pour vos demandes :

[esqa.moneris.com](http://esqa.moneris.com)

2. Dans toutes les transactions de demande de recherche de carte (Card Lookup), assurez-vous que vous utilisez un ID de commerce et un jeton API test.
3. Dans toutes les transactions de demande d'authentification 3DS des modules d'extension pour les commerçants, assurez-vous que vous utilisez un ID de commerce et un jeton API test.
4. Dans toutes les transactions de demande de recherche de code de vérification d'authentification du titulaire de carte (Cavv Lookup), assurez-vous que vous utilisez un ID de commerce et un jeton API test.

## 6.10 Passage à la phase de production avec 3-D Secure 2.2

Après avoir terminé la phase de test de votre intégration 3D Secure 2.2, procédez comme suit pour passer à la phase de production :

1. Utilisez l'URL de production comme serveur pour vos demandes :

www3.moneris.com

2. Dans toutes les transactions de demande de recherche de carte (Card Lookup), assurez-vous que vous utilisez un ID de commerce et un jeton API de production.
3. Dans toutes les transactions de demande d'authentification 3DS des modules d'extension pour les commerçants, assurez-vous que vous utilisez un ID de commerce et un jeton API de production.
4. Dans toutes les transactions de demande de recherche de code de vérification d'authentification du titulaire de carte (Cavv Lookup), assurez-vous que vous utilisez un ID de commerce et un jeton API de production.

## 6.11 Codes TransStatus de la solution 3-D Secure 2.2

Valeur	Description	Commentaires
Y	Authentifié	Le titulaire de carte a été entièrement authentifié.
D	Contestation (découplée) requise	L'authentification découplée nécessite une contestation pour authentifier le titulaire de carte.
A	Tentative d'authentification	Une preuve de tentative d'authentification a été produite.
C	Contestation requise	Le titulaire de carte nécessite une contestation pour compléter l'authentification.
U	Non authentifié	L'authentification n'a pas pu être effectuée en raison d'un problème technique ou autre.
N	Non authentifié	Non authentifié
R	Non authentifié	Non authentifié, car l'émetteur rejette l'authentification et demande que l'autorisation ne soit pas tentée.

## 6.12 3-D Secure 2.2 Codes de refus TransStatusReason communs

Les codes suivants sont renvoyés par le service 3-D Secure afin de fournir des informations supplémentaires sur l'état de la transaction 3-D Secure.

Code TransStatusReason	Description
01	Échec de l'authentification de la carte
02	Appareil inconnu
03	Appareil non pris en charge
04	Limite de fréquence d'authentification dépassée
05	Carte expirée
06	Numéro de carte invalide
07	Transaction invalide
08	Pas d'enregistrement de carte
09	Défaut de sécurité
10	Carte volée
11	Fraude soupçonnée
12	Transaction refusée non permise au titulaire de carte
13	Titulaire de carte non inscrit au service
14	La transaction a été interrompue au niveau de l'ACS
15	Confiance faible
16	Confiance moyenne
17	Confiance élevée
18	Confiance très élevée
19	Nombre maximum de contestations de l'ACS dépassé
20	Transaction de non-paiement pas prise en charge

Code TransStatusReason	Description
21	Transaction 3RI pas prise en charge
22	Problème technique de l'ACS
23	Authentification découpée requise par l'ACS mais non demandée par le demandeur 3DS
24	Délai d'expiration maximal dépassé pour l'authentification 3DS entamée par le demandeur
25	L'authentification découpée n'a pas eu suffisamment de temps pour authentifier le titulaire de la carte. L'ACS ne tentera pas de l'authentifier.
26	Authentification tentée mais non effectuée par le titulaire de la carte

**REMARQUE :** Pour obtenir une liste de tous les codes de refus TransStatus, veuillez consulter la section Référence de 3-D Secure 2.2 à l'adresse <https://developer.moneris.com>.

## 6.13 Codes de résultat liés au code de vérification d'authentification du titulaire de carte (CAVV)

Le code de vérification d'authentification du titulaire de carte (CAVV), la valeur d'authentification du titulaire de compte (AAV), et la valeur de vérification American Express (AEVV), sont les valeurs qui permettent à Visa, Mastercard et American Express de valider l'intégrité des données des transactions Visa Secure, Mastercard Identity Check et American Express SafeKey. Ces valeurs sont transmises par l'émetteur au commerçant suite à l'authentification. Le commerçant intègre ensuite la valeur CAVV, AAV ou AEVV à la demande d'autorisation en utilisant la transaction de type Achat ou Préautorisation avec la solution 3-D Secure.

Pour résumer ce processus :

1. Le commerçant effectue une demande d'authentification 3-D Secure et reçoit une valeur CAVV, AAV ou AEVV en réponse.
2. Le commerçant envoie la valeur CAVV, AAV ou AEVV à Moneris en utilisant une transaction de type Achat ou Préautorisation avec 3-D Secure et reçoit le code de résultat du CAVV dans la réponse.

Les tableaux suivants décrivent le contenu de la réponse aux données du CAVV et sa signification pour le commerçant.

### 6.13.1 Codes de résultat liés au code de vérification d'authentification du titulaire de carte (CAVV) de Visa

#### Codes de résultat liés au code de vérification d'authentification du titulaire de carte (CAVV)

Code de résultat	Message	Signification pour les commerçants
Vierge	Code de vérification d'authentification du titulaire de carte (CAVV) absent ou non vérifié	Transaction qui n'est pas d'origine Visa Secure Aucun transfert de responsabilité et le commerçant n'est pas protégé contre les débits compensatoires.
0	Les résultats de l'authentification du CAVV sont non valides.	Transaction qui n'est pas d'origine Visa Secure Aucun transfert de responsabilité et le commerçant n'est pas protégé contre les débits compensatoires.
1	Échec de la validation (authentification) du CAVV	Dans la mesure où vous avez correctement mis en œuvre le processus Visa Secure, la responsabilité de cette transaction demeure celle de l'émetteur, car les codes de raison de débits compensatoires sont couverts par Visa Secure.
2	Réussite de la validation (authentication) du code de vérification d'authentification du titulaire de carte (CAVV)	Transaction entièrement authentifiée Il y a un transfert de responsabilité, et le commerçant est protégé contre les débits compensatoires.
3, 8, A	Réussite de la validation (tentative) du code de vérification d'authentification du titulaire de carte (CAVV)	Une tentative de Visa Secure a été effectuée. Il y a un transfert de responsabilité et le commerçant est protégé contre certains débits compensatoires liés à la fraude par cartes.
4, 7, 9	Échec de la validation du CAVV; tentative	Une tentative de Visa Secure a été effectuée. Il y a un transfert de responsabilité et le commerçant est protégé contre certains débits compensatoires liés à la fraude par cartes.
6	CAVV non validé – Émetteur ne participe pas à la transaction	Une tentative de Visa Secure a été effectuée. Il y a un transfert de responsabilité et le commerçant est protégé contre certains débits compensatoires liés à la fraude par cartes.

Code de résultat	Message	Signification pour les commerçants
B	Le code de vérification d'authentification du titulaire de carte (CAVV) a réussi la validation; à titre informatif uniquement	Transaction qui n'est pas d'origine Visa Secure Aucun transfert de responsabilité et le commerçant n'est pas protégé contre les débits compensatoires.
C	Le code de vérification d'authentification du titulaire de carte (CAVV) n'a pas été validé (tentative)	Une tentative de Visa Secure a été effectuée. Il y a un transfert de responsabilité et le commerçant est protégé contre certains débits compensatoires liés à la fraude par cartes.
D	Le code de vérification d'authentification du titulaire de carte (CAVV) n'a pas été validé (authentification)	Une tentative de Visa Secure a été effectuée. Il y a un transfert de responsabilité et le commerçant est protégé contre certains débits compensatoires liés à la fraude par cartes.

### 6.13.2 Codes de résultat liés au code de vérification d'authentification du titulaire de carte (CAVV) de Mastercard

**Codes de résultat liés au code de vérification d'authentification du titulaire de carte (CAVV) de Mastercard**

Code de résultat	Message	Signification pour les commerçants
0	Authentification échouée	Il ne s'agit pas d'une transaction Mastercard Identity Check. Aucun transfert de responsabilité et le commerçant n'est pas protégé contre les débits compensatoires.
1	Authentification tentée	Une transaction Mastercard Identity Check a été tentée. Il y a un transfert de responsabilité et le commerçant est protégé contre certains débits compensatoires liés à la fraude par cartes (les cartes commerciales internationales sont exclues).
2	Authentification réussie	Transaction entièrement authentifiée Il y a un transfert de responsabilité, et le commerçant est protégé contre les débits compensatoires.

### 6.13.3 Codes de résultat liés au code de vérification d'authentification du titulaire de carte (CAVV) de American Express

#### Codes de résultat liés au code de vérification d'authentification du titulaire de carte (CAVV) de American Express

American Express SafeKey **REMARQUE** : n'est disponible que pour les commerçants acquis directement par American Express (c'est-à-dire les commerçants qui ne sont pas membres du programme OptBlue). Toutes les questions relatives aux débits compensatoires, à la responsabilité et aux différends doivent être adressées à votre représentant American Express étant donné qu'American Express est l'acquéreur officiel de ces commerçants.

Code de résultat	Description
1	AEVV échoué – Authentification, clé de l'émetteur
2	AEVV réussi – Authentification, clé de l'émetteur
3	AEVV réussi – Tentative, clé de l'émetteur
4	AEVV échoué – Tentative, clé de l'émetteur
7	AEVV échoué – Tentative, émetteur non participant, clé de réseau
8	AEVV réussi – Tentative, émetteur non participant, clé de réseau
9	AEVV échoué – Tentative, émetteur participant, serveur de contrôle d'accès (ACS) non disponible, clé de réseau
A	AEVV réussi – Tentative, émetteur participant, serveur de contrôle d'accès (ACS) non disponible, clé de réseau
U	AEVV non vérifié

## 7 Tarification multidevise (TMD)

- 7.1 À propos de la tarification multidevise (TMD)
- 7.2 Méthode de traitement des transactions avec la TMD
- 7.3 Taux d'obtention de la TMD
- 7.4 Achat utilisant la TMD
- 7.5 Achat utilisant la TMD avec 3-D Secure
- 7.6 Achat utilisant la TMD avec 3-D Secure et la chambre forte
- 7.7 Préautorisation utilisant la TMD
- 7.8 Préautorisation utilisant la TMD avec 3-D Secure
- 7.9 Préautorisation utilisant la TMD avec 3-D Secure et la chambre forte
- 7.10 Conclusion de préautorisation utilisant la TMD
- 7.11 Correction d'achat utilisant la TMD
- 7.12 Remboursement utilisant la TMD
- 7.13 Remboursement indépendant utilisant la TMD
- 7.14 Achat utilisant la TMD avec la chambre forte
- 7.15 Préautorisation utilisant la TMD avec la chambre forte
- 7.16 Remboursement indépendant utilisant la TMD avec la chambre forte
- 7.17 Codes de devise de la TMD
- 7.18 Codes d'erreur de la TMD

### 7.1 À propos de la tarification multidevise (TMD)

La tarification multidevise (TMD) est un service financier qui permet à vos entreprises d'afficher le prix de vos biens et services dans plusieurs devises tout en continuant de recevoir vos fonds et de produire vos rapports en dollars canadiens. Grâce à la TMD, les titulaires de carte peuvent magasiner, voir les prix et payer dans la devise de leur choix.

La TMD fonctionne uniquement avec les cartes Visa et Mastercard.

**REMARQUE :** Utilisez la TMD seulement pour traiter des transactions nécessitant un échange de devises étrangères; pour les transactions strictement en dollars canadiens, utilisez les demandes de transactions financières de base.

### 7.2 Méthode de traitement des transactions avec la TMD

Il existe deux méthodes pour traiter les transactions de tarification multidevises par l'intermédiaire de Passerelle Moneris :

1. **Utilisation de la transaction d'obtention de taux de la TMD** : Cette méthode est utilisée pour obtenir un taux de change et fixer ce taux précis pour une durée limitée, qui est appliqué dans une transaction ultérieure.

2. **Sans l'utilisation de la transaction d'obtention de taux de la TMD** : Cette méthode envoie une transaction de TMD sans effectuer la demande d'obtention du taux, et le taux de change est obtenu au moment du traitement.

## 7.3 Obtention du taux de la TMD

Une transaction d'obtention du taux de la TMD effectue une recherche du taux de change d'une devise étrangère et fixe ce taux de change pour l'utiliser dans une transaction financière ultérieure utilisant la TMD.

Le taux de change obtenu à la suite de cette demande de transaction est affiché dans la réponse sous la forme d'un jeton **RateToken**, et le taux de change sous-jacent est fixé pour une durée limitée.

### Définition de l'objet de transaction MCP Get Rate

```
MCPGetRate mcpGetRate = new MCPGetRate();
```

### Objet HttpsPostRequest pour les transactions d'obtention de taux de la TMD

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.SetTransaction(getRate);
```

### Champs de demande pour les transactions d'obtention de taux de la TMD (obligatoires)

Variable	Type et limites	Méthode Set
Numéro de la version de la TMD	<i>Chaîne</i> Valeur numérique La version actuelle est 1.0	getRate.setMCPVersion("MCP_VERSION_NUM");
Type de transaction de taux	<i>Chaîne</i> 1 caractère alphabétique	getRate.setRateTxnType("TRANSACTION_TYPE_VALUE");
Renseignements sur le taux de la TMD	<i>Objet</i> S. O.	getRate.setMCPRateInfo(rate);

## Champs de demande de l'objet MCP Rate Info

Au moins une des variables suivantes doit être envoyée :

Variable	Type et limites	Méthode Set
Ajouter le montant du titulaire de carte	<p><i>Tableau chaîne</i></p> <p>12 caractères numériques, 3 caractères numériques</p> <p>(la plus petite unité discrète de devise étrangère, code de devise)</p>	rate.addCardholderAmount ("FOREIGN_AMT", "FOREIGN_CURRENCY_CODE");
Ajouter le montant de règlement du commerçant	<p><i>Tableau chaîne</i></p> <p>12 caractères numériques, 3 caractères numériques</p> <p>(montant en centimes de dollars canadiens, code de devise)</p>	rate.addMerchantSettlementAmount ("CAD_AMOUNT", "FOREIGN_CURRENCY_CODE");

### Exemple d'obtention du taux de la TMD

```

package Canada;
import JavaAPI.*;
public class TestCanadaMCPGetRate
{
    public static void main(String[] args)
    {
        String store_id = "store5";
        String api_token = "yesguy";
        String processing_country_code = "CA";

        MCPGetRate getRate = new MCPGetRate();
        getRate.setMCPVersion("1.0"); //MCP Version number. Should always be 1.0
        getRate.setRateTxnType("P"); //P or R are valid values (Purchase or Refund)

        MCPRate rate = new MCPRate();
        rate.addCardholderAmount("500", "840"); //penny value amount 1.25 = 125. Foreign amount and
        //SO-4217 country currency number
        //rate.addMerchantSettlementAmount("200", "826"); //penny value amount 1.25 = 125.
        Domestic(CAD) amount and SO-4217 country currency number
        //rate.addMerchantSettlementAmount("300", "036"); //penny value amount 1.25 = 125.
        Domestic(CAD) amount and SO-4217 country currency number

        getRate.setMCPRateInfo(rate);

        HttpsPostRequest mpgReq = new HttpsPostRequest();
        mpgReq.setProcCountryCode(processing_country_code);
        mpgReq.setTestMode(true); //false for production transactions
        mpgReq.setstoreId(store_id);
        mpgReq.setApiToken(api_token);
        mpgReq.setTransaction(getRate);
        mpgReq.send();

        try
        {
            Receipt receipt = mpgReq.getReceipt();
            System.out.println("RateTxnType = " + receipt.getRateTxnType());
        }
    }
}

```

```

System.out.println("MCPRateToken = " + receipt.getMCPRateToken());
System.out.println("RateInqStartTime = " + receipt.getRateInqStartTime()); //The time (unix UTC) of when the rate is requested
System.out.println("RateInqEndTime = " + receipt.getRateInqEndTime()); //The time (unix UTC) of when the rate is returned
System.out.println("RateValidityStartTime = " + receipt.getRateValidityStartTime()); //The time (unix UTC) of when the rate is valid from
System.out.println("RateValidityEndTime = " + receipt.getRateValidityEndTime()); //The time (unix UTC) of when the rate is valid until
System.out.println("RateValidityPeriod = " + receipt.getRateValidityPeriod()); //The time in minutes this rate is valid for

System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("Message = " + receipt.getMessage());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("TimedOut = " + receipt.getTimedOut());

//RateData
for (int index = 0; index < receipt.getRatesCount(); index++)
{
    System.out.println("MCPRate = " + receipt.getMCPRate(index));
    System.out.println("MerchantSettlementCurrency = " +
receipt.getMerchantSettlementCurrency(index));
    System.out.println("MerchantSettlementAmount = " +
receipt.getMerchantSettlementAmount(index)); //Domestic(CAD) amount
    System.out.println("CardholderCurrencyCode = " + receipt.getCardholderCurrencyCode(index));
    System.out.println("CardholderAmount = " + receipt.getCardholderAmount(index)); //Foreign amount

    System.out.println("MCPErrorStatusCode = " + receipt.getMCPErrorStatusCode(index));
    System.out.println("MCPErrorMessage = " + receipt.getMCPErrorMessage(index));
}

}
catch (Exception e)
{
e.printStackTrace();
}
}
}

```

## 7.4 Achat utilisant la TMD

Une transaction d'achat vérifie que les fonds sont présents sur la carte du client, retire ces fonds de la carte et les prépare à être déposés dans le compte du commerçant.

Cette demande de transaction est la version avec la tarification multidevise (TMD) activée de la transaction financière équivalente.

### Définition de l'objet de transaction MCP Purchase

```
MCPPurchase mcpPurchase = new MCPPurchase();
```

### Objet HttpsPostRequest pour les transactions d'achat utilisant la TMD

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
```

```
mpgReq.setTransaction(mcpPurchase);
```

## Principaux champs d'objet de connexion (toutes les transactions API)

Variable	Type et limites	Méthode Set
ID de commerce	<i>Chaîne</i> S. O.	mpgReq.setstoreId(store_id);
Jeton API	<i>Chaîne</i> S. O.	mpgReq.setApiToken(api_token);

## Champs de demande liés aux transactions d'achat utilisant la TMD (obligatoires)

Variable	Type et limites	Méthode Set
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques a-Z A-Z 0-9 _ - : . @ espaces	mcpPurchase.setorderId(order_id);
Numéro de carte de crédit	<i>Chaîne</i> Maximum 20 caractères alphanumériques	mcpPurchase.setPan(pan);
Date d'expiration	<i>Chaîne</i> 4 caractères alphanumériques AAMM	mcpPurchase.setExpDate(expiry_date);
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	mcpPurchase.setCryptType(crypt);

## Champs de demande liés aux transactions d'achat utilisant la TMD (obligatoires)

Variable	Type et limites	Méthode Set
Numéro de la version de la TMD	<i>Chaîne</i> Valeur numérique La version actuelle est 1.0	mcpPurchase.setMCPVersion("MCP_VERSION_NUM");

Montant du titulaire de carte	<i>Chaîne</i> 12 caractère numérique La plus petite unité discrète de devise étrangère	mcpPurchase.setCardholderAmount ("CARDHOLDER_AMOUNT");
Code de devise du titulaire de carte	<i>Chaîne</i> 3 caractère numérique	mcpPurchase.setCardholderCurrencyCode ("CARDHOLDER_CURRENCY_CODE");

### Champs de demande liés aux transactions d'achat utilisant la TMD (facultatifs)

Variable	Type et limites	Méthode Set
ID de client	<i>Chaîne</i> 50 caractères alphanumériques  <div style="background-color: #e0f2fd; padding: 5px;"><b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : &lt;&gt;\$ % = ? ^ { } [ ] \</div>	mcpPurchase.setCustomerId(cust_id);
Descripteur dynamique	<i>Chaîne</i> 20 caractères alphanumériques  Total de 22 caractères incluant votre nom de commerçant et un séparateur  <div style="background-color: #e0f2fd; padding: 5px;"><b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : &lt;&gt;\$ % = ? ^ { } [ ] \</div>	mcpPurchase.setDynamicDescriptor(dynamic_descriptor);
Indicateur de portefeuille électronique	<i>Chaîne</i> 3 caractères alphanumériques	mcpPurchase.setWalletIndicator(wallet_indicator);
Renseignements d'identification au dossier cof	<i>Objet</i> S. O.	mcpPurchase.setCofInfo(cof);

**REMARQUE :** Cette variable est un objet imbriqué au sein d'une transaction et elle est requise lors du stockage des renseignements d'identification d'un client ou lors de l'utilisation de tels renseignements stockés. Pour plus de renseignements sur les champs de l'objet Credential on File Info, consultez la section Objet Credential File Info et variables.

Renseignements du SVA	<i>Objet</i> S. O.	mcpPurchase.setAvsInfo(avsCheck);
Renseignements du NVC	<i>Objet</i> S. O.	mcpPurchase.setCvdInfo(cvdCheck);

#### Champs de demande liés aux transactions d'achat utilisant la TMD (facultatifs)

Variable	Type et limites	Méthode Set
Jeton du taux de la TMD	<i>Chaîne</i> S. O.	mcpPurchase.setMCPRateToken("MCP_RATE_TOKEN");

#### Exemple d'achat utilisant la TMD

```

package Canada;
import JavaAPI.*;
public class TestCanadaMCPurchase
{
    public static void main(String[] args)
    {
        java.util.Date createDate = new java.util.Date();
        String order_id = "Test"+createDate.getTime();
        String store_id = "store5";
        String api_token = "yesguy";
        String amount = "5.00";
        String pan = "4242424242424242";
        String expdate = "1901"; //YYMM format
        String crypt = "7";
        String processing_country_code = "CA";
        boolean status_check = false;
        MCPPurchase mcpPurchase = new MCPPurchase();
        mcpPurchase.setOrderId(order_id);
        mcpPurchase.setAmount(amount);
        mcpPurchase.setPan(pan);
        mcpPurchase.setExpdate(expdate);
        mcpPurchase.setCryptType(crypt);
        mcpPurchase.setDynamicDescriptor("123456");
        //mcpPurchase.setWalletIndicator(""); //Refer documentation for possible values
        //optional - Credential on File details
        CofInfo cof = new CofInfo();
        cof.setPaymentIndicator("U");
        cof.setPaymentInformation("2");
        cof.setIssuerId("139X3130ASCXAS9");

        //mcpPurchase.setCofInfo(cof);
    }
}

```

```

//MCP Fields
mcpPurchase.setMCPVersion("1.0");
mcpPurchase.setCardholderAmount("500");
mcpPurchase.setCardholderCurrencyCode("840");
mcpPurchase.setMCPRateToken("P1538681661706745");

HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false for production transactions
mpgReq.setStoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(mcpPurchase);
mpgReq.setStatusCheck(status_check);

//Optional - Proxy
mpgReq.setProxy(false); //true to use proxy
mpgReq.setProxyHost("proxyURL");
mpgReq.setProxyPort("proxyPort");
mpgReq.setProxyUser("proxyUser"); //optional - domainName\user
mpgReq.setProxyPassword("proxyPassword"); //optional
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = " + receipt.getBankTotals());
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedout());
System.out.println("IsVisaDebit = " + receipt.getIsVisaDebit());
System.out.println("HostId = " + receipt.getHostId());
System.out.println("IssuerId = " + receipt.getIssuerId());

System.out.println("MerchantSettlementAmount = " +
receipt.getMerchantSettlementAmount());
System.out.println("CardholderAmount = " + receipt.getCardholderAmount());
System.out.println("CardholderCurrencyCode = " + receipt.getCardholderCurrencyCode());
System.out.println("MCPRate = " + receipt.getMCPRate());
System.out.println("MCPErrorStatusCode = " + receipt.getMCPErrorStatusCode());
System.out.println("MCPErrorMessage = " + receipt.getMCPErrorMessage());
System.out.println("HostId = " + receipt.getHostId());
}
catch (Exception e)
{
e.printStackTrace();
}
}
}

```

## 7.5 Achat utilisant la TMD avec 3-D Secure

### Définition de l'objet de transaction MCP Purchase with 3-D Secure

```
MCP CavvPurchase mcpCavvPurchase = new MCP CavvPurchase();
```

### Objet HttpsPostRequest pour les transactions d'achat utilisant la TMD avec 3-D Secure

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
```

```
mpgReq.setTransaction(mcpCavvPurchase);
```

### Principaux champs d'objet de connexion (toutes les transactions API)

Variable	Type et limites	
ID de commerce	<i>Chaîne</i> S. O.	mpgReq.setstoreId(store_id);
Jeton API	<i>Chaîne</i> S. O.	mpgReq.setApiToken(api_token);

**Champs de demande liés aux transactions d'achat utilisant la TMD avec 3-D Secure (obligatoires)**

Variable	Type et limites	Méthode Set
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques a-Z A-Z 0-9 _ - : . @ espaces	mcpCavvPurchase.setorderId(order_id);
Numéro de carte de crédit	<i>Chaîne</i> Maximum 20 caractères alphanumériques	mcpCavvPurchase.setPan(pan);
Date d'expiration	<i>Chaîne</i> 4 caractères alphanumériques AAMM	mcpCavvPurchase.setExpDate(expiry_date);
Code de vérification d'authentification du titulaire de carte	<i>Chaîne</i> 50 caractères alphanumériques	mcpCavvPurchase.setCavv(cavv);
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	mcpCavvPurchase.setCryptType(crypt);

## Champs de demande liés aux transactions d'achat utilisant la TMD (obligatoires)

Variable	Type et limites	Méthode Set
Numéro de la version de la TMD	<p><i>Chaîne</i></p> <p>Valeur numérique</p> <p>La version actuelle est 1.0</p>	mcpCavvPurchase.setMCPVersion("MCP_VERSION_NUM");
Montant du titulaire de carte	<p><i>Chaîne</i></p> <p>12 caractère numérique</p> <p>La plus petite unité discrète de devise étrangère</p>	mcpCavvPurchase.setCardholderAmount("CARDHOLDER_AMOUNT");
Code de devise du titulaire de carte	<p><i>Chaîne</i></p> <p>3 caractère numérique</p>	mcpCavvPurchase.setCardholderCurrencyCode("CARDHOLDER_CURRENCY_CODE");

## 3-D Secure 2.2 – Champs particuliers (obligatoires)

Variable	Type et limites	Méthode Set
Version de 3DS	<p><i>Chaîne</i></p> <p>10 caractères numériques</p> <p><b>REMARQUE :</b> Obligatoire pour les transactions utilisant les services de 3-D Secure d'un tiers</p>	mcpCavvPurchase.setThreeDSVersion("ThreeDSVersion"); Valeurs acceptées : 2.0.0 = protocole 3DS 2.0.0 2.1.0 = protocole 3DS 2.1.0 2.2.0 = protocole 3DS 2.2.0 2.3.0 = protocole 3DS 2.3.0
ID de transaction du serveur 3DS	<p><i>Chaîne</i></p> <p>36 caractère numérique</p> <p><b>REMARQUE :</b> Obligatoire pour les transactions utilisant les services de 3-D Secure d'un tiers, obtenue à partir de la demande de recherche code de vérification d'authentification du titulaire de carte ou de la demande d'authentification 3DS des modules d'extension pour les commerçants.</p>	mcpCavvPurchase.setThreeDSServerTransId("ThreeDSServerTransId");

## Champs de demande liés aux transactions d'achat utilisant la TMD avec 3-D Secure (facultatif)

Variable	Type et limites	Méthode Set
ID de client	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p> <div style="background-color: #e0f2f1; padding: 5px;"> <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : &lt;&gt;\$ % = ? ^{ } [ ] \</div>	mcpCavvPurchase.setCustomerId(cust_id);
Descripteur dynamique	<p><i>Chaîne</i></p> <p>20 caractères alphanumériques</p> <p>Total de 22 caractères incluant votre nom de commerçant et un séparateur</p> <div style="background-color: #e0f2f1; padding: 5px;"> <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : &lt;&gt;\$ % = ? ^{ } [ ] \</div>	mcpCavvPurchase.setDynamicDescriptor(dynamic_descriptor);
Indicateur de portefeuille électronique	<p><i>Chaîne</i></p> <p>3 caractères alphanumériques</p>	mcpCavvPurchase.setWalletIndicator(wallet_indicator);
Renseignements d'identification au dossier cof	<p><i>Objet</i></p> <p>S. O.</p> <div style="background-color: #e0f2f1; padding: 5px;"> <b>REMARQUE :</b> Cette variable est un objet imbriqué au sein d'une transaction et elle est requise lors du stockage des renseignements d'identification d'un client ou lors de l'utilisation de tels renseignements stockés. Pour plus de renseignements sur les champs de l'objet Credential on File Info, consultez la section Objet Credential File Info et variables.</div>	mcpCavvPurchase.setCofInfo(cof);
Renseignements du SVA	<p><i>Objet</i></p> <p>S. O.</p>	mcpCavvPurchase.setAvsInfo(avscheck);

Renseignements du NVC	<i>Objet</i> S. O.	mcpCavvPurchase.setCvdInfo(cvdCheck);
-----------------------	-----------------------	---------------------------------------

### Champs de demande liés aux transactions d'achat utilisant la TMD (facultatifs)

Variable	Type et limites	Méthode Set
Jeton du taux de la TMD	<i>Chaîne</i> S. O.	mcpCavvPurchase.setMCPRateToken("MCP_RATE_TOKEN");

### 3-D Secure 2.2 – Champs particuliers (facultatifs)

Variable	Type et limites	Méthode Set
ID de transaction DS	<i>Chaîne</i> 36 caractères alphanumériques  <b>REMARQUE :</b> Uniquement utilisé dans les transactions financières utilisant les services 3-D Secure d'un tiers	mcpCavvPurchase.setDsTransId("DsTransId");

### Exemple d'achat utilisant la TMD avec 3-D Secure

```

package Canada;
import JavaAPI.*;
public class TestCanadaMCP CavvPurchase
{
    public static void main(String[] args)
    {
        String store_id = "store1";
        String api_token = "yesguy1";
        java.util.Date createDate = new java.util.Date();
        String order_id = "Test"+createDate.getTime();
        String cust_id = "CUS887H67";
        String amount = "10.42";
        String pan = "4740611374762707";
        String expdate = "2201"; //YYMM
        String cavv = "BwABApFSYyd412eQQFJjAAAAAA=";
        String dynamic_descriptor = "123456";
        String processing_country_code = "CA";
        String crypt_type = "5";
        boolean status_check = false;
        MCP CavvPurchase mcpCavvPurchase = new MCP CavvPurchase();
        mcpCavvPurchase.setOrderId(order_id);
        mcpCavvPurchase.setCustId(cust_id);
        mcpCavvPurchase.setAmount(amount);
        mcpCavvPurchase.setPan(pan);
        mcpCavvPurchase.setExpdate(expdate);
        mcpCavvPurchase.setCavv(cavv);
        mcpCavvPurchase.setCryptType(crypt_type); //Mandatory for AMEX only
        mcpCavvPurchase.setDynamicDescriptor(dynamic_descriptor);
        //mcpCavvPurchase.setWalletIndicator("APP"); //set only for wallet transactions. e.g APPLE PAY
        //mcpCavvPurchase.setNetwork("Interac"); //set only for Interac e-commerce
        //mcpCavvPurchase.setData Type("3DSecure"); //set only for Interac e-commerce
        //mcpCavvPurchase.setCmId("8nAK8712sGaAkls56"); //set only for usage with Offlinx - Unique max 50 alphanumeric characters transaction id generated by merchant

        mcpCavvPurchase.setThreeDSVersion("2.2.0"); //Mandatory for financial transactions using 3rd Party 3-D Secure services.
        mcpCavvPurchase.setThreeDS ServerTransId("e11d4985-8d25-40ed-99d6-c3803fe5e68f"); //Mandatory for financial transactions using 3rd Party 3-D Secure services - obtained from MpiCavvLookup or MpiThreeDSAuthentication
        //mcpCavvPurchase.setDsTransId("12345"); //Optional - to be used only if you are using 3rd party 3ds 2.0 service
    }
}

```

```

//MCP Fields
mcpCavvPurchase.setMCPVersion("1.0");
mcpCavvPurchase.setCardholderAmount("500");
mcpCavvPurchase.setCardholderCurrencyCode("840");
mcpCavvPurchase.setMCPRateToken("P1623162875163626");

//optional - Credential on File details
CofInfo cof = new CofInfo();
cof.setPaymentIndicator("U");
cof.setPaymentInformation("2");
cof.setIssuerId("139X3130ASCXAS9");

mcpCavvPurchase.setCofInfo(cof);
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false for production transactions
mpgReq.setstoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(mcpCavvPurchase);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = " + receipt.getBankTotals());
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("CavvResultCode = " + receipt.getCavvResultCode());
System.out.println("IssuerId = " + receipt.getIssuerId());
System.out.println("ThreeDSVersion = " + receipt.getThreeDSVersion());

System.out.println("MerchantSettlementAmount = " + receipt.getMerchantSettlementAmount());
System.out.println("CardholderAmount = " + receipt.getCardholderAmount());
System.out.println("CardholderCurrencyCode = " + receipt.getCardholderCurrencyCode());
System.out.println("MCPRate = " + receipt.getMCPRate());
System.out.println("MCPErrorStatusCode = " + receipt.getMCPErrorStatusCode());
System.out.println("MCPErrorMessage = " + receipt.getMCPErrorMessage());
System.out.println("HostId = " + receipt.getHostId());
}
catch (Exception e)
{
e.printStackTrace();
}
}
}

```

## 7.6 Achat utilisant la TMD avec 3-D Secure et la chambre forte

Définition de l'objet transaction MCP Purchase with 3-D Secure and Vault

```
MCPResCavvPurchaseCC mcpResCavvPurchaseCC = new MCPResCavvPurchaseCC();
```

### Objet HttpsPostRequest pour les transactions d'achat utilisant la TMD avec 3-D Secure et la chambre forte

```
HttpsPostRequest mpgReq = new HttpsPostRequest();  
  
mpgReq.setTransaction(mcpResCavvPurchaseCC);
```

#### Principaux champs d'objet de connexion (toutes les transactions API)

Variable	Type et limites	
ID de commerce	<i>Chaîne</i> S. O.	mpgReq.setstoreId(store_id);
Jeton API	<i>Chaîne</i> S. O.	mpgReq.setApiToken(api_token);

#### Champs de demande liés aux transactions d'achat utilisant la TMD avec 3-D Secure et la chambre forte (obligatoires)

Variable	Type et limites	Méthode Set
Clé de données	<i>Chaîne</i> 25 caractères alphanumériques	mcpResCavvPurchaseCC.setData(data_key);
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques a-Z A-Z 0-9 _ - : . @ espaces	mcpResCavvPurchaseCC.setOrderId(order_id);
Numéro de carte de crédit	<i>Chaîne</i> Maximum 20 caractères alphanumériques	mcpResCavvPurchaseCC.setPan(pan);

Date d'expiration	<i>Chaîne</i> 4 caractères alphanumériques AAMM	mcpResCavvPurchaseCC.setExpDate(expiry_date);
Code de vérification d'authentification du titulaire de carte	<i>Chaîne</i> 50 caractères alphanumériques	mcpResCavvPurchaseCC.setCavv(cavv);
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	mcpResCavvPurchaseCC.setCryptType(crypt);

#### Champs de demande liés aux transactions d'achat utilisant la TMD (obligatoires)

Variable	Type et limites	Méthode Set
Numéro de la version de la TMD	<i>Chaîne</i> Valeur numérique La version actuelle est 1.0	mcpResCavvPurchaseCC.setMCPVersion("MCP_VERSION_NUM");
Montant du titulaire de carte	<i>Chaîne</i> 12 caractère numérique La plus petite unité discrète de devise étrangère	mcpResCavvPurchaseCC.setCardholderAmount("CARDHOLDER_AMOUNT");
Code de devise du titulaire de carte	<i>Chaîne</i> 3 caractère numérique	mcpResCavvPurchaseCC.setCardholderCurrencyCode("CARDHOLDER_CURRENCY_CODE");

#### 3-D Secure 2.2 – Champs particuliers (obligatoires)

Variable	Type et limites	Méthode Set
Version de 3DS	<i>Chaîne</i> 10 caractères numériques	mcpResCavvPurchaseCC.setThreeDSVersion("ThreeDSVersion");  Valeurs acceptées : 2.0.0 = protocole 3DS 2.0.0 2.1.0 = protocole 3DS 2.1.0 2.2.0 = protocole 3DS 2.2.0 2.3.0 = protocole 3DS 2.3.0
REMARQUE : Obligatoire pour les transactions utilisant les services de 3-D Secure d'un tiers		
ID de transaction du serveur 3DS	<i>Chaîne</i>	mcpResCavvPurchaseCC.setThreeDSServerTransId("ThreeDSServerTr

**REMARQUE :** Obligatoire pour les transactions utilisant les services de 3-D Secure d'un tiers, obtenue à partir de la demande de recherche code de vérification d'authentification du titulaire de carte ou de la demande d'authentification 3DS des modules d'extension pour les commerçants.

36 caractère numérique

```
ansId");
```

### Champs de demande liés aux transactions d'achat utilisant la TMD avec 3-D Secure et la chambre forte (facultatifs)

Variable	Type et limites	Méthode Set
ID de client	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p> <div style="background-color: #e0f2fd; padding: 5px;"> <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés :            &lt;&gt;\$ % = ? ^ { } [ ] \          </div>	<code>mcpResCavvPurchaseCC.setCustomerId(cust_id);</code>
Descripteur dynamique	<p><i>Chaîne</i></p> <p>20 caractères alphanumériques</p> <p>Total de 22 caractères incluant votre nom de commerçant et un séparateur</p> <div style="background-color: #e0f2fd; padding: 5px;"> <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés :            &lt;&gt;\$ % = ? ^ { } [ ] \          </div>	<code>mcpResCavvPurchaseCC.setDynamicDescriptor(dynamic_descriptor);</code>
Indicateur de portefeuille électronique	<p><i>Chaîne</i></p> <p>3 caractères alphanumériques</p>	<code>mcpResCavvPurchaseCC.setWalletIndicator(wallet_indicator);</code>
Renseignements d'identification au dossier cof	<p><i>Objet</i></p> <p>S. O.</p>	<code>mcpResCavvPurchaseCC.setCofInfo(cof);</code>

**REMARQUE :** Cette variable est un objet imbriqué au sein d'une transaction et elle est requise lors du stockage des renseignements d'identification d'un client ou lors de l'utilisation de tels renseignements stockés. Pour plus de renseignements sur les champs de l'objet Credential on File Info, consultez la section Objet Credential File Info et variables.

Renseignements du SVA	<i>Objet</i> S. O.	mcpResCavvPurchaseCC.setAvsInfo(avsCheck);
Renseignements du NVC	<i>Objet</i> S. O.	mcpResCavvPurchaseCC.setCvdInfo(cvdCheck);

### 3-D Secure 2.2 – Champs particuliers (facultatifs)

Variable	Type et limites	Méthode Set
ID de transaction DS  <b>REMARQUE :</b> Uniquement utilisé dans les transactions financières utilisant les services 3-D Secure d'un tiers	<i>Chaîne</i> 36 caractères alphanumériques	mcpResCavvPurchaseCC.setDsTransactionId("DsTransId");

### Exemple d'achat utilisant la TMD avec 3-D Secure et la chambre forte

```

package Canada;
import JavaAPI.*;
public class TestCanadaMCPResCavvPurchaseCC
{
    public static void main(String[] args)
    {
        String store_id = "monca02932";
        String api_token = "CG8kYzGgzVU5z23irgMx";
        String data_key = "xRl9O4FgrZUYdGkmqHTqiEw97";
        java.util.Date createDate = new java.util.Date();
        String order_id = "Test"+createDate.getTime();
        String amount = "1.00";
        String cust_id = "customer1"; //if sent will be submitted, otherwise cust_id from profile
        will be used
        String cavv = "AAABBjg0VhI0VniQEjRWAAAAAA";
        String processing_country_code = "CA";
        String exp_date = "1901";
        String crypt_type = "7";
        boolean status_check = false;
        MCPResCavvPurchaseCC mcpResCavvPurchaseCC = new MCPResCavvPurchaseCC();
        mcpResCavvPurchaseCC.setOrderId(order_id);
        mcpResCavvPurchaseCC.setDataKey(data_key);
        mcpResCavvPurchaseCC.setCustomerId(cust_id);
        mcpResCavvPurchaseCC.setAmount(amount);
        mcpResCavvPurchaseCC.setCavv(cavv);
        mcpResCavvPurchaseCC.setCryptType(crypt_type);
        mcpResCavvPurchaseCC.setExpDate(exp_date);
        mcpResCavvPurchaseCC.setThreeDSVersion("2.2.0"); //Mandatory for financial transactions
        using 3rd Party 3-D Secure services.
        mcpResCavvPurchaseCC.setThreeDServerTransId("e11d4985-8d25-40ed-99d6-c3803fe5e68f");
        //Mandatory for financial transactions using 3rd Party 3-D Secure services" - obtained from
        MpiCavvLookup or MpiThreeDSAuthentication
        //MCP Fields
    }
}

```

```

mcpResCavvPurchaseCC.setMCPVersion("1.0");
mcpResCavvPurchaseCC.setCardholderAmount("500");
mcpResCavvPurchaseCC.setCardholderCurrencyCode("840");
mcpResCavvPurchaseCC.setMCPRateToken("P1623439175449463");

//Mandatory - Credential on File details
CofInfo cof = new CofInfo();
cof.setPaymentIndicator("U");
cof.setPaymentInformation("2");
cof.setIssuerId("139X3130ASCXAS9");

mcpResCavvPurchaseCC.setCofInfo(cof);

HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false for production transactions
mpgReq.setStoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(mcpResCavvPurchaseCC);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("DataKey = " + receipt.getDataKey());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Message = " + receipt.getMessage());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("ResSuccess = " + receipt.getResSuccess());
System.out.println("PaymentType = " + receipt.getPaymentType());
System.out.println("CavvResultCode = " + receipt.getCavvResultCode());
System.out.println("IssuerId = " + receipt.getIssuerId());
System.out.println("ThreeDSVersion = " + receipt.getThreeDSVersion());

System.out.println("MerchantSettlementAmount = " + receipt.getMerchantSettlementAmount());
System.out.println("CardholderAmount = " + receipt.getCardholderAmount());
System.out.println("CardholderCurrencyCode = " + receipt.getCardholderCurrencyCode());
System.out.println("MCPRate = " + receipt.getMCPRate());
System.out.println("MCPErrorStatusCode = " + receipt.getMCPErrorStatusCode());
System.out.println("MCPErrorMessage = " + receipt.getMCPErrorMessage());
System.out.println("HostId = " + receipt.getHostId());
//ResolveData
System.out.println("Cust ID = " + receipt.getResCustomerId());
System.out.println("Phone = " + receipt.getResPhone());
System.out.println("Email = " + receipt.getResEmail());
System.out.println("Note = " + receipt.getResNote());
System.out.println("Masked Pan = " + receipt.getResMaskedPan());
System.out.println("Exp Date = " + receipt.getResExdate());
System.out.println("Crypt Type = " + receipt.getResCryptType());
System.out.println("Avs Street Number = " + receipt.getResAvsStreetNumber());
System.out.println("Avs Street Name = " + receipt.getResAvsStreetName());
System.out.println("Avs Zipcode = " + receipt.getResAvsZipcode());
}
catch (Exception e)
{
e.printStackTrace();
}
}
}

```

## 7.7 Préautorisation utilisant la TMD

Une transaction de préautorisation vérifie que les fonds sont présents sur la carte de crédit du client et les bloque. Les fonds sont bloqués pendant une durée déterminée par l'émetteur de la carte. Pour récupérer les fonds bloqués par une transaction de préautorisation afin de les déposer dans le compte du commerçant, une transaction de conclusion de préautorisation doit être effectuée. On ne peut conclure une préautorisation qu'une seule fois.

Cette demande de transaction est la version avec la tarification multidevise (TMD) activée de la transaction financière équivalente.

### Définition de l'objet de transaction MCP Pre-Authorization

```
MCPPreAuth mcpPreauth = new MCPPreAuth();
```

### Objet HttpsPostRequest pour les transactions de préautorisation utilisant la TMD

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
```

```
mpgReq.setTransaction(mcpPreauth);
```

### Principaux champs d'objet de connexion (toutes les transactions API)

Variable	Type et limites	
ID de commerce	<i>Chaîne</i> S. O.	mpgReq.setstoreId(store_id);
Jeton API	<i>Chaîne</i> S. O.	mpgReq.setApiToken(api_token);

### Champs de demande pour les transactions de préautorisation utilisant la TMD (obligatoires)

Variable	Type et limites	Méthode Set
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques a-Z A-Z 0-9 _ - : . @ espaces	mcpPreauth.setorderId(order_id);
Numéro de carte de crédit	<i>Chaîne</i> Maximum 20 caractères alphanumériques	mcpPreauth.setPan(pan);

Date d'expiration	<i>Chaîne</i> 4 caractères alphanumériques AAMM	mcpPreauth.setExpDate(expiry_date);
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	mcpPreauth.setCryptType(crypt);

#### Champs de demande liés aux transactions d'achat utilisant la TMD (obligatoires)

Variable	Type et limites	Méthode Set
Numéro de la version de la TMD	<i>Chaîne</i> Valeur numérique La version actuelle est 1.0	mcpPreauth.setMCPVersion("MCP_VERSION_NUM");
Montant du titulaire de carte	<i>Chaîne</i> 12 caractère numérique La plus petite unité discrète de devise étrangère	mcpPreauth.setCardholderAmount("CARDHOLDER_AMOUNT");
Code de devise du titulaire de carte	<i>Chaîne</i> 3 caractère numérique	mcpPreauth.setCardholderCurrencyCode("CARDHOLDER_CURRENCY_CODE");

#### Champs de demande pour les transactions de préautorisation utilisant la TMD (facultatifs)

Variable	Type et limites	Méthode Set
ID de client	<i>Chaîne</i> 50 caractères alphanumériques  <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : <>\$ % = ? ^ { } [ ] \	mcpPreauth.setCustId(cust_id);
Descripteur dynamique	<i>Chaîne</i>	mcpPreauth.setDynamicDescriptor(dynamic_descriptor);

<p><b>Pour les transactions de préautorisation</b> <b>REMARQUE :</b> La valeur du champ Dynamic descriptor est uniquement transférée à une conclusion de préautorisation lorsque cette dernière est effectuée par l'entremise du centre de ressources pour commerçants. Autrement, la valeur du champ doit être renvoyée durant la conclusion de préautorisation.</p>	<p>20 caractères alphanumériques</p> <p>Total de 22 caractères incluant votre nom de commerçant et un séparateur</p> <p><b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : &lt;&gt; \$ % = ? ^ { } [ ] \</p>	
<p>Indicateur de portefeuille électronique</p>	<p><i>Chaîne</i></p> <p>3 caractères alphanumériques</p>	<pre>mcpPreauth.setWalletIndicator(wallet_indicator);</pre>
<p>Autorisation finale</p> <p><b>REMARQUE :</b> Applicable uniquement aux transactions par carte Mastercard</p>	<p><i>Chaîne</i></p> <p>true/false</p>	<pre>mcpPreauth.setFinalAuth("true");</pre>
<p>Renseignements d'identification au dossier cof</p> <p><b>REMARQUE :</b> Cette variable est un objet imbriqué au sein d'une transaction et elle est requise lors du stockage des renseignements d'identification d'un client ou lors de l'utilisation de tels renseignements stockés. Pour plus de renseignements sur les champs de l'objet Credential on File Info, consultez la section Objet Credential File Info et variables.</p>	<p><i>Objet</i></p> <p>S. O.</p>	<pre>mcpPreauth.setCofInfo(cof);</pre>
<p>Renseignements du SVA</p>	<p><i>Objet</i></p> <p>S. O.</p>	<pre>mcpPreauth.setAvsInfo(avscCheck);</pre>
<p>Renseignements du NVC</p>	<p><i>Objet</i></p> <p>S. O.</p>	<pre>mcpPreauth.setCvdInfo(cvdCheck);</pre>

#### Champs de demande liés aux transactions d'achat utilisant la TMD (facultatifs)

Variable	Type et limites	Méthode Set

Jeton du taux de la TMD	<i>Chaîne</i> S. O.	mcpPreauth.setMCPRateToken ("MCP _RATE_TOKEN");
-------------------------	------------------------	--

## Exemple de préautorisation utilisant la TMD

```

package Canada;
import JavaAPI.*;
public class TestCanadaMCPPreauth
{
    public static void main(String[] args)
    {
        String store_id = "store5";
        String api_token = "yesguy";
        java.util.Date createDate = new java.util.Date();
        String order_id = "Test"+createDate.getTime();
        String amount = "5.00";
        String pan = "4242424242424242";
        String expdate = "1902";
        String crypt = "7";
        String processing_country_code = "CA";
        boolean status_check = false;
        MCPPreAuth mcpPreauth = new MCPPreAuth();
        mcpPreauth.setOrderId(order_id);
        mcpPreauth.setAmount(amount);
        mcpPreauth.setPan(pan);
        mcpPreauth.setExpdate(expdate);
        mcpPreauth.setCryptType(crypt);
        //mcpPreauth.setWalletIndicator(""); //Refer documentation for possible values
        //mcpPreauth.setFinalAuth("true");
        //optional - Credential on File details
        CofInfo cof = new CofInfo();
        cof.setPaymentIndicator("U");
        cof.setPaymentInformation("2");
        cof.setIssuerId("139X3130ASCXAS9");

        //mcpPreauth.setCofInfo(cof);

        //MCP Fields
        mcpPreauth.setMCPVersion("1.0");
        mcpPreauth.setCardholderAmount("500");
        mcpPreauth.setCardholderCurrencyCode("840");
        mcpPreauth.setMCPRateToken("P1538681661706745");

        HttpsPostRequest mpgReq = new HttpsPostRequest();
        mpgReq.setProcCountryCode(processing_country_code);
        mpgReq.setTestMode(true); //false for production transactions
        mpgReq.setstoreId(store_id);
        mpgReq.setApiToken(api_token);
        mpgReq.setTransaction(mcpPreauth);
        mpgReq.setStatusCheck(status_check);
        mpgReq.send();
        try
        {
            Receipt receipt = mpgReq.getReceipt();
            System.out.println("CardType = " + receipt.getCardType());
            System.out.println("TransAmount = " + receipt.getTransAmount());
            System.out.println("TxnNumber = " + receipt.getTxnNumber());
            System.out.println("ReceiptId = " + receipt.getReceiptId());
            System.out.println("TransType = " + receipt.getTransType());
            System.out.println("ReferenceNum = " + receipt.getReferenceNum());
            System.out.println("ResponseCode = " + receipt.getResponseCode());
            System.out.println("ISO = " + receipt.getISO());
            System.out.println("BankTotals = " + receipt.getBankTotals());
            System.out.println("Message = " + receipt.getMessage());
            System.out.println("AuthCode = " + receipt.getAuthCode());
            System.out.println("Complete = " + receipt.getComplete());
            System.out.println("TransDate = " + receipt.getTransDate());
            System.out.println("TransTime = " + receipt.getTransTime());
            System.out.println("Ticket = " + receipt.getTicket());
            System.out.println("TimedOut = " + receipt.getTimedOut());
            System.out.println("IsVisaDebit = " + receipt.getIsVisaDebit());
            //System.out.println("StatusCode = " + receipt.getStatusCode());
            //System.out.println("StatusMessage = " + receipt.getStatusMessage());
            System.out.println("IssuerId = " + receipt.getIssuerId());

            System.out.println("MerchantSettlementAmount = " + receipt.getMerchantSettlementAmount());
        }
    }
}

```

```
System.out.println("CardholderAmount = " + receipt.getCardholderAmount());
System.out.println("CardholderCurrencyCode = " + receipt.getCardholderCurrencyCode());
System.out.println("MCPRate = " + receipt.getMCPRate());
System.out.println("MCPErrorStatusCode = " + receipt.getMCPErrorStatusCode());
System.out.println("MCPErrorMessage = " + receipt.getMCPErrorMessage());
System.out.println("HostId = " + receipt.getHostId());
}
catch (Exception e)
{
e.printStackTrace();
}
}
```

## 7.8 Préautorisation utilisant la TMD avec 3-D Secure

### MCP Définition de l'objet de transaction MCP Pre-Authorization with 3-D Secure

```
MCP CavvPreAuth mcpCavvPreauth = new MCP CavvPreAuth();
```

### Objet HttpsPostRequest pour les transactions de préautorisation utilisant la TMD avec la chambre forte

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
```

```
mpgReq.setTransaction(mcpCavvPreauth);
```

### Principaux champs d'objet de connexion (toutes les transactions API)

Variable	Type et limites	
ID de commerce	<i>Chaîne</i> S. O.	mpgReq.setstoreId(store_id);
Jeton API	<i>Chaîne</i> S. O.	mpgReq.setApiToken(api_token);

### Champs de demande pour les transactions de préautorisation utilisant la TMD avec la chambre forte (obligatoires)

Variable	Type et limites	Méthode Set
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques a-Z A-Z 0-9 _ - : . @ espaces	mcpCavvPreauth.setorderId(order_id);
Numéro de carte de crédit	<i>Chaîne</i> Maximum 20 caractères alphanumériques	mcpCavvPreauth.setPan(pan);
Date d'expiration	<i>Chaîne</i>	mcpCavvPreauth.setExpDate(expiry_date);

Variable	Type et limites	Méthode Set
	4 caractères alphanumériques AAMM	
Code de vérification d'authentification du titulaire de carte	<i>Chaîne</i> 50 caractères alphanumériques	<code>mcpCavvPreauth.setCavv(cavv);</code>
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	<code>mcpCavvPreauth.setCryptType(crypt);</code>

**Champs de demande liés aux transactions d'achat utilisant la TMD (obligatoires)**

Variable	Type et limites	Méthode Set
Numéro de la version de la TMD	<i>Chaîne</i> Valeur numérique La version actuelle est 1.0	<code>mcpCavvPreauth.setMCPVersion("MCP_VERSION_NUM");</code>
Montant du titulaire de carte	<i>Chaîne</i> 12 caractère numérique La plus petite unité discrète de devise étrangère	<code>mcpCavvPreauth.setCardholderAmount("CARDHOLDER_AMOUNT");</code>
Code de devise du titulaire de carte	<i>Chaîne</i> 3 caractère numérique	<code>mcpCavvPreauth.setCardholderCurrencyCode("CARDHOLDER_CURRENCY_CODE");</code>

### 3-D Secure 2.2 – Champs particuliers (obligatoires)

Variable	Type et limites	Méthode Set
Version de 3DS  <b>REMARQUE :</b> Obligatoire pour les transactions utilisant les services de 3-D Secure d'un tiers	<i>Chaîne</i> 10 caractères numériques	<code>mcpCavvPreauth.setThreeDSVersion("ThreeDSVersion");</code> Valeurs acceptées : 2.0.0 = protocole 3DS 2.0.0 2.1.0 = protocole 3DS 2.1.0 2.2.0 = protocole 3DS 2.2.0 2.3.0 = protocole 3DS 2.3.0
ID de transaction du serveur 3DS  <b>REMARQUE :</b> Obligatoire pour les transactions utilisant les services de 3-D Secure d'un tiers, obtenue à partir de la demande de recherche code de vérification d'authentification du titulaire de carte ou de la demande d'authentification 3DS des modules d'extension pour les commerçants.	<i>Chaîne</i> 36 caractère numérique	<code>mcpCavvPreauth.setThreeDSServerTransId("ThreeDSServerTransId");</code>

### Champs de demande pour les transactions de préautorisation utilisant la TMD avec la chambre forte (facultatifs)

Variable	Type et limites	Méthode Set
ID de client	<i>Chaîne</i> 50 caractères alphanumériques  <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\	<code>mcpCavvPreauth.setCustomerId(cust_id);</code>
Descripteur dynamique	<i>Chaîne</i> 20 caractères alphanumériques  Total de 22 caractères incluant votre nom de commerçant et un séparateur	<code>mcpCavvPreauth.setDynamicDescriptor(dynamic_descriptor);</code>

	<b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\	
Indicateur de portefeuille électronique	<i>Chaîne</i> 3 caractères alphanumériques	mcpCavvPreauth.setWalletIndicator(wallet_indicator);
Renseignements d'identification au dossier cof	<i>Objet</i> S. O.	mcpCavvPreauth.setCofInfo(cof);
<b>REMARQUE :</b> Cette variable est un objet imbriqué au sein d'une transaction et elle est requise lors du stockage des renseignements d'identification d'un client ou lors de l'utilisation de tels renseignements stockés. Pour plus de renseignements sur les champs de l'objet Credential on File Info, consultez la section Objet Credential File Info et variables.		
Renseignements du SVA	<i>Objet</i> S. O.	mcpCavvPreauth.setAvsInfo(avscCheck);
Renseignements du NVC	<i>Objet</i> S. O.	mcpCavvPreauth.setCvdInfo(cvdCheck);

#### Champs de demande liés aux transactions d'achat utilisant la TMD (facultatifs)

Variable	Type et limites	Méthode Set
Jeton du taux de la TMD	<i>Chaîne</i> S. O.	mcpCavvPreauth.setMCPRateToken("MCP_RATE_TOKEN");

### 3-D Secure 2.2 – Champs particuliers (facultatifs)

Variable	Type et limites	Méthode Set
ID de transaction DS  <b>REMARQUE :</b> Uniquement utilisé dans les transactions financières utilisant les services 3-D Secure d'un tiers	<i>Chaîne</i>  36 caractères alphanumériques	mcpCavvPreauth.setDsTransId("DsTransId");

### Exemple de préautorisation utilisant la TMD avec 3-D Secure

```

package Canada;
import JavaAPI.*;
public class TestCanadaMCP_CavvPurchase
{
    public static void main(String[] args)
    {
        String store_id = "store1";
        String api_token = "yesguy1";
        java.util.Date createDate = new java.util.Date();
        String order_id = "Test"+createDate.getTime();
        String cust_id = "CUS887H67";
        String amount = "10.42";
        String pan = "4740611374762707";
        String expdate = "2201"; //YYMM
        String cavv = "BwABApFSYyd4l2eQQFjAAAAAA=";
        String dynamic_descriptor = "123456";
        String processing_country_code = "CA";
        String crypt_type = "5";
        boolean status_check = false;
        MCP_CavvPurchase mcpCavvPurchase = new MCP_CavvPurchase();
        mcpCavvPurchase.setOrderId(order_id);
        mcpCavvPurchase.setCustId(cust_id);
        mcpCavvPurchase.setAmount(amount);
        mcpCavvPurchase.setPan(pan);
        mcpCavvPurchase.setExpdate(expdate);
        mcpCavvPurchase.setCavv(cavv);
        mcpCavvPurchase.setCryptType(crypt_type); //Mandatory for AMEX only
        mcpCavvPurchase.setDynamicDescriptor(dynamic_descriptor);
        //mcpCavvPurchase.setWalletIndicator("APP"); //set only for wallet transactions. e.g APPLE PAY
        //mcpCavvPurchase.setNetwork("Interac"); //set only for Interac e-commerce
        //mcpCavvPurchase.setData("3DSecure"); //set only for Interac e-commerce
        //mcpCavvPurchase.setCmId("8nAK8712sGaAkls56"); //set only for usage with Offlinx - Unique max 50 alphanumeric characters transaction id generated by merchant

        mcpCavvPurchase.setThreeDSVersion("2.2.0"); //Mandatory for financial transactions using 3rd Party 3-D Secure services
        mcpCavvPurchase.setThreeDSServerTransId("e11d4985-8d25-40ed-99d6-c3803fe5e68f"); Mandatory for financial transactions using 3rd Party 3-D Secure services - obtained from MpiCavvLookup or MpiThreeDSAuthentication
        //mcpCavvPurchase.setDsTransId("12345"); //Optional - to be used only if you are using 3rd party 3ds 2.0 service

        //MCP Fields
        mcpCavvPurchase.setMCPVersion("1.0");
        mcpCavvPurchase.setCardholderAmount("500");
        mcpCavvPurchase.setCardholderCurrencyCode("840");
        mcpCavvPurchase.setMCPRateToken("P1623162875163626");

        //optional - Credential on File details
        CofInfo cof = new CofInfo();
        cof.setPaymentIndicator("U");
        cof.setPaymentInformation("2");
        cof.setIssuerId("139X3130ASCXAS9");

        mcpCavvPurchase.setCofInfo(cof);
    }
}

```

```

HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false for production transactions
mpgReq.setstoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(mcpCavvPurchase);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = " + receipt.getBankTotals());
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("CavvResultCode = " + receipt.getCavvResultCode());
System.out.println("IssuerId = " + receipt.getIssuerId());
System.out.println("ThreeDSVersion = " + receipt.getThreeDSVersion());

System.out.println("MerchantSettlementAmount = " + receipt.getMerchantSettlementAmount());
System.out.println("CardholderAmount = " + receipt.getCardholderAmount());
System.out.println("CardholderCurrencyCode = " + receipt.getCardholderCurrencyCode());
System.out.println("MCPRate = " + receipt.getMCPRate());
System.out.println("MCPErrorStatusCode = " + receipt.getMCPErrorStatusCode());
System.out.println("MCPErrormessage = " + receipt.getMCPErrormessage());
System.out.println("HostId = " + receipt.getHostId());
}
catch (Exception e)
{
e.printStackTrace();
}
}
}

```

## 7.9 Préautorisation utilisant la TMD avec 3-D Secure et la chambre forte

### Définition de l'objet transaction MCP Pre-Authorization with 3-D Secure and Vault

```
MCPResCavvPreauthCC mcpResCavvPreauthCC = new MCPResCavvPreauthCC();
```

### Objet HttpsPostRequest pour les transactions de préautorisation utilisant la TMD avec 3-D Secure et la chambre forte

```

HttpsPostRequest mpgReq = new HttpsPostRequest();

mpgReq.setTransaction(mcpResCavvPreauthCC);

```

**Principaux champs d'objet de connexion (toutes les transactions API)**

Variable	Type et limites	
ID de commerce	<i>Chaîne</i> S. O.	mpgReq.setstoreId(store_id);
Jeton API	<i>Chaîne</i> S. O.	mpgReq.setApiToken(api_token);

**Champs de demande liés aux transactions de préautorisation utilisant la TMD avec 3-D Secure et la chambre forte (obligatoires)**

Variable	Type et limites	Méthode Set
Clé de données	<i>Chaîne</i> 25 caractères alphanumériques	mcpResCavvPreauthCC.setData(data_key);
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques a-Z A-Z 0-9 _ - : . @ espaces	mcpResCavvPreauthCC.setOrderId(order_id);
Numéro de carte de crédit	<i>Chaîne</i> Maximum 20 caractères alphanumériques	mcpResCavvPreauthCC.setPan(pan);
Date d'expiration	<i>Chaîne</i> 4 caractères alphanumériques AAMM	mcpResCavvPreauthCC.setExpDate(expiry_date);
Code de vérification d'authentification du titulaire de carte	<i>Chaîne</i> 50 caractères alphanumériques	mcpResCavvPreauthCC.setCavv(cavv);

Variable	Type et limites	Méthode Set
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	mcpResCavvPreauthCC.setCryptType(crypt);

**Champs de demande liés aux transactions d'achat utilisant la TMD (obligatoires)**

Variable	Type et limites	Méthode Set
Numéro de la version de la TMD	<i>Chaîne</i> Valeur numérique La version actuelle est 1.0	mcpResCavvPreauthCC.setMCPVersion("MCP_VERSION_NUM");
Montant du titulaire de carte	<i>Chaîne</i> 12 caractère numérique La plus petite unité discrète de devise étrangère	mcpResCavvPreauthCC.setCardholderAmount("CARDHOLDER_AMOUNT");
Code de devise du titulaire de carte	<i>Chaîne</i> 3 caractère numérique	mcpResCavvPreauthCC.setCardholderCurrencyCode("CARDHOLDER_CURRENCY_CODE");

**3-D Secure 2.2 – Champs particuliers (obligatoires)**

Variable	Type et limites	Méthode Set
Version de 3DS  <b>REMARQUE :</b> Obligatoire pour les transactions utilisant les services de 3-D Secure d'un tiers	<i>Chaîne</i> 10 caractères numériques	mcpResCavvPreauthCC.setThreeDSVersion("ThreeDSVersion");  Valeurs acceptées :  2.0.0 = protocole 3DS 2.0.0 2.1.0 = protocole 3DS 2.1.0 2.2.0 = protocole 3DS 2.2.0 2.3.0 = protocole 3DS 2.3.0
ID de transaction du serveur 3DS  <b>REMARQUE :</b> Obligatoire pour les transactions utilisant les services de 3-D Secure d'un tiers, obtenue à partir de la demande de recherche code de vérification d'authentification du titulaire de	<i>Chaîne</i> 36 caractère numérique	mcpResCavvPreauthCC.setThreeDSServerTransId("ThreeDSServerTransId");

carte ou de la demande d'authentification 3DS des modules d'extension pour les commerçants.

## Champs de demande liés aux transactions de préautorisation utilisant la TMD avec 3-D Secure et la chambre forte ( facultatifs )

Variable	Type et limites	Méthode Set
ID de client	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p> <div style="background-color: #e0f2f1; padding: 5px;"> <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : &lt;&gt;\$%=?^{}[]\</div>	<code>mcpResCavvPreauthCC.setCustomerId(cust_id);</code>
Descripteur dynamique	<p><i>Chaîne</i></p> <p>20 caractères alphanumériques</p> <p>Total de 22 caractères incluant votre nom de commerçant et un séparateur</p> <div style="background-color: #e0f2f1; padding: 5px;"> <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : &lt;&gt;\$%=?^{}[]\</div>	<code>mcpResCavvPreauthCC.setDynamicDescriptor(dynamic_descriptor);</code>
Indicateur de portefeuille électronique	<p><i>Chaîne</i></p> <p>3 caractères alphanumériques</p>	<code>mcpResCavvPreauthCC.setWalletIndicator(wallet_indicator);</code>
Renseignements d'identification au dossier	<i>Objet</i>	<code>mcpResCavvPreauthCC.setCofInfo(cof);</code>

Variable	Type et limites	Méthode Set
cof	S. O.	
<b>REMARQUE :</b> Cette variable est un objet imbriqué au sein d'une transaction et elle est requise lors du stockage des renseignements d'identification d'un client ou lors de l'utilisation de tels renseignements stockés. Pour plus de renseignements sur les champs de l'objet Credential on File Info, consultez la section Objet Credential File Info et variables.		
Renseignements du SVA	<i>Objet</i> S. O.	mcpResCavvPreauthCC.setAvsInfo(avsCheck);
Renseignements du NVC	<i>Objet</i> S. O.	mcpResCavvPreauthCC.setCvdInfo(cvdCheck);

#### Champs de demande liés aux transactions d'achat utilisant la TMD (facultatifs)

Variable	Type et limites	Méthode Set
Jeton du taux de la TMD	<i>Chaîne</i> S. O.	mcpResCavvPreauthCC.setMCPRateToken("MCP_RATE_TOKEN");

#### 3-D Secure 2.2 – Champs particuliers (facultatifs)

Variable	Type et limites	Méthode Set
ID de transaction DS	<i>Chaîne</i> 36 caractères alphanumériques	mcpResCavvPreauthCC.setDsTransId("DsTransId");

#### Exemple de préautorisation utilisant la TMD avec 3-D Secure et la chambre forte

```
package Canada;
import JavaAPI.*;
public class TestCanadaMCPResCavvPreauthCC
{
    public static void main(String[] args)
    {
        String store_id = "store1";
```

```

String api_token = "yesguy1";
String data_key = "4INQR1A8ocxD0oafSz50LADXy";
java.util.Date createDate = new java.util.Date();
String order_id = "Test"+createDate.getTime();
String amount = "1.00";
String cust_id = "customer1"; //if sent will be submitted, otherwise cust_id from profile
will be used
String cavv = "AAABBjg0VhI0VniQEjRWAAAAAA";
String processing_country_code = "CA";
String expdate = "1901";
String crypt_type = "7";
boolean status_check = false;
MCPResCavvPreauthCC mcpResCavvPreauthCC = new MCPResCavvPreauthCC();
mcpResCavvPreauthCC.setOrderId(order_id);
mcpResCavvPreauthCC.setDataKey(data_key);
mcpResCavvPreauthCC.setCustId(cust_id);
mcpResCavvPreauthCC.setAmount(amount);
mcpResCavvPreauthCC.setCavv(cavv);
mcpResCavvPreauthCC.setExpDate(expdate);
mcpResCavvPreauthCC.setCryptType(crypt_type);

mcpResCavvPreauthCC.setThreeDSVersion("2.2.0"); //Mandatory for financial transactions using
3rd Party 3-D Secure services
mcpResCavvPreauthCC.setThreeDSServerTransId("e11d4985-8d25-40ed-99d6-c3803fe5e68f");
Mandatory for financial transactions using 3rd Party 3-D Secure services - obtained from
MpICavvLookup or MpIThreeDSAAuthentication
//MCP Fields
mcpResCavvPreauthCC.setMCPVersion("1.0");
mcpResCavvPreauthCC.setCardholderAmount("500");
mcpResCavvPreauthCC.setCardholderCurrencyCode("840");
mcpResCavvPreauthCC.setMCPRateToken("P1623165281389116");

//Mandatory - Credential on File details
CofInfo cof = new CofInfo();
cof.setPaymentIndicator("U");
cof.setPaymentInformation("2");
cof.setIssuerId("139X3130ASCXAS9");

mcpResCavvPreauthCC.setCofInfo(cof);

HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false for production transactions
mpgReq.setStoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(mcpResCavvPreauthCC);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("DataKey = " + receipt.getDataKey());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Message = " + receipt.getMessage());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("ResSuccess = " + receipt.getResSuccess());
System.out.println("PaymentType = " + receipt.getPaymentType());
System.out.println("CavvResultCode = " + receipt.getCavvResultCode());
System.out.println("IssuerId = " + receipt.getIssuerId());
System.out.println("ThreeDSVersion = " + receipt.getThreeDSVersion());

System.out.println("MerchantSettlementAmount = " + receipt.getMerchantSettlementAmount());
System.out.println("CardholderAmount = " + receipt.getCardholderAmount());
System.out.println("CardholderCurrencyCode = " + receipt.getCardholderCurrencyCode());
System.out.println("MCPRate = " + receipt.getMCPRate());
System.out.println("MCPErrorStatusCode = " + receipt.getMCPErrorStatusCode());
System.out.println("MCPErrorMessage = " + receipt.getMCPErrorMessage());
System.out.println("HostId = " + receipt.getHostId());

//ResolveData
System.out.println("Cust ID = " + receipt.getResCustomerId());
}

```

```

System.out.println("Phone = " + receipt.getResPhone());
System.out.println("Email = " + receipt.getResEmail());
System.out.println("Note = " + receipt.getResNote());
System.out.println("Masked Pan = " + receipt.getResMaskedPan());
System.out.println("Exp Date = " + receipt.getResExpdate());
System.out.println("Crypt Type = " + receipt.getResCryptType());
System.out.println("Avs Street Number = " + receipt.getResAvsStreetNumber());
System.out.println("Avs Street Name = " + receipt.getResAvsStreetName());
System.out.println("Avs Zipcode = " + receipt.getResAvsZipcode());
}
catch (Exception e)
{
e.printStackTrace();
}
}
}

```

## 7.10 Conclusion de préautorisation utilisant la TMD

Une conclusion de préautorisation récupère les fonds bloqués par une transaction de préautorisation utilisant la TMD et les préparer à être déposés dans le compte du commerçant.

Cette demande de transaction est la version avec la tarification multidevise (TMD) activée de la transaction financière équivalente.

### Définition de l'objet de transaction MCP Pre-Authorization

```
MCPCompletion mcpCompletion = new MCPCompletion();
```

### Objet HttpsPostRequest pour les transactions de conclusion de préautorisation utilisant la TMD

```
HttpsPostRequest mpgReq = new HttpsPostRequest();

mpgReq.setTransaction(mcpCompletion);
```

### Principaux champs d'objet de connexion (toutes les transactions API)

Variable	Type et limites	
ID de commerce	<i>Chaîne</i> S. O.	mpgReq.setstoreId(store_id);
Jeton API	<i>Chaîne</i> S. O.	mpgReq.setApiToken(api_token);

### Champs de demande pour les transactions de conclusion de préautorisation utilisant la TMD (obligatoires)

Variable	Type et limites	Méthode Set

ID de commande	<i>Chaîne</i> 50 caractères alphanumériques a-Z A-Z 0-9 _ - : . @ espaces	mcpCompletion.setOrderId(order_id);
Numéro de transaction	<i>Chaîne</i> 255 caractères (alphanumériques, trait d'union ou trait de soulignement) Longueur variable	mcpCompletion.setTxnNumber(txn_number);
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	mcpCompletion.setCryptType(crypt);

#### Champs de demande liés aux transactions d'achat utilisant la TMD (obligatoires)

Variable	Type et limites	Méthode Set
Numéro de la version de la TMD	<i>Chaîne</i> Valeur numérique La version actuelle est 1.0	mcpCompletion.setMCPVersion("MC_P_VERSION_NUM");
Montant du titulaire de carte	<i>Chaîne</i> 12 caractère numérique La plus petite unité discrète de devise étrangère	mcpCompletion.setCardholderAmount("CARDHOLDER_AMOUNT");
Code de devise du titulaire de carte	<i>Chaîne</i> 3 caractère numérique	mcpCompletion.setCardholderCurrencyCode("CARDHOLDER_CURRENCY_CODE");

#### Champs de demande pour les transactions de conclusion de préautorisation utilisant la TMD (facultatifs)

Variable	Type et limites	Méthode Set
ID de client	<i>Chaîne</i> 50 caractères alphanumériques  <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : < > \$ % = ? ^ { } [ ] \	mcpCompletion.setCustId(cust_id);

Descripteur dynamique	<p><i>Chaîne</i></p> <p>20 caractères alphanumériques</p> <p>Total de 22 caractères incluant votre nom de commerçant et un séparateur</p> <div style="border: 1px solid black; padding: 5px;"> <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés :          &lt;&gt;\$ % = ? ^ { } [ ] \        </div>	mcpCompletion.setDynamicDescriptor(dynamic_descriptor);

### Champs de demande liés aux transactions d'achat utilisant la TMD (facultatifs)

Variable	Type et limites	Méthode Set
Jeton du taux de la TMD	<p><i>Chaîne</i></p> <p>S. O.</p>	mcpCompletion.setMCPRateToken("MCP_RATE_TOKEN");

### Exemple de conclusion de préautorisation utilisant la TMD

```

package Canada;
import JavaAPI.*;
public class TestCanadaMCPCompletion
{
  public static void main(String[] args)
  {
    String store_id = "store5";
    String api_token = "yesguy";
    String order_id = "Test1538681966167";
    String txn_number = "696294-0_11";
    String crypt = "7";
    String cust_id = "my customer id";
    String dynamic_descriptor = "my descriptor";
    String ship_indicator = "F" ;
    String processing_country_code = "CA";
    boolean status_check = false;
    MCPCompletion mcpCompletion = new MCPCompletion();
    mcpCompletion.setOrderId(order_id);
    mcpCompletion.setTxnNumber(txn_number);
    mcpCompletion.setCryptType(crypt);
    mcpCompletion.setCustomerId(cust_id);
    mcpCompletion.setDynamicDescriptor(dynamic_descriptor);
    //mcpCompletion.setShipIndicator(ship_indicator); //optional
    //MCP Fields
    mcpCompletion.setMCPVersion("1.0");
    mcpCompletion.setCardholderAmount("500");
    mcpCompletion.setCardholderCurrencyCode("840");
    //mcpCompletion.setMCPRateToken("P1538681661706745"); //optional

    HttpsPostRequest mpgReq = new HttpsPostRequest();
    mpgReq.setProcCountryCode(processing_country_code);
    mpgReq.setTestMode(true); //false for production transactions
    mpgReq.setstoreId(store_id);
    mpgReq.setApiToken(api_token);
    mpgReq.setTransaction(mcpCompletion);
    mpgReq.setStatusCheck(status_check);
    mpgReq.send();
    try
    {
      Receipt receipt = mpgReq.getReceipt();
      System.out.println("CardType = " + receipt.getCardType());
      System.out.println("TransAmount = " + receipt.getTransAmount());
    }
  }
}

```

```

System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = " + receipt.getBankTotals());
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("IsVisaDebit = " + receipt.getIsVisaDebit());

System.out.println("MerchantSettlementAmount = " + receipt.getMerchantSettlementAmount());
System.out.println("CardholderAmount = " + receipt.getCardholderAmount());
System.out.println("CardholderCurrencyCode = " + receipt.getCardholderCurrencyCode());
System.out.println("MCPRate = " + receipt.getMCPRate());
System.out.println("MCPErrorStatusCode = " + receipt.getMCPErrorStatusCode());
System.out.println("MCPErrorMessage = " + receipt.getMCPErrorMessage());
System.out.println("HostId = " + receipt.getHostId());
}
catch (Exception e)
{
e.printStackTrace();
}
}
}

```

## 7.11 Correction d'achat utilisant la TMD

Une correction d'achat rembourse l'entièreté du montant d'un achat utilisant la TMD ou d'une conclusion de préautorisation utilisant la TMD sur la carte d'un titulaire de carte et supprime toute mention de la transaction du relevé du titulaire de carte.

Cette transaction peut être effectuée à la suite d'un achat ou d'une conclusion de préautorisation traitée la même journée, à condition que le lot contenant la transaction original soit toujours ouvert.

Le processus de traitement de la TMD utilise la fonction de fermeture automatique, et les lots sont fermés tous les jours entre 22 h et 23 h, heure de l'Est.

Cette demande de transaction est la version avec la tarification multidevise (TMD) activée de la transaction financière équivalente.

### Définition de l'objet de transaction MCP Purchase Correction

```
MCPPurchaseCorrection mcpPurchasecorrection = new MCPPurchaseCorrection();
```

### Objet HttpsPostRequest pour les transactions de correction d'achat utilisant la TMD

```
HttpsPostRequest mpgReq = new HttpsPostRequest();

mpgReq.setTransaction(mcpPurchasecorrection);
```

### Principaux champs d'objet de connexion (toutes les transactions API)

Variable	Type et limites	
ID de commerce	<i>Chaîne</i> S. O.	mpgReq.setstoreId(store_id);

Jeton API	<i>Chaîne</i>  S. O.	mpgReq.setApiToken(api_token);
-----------	----------------------------	--------------------------------

**Champs de demande liés aux transactions de correction d'achat utilisant la TMD (obligatoires)**

Variable	Type et limites	Méthode Set
ID de commande	<i>Chaîne</i>  50 caractères alphanumériques a-z A-Z 0-9 _ - : . @ espaces	mcpPurchasecorrection.setOrderId(order_id);
Numéro de transaction	<i>Chaîne</i>  255 caractères (alphanumériques, trait d'union ou trait de soulignement)  Longueur variable	mcpPurchasecorrection.setTxnNumber(txn_number);
Indicateur de commerce électronique	<i>Chaîne</i>  1 caractère alphanumérique	mcpPurchasecorrection.setCryptType(crypt);

**Champs de demande liés aux transactions de correction d'achat utilisant la TMD (facultatifs)**

Variable	Type et limites	Description
Descripteur dynamique	<i>Chaîne</i>  20 caractères alphanumériques  Total de 22 caractères incluant votre nom de commerçant et un séparateur  <div style="border: 1px solid black; padding: 5px; background-color: #e0f2e0; margin-top: 10px;"> <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : &lt;&gt;\$%=?^{}[]\ </div>	mcpPurchasecorrection.setDynamicDescriptor(dynamic_descriptor);
ID de client	<i>Chaîne</i>  50 caractères alphanumériques	mcpPurchasecorrection.setCustId(cust_id);

**REMARQUE :** Certains caractères spéciaux ne sont pas autorisés :  
<>\$%=?^{}[]\

## Exemple de transaction de correction d'achat utilisant la TMD

```
package Canada;
import JavaAPI.*;
public class TestCanadaMCPPurchaseCorrection
{
    public static void main(String[] args)
    {
        String store_id = "store5";
        String api_token = "yesguy";
        String order_id = "Test1538682314339";
        String txn_number = "696314-0_11";
        String crypt = "7";
        String dynamic_descriptor = "123456";
        String processing_country_code = "CA";
        boolean status_check = false;
        MCPPurchaseCorrection mcpPurchasecorrection = new MCPPurchaseCorrection();
        mcpPurchasecorrection.setOrderId(order_id);
        mcpPurchasecorrection.setTxnNumber(txn_number);
        mcpPurchasecorrection.setCryptType(crypt);
        mcpPurchasecorrection.setDynamicDescriptor(dynamic_descriptor);
        mcpPurchasecorrection.setCustId("my customer id");
        HttpsPostRequest mpgReq = new HttpsPostRequest();
        mpgReq.setProcCountryCode(processing_country_code);
        mpgReq.setTestMode(true); //false for production transactions
        mpgReq.setstoreId(store_id);
        mpgReq.setApiToken(api_token);
        mpgReq.setTransaction(mcpPurchasecorrection);
        mpgReq.setStatusCheck(status_check);
        mpgReq.send();
        try
        {
            Receipt receipt = mpgReq.getReceipt();
            System.out.println("CardType = " + receipt.getCardType());
            System.out.println("TransAmount = " + receipt.getTransAmount());
            System.out.println("TxnNumber = " + receipt.getTxnNumber());
            System.out.println("ReceiptId = " + receipt.getReceiptId());
            System.out.println("TransType = " + receipt.getTransType());
            System.out.println("ReferenceNum = " + receipt.getReferenceNum());
            System.out.println("ResponseCode = " + receipt.getResponseCode());
            System.out.println("ISO = " + receipt.getISO());
            System.out.println("BankTotals = " + receipt.getBankTotals());
            System.out.println("Message = " + receipt.getMessage());
            System.out.println("AuthCode = " + receipt.getAuthCode());
            System.out.println("Complete = " + receipt.getComplete());
            System.out.println("TransDate = " + receipt.getTransDate());
            System.out.println("TransTime = " + receipt.getTransTime());
            System.out.println("Ticket = " + receipt.getTicket());
            System.out.println("TimedOut = " + receipt.getTimedOut());
            System.out.println("IsVisaDebit = " + receipt.getIsVisaDebit());
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

## 7.12 Remboursement utilisant la TMD

Un remboursement rembourse entièrement ou en partie les fonds obtenus à la suite d'un achat utilisant la TMD ou d'une conclusion de préautorisation utilisant la TMD et les renvoie sur la carte du titulaire de carte.

Contrairement à une correction d'achat utilisant la TMD, la transaction initiale et le remboursement apparaîtront tous les deux sur le relevé du titulaire de carte.

Si les fonds doivent être remis sur une carte différente de celle utilisée lors de la transaction d'origine, une transaction de remboursement indépendante utilisant la TMD doit plutôt être effectuée.

Cette demande de transaction est la version avec la tarification multidevise (TMD) activée de la transaction financière équivalente.

### Définition de l'objet de transaction MCP Refund

```
MCPRefund mcpRefund = new MCPRefund();
```

### Objet HttpsPostRequest pour les transactions de remboursement utilisant la TMD

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
```

```
mpgReq.setTransaction(mcpRefund);
```

### Principaux champs d'objet de connexion (toutes les transactions API)

Variable	Type et limites	
ID de commerce	<i>Chaîne</i> S. O.	mpgReq.setstoreId(store_id);
Jeton API	<i>Chaîne</i> S. O.	mpgReq.setApiToken(api_token);

### Champs de demande liés à la transaction (obligatoires)

Variable	Type et limites	Méthode Set
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques a-z A-Z 0-9 _ - : . @ espaces	mcpRefund.setorderId(order_id);

Numéro de transaction	<i>Chaîne</i> 255 caractères (alphanumériques, trait d'union ou trait de soulignement)  Longueur variable	mcpRefund.setTxnNumber(txn_number);
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	mcpRefund.setCryptType(crypt);

#### Champs de demande liés aux transactions d'achat utilisant la TMD (obligatoires)

Variable	Type et limites	Méthode Set
Numéro de la version de la TMD	<i>Chaîne</i> Valeur numérique  La version actuelle est 1.0	mcpRefund.setMCPVersion("MCP_VERSION_NUM");
Montant du titulaire de carte	<i>Chaîne</i> 12 caractère numérique  La plus petite unité discrète de devise étrangère	mcpRefund.setCardholderAmount("CARDHOLDER_AMOUNT");
Code de devise du titulaire de carte	<i>Chaîne</i> 3 caractère numérique	mcpRefund.setCardholderCurrencyCode("CARDHOLDER_CURRENCY_CODE");

## Champs de demande liés aux transactions de remboursement indépendant utilisant la TMD (facultatifs)

Variable	Type et limites	Description
Descripteur dynamique	<p><i>Chaîne</i></p> <p>20 caractères alphanumériques</p> <p>Total de 22 caractères incluant votre nom de commerçant et un séparateur</p> <div style="border: 1px solid black; padding: 5px; background-color: #e0f2f1;"> <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : &lt;&gt;\$%=?^{}[]\ </div>	mcpRefund.setDynamicDescriptor(dynamic_descriptor);
ID de client	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p> <div style="border: 1px solid black; padding: 5px; background-color: #e0f2f1;"> <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : &lt;&gt;\$%=?^{}[]\ </div>	mcpRefund.setCustId(cust_id);

## Champs de demande liés aux transactions d'achat utilisant la TMD (facultatifs)

Variable	Type et limites	Méthode Set
Jeton du taux de la TMD	<p><i>Chaîne</i></p> <p>S. O.</p>	mcpRefund.setMCPRateToken("MCP_RATE_TOKEN");

### Exemple de remboursement utilisant la TMD

```
package Canada;
import JavaAPI.*;
public class TestCanadaMCPRefund
{
    public static void main(String[] args)
    {
        String store_id = "store5";
        String api_token = "yesguy";
        String amount = "2.00";
        String crypt = "7";
        String dynamic_descriptor = "123456";
        String custid = "mycust9";
        String order_id = "Test1534871380572";
        String txn_number = "332654-0_11";
        String processing_country_code = "CA";
        boolean status_check = false;
        MCPRefund mcpRefund = new MCPRefund();
        mcpRefund.setTxnNumber(txn_number);
        mcpRefund.setOrderId(order_id);
        mcpRefund.setCryptType(crypt);
```

```

mcpRefund.setCustomerId(custid);
mcpRefund.setDynamicDescriptor(dynamic_descriptor);

//MCP Fields
mcpRefund.setMCPVersion("1.0");
mcpRefund.setCardholderAmount("200");
mcpRefund.setCardholderCurrencyCode("840");
mcpRefund.setMCPRateToken("P1534873994652426");
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false for production transactions
mpgReq.setstoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(mcpRefund);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = " + receipt.getBankTotals());
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedOut());

System.out.println("MerchantSettlementAmount = " + receipt.getMerchantSettlementAmount());
System.out.println("CardholderAmount = " + receipt.getCardholderAmount());
System.out.println("CardholderCurrencyCode = " + receipt.getCardholderCurrencyCode());
System.out.println("MCPRate = " + receipt.getMCPRate());
System.out.println("MCPErrorStatusCode = " + receipt.getMCPErrorStatusCode());
System.out.println("MCPErrorMessage = " + receipt.getMCPErrorMessage());
System.out.println("HostId = " + receipt.getHostId());
}
catch (Exception e)
{
e.printStackTrace();
}
}
}

```

## 7.13 Remboursement indépendant utilisant la TMD

Une transaction de remboursement indépendant crédite un montant précis à la carte de crédit du titulaire de carte. Le numéro de la carte de crédit et la date d'expiration sont requis.

Vous pouvez donc rembourser des transactions n'ayant pas été traitées par l'entremise de Passerelle Moneris.

Cette demande de transaction est la version avec la tarification multidevise (TMD) activée de la transaction financière équivalente.

### Éléments dont il faut tenir compte :

En raison du potentiel de fraude, ce ne sont pas tous les comptes qui peuvent traiter ces transactions par défaut. Si vous avez besoin de ce type de transactions dans votre entreprise, vous devez en faire la demande auprès de votre gestionnaire de compte.

### Définition de l'objet de transaction MCP Independent Refund

```
MCPIndependentRefund mcpIndrefund = new MCPIndependentRefund();
```

### Objet HttpsPostRequest pour les transactions de remboursement indépendant utilisant la TMD

```
HttpsPostRequest mpgReq = new HttpsPostRequest();  
  
mpgReq.setTransaction(mcpIndrefund);
```

### Principaux champs d'objet de connexion (toutes les transactions API)

Variable	Type et limites	
ID de commerce	<i>Chaîne</i> S. O.	mpgReq.setstoreId(store_id);
Jeton API	<i>Chaîne</i> S. O.	mpgReq.setApiToken(api_token);

**Champs de demande liés aux transactions de remboursement indépendant utilisant la TMD (obligatoires)**

Variable	Type et limites	Méthode Set
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques a-z A-Z 0-9 _ - : . @ espaces	mcpIndrefund.setOrderId(order_id);
Numéro de carte de crédit	<i>Chaîne</i> Maximum 20 caractères alphanumériques	mcpIndrefund.setPan(pan);
Date d'expiration	<i>Chaîne</i> 4 caractères alphanumériques AAMM	mcpIndrefund.setExpDate(expiry_date);
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	mcpIndrefund.setCryptType(crypt);

**Champs de demande liés aux transactions d'achat utilisant la TMD (obligatoires)**

Variable	Type et limites	Méthode Set
Numéro de la version de la TMD	<i>Chaîne</i> Valeur numérique La version actuelle est 1.0	mcpIndrefund.setMCPVersion("MCP_VERSION_NUM");
Montant du titulaire de carte	<i>Chaîne</i> 12 caractère numérique La plus petite unité discrète de devise étrangère	mcpIndrefund.setCardholderAmount("CARDHOLDER_AMOUNT");
Code de devise du titulaire de carte	<i>Chaîne</i> 3 caractère numérique	mcpIndrefund.setCardholderCurrencyCode("CARDHOLDER_CURRENCY_CODE");

## Champs de demande liés aux transactions de remboursement indépendant utilisant la TMD (facultatifs)

Variable	Type et limites	Méthode Set
ID de client	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p> <div style="background-color: #e0f2f1; padding: 5px;"> <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : &lt;&gt;\$%=?^{}[]\ </div>	mcpIndrefund.setCustId(cust_id);
Descripteur dynamique	<p><i>Chaîne</i></p> <p>20 caractères alphanumériques</p> <p>Total de 22 caractères incluant votre nom de commerçant et un séparateur</p> <div style="background-color: #e0f2f1; padding: 5px;"> <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : &lt;&gt;\$%=?^{}[]\ </div>	mcpIndrefund.setDynamicDescriptor(dynamic_descriptor);

## Champs de demande liés aux transactions d'achat utilisant la TMD (facultatifs)

Variable	Type et limites	Méthode Set
Jeton du taux de la TMD	<p><i>Chaîne</i></p> <p>S. O.</p>	mcpIndrefund.setMCPRateToken("MCP_RATE_TOKEN");

## Exemple de transaction de remboursement indépendant utilisant la TMD

```

package Canada;
import JavaAPI.*;
public class TestCanadaMCPIndependentRefund
{
    public static void main(String[] args)
    {
        java.util.Date createDate = new java.util.Date();
        String order_id = "Test"+createDate.getTime();
        String store_id = "store5";
        String api_token = "yesguy";
        String cust_id = "my customer id";
        String amount = "20.00";
    }
}

```

```

String pan = "4242424242424242";
String expdate = "1901"; //YYMM
String crypt = "7";
String processing_country_code = "CA";
boolean status_check = false;
MCPIndependentRefund mcpIndrefund = new MCPIndependentRefund();
mcpIndrefund.setOrderId(order_id);
mcpIndrefund.setCustId(cust_id);
mcpIndrefund.setAmount(amount);
mcpIndrefund.setPan(pan);
mcpIndrefund.setExpdate(expdate);
mcpIndrefund.setCryptType(crypt);
mcpIndrefund.setDynamicDescriptor("123456");

//MCP Fields
mcpIndrefund.setMCPVersion("1.0");
mcpIndrefund.setCardholderAmount("500");
mcpIndrefund.setCardholderCurrencyCode("840");
mcpIndrefund.setMCPRateToken("R1538679861330690");
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false for production transactions
mpgReq.setstoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(mcpIndrefund);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = " + receipt.getBankTotals());
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("IsVisaDebit = " + receipt.getIsVisaDebit());

System.out.println("MerchantSettlementAmount = " + receipt.getMerchantSettlementAmount());
System.out.println("CardholderAmount = " + receipt.getCardholderAmount());
System.out.println("CardholderCurrencyCode = " + receipt.getCardholderCurrencyCode());
System.out.println("MCPRate = " + receipt.getMCPRate());
System.out.println("MCPErrorStatusCode = " + receipt.getMCPErrorStatusCode());
System.out.println("MCPErrorMessage = " + receipt.getMCPErrorMessage());
System.out.println("HostId = " + receipt.getHostId());
}
catch (Exception e)
{
e.printStackTrace();
}
}
}

```

## 7.14 Achat utilisant la TMD avec la chambre forte

Cette transaction utilise la clé de donnée afin de trouver un profil de carte de crédit enregistré précédemment dans la chambre forte. Les renseignements enregistrés dans le profil sont ensuite utilisés pour effectuer une transaction d'achat.

La clé de données peut être une clé temporaire générée par la transformation en jetons hébergée, ou une clé permanente provenant de la chambre forte.

Cette demande de transaction est la version avec la tarification multidevise (TMD) activée de la transaction financière équivalente.

**Définition de l'objet de transaction MCP Purchase with Vault**

```
MCPResPurchaseCC mcpResPurchaseCC = new MCPResPurchaseCC();
```

**Objet HttpsPostRequest pour les transactions d'achat utilisant la TMD avec la chambre forte**

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
```

```
mpgReq.setTransaction(mcpResPurchaseCC);
```

**Principaux champs d'objet de connexion (toutes les transactions API)**

Variable	Type et limites	
ID de commerce	<i>Chaîne</i> S. O.	mpgReq.setstoreId(store_id);
Jeton API	<i>Chaîne</i> S. O.	mpgReq.setApiToken(api_token);

**Champs de demande liés aux transactions d'achat utilisant la TMD avec la chambre forte (obligatoires)**

Variable	Type et limites	Méthode Set
Clé de données	<i>Chaîne</i> 25 caractères alphanumériques	mcpResPurchaseCC.setData(data_key);
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques a-z A-Z 0-9 _ - : . @ espaces	mcpResPurchaseCC.setOrderId(order_id);
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	mcpResPurchaseCC.setCryptType(crypt);

## Champs de demande liés aux transactions d'achat utilisant la TMD (obligatoires)

Variable	Type et limites	Méthode Set
Numéro de la version de la TMD	<p><i>Chaîne</i></p> <p>Valeur numérique</p> <p>La version actuelle est 1.0</p>	<code>mcpResPurchaseCC.setMCPVersion("MCP_VERSION_NUM");</code>
Montant du titulaire de carte	<p><i>Chaîne</i></p> <p>12 caractère numérique</p> <p>La plus petite unité discrète de devise étrangère</p>	<code>mcpResPurchaseCC.setCardholderAmount("CARDHOLDER_AMOUNT");</code>
Code de devise du titulaire de carte	<p><i>Chaîne</i></p> <p>3 caractère numérique</p>	<code>mcpResPurchaseCC.setCardholderCurrencyCode("CARDHOLDER_CURRENCY_CODE");</code>

## Champs de demande liés aux transactions de d'achat utilisant la TMD avec la chambre forte (facultatifs)

Variable	Type et limites	Méthode Set
ID de client	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p> <p><b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés :  <code>&lt;&gt; \$ % = ? ^ { } [ ] \</code></p>	<code>mcpResPurchaseCC.setCustomerId(cust_id);</code>
Renseignements d'identification au dossier cof	<p><i>Objet</i></p> <p>S. O.</p> <p><b>REMARQUE :</b> Cette variable est un objet imbriqué au sein d'une transaction et elle est requise lors du stockage des renseignements d'identification d'un client ou lors de l'utilisation de tels renseignements stockés. Pour plus de renseignements sur les champs de l'objet Credential on File Info, consultez la section Objet Credential File Info et variables.</p>	<code>mcpResPurchaseCC.setCofInfo(cof);</code>

Renseignements du SVA	<i>Objet</i> S. O.	mcpResPurchaseCC.setAvsInfo(av sCheck);
Renseignements du NVC	<i>Objet</i> S. O.	mcpResPurchaseCC.setCvdInfo(cv dCheck);

### Champs de demande liés aux transactions d'achat utilisant la TMD (facultatifs)

Variable	Type et limites	Méthode Set
Jeton du taux de la TMD	<i>Chaîne</i> S. O.	mcpResPurchaseCC.setMCPRateToke n ("MCP_RATE_TOKEN");

### Exemple d'achat utilisant la TMD avec la chambre forte

```

package Canada;
import JavaAPI.*;
public class TestCanadaMCPResPurchaseCC
{
    public static void main(String[] args)
    {
        java.util.Date createDate = new java.util.Date();
        String order_id = "Test"+createDate.getTime();
        String store_id = "store5";
        String api_token = "yesguy";
        String data_key = "800XGiwxgvfbZngigVFeld9d2";
        String cust_id = "customer1"; //if sent will be submitted, otherwise cust_id from profile
        will be used
        String crypt_type = "1";
        String descriptor = "my descriptor";
        String processing_country_code = "CA";
        String expdate = "1512"; //For Temp Token
        boolean status_check = false;
        MCPResPurchaseCC mcpResPurchaseCC = new MCPResPurchaseCC();
        mcpResPurchaseCC.setDataKey(data_key);
        mcpResPurchaseCC.setOrderId(order_id);
        mcpResPurchaseCC.setCustomerId(cust_id);
        mcpResPurchaseCC.setCryptType(crypt_type);
        //resPurchaseCC.setDynamicDescriptor(descriptor);
        //resPurchaseCC.setExpDate(expdate); //Temp Tokens only

        //MCP Fields
        mcpResPurchaseCC.setMCVersion("1.0");
        mcpResPurchaseCC.setCardholderAmount("500");
        mcpResPurchaseCC.setCardholderCurrencyCode("840");
        mcpResPurchaseCC.setMCPRateToken("P1538679861174342");
        //Mandatory - Credential on File details
        CofInfo cof = new CofInfo();
        cof.setPaymentIndicator("U");
        cof.setPaymentInformation("2");

        mcpResPurchaseCC.setCofInfo(cof);

        HttpsPostRequest mpgReq = new HttpsPostRequest();
        mpgReq.setProcCountryCode(processing_country_code);
        mpgReq.setTestMode(true); //false for production transactions
        mpgReq.setstoreId(store_id);
        mpgReq.setApiToken(api_token);
        mpgReq.setTransaction(mcpResPurchaseCC);
        mpgReq.setStatusCheck(status_check);
        mpgReq.send();
        try
        {
            Receipt receipt = mpgReq.getReceipt();
            System.out.println("DataKey = " + receipt.getDataKey());
            System.out.println("ReceiptId = " + receipt.getReceiptId());
        }
    }
}

```

```

System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Message = " + receipt.getMessage());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("ResSuccess = " + receipt.getResSuccess());
System.out.println("PaymentType = " + receipt.getPaymentType());
System.out.println("IsVisaDebit = " + receipt.getIsVisaDebit());
System.out.println("Cust ID = " + receipt.getResCustomerId());
System.out.println("Phone = " + receipt.getResPhone());
System.out.println("Email = " + receipt.getResEmail());
System.out.println("Note = " + receipt.getResNote());
System.out.println("Masked Pan = " + receipt.getResMaskedPan());
System.out.println("Exp Date = " + receipt.getResExpdate());
System.out.println("Crypt Type = " + receipt.getResCryptType());
System.out.println("Avs Street Number = " + receipt.getResAvsStreetNumber());
System.out.println("Avs Street Name = " + receipt.getResAvsStreetName());
System.out.println("Avs Zipcode = " + receipt.getResAvsZipcode());
System.out.println("IssuerId = " + receipt.getIssuerId());

System.out.println("MerchantSettlementAmount = " + receipt.getMerchantSettlementAmount());
System.out.println("CardholderAmount = " + receipt.getCardholderAmount());
System.out.println("CardholderCurrencyCode = " + receipt.getCardholderCurrencyCode());
System.out.println("MCPRate = " + receipt.getMCPRate());
System.out.println("MCPErrorStatusCode = " + receipt.getMCPErrorStatusCode());
System.out.println("MCPErrorMessage = " + receipt.getMCPErrorMessage());
System.out.println("HostId = " + receipt.getHostId());
}
catch (Exception e)
{
e.printStackTrace();
}
}
}

```

## 7.15 Préautorisation utilisant la TMD avec la chambre forte

Cette transaction utilise la clé de donnée afin de trouver un profil de carte de crédit enregistré précédemment dans la chambre forte. Les détails enregistrés dans le profil sont ensuite utilisés pour effectuer une transaction de préautorisation.

La clé de données peut être une clé temporaire générée par la transformation en jetons hébergée, ou une clé permanente provenant de la chambre forte.

Cette demande de transaction est la version avec la tarification multidevise (TMD) activée de la transaction financière équivalente.

**Définition de l'objet de transaction MCP Pre-Authorization with Vault**

```
MCPResPreauthCC mcpResPreauthCC = new MCPResPreauthCC();
```

**Objet HttpsPostRequest pour les transactions de préautorisation utilisant la TMD avec la chambre forte**

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
```

```
mpgReq.setTransaction(mcpResPreauthCC);
```

**Principaux champs d'objet de connexion (toutes les transactions API)**

Variable	Type et limites	
ID de commerce	<i>Chaîne</i> S. O.	mpgReq.setstoreId(store_id);
Jeton API	<i>Chaîne</i> S. O.	mpgReq.setApiToken(api_token);

**Champs de demande pour les transactions de préautorisation utilisant la TMD avec la chambre forte (obligatoires)**

Variable	Type et limites	Méthode Set
Clé de données	<i>Chaîne</i> 25 caractères alphanumériques	mcpResPreauthCC.setData(data_key);
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques a-z A-Z 0-9 _ - : . @ espaces	mcpResPreauthCC.setOrderId(order_id);
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	mcpResPreauthCC.setCryptType(crypt);

## Champs de demande liés aux transactions d'achat utilisant la TMD (obligatoires)

Variable	Type et limites	Méthode Set
Numéro de la version de la TMD	<p><i>Chaîne</i></p> <p>Valeur numérique</p> <p>La version actuelle est 1.0</p>	<code>mcpResPreauthCC.setMCPVersion("MCP_VERSION_NUM");</code>
Montant du titulaire de carte	<p><i>Chaîne</i></p> <p>12 caractère numérique</p> <p>La plus petite unité discrète de devise étrangère</p>	<code>mcpResPreauthCC.setCardholderAmount("CARDHOLDER_AMOUNT");</code>
Code de devise du titulaire de carte	<p><i>Chaîne</i></p> <p>3 caractère numérique</p>	<code>mcpResPreauthCC.setCardholderCurrencyCode("CARDHOLDER_CURRENCY_CODE");</code>

## Champs de demande pour les transactions de préautorisation utilisant la TMD avec la chambre forte (facultatifs)

Variable	Type et limites	Méthode Set
ID de client	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p> <p><b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés :  <code>&lt;&gt;\$%=?^{}[]\</code></p>	<code>mcpResPreauthCC.setCustId(cust_id);</code>
Descripteur dynamique	<p><i>Chaîne</i></p> <p>20 caractères alphanumériques</p> <p>Total de 22 caractères incluant votre nom de commerçant et un séparateur</p> <p><b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés :  <code>&lt;&gt;\$%=?^{}[]\</code></p>	<code>mcpResPreauthCC.setDynamicDescriptor(dynamic_descriptor);</code>
Autorisation finale	<i>Chaîne</i>	<code>mcpResPreauthCC.setFinalAuth("true");</code>

<b>REMARQUE :</b> Applicable uniquement aux transactions par carte Mastercard	true/false	
Renseignements d'identification au dossier cof	<i>Objet</i> S. O.	mcpResPreauthCC.setCofInfo(cof);
<b>REMARQUE :</b> Cette variable est un objet imbriqué au sein d'une transaction et elle est requise lors du stockage des renseignements d'identification d'un client ou lors de l'utilisation de tels renseignements stockés. Pour plus de renseignements sur les champs de l'objet Credential on File Info, consultez la section Objet Credential File Info et variables.		
Renseignements du SVA	<i>Objet</i> S. O.	mcpResPreauthCC.setAvsInfo(avscCheck);
Renseignements du NVC	<i>Objet</i> S. O.	mcpResPreauthCC.setCvdInfo(cvdCheck);

#### Champs de demande liés aux transactions d'achat utilisant la TMD (facultatifs)

Variable	Type et limites	Méthode Set
Jeton du taux de la TMD	<i>Chaîne</i> S. O.	mcpResPreauthCC.setMCPRateToken("MCP_RATE_TOKEN");

#### Exemple d'une transaction de préautorisation utilisant la TMD avec la chambre forte

```

package Canada;
import JavaAPI.*;
public class TestCanadaMCPResPreauthCC
{
    public static void main(String[] args)
    {
        java.util.Date createDate = new java.util.Date();
        String order_id = "Test"+createDate.getTime();
        String store_id = "store5";
        String api_token = "yesguy";
        String data_key = "rS7DbroQHJmJxdBfXFXiauQc4";
        String amount = "1.00";
        String cust_id = "customer1"; //if sent will be submitted, otherwise cust_id from profile
        //will be used
        String crypt_type = "1";
        String dynamic_descriptor = "my descriptor";
        String processing_country_code = "CA";
        String expdate = "1712"; //For Temp Token
        boolean status_check = false;
    }
}

```

```

MCPResPrauthCC mcpResPrauthCC = new MCPResPrauthCC();
mcpResPrauthCC.setDataKey(data_key);
mcpResPrauthCC.setOrderId(order_id);
mcpResPrauthCC.setCustId(cust_id);
mcpResPrauthCC.setAmount(amount);
mcpResPrauthCC.setCryptType(crypt_type);
mcpResPrauthCC.setDynamicDescriptor(dynamic_descriptor);
//mcpResPrauthCC.setExpDate(expdate); //Temp Tokens only
//mcpResPrauthCC.setFinalAuth("true");

//MCP Fields
mcpResPrauthCC.setMCPVersion("1.0");
mcpResPrauthCC.setCardholderAmount("500");
mcpResPrauthCC.setCardholderCurrencyCode("840");
mcpResPrauthCC.setMCPRateToken("P1538681661706745");

//Mandatory - Credential on File details
CofInfo cof = new CofInfo();
cof.setPaymentIndicator("U");
cof.setPaymentInformation("2");
cof.setIssuerId("139X3130ASCXAS9");

mcpResPrauthCC.setCofInfo(cof);
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false for production transactions
mpgReq.setstoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(mcpResPrauthCC);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("DataKey = " + receipt.getDataKey());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Message = " + receipt.getMessage());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("ResSuccess = " + receipt.getResSuccess());
System.out.println("PaymentType = " + receipt.getPaymentType());
System.out.println("IsVisaDebit = " + receipt.getIsVisaDebit());
System.out.println("IsCorporate = " + receipt.getCorporateCard());
System.out.println("Cust ID = " + receipt.getResCustId());
System.out.println("Phone = " + receipt.getResPhone());
System.out.println("Email = " + receipt.getResEmail());
System.out.println("Note = " + receipt.getResNote());
System.out.println("Masked Pan = " + receipt.getResMaskedPan());
System.out.println("Exp Date = " + receipt.getResExpdate());
System.out.println("Crypt Type = " + receipt.getResCryptType());
System.out.println("Avs Street Number = " + receipt.getResAvsStreetNumber());
System.out.println("Avs Street Name = " + receipt.getResAvsStreetName());
System.out.println("Avs Zipcode = " + receipt.getResAvsZipcode());
System.out.println("IssuerId = " + receipt.getIssuerId());

System.out.println("MerchantSettlementAmount = " + receipt.getMerchantSettlementAmount());
System.out.println("CardholderAmount = " + receipt.getCardholderAmount());
System.out.println("CardholderCurrencyCode = " + receipt.getCardholderCurrencyCode());
System.out.println("MCPRate = " + receipt.getMCPRate());
System.out.println("MCPErrorStatusCode = " + receipt.getMCPErrorStatusCode());
System.out.println("MCPErrorMessage = " + receipt.getMCPErrorMessage());
System.out.println("HostId = " + receipt.getHostId());
}
catch (Exception e)
{
e.printStackTrace();
}
}
}

```

## 7.16 Remboursement indépendant utilisant la TMD avec la chambre forte

Cette transaction utilise la clé de donnée afin de trouver un profil de carte de crédit enregistré précédemment dans la chambre forte. Les détails enregistrés dans le profil sont ensuite utilisés pour effectuer une transaction de remboursement indépendant.

Cette demande de transaction est la version avec la tarification multidevise (TMD) activée de la transaction financière équivalente.

### Éléments dont il faut tenir compte :

En raison du potentiel de fraude, ce ne sont pas tous les comptes qui peuvent traiter ces transactions par défaut. Si vous avez besoin de ce type de transactions dans votre entreprise, vous devez en faire la demande auprès de votre gestionnaire de compte.

### Définition de l'objet de transaction MCP Independent Refund with Vault

```
MCPResIndRefundCC mcpResIndRefundCC = new MCPResIndRefundCC();
```

### Objet HttpsPostRequest pour les transactions de remboursement indépendant utilisant la TMD avec la chambre forte

```
HttpsPostRequest mpgReq = new HttpsPostRequest();  
mpgReq.setTransaction(mcpResIndRefundCC);
```

**Principaux champs d'objet de connexion (toutes les transactions API)**

Variable	Type et limites	
ID de commerce	<i>Chaîne</i> S. O.	mpgReq.setstoreId(store_id);
Jeton API	<i>Chaîne</i> S. O.	mpgReq.setApiToken(api_token);

**Champs de demande liés aux transactions de remboursement indépendant utilisant la TMD avec la chambre forte (obligatoires)**

Variable	Type et limites	Méthode Set
Clé de données	<i>Chaîne</i> 25 caractères alphanumériques	mcpResIndRefundCC.setData(data_key);
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques a-z A-Z 0-9 _ - : . @ espaces	mcpResIndRefundCC.setOrderId(order_id);
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	mcpIndrefund.setCryptType(crypt);

**Champs de demande liés aux transactions d'achat utilisant la TMD (obligatoires)**

Variable	Type et limites	Méthode Set
Numéro de la version de la TMD	<i>Chaîne</i> Valeur numérique La version actuelle est 1.0	mcpResIndRefundCC.setMCPVersion("MCP_VERSION_NUM");
Montant du titulaire de carte	<i>Chaîne</i> 12 caractère numérique La plus petite unité discrète de devise étrangère	mcpResIndRefundCC.setCardholderAmount("CARDHOLDER_AMOUNT");

Code de devise du titulaire de carte	<i>Chaîne</i> 3 caractère numérique	mcpResIndRefundCC.setCardholderCurrencyCode ("CARDHOLDER_CURRENCY_CODE");
--------------------------------------	--	---

### Champs de demande liés aux transactions de remboursement indépendant utilisant la TMD avec la chambre forte (facultatifs)

Variable	Type et limites	Méthode Set
ID de client	<i>Chaîne</i> 50 caractères alphanumériques  <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\	mcpResIndRefundCC.setCustId(cust_id);

### Champs de demande liés aux transactions d'achat utilisant la TMD (facultatifs)

Variable	Type et limites	Méthode Set
Jeton du taux de la TMD	<i>Chaîne</i> S. O.	mcpResIndRefundCC.setMCPRateToken ("MCP_RATE_TOKEN");

### Exemple de remboursement indépendant utilisant la TMD avec la chambre forte

```
package Canada;
import JavaAPI.*;
public class TestCanadaMCPResIndRefundCC
{
    public static void main(String[] args)
    {
        java.util.Date createDate = new java.util.Date();
        String order_id = "Test"+createDate.getTime();
        String store_id = "store5";
        String api_token = "yesguy";
        String data_key = "rS7DbroQHJmJxdBfXFXiauQc4";
        String amount = "1.00";
        String cust_id = "customer1";
        String crypt_type = "1";
        String processing_country_code = "CA";
        boolean status_check = false;
        MCPResIndRefundCC mcpResIndRefundCC = new MCPResIndRefundCC();
        mcpResIndRefundCC.setDataKey(data_key);
        mcpResIndRefundCC.setOrderId(order_id);
        mcpResIndRefundCC.setCustId(cust_id);
        mcpResIndRefundCC.setAmount(amount);
        mcpResIndRefundCC.setCryptType(crypt_type);

        //MCP Fields
        mcpResIndRefundCC.setMCPVersion("1.0");
        mcpResIndRefundCC.setCardholderAmount("500");
        mcpResIndRefundCC.setCardholderCurrencyCode("840");
        mcpResIndRefundCC.setMCPRateToken("R1538679861330690");
        HttpsPostRequest mpgReq = new HttpsPostRequest();
        mpgReq.setProcCountryCode(processing_country_code);
        mpgReq.setTestMode(true); //false for production transactions
        mpgReq.setstoreId(store_id);
        mpgReq.setApiToken(api_token);
        mpgReq.setTransaction(mcpResIndRefundCC);
        mpgReq.setStatusCheck(status_check);
        mpgReq.send();
        try
        {
            Receipt receipt = mpgReq.getReceipt();
        }
```

```

System.out.println("DataKey = " + receipt.getDataKey());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Message = " + receipt.getMessage());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("ResSuccess = " + receipt.getResSuccess());
System.out.println("PaymentType = " + receipt.getPaymentType());
System.out.println("IsVisaDebit = " + receipt.getIsVisaDebit());
System.out.println("Cust ID = " + receipt.getResCustId());
System.out.println("Phone = " + receipt.getResPhone());
System.out.println("Email = " + receipt.getResEmail());
System.out.println("Note = " + receipt.getResNote());
System.out.println("Masked Pan = " + receipt.getResMaskedPan());
System.out.println("Exp Date = " + receipt.getResExpdate());
System.out.println("Crypt Type = " + receipt.getResCryptType());
System.out.println("Avs Street Number = " + receipt.getResAvsStreetNumber());
System.out.println("Avs Street Name = " + receipt.getResAvsStreetName());
System.out.println("Avs Zipcode = " + receipt.getResAvsZipcode());

System.out.println("MerchantSettlementAmount = " + receipt.getMerchantSettlementAmount());
System.out.println("CardholderAmount = " + receipt.getCardholderAmount());
System.out.println("CardholderCurrencyCode = " + receipt.getCardholderCurrencyCode());
System.out.println("MCPRate = " + receipt.getMCPRate());
System.out.println("MCPErrorStatusCode = " + receipt.getMCPErrorStatusCode());
System.out.println("MCPErrorMessage = " + receipt.getMCPErrorMessage());
System.out.println("HostId = " + receipt.getHostId());
}
catch (Exception e)
{
e.printStackTrace();
}
}
}

```

## 7.17 Codes de devise de la TMD

Pour les symboles monétaires, consultez la page <https://justforex.com/education/currencies>.

**REMARQUE :** Cette documentation contient des liens vers des sites Web détenus et exploités par des tierces parties. Si vous utilisez ces liens, vous quittez notre site Web. Ces liens sont fournis à titre informatif et pour votre commodité uniquement. Le contenu de ces sites Web ou de ces tiers n'est pas approuvé par Solutions Moneris. Solutions Moneris n'a aucun contrôle sur le contenu des sites Web liés et n'est pas responsable de ces sites Web, de leur contenu ou de leur disponibilité. Si vous décidez d'accéder à des sites Web tiers et d'utiliser les renseignements qu'ils contiennent, vous le faites entièrement à vos propres risques.

Code de devise (ISO)	Devise et acronyme
008	Lek albanais (ALL)
012	Dinar algérien (DZD)
032	Peso argentin (ARS)
036	Dollar australien (AUD)
048	Dinar de Bahreïn (BHD)
050	Taka (BDT)
052	Dollar de Barbade (BBD)
060	Dollar des Bermudes (BMD)
064	Ngultrum (BTN)
068	Boliviano (BOB)
084	Dollar de Belize (BZD)
090	Dollar des îles Salomon (SBD)
096	Dollar de Brunei (BND)
108	Franc du Burundi (BIF)
132	Escudo de Cabo Verde (CVE)
136	Dollar des îles Caïmans (KYD)
144	Roupie de Sri Lanka (LKR)
152	Peso chilien (CLP)

Code de devise (ISO)	Devise et acronyme
156	Yuan (CNY)
170	Peso colombien (COP)
174	Franc des Comores (KMF)
188	Colon de Costa Rica (CRC)
191	Kuna (HRK)
192	Peso cubain (CUP)
203	Couronne tchèque (CZK)
208	Couronne danoise (DKK)
214	Peso dominicain (DOP)
222	Colon du Salvador (SVC)
242	Dollar de Fidji (FJD)
262	Franc de Djibouti (DJF)
270	Dalasi (GMD)
292	Livre de Gibraltar (GIP)
320	Quetzal (GTQ)
324	Franc guinéen (GNF)
328	Dollar de Guyane (GYD)
332	Gourde haïtienne (HTG)

Code de devise (ISO)	Devise et acronyme
340	Lempira hondurien (HNL)
344	Dollar de Hong Kong (HKD)
348	Forint hongrois (HUF)
352	Couronne islandaise (ISK)
356	Roupie indienne (INR)
360	Roupie indonésienne (IDR)
376	Shekel israélien (ILS)
388	Dollar jamaïcain (JMD)
392	Yen (JPY)
398	Tenge (KZT)
400	Dinar jordanien (JOD)
404	Shilling du Kenya (KES)
410	Won de la Corée du Sud (KRW)
414	Dinar koweïtien (KWD)
418	Kip (LAK)
426	Loti lesothan (LSL)
430	Dollar libérien (LRD)
446	Pataca (MOP)

Code de devise (ISO)	Devise et acronyme
454	Kwacha malawien (MWK)
458	Ringgit de Malaisie (MYR)
462	Rufiyaa (MVR)
480	Roupie de Maurice (MUR)
484	Peso mexicain (MXN)
498	Leu de Moldavie (MDL)
504	Dirham marocain (MAD)
512	Rial Omani (OMR)
516	Dollar namibien (NAD)
524	Roupie du Népal (NPR)
532	Florin des Antilles néerlandaises (ANG)
533	Florin d'Aruba (AWG)
548	Vatu (VUV)
554	Dollar néo-zélandais (NZD)
558	Cordoba Oro (NIO)
566	Naira nigérien (NGN)
578	Couronne norvégienne (NOK)
586	Roupie du Pakistan (PKR)

Code de devise (ISO)	Devise et acronyme
598	Kina (PGK)
600	Guarani (PYG)
604	Nuevo sol péruvien (PEN)
608	Peso philippin (PHP)
634	Riyal du Qatar (QAR)
643	Rouble russe (RUB)
646	Franc du Rwanda (RWF)
654	Livre de Sainte-Hélène
682	Riyal saoudien (SAR)
690	Roupie des Seychelles (SCR)
694	Leone sierra-léonais (SLL)
702	Dollar de Singapour (SGD)
704	Dong (VND)
710	Rand sud-africain (ZAR)
748	Lilangeni (SZL)
752	Couronne suédoise (SEK)
756	Franc Suisse (CHF)

Code de devise (ISO)	Devise et acronyme
764	Baht (THB)
780	Dollar de la Trinité et de Tobago (TTD)
784	Dirham des Émirats arabes unis (AED)
788	Dinar tunisien (TND)
800	Shilling ougandais (UGX)
807	Denar (MKD)
818	Livre égyptienne (EGP)
826	Livre sterling (GBP)
834	Shilling de Tanzanie (TZS)
840	Dollar des États-Unis (USD)
858	Peso uruguayen (UYU)
860	Sum (UZS)
882	Tala (WST)
901	Nouveau dollar de Taiwan (TWD)
929	Ouguija mauritanien (MRU)
933	Rouble bélorusse (BYN)
934	Manat turkmène (TMT)
941	Dinar serbe (RSD)

Code de devise (ISO)	Devise et acronyme
943	Metical (MZN)
944	Manat azerbaïdjanaïs (AZN)
946	Nouveau Leu (RON)
949	Livre turque (TRY)
951	Dollar des Caraïbes orientales (XCD)
952	Franc CFA (XOF)
953	Franc Pacifique (XPF)
967	Kwacha zambien (ZNW)
968	Dollar de Surinam (SRD)
969	Ariary malgache (MGA)
971	Afghani (AFN)
972	Somoni (TJS)
973	Kwanza (AOA)
975	Lev bulgare (BGN)
977	Mark convertible (BAM)
978	Euro (EUR)
981	Lari géorgien (GEL)
985	Zloty (PLN)

Code de devise (ISO)	Devise et acronyme
986	Real brésilien (BRL)

## 7.18 Codes d'erreur de la TMD

Code d'erreur	Description
200	OK (aucune valeur ne sera renvoyée dans le message d'erreur de la TMD)
500	Erreur en amont
1000	Format JSON non valide
1003	txnType non valide détecté : <invalid txnType> veuillez entrer ACHAT (PURCHASE) ou REMBOURSEMENT (REFUND)
1005	Combinaison rateInquiryId-txnType non valide
1007	Avertissement : au moins l'un des éléments cardHolderCurrency ou merchantSettlementCurrency doit avoir une valeur autre que zéro
1008	Le montant du titulaire de la carte ne doit pas être zéro
1009	Montants négatifs détectés
1010	Devise du titulaire de la carte non prise en charge détectée : <unsupported currency>
1015	rateInquiryId non valide
1016	ID de commerçant non pris en charge

## 8 Versements Visa

- 8.1 À propos de la solution Versements Visa
- 8.2 Types de transaction Versements Visa
- 8.3 Envoi de transactions avec la solution Versements Visa
- 8.4 Recherche de plan de versements
- 8.5 Recherche de plan de versements dans la chambre forte
- 8.6 Objet Installment Info

### 8.1 À propos de la solution Versements Visa

La solution Versements Visa permet aux émetteurs d'offrir aux titulaires de carte de payer leurs achats en plusieurs versements. Lorsqu'un titulaire de carte accepte un plan de versements, le commerçant reçoit le paiement entier, et le titulaire de carte paie l'émetteur en fonction du plan de versements.

Pour obtenir une liste des définitions des champs de demande et de réponse, consultez la section B.3 Définition des champs de réponse – Versements Visa.

### 8.2 Types de transaction Versements Visa

Les transactions financières prenant en charge la solution Versements Visa sont les suivantes :

- Achat
- Préautorisation
- Conclusion de préautorisation
- Correction d'achat
- Remboursement
- Achat avec la chambre forte (ResPurchaseCC)
- Préautorisation avec la chambre forte (ResPreauthCC)

**REMARQUE :** Les versements de la solution Versements Visa ne sont pas pris en charge par les transactions de remboursement indépendant.

**AVERTISSEMENT :** N'envoyez pas l'objet Installment Info lors d'une transaction qui n'offre pas la fonction Versements Visa, car la transaction pourrait échouer

## 8.3 Envoi de transactions avec la solution Versements Visa

L'envoi de transactions avec la solution Versements Visa comprend les étapes suivantes :

1. envoi de la demande de transaction Recherche de plan de versements ou Recherche de plan de versements dans la chambre forte (pour les transactions avec la chambre forte) afin d'obtenir les données **installment plan ID**, **installment plan reference** et **terms and conditions version** dans la réponse;
2. à l'aide des données obtenues dans la réponse ci-dessus, envoie de l'objet **Installment Info** pour les transactions d'achat ou de préautorisation (pour les transactions avec la chambre forte, utiliser les transactions d'achat avec la chambre forte et de préautorisation avec la chambre forte);

lors de la conclusion d'une transaction de préautorisation, ou lors d'une transaction d'annulation d'achat ou de remboursement, ainsi que pour le reste de l'API unifiée, les transactions précédentes sont référencées au moyen des valeurs **order ID** et **transaction number**, ou **data key** pour les transactions avec la chambre forte.

**REMARQUE :** Les versements de la solution Versements Visa ne sont pas pris en charge par les transactions de remboursement indépendant.

## 8.4 Recherche de plan de versements

Les transactions de recherche de plan de versements sont utilisées pour obtenir les renseignements nécessaires aux transactions financières avec Versements Visa.

### Définition de l'objet de transaction **Installment Plan Lookup**

```
InstallmentLookup installmentLookup = new InstallmentLookup();
```

### Objet **HttpsPostRequest** pour les transactions de recherche d'un plan de versement

```
HttpsPostRequest mpgReq = new HttpsPostRequest();

mpgReq.setTransaction(installmentLookup);
```

### Principaux champs d'objet de connexion (toutes les transactions API)

Variable	Type et limites	
ID de commerce	<i>Chaîne</i> S. O.	mpgReq.setstoreId(store_id);
Jeton API	<i>Chaîne</i> S. O.	mpgReq.setApiToken(api_token);

### Champs de demande liés aux transactions de recherche de plan de versements (obligatoires)

Variable	Type et limites	Méthode Set
ID de commande	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p> <p>a-Z A-Z 0-9 _ - : . @ espaces</p>	installmentLookup.setOrderId(order_id);
Montant	<p><i>Chaîne</i></p> <p>10 caractères décimaux</p> <p>Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal</p>	installmentLookup.setAmount(amount);
	<b>EXAMPLE : 1 234 567,89</b>	
Numéro de carte de crédit	<p><i>Chaîne</i></p> <p>Maximum 20 caractères alphanumériques</p>	installmentLookup.setPan(pan);
Date d'expiration	<p><i>Chaîne</i></p> <p>4 caractères alphanumériques</p> <p>AAMM</p>	installmentLookup.setExpDate(expiry_date);

### Exemple d'une transaction de recherche de plan de versements

```

package Canada;
import JavaAPI.*;
public class TestCanadaInstallmentLookup
{
    public static void main(String[] args)
    {
        java.util.Date createDate = new java.util.Date();
        String order_id = "Test"+createDate.getTime();
        String store_id = "monca03650";
        String api_token = "7Yw0MPTlhjBRcZiE6837";
        String amount = "600.00";
        String pan = "4761270070000310";
        String expdate = "2212"; //YYMM format

        InstallmentLookup InstallmentLookup = new InstallmentLookup();
        InstallmentLookup.setOrderId(order_id);
        InstallmentLookup.setAmount(amount);
        InstallmentLookup.setPan(pan);
        InstallmentLookup.setExpdate(expdate);

        HttpsPostRequest mpgReq = new HttpsPostRequest();
    }
}

```

```

mpgReq.setTestMode(true); //false for production transactions
mpgReq.setStoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(InstallmentLookup);

//Optional - Proxy
mpgReq.setProxy(false); //true to use proxy
mpgReq.setProxyHost("proxyURL");
mpgReq.setProxyPort("proxyPort");
mpgReq.setProxyUser("proxyUser"); //optional - domainName\User
mpgReq.setProxyPassword("proxyPassword"); //optional
mpgReq.send();

try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = " + receipt.getBankTotals());
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedout());
System.out.println("IsVisaDebit = " + receipt.getIsVisaDebit());

EligibleInstallmentPlans eligibleInstallmentPlans = receipt.getEligibleInstallmentPlans();

int planCount = eligibleInstallmentPlans.getPlanCount();
PlanDetails [] installmentPlans = eligibleInstallmentPlans.getInstallmentPlans();

for (int i = 0; i < planCount; i++)
{
System.out.println("\nPlanId = " + installmentPlans[i].getPlanId() + "\n");
System.out.println("PlanIdRef = " + installmentPlans[i].getPlanIdRef());
System.out.println("Name = " + installmentPlans[i].getName());
System.out.println("Type = " + installmentPlans[i].getType());
System.out.println("NumInstallments = " + installmentPlans[i].getNumInstallments());
System.out.println("InstallmentFrequency = " +
installmentPlans[i].getInstallmentFrequency());

TAC tac = installmentPlans[i].getTac();

TACDetails [] tacDetailsList = tac.getTacDetailsList();
int tacCount = tac.getTacCount();

for (int j = 0; j < tacCount; j++)
{
TACDetails tacDetails = tacDetailsList[j];

System.out.println("\nText = " + tacDetails.getText() +"\n");
System.out.println("Url = " + tacDetails.getUrl());
System.out.println("Version = " + tacDetails.getVersion());
System.out.println("LanguageCode = " + tacDetails.getLanguageCode());
}

PromotionInfo promotionInfo = installmentPlans[i].getPromotionInfo();

System.out.println("\nPromotionCode = " + promotionInfo.getPromotionCode() +" \n");
System.out.println("PromotionId = " + promotionInfo.getPromotionId());

FirstInstallment firstInstallment = installmentPlans[i].getFirstInstallment();

System.out.println("\nUpfrontFee = " + firstInstallment.getUpfrontFee() +"\n");
System.out.println("InstallmentFee = " + firstInstallment.getInstallmentFee());
System.out.println("Amount = " + firstInstallment.getAmount());

LastInstallment lastInstallment = installmentPlans[i].getLastInstallment();

System.out.println("\nInstallmentFee = " + lastInstallment.getInstallmentFee());
}

```

```
System.out.println("Amount = " + lastInstallment.getAmount());  
System.out.println("\nAPR = " + installmentPlans[i].getAPR());  
System.out.println("\nTotalFees = " + installmentPlans[i].getTotalFees());  
System.out.println("\nTotalPlanCost = " + installmentPlans[i].getTotalPlanCost());  
}  
}  
catch (Exception e)  
{  
e.printStackTrace();  
}  
}  
}
```

## 8.5 Recherche de plan de versements dans la chambre forte

Les transactions de recherche de plan de versements dans la chambre forte sont utilisées pour obtenir les renseignements nécessaires au traitement de transactions financières avec des versements lorsqu'un jeton enregistré dans la chambre forte de Moneris est utilisé.

### Définition de l'objet de transaction Vault Installment Plan Lookup

```
ResInstallmentLookup resInstallmentLookup= new ResInstallmentLookup();
```

### Objet HttpsPostRequest pour les transactions de recherche d'un plan de versements dans la chambre forte

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
```

```
mpgReq.setTransaction(resInstallmentLookup);
```

### Principaux champs d'objet de connexion (toutes les transactions API)

Variable	Type et limites	
ID de commerce	<i>Chaîne</i> S. O.	mpgReq.setstoreId(store_id);
Jeton API	<i>Chaîne</i> S. O.	mpgReq.setApiToken(api_token);

### Champs de demande pour les transactions recherche d'un plan de versements dans la chambre forte (obligatoires)

Variable	Type et limites	Méthode Set
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques a-Z A-Z 0-9 _ - : . @ espaces	resInstallmentLookup.setorderId(order_id);
Montant	<i>Chaîne</i> 10 caractères décimaux	resInstallmentLookup.setAmount(amount);

Variable	Type et limites	Méthode Set
	Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal  <b>EXEMPLE : 1 234 567,89</b>	
Clé de données	<i>Chaîne</i> 25 caractères alphanumériques	<code>resInstallmentLookup.setData(data_key);</code>
Date d'expiration	<i>Chaîne</i> 4 caractères alphanumériques AAMM	<code>resInstallmentLookup.setExpDate(expiry_date);</code>

**Exemple d'une transaction de recherche de plan de versements dans la chambre forte**

```

package Canada;
import JavaAPI.*;
public class TestCanadaInstallmentLookup
{
    public static void main(String[] args)
    {
        java.util.Date createDate = new java.util.Date();
        String order_id = "Test"+createDate.getTime();
        String store_id = "monca03650";
        String api_token = "7Yw0MPTlhjBRcZiE6837";
        String amount = "600.00";
        String expdate = "2212"; //YYMM format

        InstallmentLookup InstallmentLookup = new InstallmentLookup();
        InstallmentLookup.setOrderId(order_id);
        InstallmentLookup.setAmount(amount);
        InstallmentLookup.setExpdate(expdate);

        HttpsPostRequest mpgReq = new HttpsPostRequest();
        mpgReq.setTestMode(true); //false for production transactions
        mpgReq.setStoreId(store_id);
        mpgReq.setApiToken(api_token);
        mpgReq.setTransaction(InstallmentLookup);

        //Optional - Proxy
        mpgReq.setProxy(false); //true to use proxy
        mpgReq.setProxyHost("proxyURL");
        mpgReq.setProxyPort("proxyPort");
        mpgReq.setProxyUser("proxyUser"); //optional - domainName\User
        mpgReq.setProxyPassword("proxyPassword"); //optional
        mpgReq.send();

        try
        {
            Receipt receipt = mpgReq.getReceipt();
            System.out.println("CardType = " + receipt.getCardType());
            System.out.println("TransAmount = " + receipt.getTransAmount());
            System.out.println("TxnNumber = " + receipt.getTxnNumber());
        }
    }
}

```



## 8.6 Objet Installment Info

Lors de l'envoi d'une transaction d'achat ou de préautorisation avec la solution Versements Visa, l'objet Info Installment est inclus dans la demande. Cet objet utilise les renseignements reçus en réponse de la transaction de recherche de plan de versements.

Pour obtenir une liste des définitions des champs de demande et de réponse, consultez la section B.3 Définition des champs de réponse – Versements Visa.

### Champs de demande de l'objet Installment Info

Variable	Type et limites	Méthode Set
ID du plan de versements	<i>Chaîne</i> 36 caractères alphanumériques Longueur fixe	<code>installmentLookup.setPlanId("VALUE");</code>
Référence du plan de versements	<i>Chaîne</i> 10 caractères alphanumériques Longueur fixe	<code>installmentLookup.setPlanIdRef("VALUE");</code>
Version des modalités	<i>Chaîne</i> 10 caractères alphanumériques Longueur variable (de 1 à 10 caractères)	<code>installmentLookup.setTacVersion("VALUE");</code>

**AVERTISSEMENT :** N'envoyez pas l'objet Installment Info lors d'une transaction qui n'offre pas la fonction Versements Visa, car la transaction pourrait échouer

## **9 Outils de protection contre la fraude en ligne**

- 9.1 Service de vérification d'adresse
- 9.2 Numéro de vérification de carte (NVC)
- 9.3 Outil de gestion des risques transactionnels

## 9.1 Service de vérification d'adresse

- 9.1.1 À propos du service de vérification d'adresse (SVA)
- 9.1.2 Objet AVS Info
- 1. Objet AVS Information
- 9.1.3 Codes de réponse du SVA
- 9.1.4 Exemple d'une transaction utilisant le SVA

### 9.1.1 À propos du service de vérification d'adresse (SVA)

Le service de vérification d'adresse (SVA) est un outil de prévention de la fraude facultatif offert par les banques émettrices; cet outil vérifie l'adresse du titulaire de carte dans le cadre du processus d'autorisation de la transaction. L'adresse du SVA est ensuite comparée à l'adresse enregistrée dans le dossier de la banque émettrice. Le SVA vérifie si le numéro d'immeuble, le nom de la rue et le code postal correspondent. La banque émettrice renvoie un code de résultat indiquant si les données correspondent ou non. Peu importe le code de résultat obtenu, la carte de crédit est autorisée par la banque émettrice.

La réponse reçue du SVA se veut une mesure de sécurité et de protection contre la fraude, mais la réponse elle-même n'a aucune incidence sur la conclusion de la transaction. Une fois la réponse reçue, le choix de traiter une transaction ou non est entièrement laissé au commerçant. Les réponses reçues **ne sont pas** des indications strictes concernant l'approbation ou le refus d'une transaction.

Le SVA est pris en charge pour les transactions suivantes :

- Achat (de base et par glissement de la bande magnétique)
- Préautorisation (de base)
- Réautorisation (de base)
- ResAddCC (ajout d'une carte de crédit à la chambre forte)
- ResUpdateCC (mise à jour d'une carte de crédit dans la chambre forte)

#### Éléments dont il faut tenir compte :

- Le SVA est pris en charge par Visa, Mastercard, American Express, Discover et JCB.
- Lorsque vous testez le SVA, utilisez **seulement** les numéros de carte test de Visa 42424242424242 ou 400555444444403 ainsi que les montants indiqués dans le document Simulator eFraud Response Codes (Simulateur de codes de réponse pour la fraude en ligne, offert en anglais seulement) qui se trouve dans le portail pour développeurs de Moneris (<https://developer.moneris.com>).  
L'ID de commerce « store5 » est configuré pour prendre en charge les tests de SVA.

### 9.1.2 Objet AVS Info

#### Définition de l'objet AVSInfo

```
AvsInfo avsCheck = new AvsInfo();
```

#### Méthode Set pour l'objet de transaction

```
<transaction>.setAvsInfo(avsCheck);
```

Variable	Type et limites	Méthode Set	Description
Numéro d'immeuble pour le SVA	<p><i>Chaîne</i> 19 caractères alphanumériques</p> <p><b>REMARQUE :</b> Cette limite de caractère combine le nombre total de caractères autorisés pour le numéro d'immeuble et le nom de rue pour le SVA.</p>	avsCheck.setAvsStreetNumber("212");	Numéro d'immeuble du titulaire de carte
Nom de rue pour le SVA	<p><i>Chaîne</i> 19 caractères alphanumériques</p> <p><b>REMARQUE :</b> Cette limite de caractère combine le nombre total de caractères autorisés pour le numéro d'immeuble et le nom de rue pour le SVA.</p>	avsCheck.setAvsStreetName("Payton Street");	Nom de la rue du titulaire de carte
Code postal pour le SVA	<p><i>Chaîne</i> 9 caractères alphanumériques</p>	avsCheck.setAvsZipCode("M1M1M1");	Code postal du titulaire de carte

### 9.1.3 Codes de réponse du SVA

Vous trouverez ci-dessous une liste complète des codes de réponse du SVA. Vous pouvez les obtenir en appelant la méthode `receipt.GetAvsResultCode()`.

Code	Visa	Mastercard/Discover	American Express/JCB
A	Adresse pour le SVA uniquement (correspondance partielle)	L'adresse municipale correspond, mais pas le code postal.	L'adresse de facturation correspond, mais pas le code postal.
D	Ne s'applique plus à Visa	S. O.	Le nom du client est incorrect. Le code postal ou ZIP correspond.
E	S. O.	S. O.	Le nom du client est incorrect. L'adresse de facturation et le code postal ou ZIP correspondent.
F	Ne s'applique plus à Visa	S. O.	Le nom du client est incorrect. L'adresse de facturation correspond.
G	Ne s'applique plus à Visa	Les renseignements liés à l'adresse n'ont pas été vérifiés pour une transaction internationale.	S. O.
I	Les renseignements liés à l'adresse n'ont pas été vérifiés.	S. O.	S. O.
K	S. O.	S. O.	Le nom du client correspond.
L	S. O.	S. O.	Le nom du client et le code postal ou ZIP correspondent.
M	Ne s'applique plus à Visa	S. O.	Le nom du client, l'adresse de facturation et le code postal ou ZIP correspondent.
N	Aucune correspondance du SVA	Ni l'adresse municipale ni le code postal ou ZIP ne correspondent.	L'adresse de facturation et le code postal ou ZIP ne correspondent pas.
O	S. O.	S. O.	Le nom du client et l'adresse de facturation correspondent.

P	L'acquéreur de transactions a envoyé le code postal ou ZIP et l'adresse municipale, mais l'adresse municipale n'a pas été vérifiée en raison de formats incompatibles.	S. O.	S. O.
R	Résultat du SVA indéterminé (réessai)	Nouvel essai : le système n'a pas été en mesure de traiter la demande.	Le système n'est pas disponible. Essayez de nouveau.
S	Ne s'applique plus à Visa	Le SVA n'est actuellement pas pris en charge.	Le SVA n'est actuellement pas pris en charge.
T	S. O.	Le code ZIP de neuf chiffres correspond, mais pas l'adresse.	S. O.
U	Vérification du SVA impossible	Aucune donnée n'a été fournie par le système d'autorisation ni par celui de l'émetteur.	Les renseignements ne sont pas disponibles.
W	Ne s'applique plus à Visa	Pour les adresses des États-Unis, le code ZIP de neuf chiffres correspond, mais pas l'adresse.  Pour les adresses ailleurs qu'aux États-Unis, le code postal correspond, mais pas l'adresse.	Le nom du client, l'adresse de facturation et le code postal correspondent parfaitement.
X	Ne s'applique plus à Visa	Pour les adresses des États-Unis, le code ZIP de neuf chiffres et l'adresse correspondent.  Pour les adresses ailleurs qu'aux États-Unis, le code postal et l'adresse correspondent.	S. O.
Y	Correspondance complète du SVA	L'adresse de facturation et le code postal ou ZIP correspondent.	L'adresse de facturation et le code postal ou ZIP correspondent.
Z	Code postal ou ZIP pour le SVA uniquement (correspondance partielle)	Pour les adresses des États-Unis, le code ZIP de cinq chiffres correspond, mais pas l'adresse	Le code postal ou ZIP correspond, mais pas l'adresse de facturation.

#### 9.1.4 Exemple d'une transaction utilisant le SVA

Ceci est un exemple de code .NET montrant l'intégration du SVA dans une transaction d'achat. Les renseignements de l'objet Purchase non pertinents pour le SVA ont été retirés.

Pour en savoir plus sur les transactions d'achat, consultez la section 2.1 Achat.

##### Exemple d'une transaction d'achat avec les renseignements du SVA (objet AVS Information)

```
AvsInfo avsCheck = new AvsInfo();
avsCheck.setAvsStreetNumber("212");
avsCheck.setAvsStreetName("Payton Street");
avsCheck.setAvsZipCode("M1M1M1");
avsCheck.setAvsEmail("test@host.com");
avsCheck.setAvsHostname("hostname");
avsCheck.setAvsBrowser("Mozilla");
avsCheck.setAvsShiptoCountry("CAN");
avsCheck.setAvsShipMethod("G");
avsCheck.setAvsMerchProdSku("123456");
avsCheck.setAvsCustIp("192.168.0.1");
avsCheck.setAvsCustPhone("5556667777");

Purchase purchase = new Purchase();
purchase.setAvsInfo(avsCheck);
```

## 9.2 Numéro de vérification de carte (NVC)

- 9.2.1 À propos du numéro de vérification de carte (NVC)
- 9.2.3 Objet CVD Information
- 9.2.4 Codes de résultat liés au NVC
- 9.2.5 Exemple d'une transaction d'achat avec l'objet CVD Info

### 9.2.1 À propos du numéro de vérification de carte (NVC)

Le numéro de vérification de carte (NVC) est un numéro additionnel imprimé sur les cartes de crédit et permet de vérifier un élément de plus lors du processus de vérification de l'identité d'un titulaire de carte durant une transaction.

La réponse reçue à propos du NVC se veut une mesure de sécurité et de protection contre la fraude, mais la réponse elle-même n'a aucune incidence sur la conclusion de la transaction. Lorsque la réponse est reçue, le choix de traiter une transaction ou non est entièrement laissé au commerçant. Ces réponses **ne doivent pas** déterminer qu'une transaction sera approuvée ou refusée.

Le NVC est pris en charge pour les transactions suivantes :

- Achat (de base, avec la chambre forte et par glissement de la bande magnétique)
- Préautorisation (de base et avec la chambre forte)
- Réautorisation

#### Éléments dont il faut tenir compte :

- Le NVC est uniquement pris en charge par Visa, Mastercard, American Express, Discover, JCB et UnionPay.  
Pour les cartes UnionPay, aucune réponse ne sera renvoyée concernant le NCV : l'émetteur approuvera ou refusera plutôt la transaction en fonction du résultat.
- Lorsque vous testez le NCV, utilisez **seulement** les numéros de carte test de Visa 4242424242424242 ou 400555444444403 ainsi que les montants indiqués dans le document Simulator eFraud Response Codes (Simulateur de codes de réponse pour la fraude en ligne, offert en anglais seulement) qui se trouve dans le portail pour développeurs de Moneris (<https://developer.moneris.com>).
- L'ID de commerce « store5 » est configuré pour prendre en charge les tests du NVC.

### 9.2.2 Transactions nécessitant le NCV

L'objet du numéro de vérification de carte (NVC) est requis dans les situations suivantes :

- les transactions initiales durant lesquelles les renseignements d'identification du titulaire de carte sont enregistrées au dossier (les transactions de suivi subséquentes n'ont pas besoin du NVC);
- toute transaction d'achat, de préautorisation ou de vérification de carte durant laquelle vous n'enregistrez pas les renseignements d'identification du titulaire de carte.

### 9.2.3 Objet CVD Information

**REMARQUE :** Le NVC doit uniquement être transmis à Passerelle Moneris. Il ne doit **jamais** être enregistré pour des transactions ultérieures ou être affiché sur le reçu.

#### Définition de l'objet CvdlInfo

```
CvdInfo cvdCheck = new CvdInfo();

$cvdTemplate = array(
    'cvd_indicator' => $cvd_indicator,
    'cvd_value' => $cvd_value
);

$mpgCvdInfo = new mpgCvdInfo ($cvdTemplate);
```

#### Méthode Set pour l'objet de transaction

```
transaction.setCvdInfo(cvdCheck);

$mpgTxn->setCvdInfo($mpgCvdInfo);
```

#### Champs de demande de l'objet CVD Info (obligatoires)

Variable	Type et limites	Description
Indicateur du NVC <cvd_indicator>	<i>Chaîne</i> 1 caractère numérique	Indique la présence d'un NVC  Valeurs possibles :  0 = Le CVC a été volontairement contourné ou n'a pas été fourni par le commerçant. 1 = Le CVC est présent. 2 = Le CVC est affiché sur la carte, mais il est illisible. 9 = Le titulaire de carte indique qu'il n'y a pas de CVC sur sa carte.

Variable	Type et limites	Description
Valeur du NVC <cvd_value>	<i>Chaîne</i> 4 caractères numériques	NVC imprimé sur la carte  <b>REMARQUE :</b> Le NVC doit uniquement être transmis à Passerelle Moneris. Il ne doit <b>jamais</b> être enregistré pour des transactions ultérieures ou être affiché sur le reçu.

Tableau 1 Objet CVD Info – Champs obligatoires

Variable	Type et limites	Méthode Set	Description
Indicateur du NVC	<i>Chaîne</i> 1 caractère numérique	<code>cvdCheck.setCvdIndicator("1");</code>	Indique la présence d'un NVC  Valeurs possibles :  0 : Le CVC a été volontairement contourné ou n'a pas été fourni par le commerçant.  1 : Le CVC est présent.  2 : Le CVC est affiché sur la carte, mais il est illisible.  9 : Le titulaire de carte indique qu'il n'y a pas de CVC sur sa carte.
Valeur du NVC	<i>Chaîne</i> 4 caractères numériques	<code>cvdCheck.setCvdValue("099");</code>	NVC imprimé sur la carte de crédit  <b>REMARQUE :</b> Le NVC doit uniquement être transmis à Passerelle Moneris. Il ne doit <b>jamais</b> être enregistré pour des transactions ultérieures ou être affiché sur le reçu.

#### 9.2.4 Codes de résultat liés au NVC

La vérification du NVC est offerte pour les transactions par carte Visa, Mastercard, Discover, American Express, JCB et UnionPay.

Code	Description
M	Correspondance
N	Aucune contribution de l'employeur
P	Non traité
S	NVC présent sur la carte, mais le commerçant a indiqué qu'il ne l'état pas
U	Émetteur qui n'utilise pas le NVC
Y	Correspondance pour American Express et JCB seulement
D	Code de sécurité non valide pour American Express et JCB seulement
Autre	Code de réponse non valide

### 9.2.5 Exemple d'une transaction d'achat avec l'objet CVD Info

Ceci est un exemple de code .NET montrant l'intégration du NVC dans une transaction d'achat. Les renseignements de l'objet Purchase non pertinents pour le NVC ont été retirés.

#### Exemple d'une transaction d'achat avec les renseignements du NVC (objet CVD Information)

```
CvdInfo cvdCheck = new CvdInfo();
cvdCheck.setCvdIndicator("1");
cvdCheck.setCvdValue("099");

Purchase purchase = new Purchase();
purchase.setCvdInfo(cvdCheck);
```

## 9.3 Outil de gestion des risques transactionnels

- 9.3.1 À propos de l'outil de gestion des risques transactionnels
- 9.3.2 Introduction aux demandes d'information (queries)
- 9.3.3 Demande d'information sur la session (Session Query)
- 9.3.4 Demande d'information sur les attributs (Attribute Query)
- 9.3.6 Ajout de balises de profilage à votre site Web
- 9.3.6 Ajout de balises de profilage à votre site Web

L'outil de gestion des risques transactionnels peut uniquement être utilisés dans le cadre d'une **intégration canadienne**.

### 9.3.1 À propos de l'outil de gestion des risques transactionnels

L'outil de gestion des risques transactionnels fournit des renseignements supplémentaires pour vous aider à détecter les transactions frauduleuses. Afin de maximiser les avantages de l'outil de gestion des risques transactionnels, nous vous recommandons fortement de faire ce qui suit :

- Réfléchissez sérieusement à la logique applicative et aux processus que vous devez mettre en place en ce qui a trait la gestion des réponses envoyées par l'outil de gestion des risques transactionnels.
- Mettez en place les autres outils de protection contre la fraude offerts par Passerelle Moneris (p. ex., le SVA, le NVC, Vérifié par Visa, Mastercard Securecode et American Express SafeKey).

### 9.3.2 Introduction aux demandes d'information (queries)

Voici les deux types de transactions associées à l'outils de gestion des risques transactionnels :

- Les demandes d'information sur la session (page 382)
- Les demandes d'information sur les attributs (page 389)

Les demandes d'information sur la session et les demandes d'information sur les attributs sont toutes les deux utilisées au moment de la transaction pour évaluer les risques.

Moneris vous recommande d'utiliser principalement les demandes d'information sur la session puisqu'elles utilisent les empreintes digitales ainsi que d'autres renseignements transactionnels afin de fournir une cote de risque.

Pour utiliser les demandes d'information sur la session, vous devez faire ce qui suit :

- Vous devez ajouter des balises sur votre site Web afin de recueillir les renseignements sur les empreintes.
- Vous devez effectuer une transaction de demande d'information sur la session.

Si vous n'êtes pas en mesure d'obtenir les renseignements requis pour effectuer une demande d'information sur la session (comme l'empreinte digitale), utilisez plutôt les demandes d'information sur les attributs.

### 9.3.3 Demande d'information sur la session (Session Query)

Lorsqu'une session de profilage est entamée sur l'appareil d'un client, l'API de demande d'information sur la session est utilisée au moment de la transaction ou pour connaître l'identifiant d'un appareil ou son « empreinte digitale », une liste d'attributs et une évaluation du risque au sujet de l'appareil du client.

#### Définition de l'objet de transaction Session Query

```
SessionQuery sq = new SessionQuery();
```

#### Objet HttpsPostRequest pour les transactions de demande d'information sur la session

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
```

```
mpgReq.setTransaction(sq);
```

#### Valeurs liées aux transactions de demande d'information sur la session

**Tableau 10 : Valeurs obligatoires pour l'objet de transaction Session Query**

Valeur	Type	Limites	Méthode Set
	Description		
ID de session	Chaîne	9 caractères décimaux Caractères autorisés : [a-z], [A-Z], 0-9, _, -	sq.setSessionId(session_id);
Identifiant de la session du serveur Web généré lorsqu'une session de profilage d'appareil commence			
Type de service	Chaîne	9 caractères décimaux	sq.setServiceType(service_type);
Indique les champs de sortie inclus dans la réponse session -- renvoie les attributs concernant l'adresse IP et l'appareil			
Type d'événement	Chaîne	Paiement	sq.setEventType(service_type);
Définit le type de transactions ou d'événement à des fins de production de rapports payment – Achat de biens et de services			
Numéro de carte de crédit (PAN)	Chaîne	20 caractère numérique Sans espace ou tiret	sq.setPan(pan);
La plupart des numéros de carte de crédit d'aujourd'hui sont composés de 16 chiffres, mais certains numéros à 13 chiffres sont toujours acceptés par certains émetteurs. La limite de ce champ est de 20 chiffres pour tenir compte des futures prolongations des numéros de carte et la prise en charge possible des marques de carte privée.			

Valeur	Type	Limites	Méthode Set
			Description
Rue 1 de l'adresse du compte	Chaîne	32 caractères alphanumériques	<code>sq.setAccountAddressStreet2("4th Flr West Tower");</code>
			Première partie de la rue de l'adresse de facturation
Rue 2 de l'adresse du compte	Chaîne	32 caractères alphanumériques	<code>sq.setAccountAddressStreet2("4th Flr West Tower");</code>
			Deuxième partie de la rue de l'adresse de facturation
Ville de l'adresse du compte	Chaîne	50 caractères alphanumériques	<code>sq.setAccountAddressCity("Toronto");</code>
			Ville de l'adresse de facturation
Province ou état de l'adresse du compte	Chaîne	64 caractères alphanumériques	<code>sq.setAccountAddressState("Ontario");</code>
			Province ou état de l'adresse de facturation
Pays de l'adresse du compte	Chaîne	2 caractères alphanumériques	<code>sq.setAccountAddressCountry("CA");</code>
			Code ISO2 représentant le pays de l'adresse de facturation
Code postal ou ZIP de l'adresse du compte	Chaîne	8 caractères alphanumériques	<code>sq.setAccountAddressZip("M8X2X2");</code>
			Code postal ou ZIP de l'adresse de facturation
Rue 1 de l'adresse d'expédition du compte	Chaîne	32 caractères alphanumériques	<code>sq.setShippingAddressStreet1("3300 Bloor St W");</code>
			Première partie de la rue de l'adresse d'expédition
Rue 2 de l'adresse d'expédition du compte (Shipping address street 2)	Chaîne	32 caractères alphanumériques	<code>sq.setShippingAddressStreet2("4th Flr West Tower");</code>
			Deuxième partie de la rue de l'adresse d'expédition
Ville de l'adresse d'expédition	Chaîne	50 caractères alphanumériques	<code>sq.setShippingAddressCity("Toronto");</code>
			Ville de l'adresse d'expédition

Valeur	Type	Limites	Méthode Set
			Description
Province ou état de l'adresse d'expédition	Chaîne	64 caractères alphanumériques	<code>sq.setShippingAddressState("Ontario");</code>
Province ou état de l'adresse d'expédition			
Pays de l'adresse d'expédition	Chaîne	2-character alphanumeric	<code>sq.setShippingAddressCountry("CA");</code>
	Code ISO2 représentant le pays de l'adresse d'expédition		
Code postal ou ZIP de l'adresse d'expédition	Chaîne	8 caractères alphanumériques	<code>sq.setAccountAddressZip("M8X2X2");</code>
	Code postal ou ZIP de l'adresse d'expédition		
Attribut 1-5 local	Chaîne	255 caractères alphanumériques	<code>sq.setLocalAttrib1("a");</code>
	Ces cinq attributs peuvent être utilisés pour transférer des données d'attribut personnalisées. Vous pouvez utiliser ces attributs pour mettre en corrélation certaines données et les renseignements fournis par l'appareil.		
Montant de la transaction	Chaîne	255 caractères alphanumériques Doit comporter 2 décimales	<code>sq.setTransactionAmount("1.00");</code>
	Montant numérique de la transaction		
Devise de la transaction	Chaîne	10 caractère numérique	<code>sq.setTransactionCurrency("CAN");</code>
	Il s'agit de la devise de la transaction. Si la valeur TransactionAmount est incluse, la valeur TransactionCurrency est requise.  Voici les valeurs à utiliser : CAD – 124 USD – 840		

**Tableau 11 : Valeurs facultatives pour l'objet de transaction Session Query**

Valeur	Type	Limites	Méthode Set
			Description
Identifiant du	Chaîne	255 caractères alphanumériques	<code>sq.setAccountLogin("13195417-8CA0-46cd-960D-14C158E4DBB2");</code>

## 9. Outils de protection contre la fraude en ligne

Valeur	Type	Limites		Méthode Set
		Description		
compte	Nom de l'identifiant du compte			
Algorithme de hachage du mot de passe	Chaîne	40 caractères alphanumériques	sq.setPasswordHash ("489c830f10f7c601d30599a0dea f66e64d2aa50a");	L'entrée doit être un algorithme de hachage SHA-2 du mot de passe en format hexadécimal. Cette valeur vérifie si le mot de passe est sur une liste de vérification.
Numéro de compte	Chaîne	255 caractères alphanumériques	sq.setAccountNumber ("3E17A905-AC8A-4c8d-A417-3DADA2A55220");	Numéro du compte
Nom du compte	Chaîne	255 caractères alphanumériques	sq.setAccountName ("4590FCC0-DF4A-44d9-A57B-AF9DE98B84DD");	Nom du compte (ou concaténation du prénom et nom de famille du détenteur du compte)
Adresse courriel du compte	Chaîne	100 caractères alphanumériques	sq.SetAccountEmail ("3CAE72EF-6B69-4a25-93FE-2674735E78E8@test.threatmetrix.com");	Il s'agit de l'adresse courriel entrée dans le formulaire pour ce contact. Elle permet de vérifier s'il s'agit de l'ID de l'adresse courriel d'un compte à haut risque.
Numéro de téléphone du compte	Chaîne	32 caractères alphanumériques		
	Numéro de téléphone du contact, y compris le code du pays et régional Ne doit contenir aucun espace blanc			
	Doit avoir le format suivant : 0..9,<space>,(),[,] Les accolades doivent être jumelées.			
Rue 1 de l'adresse (address street 2)	Chaîne	32 caractères alphanumériques		
	Première partie de la rue de l'adresse du compte			
Rue de l'adresse 2	Chaîne	32 caractères alphanumériques		
	Deuxième partie de la rue de l'adresse du compte			
Ville de l'adresse	Chaîne	50 caractères alphanumériques		
	Ville de l'adresse du compte			
Province ou état	Chaîne	64 caractères alphanumériques		

Valeur	Type	Limites	Méthode Set
	Description		
de l'adresse	Province ou état de l'adresse du compte		
Pays	Chaîne	2 caractères alphanumériques	
	Code ISO2 à 2 caractères représentant le pays de l'adresse du compte		
Code postal ou ZIP de l'adresse	Chaîne	8 caractères alphanumériques	
	Code postal ou ZIP de l'adresse du compte		
Rue 1 de l'adresse d'expédition (ship address street 2)	Chaîne	32 caractères alphanumériques	
	Première partie de la rue de l'adresse d'expédition		
Rue 2 de l'adresse d'expédition (ship address street 2)	Chaîne	32 caractères alphanumériques	
	Deuxième partie de la rue de l'adresse d'expédition		
Ville de l'adresse d'expédition	Chaîne	50 caractères alphanumériques	
	Ville de l'adresse d'expédition		
Province ou état de l'adresse d'expédition	Chaîne	64 caractères alphanumériques	
	Province ou état de l'adresse d'expédition		
Pays de l'adresse d'expédition	Chaîne	2 caractères alphanumériques	
	Code ISO2 à 2 caractères représentant le pays de l'adresse d'expédition		
Code postal ou	Chaîne	8 caractères alphanumériques	

Valeur	Type	Limites	Méthode Set
	Description		
ZIP de l'adresse d'expédition		Code postal ou ZIP de l'adresse d'expédition	
Algorithme de hachage du numéro de carte de crédit	Chaîne	255 caractères alphanumériques	Il s'agit de l'algorithme de hachage SHA-2 (en format hexadécimal) du numéro de carte de crédit.
Attribut 1-8 personnalisé	Chaîne	255 caractères alphanumériques	Ces huit attributs peuvent être utilisés pour transférer des données personnalisées, qui peuvent ensuite être utilisés selon les règles.

#### Exemple d'une transaction de demande d'information sur la session – CA

```

package Canada;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.Map;
import JavaAPI.*;
public class TestCanadaRiskCheckSession
{
    public static void main(String[] args)
    {
        String store_id = "moneris";
        String api_token = "hurgle";
        java.util.Date createDate = new java.util.Date();
        String order_id = "Test"+createDate.getTime();
        String session_id = "abc123";
        String service_type = "session";
        //String event_type = "LOGIN";
        String processing_country_code = "CA";
        boolean status_check = false;
        SessionQuery sq = new SessionQuery();
        sq.setOrderId(order_id);
        sq.setSessionId(session_id);
        sq.setServiceType(service_type);
        sq.setEventType(service_type);
        //sq.setPolicy("");
        //sq.setDeviceId("4EC40DE5-0770-4fa0-BE53-981C067C598D");
        sq.setAccountLogin("13195417-8CA0-46cd-960D-14C158E4DBB2");
        sq.setPasswordHash("489c830f10f7c601d30599a0deaf66e64d2aa50a");
        sq.setAccountNumber("3E17A905-AC8A-4c8d-A417-3DADA2A55220");
        sq.setAccountName("4590FCC0-DF4A-44d9-A57B-AF9DE98B84DD");
        sq.setAccountEmail("3CAE72EF-6B69-4a25-93FE-2674735E78E8@test.threatmetrix.com");

        //sq.setAccountTelephone("5556667777");
        sq.setPan("4242424242424242");
        //sq.setAccountAddressStreet1("3300 Bloor St W");
        //sq.setAccountAddressStreet2("4th Flr West Tower");
        //sq.setAccountAddressCity("Toronto");
        //sq.setAccountAddressState("Ontario");
        //sq.setAccountAddressCountry("CA");
        //sq.setAccountAddressZip("M8X2X2");
        //sq.setShippingAddressStreet1("3300 Bloor St W");
        //sq.setShippingAddressStreet2("4th Flr West Tower");
        //sq.setShippingAddressCity("Toronto");
        //sq.setShippingAddressState("Ontario");
    }
}

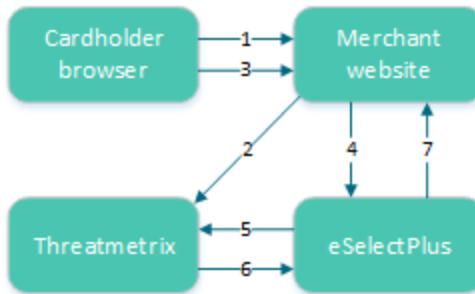
```

```

//sq.setShippingAddressCountry("CA");
//sq.setShippingAddressZip("M8X2X2");
//sq.setLocalAttrib1("a");
//sq.setLocalAttrib2("b");
//sq.setLocalAttrib3("c");
//sq.setLocalAttrib4("d");
//sq.setLocalAttrib5("e");
//sq.setTransactionAmount("1.00");
//sq.setTransactionCurrency("840");
//set SessionAccountInfo
sq.setTransactionCurrency("CAN");
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false for production transactions
mpgReq.setstoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(sq);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
String[] rules;
Hashtable<String, String> results = new Hashtable<String, String>();
Receipt receipt = mpgReq.getReceipt();
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("Message = " + receipt.getMessage());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
results = receipt.getRiskResult();
Iterator<Map.Entry<String, String>> response = results.entrySet().iterator();
while (response.hasNext())
{
Map.Entry<String, String> entry = response.next();
System.out.println(entry.getKey().toString() + " = " + entry.getValue().toString());
}
rules = receipt.getRiskRules();
for (int i = 0; i < rules.length; i++)
{
System.out.println("RuleName = " + rules[i]);
System.out.println("RuleCode = " + receipt.getRuleCode(rules[i]));
System.out.println("RuleMessageEn = " + receipt.getRuleMessageEn(rules[i]));
System.out.println("RuleMessageFr = " + receipt.getRuleMessageFr(rules[i]));
}
}
catch (Exception e)
{
e.printStackTrace();
}
}
}

```

### 9.3.3.1 Flux d'une transaction de demande d'information sur la session



**Image 3 : Flux d'une transaction de demande d'information sur la session**

1. Le titulaire de carte se connecte au site Web du commerçant.
2. Une fois la page chargée dans le navigateur Web du titulaire de carte, des balises spéciales intégrées au site permettent de recueillir des renseignements de l'appareil et de les envoyer à ThreatMetrix en tant qu'empreinte digitale de l'appareil.
3. Le client effectue une transaction.
4. L'application du site Web du commerçant effectue une transaction de demande d'information sur la session à Passerelle Moneris en utilisant le même ID de session que celui inclus dans l'empreinte digitale de l'appareil. Cet appel doit avoir lieu dans les 30 minutes suivant le profilage (2).
5. Passerelle Moneris transfère les données de la demande d'information sur la session à ThreatMetrix.
6. ThreatMetrix utilise les données de la demande d'information sur la session ainsi que les renseignements liés à l'empreinte digitale de l'appareil pour évaluer la transaction selon les règles établies. Une cote est générée en fonction des règles.
7. Le commerçant utilise les renseignements reçus dans son analyse de risque pour prendre une décision. Il peut décider de conclure la transaction de paiement du titulaire de carte ou bien de l'annuler.

### 9.3.4 Demande d'information sur les attributs (Attribute Query)

La demande d'information sur les attributs permet d'évaluer le risque d'une transaction en utilisant les identifiants liés à la transaction, comme l'adresse courriel et le numéro de carte. Contrairement à la demande d'information sur la séance, cette demande ne nécessite pas de renseignements sur l'empreinte digitale de l'appareil.

#### Définition de l'objet de transaction AttributeQuery

```
AttributeQuery aq = new AttributeQuery();
```

#### Objet HttpsPostRequest pour les transactions de demande d'information sur les attributs

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setTransaction(aq);
```

#### Valeurs liées aux transactions de demande d'information sur les attributs

**Tableau 12 : Valeurs obligatoires pour l'objet de transaction Attribute Query**

Valeur	Type	Limites	Méthode Set
	Description		
Type de service	Chaîne	S. O.	<code>aq.setServiceType(service_type);</code>
		Indique les champs de sortie inclus dans la réponse session -- renvoie les attributs concernant l'adresse IP et l'appareil	
ID de l'appareil (device ID)	Chaîne	36 caractères alphanumériques	<code>aq.setDeviceId("");</code>
		Identifiant d'appareil unique généré par un appel antérieur à l'API de demande d'information sur la session de ThreatMetrix	
Numéro de carte de crédit	Chaîne	20 caractère numérique Sans espace ou tiret	<code>aq.setPan(pan);</code>
		La plupart des numéros de carte de crédit d'aujourd'hui sont composés de 16 chiffres, mais certains numéros à 13 chiffres sont toujours acceptés par certains émetteurs. La limite de ce champ est de 20 chiffres pour tenir compte des futures prolongations des numéros de carte et la prise en charge possible des marques de carte privée.	
Adresse IP	Chaîne	64 caractères alphanumériques	<code>aq.setIPAddress("192.168.0.1");</code>
		Il s'agit de l'adresse IP réelle. Les résultats seront notamment <code>true_ip_geo</code> et <code>true_ip_score</code> .	
Adresse IP transférée	Chaîne	64 caractères alphanumériques	<code>aq.setIPForwarded("192.168.1.0");</code>
		Il s'agit de l'adresse IP du serveur mandataire (proxy). Si l'adresse IP est fournie, les résultats seront <code>proxy_ip_geo</code> et <code>proxy_ip_score</code> .  Si aucune adresse IP n'est fournie, cette adresse IP sera considérée comme l'adresse IP réelle et les résultats seront notamment <code>true_ip_geo</code> et <code>true_ip_score</code> .	
Rue 1 de l'adresse du compte	Chaîne	32 caractères alphanumériques	<code>aq.setAccountAddressStreet1("330 0 Bloor St W");</code>
		Première partie de la rue de l'adresse de facturation	
Rue 2 de l'adresse du compte	Chaîne	32 caractères alphanumériques	<code>aq.setAccountAddressStreet2("4th Flr West Tower");</code>
		Deuxième partie de la rue de l'adresse de facturation	
Ville de l'adresse du	Chaîne	50 caractères alphanumériques	<code>aq.setAccountAddressCity("Toronto");</code>

Valeur	Type	Limites	Méthode Set
	Description		
compte	Ville de l'adresse de facturation		
Province ou état de l'adresse du compte	Chaîne	64 caractères alphanumériques	aq.setAccountAddressState ("Ontario");
	Province ou état de l'adresse de facturation		
Pays de l'adresse du compte	Chaîne	2 caractères alphanumériques	aq.setAccountAddressCountry ("CA");
	Code ISO2 représentant le pays de l'adresse de facturation		
Code postal ou ZIP de l'adresse du compte	Chaîne	8 caractères alphanumériques	aq.setAccountAddressZip ("M8X2X2");
	Code postal ou ZIP de l'adresse de facturation		
Rue 1 de l'adresse d'expédition du compte	Chaîne	32 caractères alphanumériques	aq.setShippingAddressStreet1 ("3300 Bloor St W");
	Pays de l'adresse du compte		
Rue 2 de l'adresse d'expédition	Chaîne	32 caractères alphanumériques	aq.setShippingAddressStreet2 ("4th Flr West Tower");
	Deuxième partie de la rue de l'adresse d'expédition		
Ville de l'adresse d'expédition	Chaîne	50 caractères alphanumériques	aq.setShippingAddressCity ("Toronto");
	Ville de l'adresse d'expédition		
Province ou état de l'adresse d'expédition	Chaîne	64 caractères alphanumériques	aq.setShippingAddressState ("Ontario");
	Province ou état de l'adresse d'expédition		
Pays de l'adresse d'expédition	Chaîne	2 caractères alphanumériques	aq.setShippingAddressCountry ("CA");
	Code ISO2 représentant le pays de l'adresse d'expédition		
Code postal ou ZIP de l'adresse du compte	Chaîne	8 caractères alphanumériques	aq.setAccountAddressZip ("M8X2X2");
	Code postal ou ZIP de l'adresse d'expédition		

**Exemple d'une transaction de demande d'information sur les attributs**

```

String store_id = "moneris";
String api_token = "hurgle";
java.util.Date createDate = new java.util.Date();
String order_id = "Test"+createDate.getTime();
String service_type = "session";
String processing_country_code = "CA";
boolean status_check = false;

AttributeQuery aq = new AttributeQuery();
aq.setOrderId(order_id);
aq.setServiceType(service_type);
aq.setDeviceId("");
aq.setAccountLogin("13195417-8CA0-46cd-960D-14C158E4DBB2");
aq.setPasswordHash("489c830f10f7c601d30599a0deaf66e64d2aa50a");
aq.setAccountNumber("3E17A905-AC8A-4c8d-A417-3DADA2A55220");
aq.setAccountName("4590FCC0-DF4A-44d9-A57B-AF9DE98B84DD");
aq.setAccountEmail("3CAE72EF-6B69-4a25-93FE-2674735E78E8@test.threatmetrix.com");
//aq.setCCNumberHash("4242424242424242");
//aq.setIPAddress("192.168.0.1");
//aq.setIPForwarded("192.168.1.0");
aq.setAccountAddressStreet1("3300 Bloor St W");
aq.setAccountAddressStreet2("4th Flr West Tower");
aq.setAccountAddressCity("Toronto");
aq.setAccountAddressState("Ontario");
aq.setAccountAddressCountry("CA");
aq.setAccountAddressZip("M8X2X2");
aq.setShippingAddressStreet1("3300 Bloor St W");
aq.setShippingAddressStreet2("4th Flr West Tower");
aq.setShippingAddressCity("Toronto");
aq.setShippingAddressState("Ontario");
aq.setShippingAddressCountry("CA");
aq.setShippingAddressZip("M8X2X2");

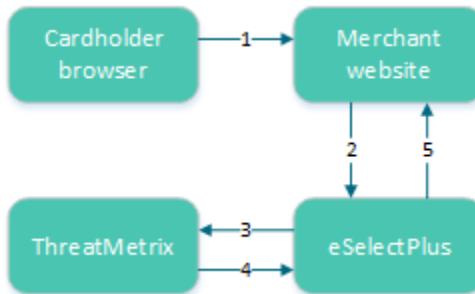
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setTransaction(aq);
mpgReq.send();

try
{
    String[] rules;
    Hashtable<String, String> results = new Hashtable<String, String>();
    Receipt receipt = mpgReq.getReceipt();
    System.out.println("ResponseCode = " + receipt.getResponseCode());
    System.out.println("Message = " + receipt.getMessage());
    System.out.println("TxnNumber = " + receipt.getTxnNumber());

    results = receipt.getRiskResult();
    Iterator<Map.Entry<String, String>> response = results.entrySet().iterator();
    while (response.hasNext())
    {
        Map.Entry<String, String> entry = response.next();
        System.out.println(entry.getKey().toString() + " = " +
        entry.getValue().toString());
    }
    rules = receipt.getRiskRules();
    for (int i = 0; i < rules.length; i++)
    {
        System.out.println("RuleName = " + rules[i]);
        System.out.println("RuleCode = " + receipt.getRuleCode(rules[i]));
        System.out.println("RuleMessageEn = " + receipt.getRuleMessageEn(rules[i]));
        System.out.println("RuleMessageFr = " + receipt.getRuleMessageFr(rules[i]));
    }
}

```

### 9.3.4.1 Flux d'une transaction de demande d'information sur les attributs



**Image 4 : Flux d'une transaction de demande d'information sur les attributs**

8. Le titulaire de carte se connecte sur le site Web du commerçant et effectue une transaction.
9. L'application Web du commerçant effectue une transaction de demande d'information sur les attributs incluant l'ID de session à Passerelle Moneris.
10. Passerelle Moneris transfère les données de la demande d'information sur les attributs à ThreatMetrix.
11. ThreatMetrix utilise les données de la demande sur les attributs pour évaluer la transaction selon les règles établies. Une cote est générée en fonction des règles.
12. Le commerçant utilise les renseignements reçus dans son analyse de risque pour prendre une décision. Il peut décider de conclure la transaction de paiement du titulaire de carte ou bien de l'annuler.

### 9.3.5 Gérer les réponses

Lorsque vous examinez les réponses et que vous déterminez la façon de gérer la transaction, nous vous recommandons d'utiliser les renseignements suivants de façon manuelle ou automatique par l'entremise de la logique de votre site Web :

- Cotation des risques
- Règles déclenchées (p. ex. codes de la règle, noms de la règle, messages de la règle)
- Résultats envoyés par Vérifié par Visa, Mastercard Securecode, le SVA, le NVC et l'autorisation de la banque
- Codes de réponses de l'outil de gestions des risques transactionnels inclus dans les processus automatiques

### 9.3.5.1 Champs de réponse de l'outil de gestion des risques transactionnels (TRMT)

Tableau 13 : Valeurs de réponse de l'objet Receipt liées à l'outil de gestion des risques transactionnels

Valeur	Type	Limites	Méthode Get
	Définition		
Code de réponse	Chaîne	3 caractères alphanumériques	<code>receipt.getResponseCode();</code>
001 = Succès 980 = Erreur de données 982 = ID de commande dupliqué 983 = Transaction non valide 984 = Transaction évaluée précédemment 985 = Description d'activité non valide 986 = Description d'incidence non valide 987 = Description de confidence non valide 988 = Impossible de trouver la transaction précédente			
Message	Chaîne	S. O.	<code>receipt.getMessage();</code>
Messages de réponse			
Type d'événement	Chaîne	S. O.	
Type de transaction ou d'événement renvoyé dans la réponse			
Org ID	Chaîne	S. O.	
Identifiant unique de transaction généré par ThreatMetrix			
Politique	Chaîne	S. O.	
Il s'agit de la politique utilisée pour la demande d'information sur la session qui sera renvoyée avec la demande. Si la politique n'a pas été incluse, le nom par défaut de la politique est renvoyé.			
Cote de la politique	Chaîne	S. O.	
Il s'agit de la somme de la pondération de tous les risques décelés par les règles déclenchées en fonction de la fourchette établie par la politique sélectionnée [-100...100].			
Durée de	Chaîne	S. O.	

Valeur	Type	Limites	Méthode Get
	Définition		
la demande	Durée de traitement de la transaction		
ID de la demande	Chaîne	S. O.	
	Numéro unique qui sera toujours renvoyé avec la demande de retour		
Résultat de la demande	Chaîne	S. O.	<code>receipt.getRiskResult();</code>
	Voir la section 9.3.5.1 (page 410).		
État de l'évaluation	Chaîne	S. O.	
	État de la transaction selon les évaluations et les cotes de risque		
Évaluation de risque	Chaîne	S. O.	
	Évaluation de la transaction selon les évaluations et les cotes de risque		
Type de service	Chaîne	S. O.	
	Type de service qui sera toujours inclus dans la réponse de la demande d'information sur les attributs		
ID de session	Chaîne	S. O.	
	Identifiant temporaire unique à chaque visiteur qui sera inclus dans la demande de retour		
Sommaire de la cote de risque	Chaîne	S. O.	
	Basé sur toutes les valeurs retournées dans la fourchette établie [-100 ... 100]		
ID de transaction	Chaîne	S. O.	
	Identifiant de la transaction qui sera toujours inclus dans la réponse lorsqu'il est inclus dans l'entrée		
Session inconnue	Chaîne	S. O.	
	Si cette valeur est présente, elle est toujours « yes » (oui). Elle indique qu'un ID de session a été transmis, mais qu'il n'a pas été trouvé.		

Tableau 14 : Descriptions des codes de réponse

Valeur	Définition
001	Réussite
981	Erreur de données

Valeur	Définition
982	ID de commande dupliqué
983	Transaction non valide
984	Transaction évaluée précédemment
985	Description d'activité non valide
986	Description d'incidence non valide
987	Description de confidence non valide
988	Impossible de trouver la transaction précédente

**Tableau 15 : Descriptions et valeurs des résultats de la demande**

Valeur	Définition
fail_duplicate_entities_of_same_type	Plusieurs entités de la même catégorie ont été indiquées, par exemple, password_hash a été indiqué deux fois.
fail_incomplete	ThreatMetrix n'a pas pu traiter la demande en raison de données incomplètes ou incorrectes.
fail_invalid_account_number	Le format du numéro de compte fourni était non valide.
fail_invalid_characters	Des caractères non valides ont été soumis.
fail_invalid_charset	La valeur character set était non valide.
fail_invalid_currency_code	Le format de la valeur currency_code était non valide.
fail_invalid_currency_format	Le format de la valeur currency_format était non valide.
fail_invalid_telephone_number	Le format du numéro de téléphone fourni était non valide.
fail_access	ThreatMetrix n'a pas pu traiter la demande, car la vérification de l'API a échoué.
fail_internal_error	ThreatMetrix a rencontré une erreur lors du traitement de la demande.

Valeur	Définition
fail_invalid_device_id	Le format de la valeur device_id fournie était invalide.
fail_invalid_email_address	Le format de l'adresse courriel fournie était non valide.
fail_invalid_fuzzy_device_id	Le format de la valeur fuzzy_device_id était non valide.
fail_invalid_ip_address_parameter	Le format du paramètre ip_address fourni était non valide.
fail_invalid_parameter	Le format du paramètre était non valide, ou la valeur dépasse les limites.
fail_invalid_sha_hash	Le format d'un paramètre précisé comme un hachage sha était non valide. Le hachage sha comprenait le hachage sha1/2/3.
fail_invalid_submitter_id	Le format de l'ID du demandeur était non valide ou la valeur dépassait les limites.
fail_no_policy_configured	Aucune politique n'a été configurée en fonction de la valeur org_id.
fail_not_enough_params	Le nombre d'attributs de l'appareil collectés pendant le profilage était insuffisant pour effectuer une comparaison d'empreintes.
fail_parameter_overlength	La valeur du paramètre était trop longue.
fail_temporarily_unavailable	La demande a échoué, car le service est temporairement indisponible.
fail_too_many_instances_of_same_parameter	Des valeurs multiples étaient attribuées à des paramètres qui acceptent uniquement une.
fail_verification	La limite de requête de l'API a été atteinte
success	ThreatMetrix a réussi à traiter la demande.

### 9.3.5.2 Comprendre la cotation des risques

Pour chaque demande de session ou demande d'attribut, une cotation dont la valeur se situe entre -100 et +100 est renvoyée en fonction des règles qui ont été déclenchées pour la transaction.

Le Tableau 16 définit les fourchettes de cotation de risque.

**Tableau 16 : Définition des cotations de risque des demandes de session et des demandes d'attributs**

Cotation des risques	Définition de Visa
De -100 à -1	Un score plus faible signifie une plus grande probabilité que la transaction soit frauduleuse.
0	Transaction neutre
De 1 à 100	Un score plus élevé signifie une plus faible une probabilité que la transaction soit frauduleuse. <b>Remarque :</b> Toutes les transactions de commerce électronique comportent un certain niveau de risque. Il est donc rare de voir une cotation du risque avec une valeur positive élevée.

Lors de l'évaluation du risque d'une transaction, la cotation du risque donne un premier indicateur du risque potentiel que la transaction soit frauduleuse. Étant donné que certaines des règles appliquées à chaque transaction peuvent ne pas concerner la situation de votre entreprise, examinez les règles qui ont été déclenchées lors de la transaction avant de déterminer comment la traiter.

### 9.3.5.3 Comprendre les codes de règle, les noms de règle et les messages de règle

Les codes de règle, les noms de règle et les messages de règle contiennent des détails sur les règles qui ont été déclenchées pendant l'évaluation des renseignements fournis dans la demande d'information de session ou d'attribut. Chaque code de règle possède un nom de règle et un message de règle. Le nom de la règle et le message de la règle sont généralement similaires. Le Tableau 17 contient des renseignements supplémentaires sur chaque règle.

Lorsque vous évaluez le risque d'une transaction, il est recommandé de revoir les règles qui ont été déclenchées pour cette transaction et d'en évaluer la pertinence pour votre entreprise. (C'est-à-dire, quel est le lien avec les habitudes d'achat typiques de votre clientèle).

Si vous automatissez la totalité ou une partie des processus décisionnels liés au traitement des réponses, vous pourriez vouloir utiliser les codes de règle. Si vous documentez des processus manuels, vous pouvez vous référer au nom de la règle ou au message de la règle, qui sont plus faciles à utiliser.

**Tableau 17 : Noms, numéros et messages de règles**

Nom de règle	Numéro de règle	Message de règle
	Explication de la règle	
Listes blanches		
DeviceWhitelisted	WL001	Appareil sur la liste blanche

## 9. Outils de protection contre la fraude en ligne

Nom de règle	Numéro de règle	Message de règle
	Explication de la règle	
	L'appareil est sur la liste blanche. Cela signifie que l'appareil est signalé comme étant toujours « ok ».	
	<b>Remarque :</b> Cette règle n'est pas offerte actuellement.	
IPWhitelisted	WL002	Adresse IP sur la liste blanche
	L'adresse IP est sur la liste blanche. Cela signifie que l'appareil est signalé comme étant toujours « ok ».	
	<b>Remarque :</b> Cette règle n'est pas offerte actuellement.	
EmailWhitelisted	WL003	Adresse courriel sur la liste blanche
	L'adresse courriel est sur la liste blanche. Cela signifie que l'appareil est signalé comme étant toujours « ok ».	
	<b>Remarque :</b> Cette règle n'est pas offerte actuellement.	
Vélocité de l'événement		
2DevicePayment	EV003	Vélocité des paiements de l'appareil de 2
	Plusieurs paiements ont été détectés à partir de cet appareil au cours des dernières 24 heures.	
2IPPPaymentVelocity	EV006	Vélocité des paiements de l'adresse IP de 2
	Plusieurs paiements ont été détectés à partir de cette adresse IP au cours des dernières 24 heures.	
2ProxyPaymentVelocity	EV008	Vélocité des paiements du serveur mandataire de 2
	L'appareil a utilisé trois serveurs mandataires différents ou plus sur une période de 24 heures. Il peut s'agir d'un risque ou d'une personne utilisant un serveur mandataire d'entreprise légitime.	
Courriel		
3EmailPerDeviceDay	EM001	3 courriels pour l'ID de l'appareil en un jour
	Cet appareil a présenté trois courriels différents au cours des dernières 24 heures.	
3EmailPerDeviceWeek	EM002	3 courriels pour l'ID de l'appareil en une semaine
	Cet appareil a présenté 3 courriels différents au cours de la dernière semaine.	

Nom de règle	Numéro de règle		Message de règle
	Explication de la règle		
3DevciePerEmailDay	EM003	3 ID d'appareils pour une même adresse courriel en un jour	
	Cette adresse courriel a été présentée à partir de trois appareils différents au cours des dernières 24 heures.		
3DevciePerEmailWeek	EM004	3 ID d'appareils pour une même adresse courriel en une semaine	
	Cette adresse courriel a été présentée à partir de trois appareils différents au cours de la dernière semaine.		
EmailDistanceTravelled	EM005	Distance parcourue par le courriel	
	Cette adresse email a été associée à différents emplacements physiques dans une courte période de temps.		
3EmailPerSmartIDHour	EM006	3 courriels pour la valeur SmartID en 1 heure	
	La valeur SmartID de cet appareil a été associée à trois adresses courriel différentes en une heure.		
GlobalEMailOverOneMonth	EM007	Courriel global il y a plus d'un mois	
		L'adresse courriel été utilisée dans la transaction il y a plus de 30 jours. Cela indique généralement que la transaction est moins risquée.	
		<b>Remarque :</b> Cette règle est définie de manière à ne pas avoir d'incidence sur la cote de la police ou la cote de risque.	
ComputerGeneratedEmailAddress	EM008	Adresse courriel produite par ordinateur	
		Cette transaction a utilisé une adresse électronique produite par ordinateur.	
Numéro de compte			
3AccountNumberPerDeviceDay	AN001	3 numéros de compte pour l'appareil en un jour	
	Cet appareil a présenté trois comptes utilisateurs différents au cours des dernières 24 heures.		
3AccountNumberPerDeviceWeek	AN002	3 numéros de compte pour l'appareil en un jour	
	Cet appareil a présenté trois adresses courriel différentes au cours de la dernière semaine.		
3DevciePerAccountNumberDay	AN003	3 ID d'appareils pour une même adresse courriel en un jour	

## 9. Outils de protection contre la fraude en ligne

Nom de règle	Numéro de règle	Message de règle
	Explication de la règle	
	Ce compte d'utilisateur a été utilisé à partir de trois appareils différents au cours des dernières 24 heures.	
3DevciePerAccountNumberWeek	AN004	3 ID d'appareils pour une même adresse courriel en une semaine
	Ce numéro de carte a été utilisé à partir de trois appareils différents au cours de la dernière semaine.	
AccountNumberDistanceTravelled	AN005	Distance parcourue par le numéro de compte
	Ce numéro de carte a été utilisé à partir de différents emplacements physiques dans une courte période de temps.	
Carte de crédit ou paiements		
3CreditCardPerDeviceDay	CP001	3 cartes de crédit pour l'appareil en un jour
	Cet appareil a utilisé trois cartes de crédit en 24 heures.	
3CreditCardPerDeviceWeek	CP002	3 cartes de crédit pour l'appareil en une semaine
	Cet appareil a utilisé trois cartes de crédit en une semaine.	
3DevicePerCreditCardDay	CP003	3 ID d'appareils pour une même carte de crédit en un jour
	Cette carte de crédit a été utilisée sur trois appareils différents en 24 heures.	
3DevciePerCreditCardWeek	CP004	3 ID d'appareils pour une même carte de crédit en une semaine
	Cette carte de crédit a été utilisé sur trois appareils différents en une semaine.	
CreditCardDistanceTravelled	CP005	La carte de crédit a voyagé
	La carte de crédit a été utilisé à différents emplacements physiques dans une courte période de temps.	
CreditCardShipAddressGeoMismatch	CP006	L'adresse de la carte de crédit et l'adresse d'expédition ne correspondent pas
	La carte de crédit a été émise dans une région différente de l'adresse d'expédition fournie.	
CreditCardBillAddressGeoMismatch	CP007	L'adresse de la carte de crédit et l'adresse de facturation ne correspondent pas

Nom de règle	Numéro de règle	Message de règle
	Explication de la règle	
CreditCardDeviceGeoMismatch	CP008	L'emplacement de la carte de crédit et de l'appareil ne correspondent pas
		L'appareil est situé dans une région différente de celle où la carte a été émise.
CreditCardBINShipAddressGeoMismatch	CP009	Le lieu d'émission de la carte de crédit et l'adresse d'expédition ne correspondent pas.
		La carte de crédit a été émise dans une région différente de l'adresse d'expédition fournie.
CreditCardBINBillAddressGeoMismatch	CP010	L'adresse d'émission de la carte de crédit de la carte de crédit et l'adresse de facturation ne correspondent pas
		La carte de crédit a été émise dans une région différente de l'adresse de facturation fournie.
CreditCardBINDeviceGeoMismatch	CP011	Le lieu d'émission de la carte de crédit et l'emplacement de l'appareil ne correspondent pas.
		L'appareil est situé dans une région différente de celle où la carte a été émise.
TransactionValueDay	CP012	Seuil quotidien de la valeur de la transaction
		La valeur de la transaction dépasse le seuil quotidien.
TransactionValueWeek	CP013	Seuil hebdomadaire de la valeur de la transaction
		La valeur de la transaction dépasse le seuil hebdomadaire.
Règles du serveur mandataire		
3ProxyPerDeviceDay	PX001	3 adresses IP de serveur mandataire en un jour
		Cet appareil a utilisé trois serveurs mandataires différents au cours des dernières 24 heures.
AnonymousProxy	PX002	Adresse IP de serveur mandataire anonyme
		Cet appareil utilise un serveur mandataire anonyme
UnusualProxyAttributes	PX003	Attributs de serveur mandataire inhabituels

## 9. Outils de protection contre la fraude en ligne

Nom de règle	Numéro de règle	Message de règle
	Explication de la règle	
	Cette transaction provient d'une source avec des attributs de serveur mandataire inhabituels.	
AnonymousProxy	PX004	Serveur mandataire anonyme
	Cet appareil se connecte par l'entremise d'une connexion mandataire anonyme.	
HiddenProxy	PX005	Serveur mandataire masqué
	Cet appareil se connecte par l'entremise d'un serveur mandataire masqué.	
OpenProxy	PX006	Serveur mandataire ouvert
	Cette transaction provient d'une source qui utilise un serveur mandataire ouvert.	
TransparentProxy	PX007	Serveur mandataire transparent
	Cette transaction provient d'une source qui utilise un serveur mandataire transparent.	
DeviceProxyGeoMismatch	PX008	Correspondance entre le serveur mandataire et la géolocalisation réelle
	Cet appareil se connecte par l'entremise d'un serveur mandataire qui ne correspond pas à la véritable géolocalisation de l'appareil.	
ProxyTrueISPMismatch	PX009	Correspondance entre le serveur mandataire et l'adresse IP réelle
	Cet appareil se connecte par l'entremise d'un serveur mandataire qui ne correspond pas à la véritable adresse IP de l'appareil.	
ProxyTrueOrganizationMismatch	PX010	Correspondance entre le serveur mandataire et l'organisation réelle
	Les renseignements du serveur mandataire et du véritable fournisseur de services Internet pour cette source ne correspondent pas.	
DeviceProxyRegionMismatch	PX011	Correspondance entre le serveur mandataire et la région réelle
	Les renseignements sur le serveur mandataire et la région de l'appareil ne correspondent pas.	
ProxyNegativeReputation	PX012	L'adresse IP du serveur mandataire est signalée comme étant risquée dans le réseau de réputation.

Nom de règle	Numéro de règle	Message de règle
	Explication de la règle	
	Cet appareil se connecte à partir d'un serveur mandataire dont la réputation est négative.	
SatelliteProxyISP	PX013	Serveur mandataire par satellite
	Cette transaction provient d'une source qui utilise un serveur mandataire par satellite.	
Géolocalisation		
DeviceCountriesNotAllowed	GE001	La géolocalisation véritable fait partie de la liste noire des pays non autorisés.
	Cet appareil se connecte depuis un emplacement géographique à haut risque.	
DeviceCountriesNotDefined	GE002	La géolocalisation véritable fait partie de la liste blanche négative des pays non autorisés.
	L'appareil provient d'une région qui ne figure pas sur la liste blanche des régions acceptées.	
DeviceProxyGeoMismatch	GE003	La géolocalisation véritable est différente de la géolocalisation du serveur mandataire.
	L'emplacement géographique véritable de cet appareil est différente de l'emplacement géographique du serveur mandataire.	
DeviceAccountGeoMismatch	GE004	L'adresse du compte est différente de la géolocalisation véritable.
	Cet appareil a présenté une adresse de facturation de compte qui ne correspond pas à la géolocalisation de l'appareil.	
DeviceShipGeoMismatch	GE005	Incohérence entre la géolocalisation de l'appareil et l'adresse d'expédition
	L'emplacement de l'appareil et l'adresse d'expédition ne correspondent pas.	
DeviceShipGeoMismatch	GE006	Incohérence entre la géolocalisation de l'appareil et l'adresse d'expédition
	L'emplacement de l'appareil et l'adresse d'expédition ne correspondent pas.	
Appareil		
SatelliteISP	DV001	Fournisseur de service internet par satellite

## 9. Outils de protection contre la fraude en ligne

Nom de règle	Numéro de règle	Message de règle
	Explication de la règle	
	Cette transaction provient d'une source qui utilise un fournisseur de service Internet par satellite.	
MidsessionChange	DV002	Session modifiée au milieu de la session
	Cet appareil a modifié les détails et les identificateurs de la session au milieu d'une session.	
LanguageMismatch	DV003	Langue ne correspond pas
	La langue de l'utilisateur ne correspond pas à la langue principale de la région de la véritable adresse IP.	
NoDeviceID	DV004	Aucun ID d'appareil
	Aucun ID d'appareil n'était disponible pour cette transaction.	
Dial-upConnection	DV005	Connexion commutée
	Cet appareil utilise une connexion commutée moins facilement identifiable.	
DeviceNegativeReputation	DV006	Appareil sur la liste noire du réseau de réputation
	Cet appareil a une mauvaise réputation connue, signalée sur le réseau de fraude.	
DeviceGlobalBlacklist	DV007	Appareil sur la liste noire globale
	Cet appareil a été signalé sur la liste noire globale des appareils posant problème.	
DeviceCompromisedDay	DV008	Appareil compromis au cours de la dernière journée
	Cet appareil a été signalé comme étant compromis au cours des dernières 24 heures.	
DeviceCompromisedHour	DV009	Appareil compromis au cours de la dernière heure
	Cet appareil a été signalé comme étant compromis au cours de la dernière heure.	
FlashImagesCookiesDisabled	DV010	Témoins (cookies) des images Flash désactivés
	Les fonctions ou identificateurs clés du navigateur ont été désactivés sur cet appareil.	
FlashCookiesDisabled	DV011	Témoins (cookies) Flash désactivés

<b>Nom de règle</b>	<b>Numéro de règle</b>	<b>Message de règle</b>
	<b>Explication de la règle</b>	
	Les fonctions ou identificateurs clés du navigateur ont été désactivés sur cet appareil.	
FlashDisabled	DV012	Flash désactivé
	Les fonctions ou identificateurs clés du navigateur ont été désactivés sur cet appareil.	
ImagesDisabled	DV013	Images désactivées
	Les fonctions ou identificateurs clés du navigateur ont été désactivés sur cet appareil.	
CookiesDisabled	DV014	Témoins (cookies) désactivés
	Les fonctions ou identificateurs clés du navigateur ont été désactivés sur cet appareil.	
DeviceDistanceTravelled	DV015	Distance parcourue par l'appareil
	L'appareil a été utilisé à plusieurs emplacements physiques dans une courte période de temps.	
PossibleCookieWiping	DV016	Suppression des témoins (cookies)
	Cet appareil semble supprimer les témoins (cookies) après chaque session.	
PossibleCookieCopying	DV017	Copie de témoins (cookies) possible
	Cet appareil semble copier les témoins (cookies).	
PossibleVPNConnection	DV018	Utilisation possible d'une connexion VPN
	Cet appareil utilise possiblement une connexion VPN.	

### 9.3.5.4 Exemples de réponses de risque

#### Demande de session

Exemple de réponse de risque – Demande de session
<pre>&lt;?xml version="1.0"?&gt; &lt;response&gt; &lt;receipt&gt; &lt;ResponseCode&gt;001&lt;/ResponseCode&gt; &lt;Message&gt;Success&lt;/Message&gt; &lt;Result&gt; &lt;session_id&gt;abc123&lt;/session_id&gt; &lt;unknown_session&gt;yes&lt;/unknown_session&gt; &lt;event_type&gt;payment&lt;/event_type&gt; &lt;service_type&gt;session&lt;/service_type&gt; &lt;policy_score&gt;-25&lt;/policy_score&gt; &lt;transaction_id&gt;riskcheck42&lt;/transaction_id&gt; &lt;org_id&gt;11kue096&lt;/org_id&gt; &lt;request_id&gt;91C1879B-33D4-4D72-8FCB-B60A172B3CAC&lt;/request_id&gt; &lt;risk_rating&gt;medium&lt;/risk_rating&gt;</pre>

```

<request_result>success</request_result>
<summary_risk_score>-25</summary_risk_score>
<Policy>default</policy>
<review_status>review</review_status>
</Result>
<Rule>
<RuleName>ComputerGeneratedEMail</RuleName>
<RuleCode>UN001</RuleCode>
<RuleMessageEn>Unknown Rule</RuleMessageEn>
<RuleMessageFr>Regle Inconnus</RuleMessageFr>
</Rule>
<Rule>
<RuleName>NoDeviceID</RuleName>
<RuleCode>DV004</RuleCode>
<RuleMessageEn>No Device ID</RuleMessageEn>
<RuleMessageFr>null</RuleMessageFr>
</Rule>
</receipt>
</response>

```

## Demande d'attribut

### Exemple de réponse de risque – Demande d'attribut

```

<?xml version="1.0"?>
<response>
<receipt>
<ResponseCode>001</ResponseCode>
<Message = Success</Message>
<Result>
<org_id>11kue096</org_id>
<request_id>443D7FB5-CC5C-4917-A57E-27EAC824069C</request_id>
<service_type>session</service_type>
<risk_rating>medium</risk_rating>
<summary_risk_score>-25</summary_risk_score>
<request_result>success</request_result>
<policy>default</policy>
<policy_score>-25</policy_score>
<transaction_id>riskcheck19</transaction_id>
<review_status>review</review_status>
</Result>
<Rule>
<RuleName>ComputerGeneratedEMail</RuleName>
<RuleCode>UN001</RuleCode>
<RuleMessageEn>Unknown Rule</RuleMessageEn>
<RuleMessageFr>Regle Inconnus</RuleMessageFr>
</Rule>
<Rule>
<RuleName>NoDeviceID</RuleName>
<RuleCode>DV004</RuleCode>
<RuleMessageEn>No Device ID</RuleMessageEn>
<RuleMessageFr>null</RuleMessageFr>
</Rule>
</receipt>
</response>

```

### 9.3.6 Ajout de balises de profilage à votre site Web

Placez les balises de profilage sur une page HTML desservie par votre application Web de façon à ce que ThreatMetrix puisse recueillir des renseignements sur l'appareil à partir du navigateur Web du client. Les balises doivent être placées sur une page qu'un visiteur afficherait dans une fenêtre de navigateur pendant 3 à 5 secondes (comme une page qui demande à l'utilisateur d'entrer des données). Une fois le profil de l'appareil établi, une demande de session peut être utilisée pour obtenir des renseignements détaillés sur l'appareil afin d'évaluer les risques avant de soumettre une transaction de paiement.

Il existe deux balises de profilage qui nécessitent deux variables : `org_id` et `session_id`. La variable `session_id` doit correspondre à la valeur ID value qui doit être transmise dans la transaction de demande de session. Les valeurs `org_id` valides sont :

**11kue096**

Environnement de tests d'assurance qualité

**Ibhqgx47**

Environnement de production

Voici un exemple HTML des balises de profilage.

**REMARQUE :** Votre site doit remplacer la balise `<my_session_id>` dans l'exemple de code par une valeur alphanumérique unique chaque fois que vous prenez l'empreinte d'un nouveau client.

```
<p style="background:url(https://h.onlinemetrix.net/fp/clear.png?org_id=11kue096&session_id=<my_session_id>&m=1)">
</p>



<script src="https://h.onlinemetrix.net/fp/check.js?org_id=11kue096&session_id=<my_session_id>" type="text/javascript">
</script>

<object type="application/x-shockwave-flash"
        data="https://h.onlinemetrix.net/fp/fp.swf?org_id=11kue096&session_id=<my_session_id>"
        width="1" height="1" id="obj_id">
    <param name="movie" value="https://h.onlinemetrix.net/fp/fp.swf?org_id=11kue096&session_id=<my_session_id>" />
    <div></div>
</object>
```

## 9.4 Intégration de tous les outils de protection contre la fraude offerts

- 9.4.1 Options de mise en œuvre de l'outil de gestion des risques transactionnels (TRMT)
- 9.4.2 Liste de contrôle de mise en œuvre
- 9.4.3 Prise de décision

Pour minimiser les activités frauduleuses lors des transactions en ligne, Moneris vous recommande de mettre en œuvre tous les outils de lutte contre la fraude offerts par Passerelle Moneris. Ces outils sont expliqués ci-dessous :

### **Service de vérification d'adresse (SVA)**

Vérifie les informations relatives à l'adresse de facturation du titulaire de la carte

### **Vérifié par Visa, MasterCard Secure Code et Amex SafeKey (VbV, MCSC et SafeKey)**

Authentifie le titulaire de la carte lors d'une transaction en ligne

### **Numéro de vérification de carte (NVC)**

Valide que le titulaire de la carte est en possession d'une carte de crédit authentique lors de la transaction

Veuillez noter que toutes les réponses renvoyées par ces méthodes de vérification sont destinées à renforcer la sécurité et la prévention des fraudes. La réponse elle-même n'a aucune incidence sur la conclusion d'une transaction. Une fois la réponse reçue, le choix de traiter une transaction ou non est entièrement laissé au commerçant.

### **9.4.1 Options de mise en œuvre de l'outil de gestion des risques transactionnels (TRMT)**

#### **Option A**

Traitez une demande de l'outil de gestion des risques transactionnels et obtenez la réponse. Vous pouvez alors décider de poursuivre la transaction, de l'interrompre ou d'utiliser des fonctions de protection contre la fraude supplémentaires.

Si vous souhaitez utiliser des fonctions de protection contre la fraude supplémentaires, effectuez l'une ou les deux actions suivantes pour vous aider à décider si vous devez poursuivre la transaction ou l'annuler :

- Traitez une transaction Vérifié par Visa, SecureCode de Mastercard ou SafeKey et obtenez la réponse. Le commerçant prend alors la décision de poursuivre ou d'annuler la transaction.
- Traitez une transaction financière comprenant des détails de SVA ou de NVC et obtenez la réponse. Le commerçant prend alors la décision de poursuivre ou d'annuler la transaction.

#### **Option B**

1. Traitez une demande de l'outil de gestion des risques transactionnels et obtenez la réponse.
2. Traitez une transaction Vérifié par Visa, SecureCode de Mastercard ou SafeKey et obtenez la réponse.

3. Traitez une transaction financière comprenant des détails de SVA ou de NVC et obtenez la réponse.
4. Le commerçant prend ensuite une décision ponctuelle en fonction des réponses reçues des outils de protection contre la fraude.

#### **9.4.2 Liste de vérification de mise en œuvre**

Les listes de vérification suivantes présentent les tâches de haut niveau qui sont requises dans le cadre de la mise en œuvre de votre outil de gestion des risques transactionnels. Chaque organisation ayant ses propres exigences en matière d'application des changements de système et de processus, cette liste n'est qu'une ligne directrice et ne couvre pas tous les aspects de votre projet.

**Téléchargez et consultez l'ensemble des API et des guides d'intégration applicables.**

Veuillez consulter les sections du présent document qui se rapportent à la fonction suivante.

**Tableau 18 : Documentation de l'API**

Document et API	Utilisez le document si...
Guide d'intégration de l'outil de gestion des risques transactionnels (No de Section)	Vous mettez en œuvre ou mettez à jour votre intégration pour l'outil de gestion des risques transactionnels.
Modules d'extension pour les commerçants de Moneris – Vérifié par Visa, Mastercard SecureCode ou American Express SafeKey – Guide d'intégration de l'API Java	Vous mettez en œuvre ou mettez à jour la solution Vérifié par Visa, Mastercard SecureCode ou American Express SafeKey.
Transaction de base avec le SVA et le NVC (No de Section)	Vous mettez en œuvre ou mettez à jour le traitement des transactions, du SVA ou du NVC.

#### **Conception de votre flux de transactions et de vos processus opérationnels**

Lors de la conception de votre flux de transactions, songez aux scénarios que vous aimeriez voir automatisés et à ceux que vous aimeriez voir traités manuellement par vos employés.

Les sections Comprendre le flux de transactions de gestion du risque transactionnel et Gérer les réponses (page 393) peuvent vous aider à concevoir vos flux de transactions et vos processus.

Voici les éléments dont il faut tenir compte lors de la conception de vos processus :

- Les processus permettant d'aviser les personnes de votre organisation lorsqu'une maintenance est prévue pour Passerelle Moneris
- Le traitement des remboursements, des commandes annulées, etc.
- La communication avec les clients lorsque vous n'expédierez pas les marchandises en raison de fraude présumée, de marchandises en rupture de stock, etc.

#### **Compléter votre conception et vos essais**

- Le guide d'intégration de l'API de Passerelle Moneris fournit les détails techniques nécessaires à la conception et aux tests. Assurez-vous de suivre les instructions de test et les données fournies.

### Si vous êtes un intégrateur

- Assurez-vous que votre solution répond aux exigences des normes PCI-DSS ou PA-DSS, le cas échéant.
- Envoyez un courriel à [eproducts@moneris.com](mailto:eproducts@moneris.com) avec l'objet « Demande de certification ».
- Concevez du matériel pour que vos clients soient installés le plus rapidement possible avec votre solution et un compte Moneris. Veuillez inclure des renseignements tels que :
  - Les étapes qu'ils doivent suivre pour entrer leur ID de magasin ou les renseignements liés au jeton API dans votre solution.
  - Tous les services facultatifs que vous prenez en charge par l'entremise de Passerelle Moneris (tels que l'outil de gestion des risques transactionnels, le SVA, le NVC, Vérifié par Visa, SecureCode de Mastercard, SafeKey, etc.) afin que les clients puissent demander ces fonctions.

#### 9.4.3 Prise de décision

En fonction de vos politiques et processus commerciaux, les renseignements obtenus grâce aux outils de protection contre la fraude (tels que SVA , NVC, Vérifié par Visa, SecureCode de Mastercard, SafeKey et l'outil de gestion des risques transactionnels) peuvent vous aider à prendre une décision éclairée quant à l'acceptation ou le refus d'une transaction en raison de son caractère potentiellement frauduleux.

Si vous ne voulez pas poursuivre une transaction potentiellement frauduleuse, vous devez informer le client que vous ne poursuivrez pas sa transaction.

Si vous tentez de procéder à une authentification supplémentaire en utilisant les outils de protection contre la fraude à votre disposition, mais que vous avez reçu une réponse d'approbation, annulez la transaction financière en procédant de l'une des manières suivantes :

- Si la transaction originale est un achat, utilisez une transaction de correction ou de remboursement d'achat. Vous aurez besoin des numéros de commande et de transaction originaux.
- Si la transaction originale est une préautorisation, effectuez une transaction de conclusion de 0,00 \$.

## 10 Intégration d'Apple Pay et de Google Pay<sup>MC</sup>

- 10.1 À propos de l'intégration d'Apple Pay et de Google Pay<sup>MC</sup>
- 10.2 Sommaire du processus de transaction Apple Pay
- 10.3 Sommaire du processus de transaction Google Pay<sup>MC</sup>
- 10.4 À propos de l'intégration de l'API d'Apple Pay et de Google Pay<sup>MC</sup>
- 10.5 Achat utilisant le code de vérification d'authentification du titulaire de carte – Apple Pay et Google Pay<sup>MC</sup>
- 10.6 Préautorisation utilisant le code de vérification d'authentification du titulaire de carte – Apple Pay et Google Pay<sup>MC</sup>

### 10.1 À propos de l'intégration d'Apple Pay et de Google Pay<sup>MC</sup>

Passerelle Moneris permet aux commerçants de traiter des transactions dans des applications fonctionnant sur des appareils mobiles iOS (Apple Pay) ou Android (Google Pay<sup>MC</sup>), et dans un navigateur lorsqu'ils utilisent le navigateur Web Safari (Apple Pay, en utilisant des appareils Apple uniquement) ou le navigateur Web Chrome (Google Pay<sup>MC</sup>).

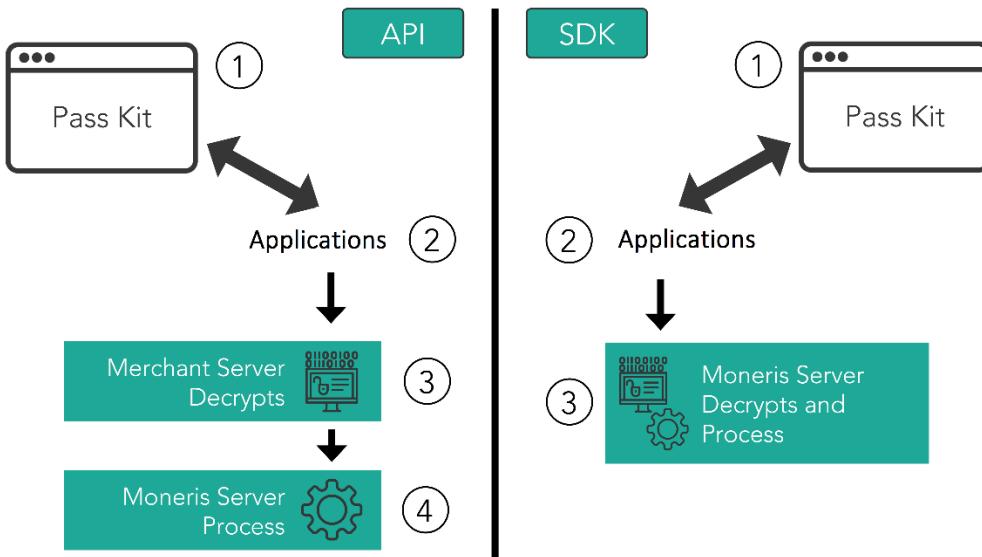
Solutions Moneris offre deux méthodes d'intégration pour le traitement des transactions Apple Pay et Google Pay<sup>MC</sup>. Les commerçants peuvent choisir d'utiliser l'une des deux méthodes suivantes :

- la méthode utilisant la trousse de développement logiciel (SDK), ou
- la méthode API (où le déchiffrage des données utiles de la transaction est géré par les commerçants). Bien que les deux méthodes offrent les mêmes fonctionnalités de paiement de base, leurs mises en œuvre sont différentes.

Ce guide ne traite que de la méthode API; pour plus de renseignements sur la méthode d'intégration utilisant la trousse SDK, consultez le portail des développeurs de Moneris à l'adresse <https://developer.moneris.com>.

### 10.2 Sommaire du processus de transaction Apple Pay

Pour les méthodes d'intégration mobile API et SDK au sein de l'application, l'application iOS du commerçant utilise le cadre PassKit d'Apple pour demander et recevoir les données de paiement chiffrées d'Apple. Lorsque les détails du paiement sont renvoyés sous leur forme chiffrée, ils peuvent être déchiffrés et traités par Passerelle Moneris de l'une des deux méthodes suivantes : la méthode SDK ou API.



### Étapes du processus de paiement Apple Pay

#### API

1. L'application mobile ou la page Web du commerçant demande et reçoit les données chiffrées.
2. Les données chiffrées sont envoyées au serveur du commerçant, où elles sont déchiffrées.
3. Passerelle Moneris reçoit les données déchiffrées du serveur du commerçant et traite la transaction Cvv Purchase – Apple Pay and Google Pay<sup>MC</sup> ou Cvv Pre-Authorization – Apple Pay & Google Pay<sup>MC</sup> à la page 420transaction
  - a. Veuillez vous assurer que l'indicateur de portefeuille est correctement rempli avec la bonne valeur (APP pour Apple Pay In-App ou APW pour Apple Pay on the Web).

#### Trousse SDK

1. L'application mobile ou la page Web du commerçant demande et reçoit les données chiffrées.
2. Les données chiffrées sont envoyées du serveur du commerçant au serveur de Passerelle Moneris où elles sont déchiffrées et traitées.

Ce guide ne traite que de la méthode API; pour plus de renseignements sur la méthode d'intégration utilisant la trousse SDK, consultez le portail des développeurs de Moneris à l'adresse <https://developer.moneris.com>.

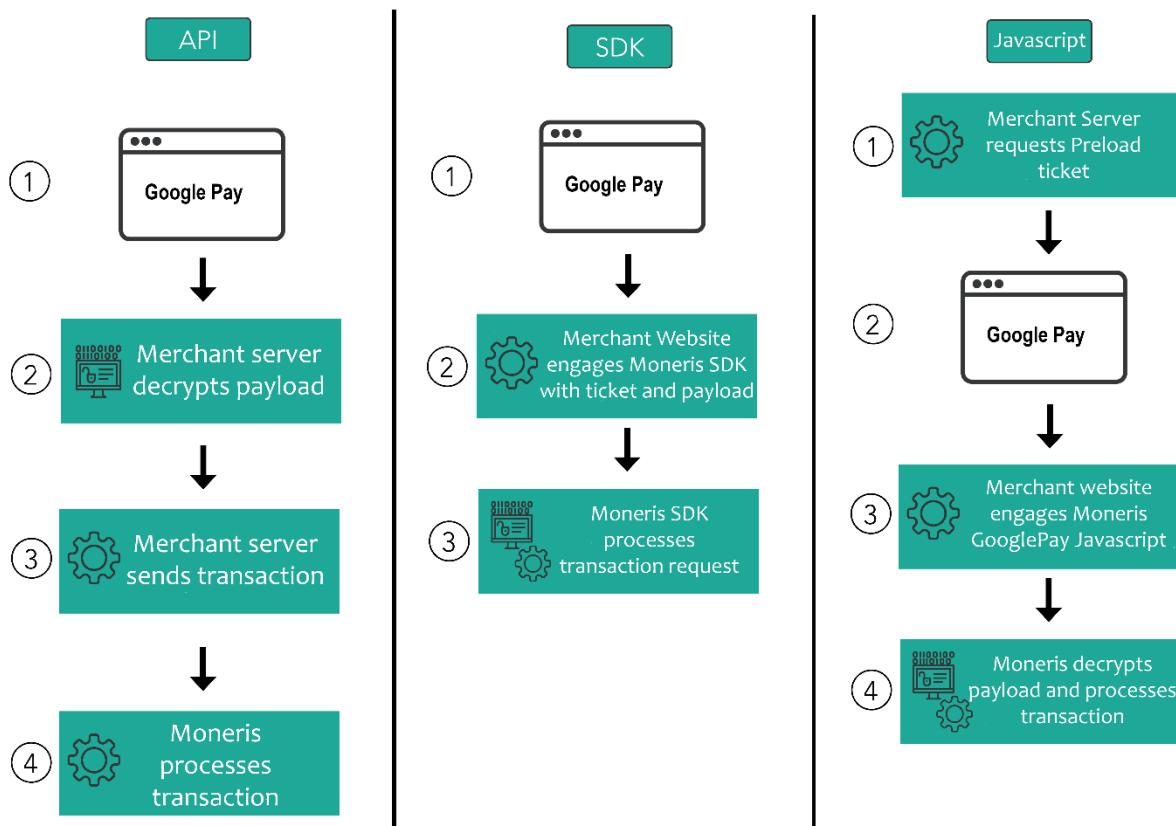
### 10.3 Sommaire du processus de transaction Google PayMC

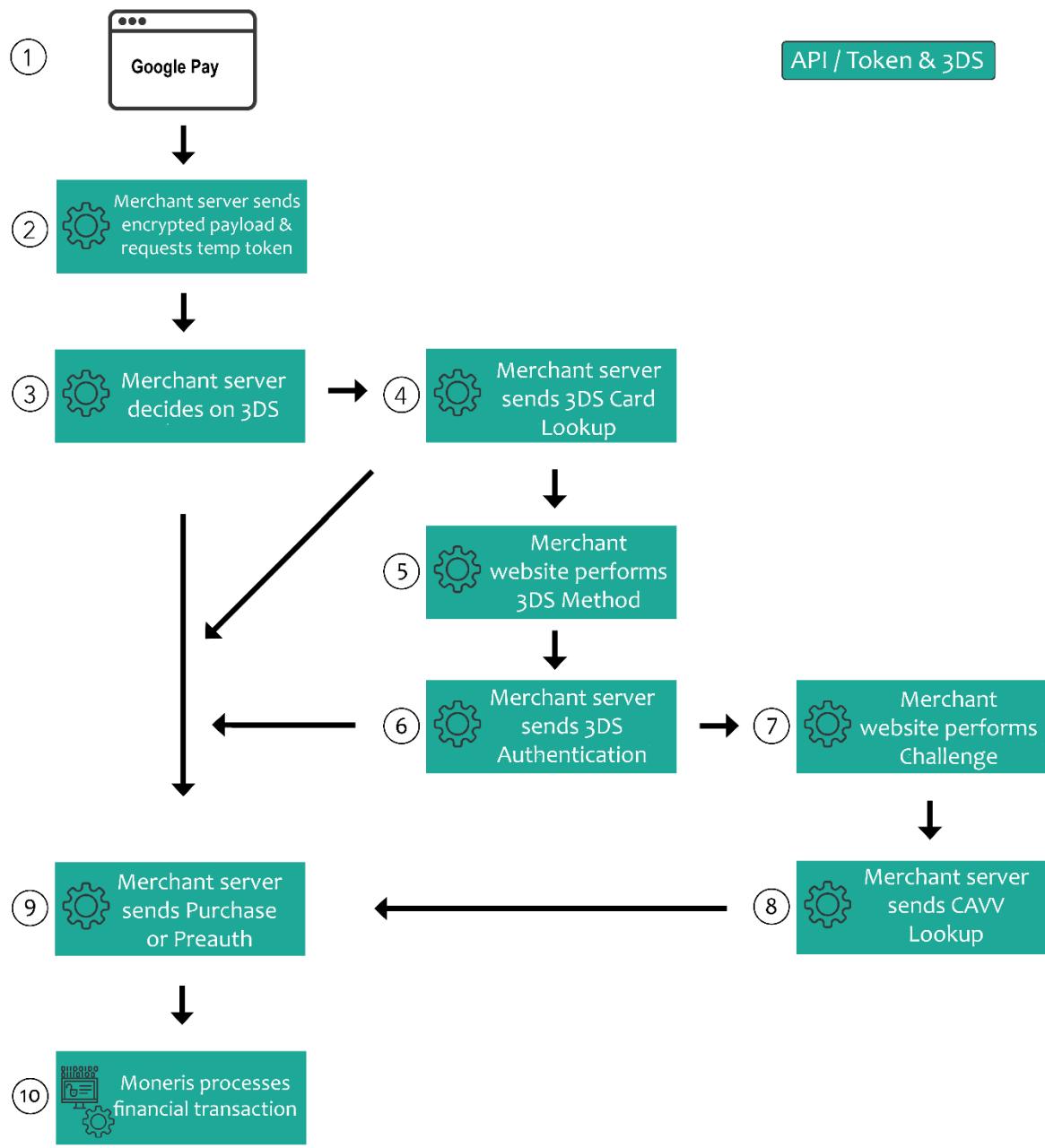
Moneris propose quatre méthodes d'intégration pour traiter les transactions avec les portefeuilles Google Pay. Toutes les intégrations utilisent le cadre de Google PayMD pour demander et recevoir des données de paiement chiffrées de la part de Google. Lorsque les données de paiement sont renvoyées sous forme chiffrée, le commerçant peut déchiffrer les données utiles locales sur son serveur ou les transmettre à Moneris pour qu'elles soient déchiffrées.

Moneris recommande aux commerçants d'utiliser le processus 3DS sur les jetons temporaires de Google Pay pour le type de carte sous-jacent de FPAN afin de réduire le risque de fraude et de rétrofacturation. Les commerçants peuvent tenter ou ignorer le processus 3DS si le type de carte sous-jacent est DPAN; il se peut toutefois que l'émetteur de la carte ne prenne pas en charge la solution 3DS

pour un PAN d'appareil; Moneris recommande donc d'utiliser la recherche de carte 3DS pour s'assurer que la solution 3DS est prise en charge.

## Méthodes et processus d'intégration de Google PayMC





## **Données utiles locales déchiffrées (serveur du commerçant vers l'API de Passerelle Moneris)**

Le commerçant déchiffre les données utiles locales de Google Pay, puis traite une transaction financière standard par l'entremise de l'appel API de Passerelle Moneris avec les données de la carte. Ceci est utilisé par les solutions dans une application et en ligne.

**REMARQUE:** Dans la méthode API où le serveur du commerçant est responsable du déchiffrement des données, le commerçant doit signer une entente avec Google directement. Google peut alors lui fournir les clés pour déchiffrer les données.

1. L'application mobile ou la page Web du commerçant demande et reçoit les données chiffrées de Google.
2. Les données chiffrées sont envoyées du serveur du commerçant au serveur de Passerelle Moneris par l'intermédiaire du SDK où elles sont déchiffrées et traitées.

## **Données utiles locales chiffrées (site Web du commerçant vers le SDK de Moneris)**

Le commerçant transmet les données utiles locales chiffrées au SDK Google Pay de Moneris. Ceci est utilisé uniquement pour les solutions dans une application. Les fichiers du SDK se trouvent sur le [Github de Moneris](#) et les instructions d'intégration sur le [portail pour développeurs de Moneris](#).

1. L'application mobile ou la page Web du commerçant demande et reçoit les données chiffrées de Google.
2. Les données chiffrées sont envoyées du serveur du commerçant au serveur de Passerelle Moneris par l'intermédiaire du SDK où elles sont déchiffrées et traitées.

## **Données utiles locales chiffrées (site Web du commerçant vers JavaScript de Moneris)**

Le commerçant modifie le Javascript de Google Pay pour utiliser Google Pay de Moneris pour le traitement du paiement. Un préchargement facultatif est inclus dans l'API de Passerelle Moneris spécifique à cette méthode d'intégration. Consultez le guide complet qui se trouve dans le [Github de Moneris](#) ou les instructions sur le [portail pour développeurs de Moneris](#).

1. Le serveur du commerçant remplit une demande de préchargement et reçoit un billet (étape facultative).
2. L'application mobile ou la page Web du commerçant demande et reçoit les données chiffrées de Google.
3. Les données chiffrées sont envoyées de l'application ou du site Web du commerçant à la passerelle Moneris par l'intermédiaire du JavaScript de Moneris intégré où elles sont déchiffrées et traitées.

## **Données utiles locales chiffrées avec 3DS (serveur du commerçant vers l'API de Passerelle Moneris)**

Le commerçant transmet les données utiles locales chiffrées via l'API de Passerelle Moneris afin de déchiffrer les données de la carte et de les transformer temporairement en jeton. Ce jeton temporaire est utilisé pour l'authentification 3D-Secure et la transaction financière qui s'ensuit.

**REMARQUE:** Moneris recommande aux commerçants d'utiliser le processus 3DS pour les jetons temporaires Google Pay avec un type de carte sous-jacent FPAN afin de réduire le risque de fraude et de rétrofacturation. Les commerçants peuvent tenter ou ignorer le processus 3DS si le type de carte sous-jacent est DPAN; il se peut toutefois que l'émetteur de la carte ne prenne pas en charge la solution 3DS pour un PAN d'appareil; Moneris recommande donc d'utiliser la recherche de carte 3DS pour s'assurer que la solution 3DS est prise en charge.

1. L'application ou la page Web du commerçant demande et reçoit les données chiffrées de Google.
2. Les données chiffrées sont envoyées du serveur du commerçant à Passerelle Moneris par l'intermédiaire d'un jeton temporaire Google Pay. Moneris renvoie un jeton de paiement temporaire dans la réponse et une méthode GooglePayPaymentMethod indiquant le type de données de carte sous-jacentes (FPAN ou DPAN).
3. Le serveur du commerçant choisit d'effectuer ou non l'authentification 3DS. S'il choisit d'ignorer l'authentification 3DS, le serveur du commerçant peut passer à l'étape 9 et effectuer immédiatement une transaction financière.
4. Le serveur du commerçant envoie le jeton temporaire dans une demande de recherche de carte 3DS. Moneris répond en indiquant si la carte sous-jacente prend en charge l'authentification 3DS ou non et en donnant les détails de la méthode 3DS, le cas échéant.  
Pour les cartes qui ne prennent pas en charge le 3DS, le serveur du commerçant doit ignorer le reste des étapes du processus d'authentification 3DS et passer à l'étape 9.
5. Si elle est disponible, le serveur du commerçant et l'application exécutent la méthode 3DS en utilisant les données 3DSMethodData et 3DSMethodURL (voir la section « Manipuler la méthode 3DS pour l'empreinte digitale des appareils » à la page 239).
6. Le serveur du commerçant envoie une demande d'authentification 3DS (canal de navigation) à Passerelle Moneris (voir la section « Implémentation de la demande d'authentification MPI 3DS » à la page **Error! Bookmark not defined.**).

Si Moneris répond avec un résultat positif (sans friction), le serveur du commerçant reçoit les valeurs CAVV et ECI de la réponse d'authentification 3DS elle-même. Vous n'avez pas besoin de suivre le processus de contestation et vous pouvez passer à l'étape 9.

Si Moneris répond qu'une invite de contestation (friction) est nécessaire, passez à l'étape suivante.

7. Le serveur du commerçant et l'application procèdent à la contestation 3DS (voir la section «Traitement du flux de contestation » à la page **Error! Bookmark not defined.**).
8. Le serveur du commerçant envoie une demande de recherche de CAVV pour récupérer la valeur d'authentification (CAVV) et l'indicateur de commerce électronique (crypt\_type) une fois la contestation terminée.
9. Le serveur du commerçant effectue une transaction financière de type GooglePayTokenPreauth ou GooglePayTokenPurchase. Le CAVV et l'indicateur de commerce électronique (crypt\_type) sont inclus comme suit :

Si le 3DS a été ignoré précédemment, omettez le champ CAVV et utilisez l'ICE approprié pour votre type de transaction.

Si l'authentification 3DS a été effectuée avec succès, fournissez le CAVV à partir de votre réponse d'authentification 3DS ou de recherche de CAVV.

## 10.4 À propos de l'intégration de l'API d'Apple Pay et de Google Pay<sup>MC</sup>

Une intégration par API sert à établir un lien de communication entre votre serveur de commerçant et le serveur de Moneris. Les API sont nécessaires pour traiter toute transaction, et c'est pourquoi les API de Apple Pay et de Google Pay<sup>MC</sup> sont également incluses dans l'intégration SDK.

Si le commerçant choisit d'utiliser la méthode d'intégration par API uniquement, il doit déchiffrer lui-même les données avant de les envoyer à Passerelle Moneris pour qu'elles soient traitées. Comme ce processus est compliqué, Moneris recommande que seules les entreprises ayant une expertise et un système de traitement des paiements déjà intégré utilisent la méthode d'intégration par API; tous les autres commerçants devraient utiliser la méthode d'intégration SDK de Moneris pour Apple Pay ou Google Pay<sup>MC</sup>.

### 10.4.1 Types de transaction utilisées pour Apple Pay et Google Pay<sup>MC</sup>

Dans l'API de Passerelle Moneris, il existe deux types de transaction qui vous permettent de traiter les données de transaction déchiffrées provenant d'Apple Pay et de Google Pay<sup>MC</sup> :

- 11.5 Achat utilisant le code de vérification d'authentification du titulaire de carte – Apple Pay et Google Pay<sup>MC</sup>
- 11.6 Préautorisation utilisant le code de vérification d'authentification du titulaire de carte – Apple Pay et Google Pay<sup>MC</sup>

Commerce électronique par carte INTERAC<sup>MD</sup> **REMARQUE :** Cette fonction peut uniquement être utilisée pour les transactions d'achat utilisant le code de vérification d'authentification.

Remarque : Si un jeton de paiement déchiffré contient un numéro de carte d'appareil (DPAN), il n'est pas recommandé d'effectuer une tentative d'authentification 3DS, car il se peut qu'elle ne soit pas prise en charge. Un DPAN est un jeton spécifique à l'appareil fourni par le fournisseur du portefeuille pour identifier la carte sous-jacente associée au numéro de carte de financement (FPAN) du titulaire de la

carte.

Après avoir traité la transaction initiale en effectuant un achat ou une préautorisation utilisant le code de vérification d'authentification du titulaire de carte, vous pouvez ensuite, au besoin, traiter l'une des transactions suivantes :

- Remboursement
- Conclusion de préautorisation
- Correction d'achat

## 10.5 Achat utilisant le code de vérification d'authentification du titulaire de carte – Apple Pay et Google Pay<sup>MC</sup>

Une transaction d'achat utilisant le code de vérification d'authentification du titulaire de carte pour Apple Pay et Google Pay<sup>MC</sup> suit un modèle 3-D Secure, mais ne requiert pas de module d'extension pour les commerçants. Une fois les données de la transactions été déchiffrées, cette transaction vérifie que les fonds requis sont présents sur la carte du client, retire ces fonds de la carte et les prépare à être déposés dans le compte du commerçant.

Pour effectuer l'authentification 3-D Secure, le module d'extension de Moneris pour les commerçants ou tout autre module d'extension tiers peut être utilisé.

En plus des transactions 3-D Secure, cette transaction peut également être utilisée pour traiter les transactions Apple Pay et Google Pay<sup>MC</sup>. Cette transaction s'applique uniquement si vous choisissez de l'intégrer directement à Apple Wallet ou à Google Wallet (si vous n'utilisez pas les trousseuses SDK Apple Pay ou Google Pay<sup>MC</sup> de Moneris).

Référez-vous aux portails pour développeurs d'Apple ou de Google pour en savoir plus sur l'intégration directe à leurs portefeuilles et recueillir les données utiles.

**AVERTISSEMENT :** Moneris déconseille fortement l'utilisation de cadres pour l'intégration de la solution 3-D Secure et ne peut garantir leur fiabilité lors du traitement des transactions dans l'environnement de production.

### Définition de l'objet de transaction Cavv Purchase for Apple Pay & Google Pay™

```
CavvPurchase cavvPurchase = new CavvPurchase();
```

### Objet HttpsPostRequest pour les transactions d'achat utilisant le code de vérification d'authentification du titulaire de carte pour Apple Pay et Google Pay<sup>MC</sup>

```
HttpsPostRequest mpgReq = new HttpsPostRequest();  
  
mpgReq.setTransaction(cavvPurchase);
```

**Champs de demande liés aux transactions d'achat utilisant le code de vérification d'authentification du titulaire de carte pour Apple Pay et Google PayMC (obligatoires)**

Variable	Type et limites	Méthode Set
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques a-Z A-Z 0-9 _ - : . @ espaces	cavvPurchase.setOrderId(order_id);
Montant	<i>Chaîne</i> 10 caractères décimaux Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal <b>EXEMPLE : 1 234 567,89</b>	cavvPurchase.setAmount(amount);
Numéro de carte de crédit	<i>Chaîne</i> Maximum 20 caractères alphanumériques	cavvPurchase.setPan(pan);
Date d'expiration	<i>Chaîne</i> 4 caractères alphanumériques AAMM	cavvPurchase.setExpDate(expiry_date);
Code de vérification d'authentification du titulaire de carte	<i>Chaîne</i> 50 caractères alphanumériques	cavvPurchase.setCavv(cavv);
<b>REMARQUE :</b> Pour les transactions d'achat et de préautorisation Apple Pay et Google Pay <sup>MC</sup> utilisant le code de vérification		

Variable	Type et limites	Méthode Set
d'authentification du titulaire de carte, le champ CAVV contient le cryptogramme déchiffré. Pour plus de renseignements, consultez l'annexe A Définition des champs de demande.		
Indicateur de commerce électronique  <b>REMARQUE :</b> Pour les transactions d'achat et de préautorisation Apple Pay et Google Pay <sup>MC</sup> utilisant le code de vérification d'authentification du titulaire de carte, l'indicateur de commerce électronique est un champ obligatoire contenant la valeur reçue des données déchiffrées ou une valeur par défaut de 5. Si vous obtenez une valeur à deux caractères (par exemple, 05 ou 07) à partir des données, supprimez le 0 initial et envoyez-nous simplement le deuxième caractère. Pour plus de renseignements, consultez l'annexe A Définition des champs de demande.	<i>Chaîne</i> 1 caractère alphanumérique  <code>cavvPurchase.setCryptType(crypt);</code>	

Les champs suivants sont requis pour Apple Pay et Google Pay uniquement :

Variable	Type et limites	
Réseau	<i>Chaîne</i> Caractère alphabétique  <code>cavv_purchase.setNetwork(network);</code>	
Type de données	<i>Chaîne</i> 3 caractères alphanumériques  <code>cavv_purchase.setDataType(data_type);</code>	

**Champs de demande liés aux transactions d'achat utilisant le code de vérification d'authentification du titulaire de carte pour Apple Pay et Google PayMC (facultatifs)**

Variable	Type et limites	Méthode Set
Vérification d'état	<i>Valeur booléenne</i> true/false	mpgReq.setStatusCheck(status_check);
ID de client	<i>Chaîne</i> 50 caractères alphanumériques  <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : <>\$ % = ? ^{ } [ ] \	cavvPurchase.setCustId(cust_id);
Descripteur dynamique	<i>Chaîne</i> 20 caractères alphanumériques  Total de 22 caractères incluant votre nom de commerçant et un séparateur  <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : <>\$ % = ? ^{ } [ ] \	cavvPurchase.setDynamicDescriptor(dynamic_descriptor);
ID de correspondance de carte	<i>Chaîne</i> 50 caractères alphanumériques  <b>REMARQUE :</b> Applicables à Offlinx <sup>MC</sup> seulement, chaque transaction doit avoir une valeur unique	cavvPurchase.setCmId(transaction_id);
Renseignements du client	<i>Objet</i> S. O.	cavvPurchase.setCustInfo(customer);

**Exemple d'achat utilisant le code de vérification d'authentification du titulaire de carte pour Apple Pay et Google PayMC**

```
package Canada;
import JavaAPI.*;
public class TestCanadaCavvPurchase
{
    public static void main(String[] args)
    {
        String store_id = "store5";
        String api_token = "yesguy";
```

```

java.util.Date createDate = new java.util.Date();
String order_id = "Test"+createDate.getTime();
String cust_id = "CUS887H67";
String amount = "10.42";
String pan = "4740611374762707";
String expdate = "1901"; //YYMM
String cavv = "BwABApFSYyd4l2eQQFjAAAAAAA=";
String dynamic_descriptor = "123456";
String processing_country_code = "CA";
String crypt_type = "5";
boolean status_check = false;
boolean foreign_indicator= true; //New Foreign Indicator field

CavvPurchase cavvPurchase = new CavvPurchase();
cavvPurchase.setOrderId(order_id);
cavvPurchase.setCustId(cust_id);
cavvPurchase.setAmount(amount);
cavvPurchase.setPan(pan);
cavvPurchase.setExpdate(expdate);
cavvPurchase.setCavv(cavv);
cavvPurchase.setCryptType(crypt_type); //Mandatory for AMEX only
cavvPurchase.setDynamicDescriptor(dynamic_descriptor);
//cavvPurchase.setWalletIndicator("APP"); //set only for wallet transactions. e.g APPLE PAY
//cavvPurchase.setNetwork("Interac"); //set only for Interac e-commerce
//cavvPurchase.setData-Type("3DSecure"); //set only for Interac e-commerce
//cavvPurchase.setCmId("8nAK8712sGaAkls56"); //set only for usage with Offlinx - Unique max
50 alphanumeric characters transaction id generated by merchant

cavvPurchase.setThreeDSVersion("2.2.0"); //Mandatory for financial transactions using 3rd
Party 3-D Secure services.
cavvPurchase.setThreeDSServerTransId("e11d4985-8d25-40ed-99d6-c3803fe5e68f"); //Mandatory
for financial transactions using 3rd Party 3-D Secure services - obtained from
MpICavyLookup or MpIThreeDSAAuthentication
//cavvPurchase.setDsTransId("12345");//Optional - to be used only if you are using 3rd
party 3ds service
cavvPurchase.setForeignIndicator(foreign_indicator);

// TrId and TokenCryptogram are optional, refer documentation for more details.
cavvPurchase.setTrId("50189815682");
cavvPurchase.setTokenCryptogram("APmbM/411e0uAAH+s6xMAAADFA==");

//optional - Credential on File details
CofInfo cof = new CofInfo();
cof.setPaymentIndicator("U");
cof.setPaymentInformation("2");
cof.setIssuerId("139X3130ASCXAS9");

cavvPurchase.setCofInfo(cof);
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false for production transactions
mpgReq.setstoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(cavvPurchase);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = " + receipt.getBankTotals());
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("CavvResultCode = " + receipt.getCavvResultCode());
System.out.println("IssuerId = " + receipt.getIssuerId());
System.out.println("ThreeDSVersion = " + receipt.getThreeDSVersion());
System.out.println("SourcePanLast4 = " + receipt.getSourcePanLast4());
System.out.println("Advice Code= " + receipt.getAdviceCode());
}

```

```
        catch (Exception e)
        {
            e.printStackTrace ();
        }
    }
}
```

## 10.6 Préautorisation utilisant le code de vérification d'authentification du titulaire de carte – Apple Pay et Google Pay<sup>MC</sup>

Une transaction de préautorisation utilisant le code de vérification d'authentification du titulaire de carte pour Apple Pay et Google Pay<sup>MC</sup> suit un modèle 3-D Secure, mais ne requiert pas de module d'extension pour les commerçants. Une fois les données de la transaction déchiffrées, cette transaction vérifie que les fonds requis sont présents sur la carte du client et les bloque. Pour préparer les fonds à être déposés sur le compte du commerçant, veuillez effectuer une transaction de conclusion de préautorisation.

Pour effectuer l'authentification 3-D Secure, le module d'extension de Moneris pour les commerçants ou tout autre module d'extension tiers peut être utilisé.

En plus des transactions 3-D Secure, cette transaction peut également être utilisée pour traiter les transactions Apple Pay et Google Pay<sup>MC</sup>. Cette transaction s'applique uniquement si vous choisissez de l'intégrer directement à Apple Wallet ou à Google Wallet (si vous n'utilisez pas les trousseaux SDK Apple Pay ou Google Pay<sup>MC</sup> de Moneris).

Référez-vous aux portails pour développeurs d'Apple ou de Google pour en savoir plus sur l'intégration directe à leurs portefeuilles et recueillir les données utiles.

Commerce électronique par carte INTERAC<sup>MD</sup> **REMARQUE :** Cette fonction peut uniquement être utilisée pour les transactions d'achat utilisant le code de vérification d'authentification.

**AVERTISSEMENT :** Moneris déconseille fortement l'utilisation de cadres pour l'intégration de la solution 3-D Secure et ne peut garantir leur fiabilité lors du traitement des transactions dans l'environnement de production.

### Définition de l'objet de transaction Cavv Pre-Authorization for Apple Pay & Google Pay<sup>TM</sup>

```
CavvPreAuth cavvPreauth = new CavvPreAuth();
```

### Objet HttpsPostRequest pour les transactions de préautorisation utilisant le code de vérification d'authentification du titulaire de carte pour Apple Pay et Google Pay<sup>MC</sup>

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.SetTransaction(cavvPreauth);
```

**Champs de demande liés aux transactions de préautorisation utilisant le code de vérification d'authentification du titulaire de carte pour Apple Pay et Google PayMC (obligatoires)**

Variable	Type et limites	Méthode Set
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques a-Z A-Z 0-9 _ - : . @ espaces	cavvPreauth.setOrderId(order_id);
Montant	<i>Chaîne</i> 10 caractères décimaux Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal <b>EXEMPLE : 1 234 567,89</b>	cavvPreauth.setAmount(amount);
Numéro de carte de crédit	<i>Chaîne</i> Maximum 20 caractères alphanumériques	cavvPreauth.setPan(pan);
Code de vérification d'authentification du titulaire de carte	<i>Chaîne</i> 50 caractères alphanumériques	cavvPreauth.setCavv(cavv);
<b>REMARQUE :</b> Pour les transactions d'achat et de préautorisation Apple Pay et Google Pay <sup>MC</sup> utilisant le code de vérification d'authentification du titulaire de carte, le champ CAVV contient le cryptogramme déchiffré. Pour plus de renseignements, consultez l'annexe A Définition des champs de demande.		
Date d'expiration	<i>Chaîne</i> 4 caractères alphanumériques AAMM	cavvPreauth.setExpDate(expiry_date);

Variable	Type et limites	Méthode Set
Indicateur de commerce électronique  <b>REMARQUE :</b> Pour les transactions d'achat et de préautorisation Apple Pay et Google Pay <sup>MC</sup> utilisant le code de vérification d'authentification du titulaire de carte, l'indicateur de commerce électronique est un champ obligatoire contenant la valeur reçue des données déchiffrées ou une valeur par défaut de 5. Si vous obtenez une valeur à deux caractères (par exemple, 05 ou 07) à partir des données, supprimez le 0 initial et envoyez-nous simplement le deuxième caractère. Pour plus de renseignements, consultez l'annexe A Définition des champs de demande.	<i>Chaîne</i> 1 caractère alphanumérique	cavvPreauth.setCryptType(crypt);

**Champs de demande liés aux transactions de préautorisation utilisant le code de vérification d'authentification du titulaire de carte pour Apple Pay et Google Pay<sup>MC</sup> ( facultatifs )**

Variable	Type et limites	Méthode Set
Vérification d'état	<i>Valeur booléenne</i> true/false	mpgReq.setStatusCheck(status_check);
ID de client	<i>Chaîne</i> 50 caractères alphanumériques  <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\	cavvPreauth.setCustId(cust_id);
Descripteur dynamique  <b>Pour les transactions de préautorisation</b> <b>REMARQUE :</b> La valeur du champ Dynamic descriptor est uniquement	<i>Chaîne</i> 20 caractères alphanumériques Total de 22 caractères incluant votre nom de	cavvPreauth.setDynamicDescriptor(dynamic_descriptor);

Variable	Type et limites	Méthode Set
transférée à une conclusion de préautorisation lorsque cette dernière est effectuée par l'entremise du centre de ressources pour commerçants. Autrement, la valeur du champ doit être renvoyée durant la conclusion de préautorisation.	commerçant et un séparateur  <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : <> \$ % = ? ^ { } [ ] \	
ID de correspondance de carte  <b>REMARQUE :</b> Applicables à Offlinx <sup>MC</sup> seulement, chaque transaction doit avoir une valeur unique	<i>Chaîne</i> 50 caractères alphanumériques	cavvPreauth.setCmId(transaction_id);
Réseau  <b>REMARQUE :</b> Cette variable de demande est obligatoire pour les transactions de commerce électronique INTERAC <sup>MD</sup> effectuées via Apple Pay ou Google Pay <sup>MC</sup> uniquement, et ne doit pas être utilisée pour les transactions par carte de crédit.	<i>Chaîne</i> Caractère alphabétique	cavvPurchase.setNetwork(network);
Type de données  <b>REMARQUE :</b> Cette variable de demande est obligatoire pour les transactions de commerce électronique INTERAC <sup>MD</sup> effectuées via Apple Pay ou Google Pay <sup>MC</sup> uniquement, et ne doit pas être utilisée pour les transactions par carte de crédit.	<i>Chaîne</i> 3 caractères alphanumériques	cavvPurchase.setDataTyppe(data_type);

#### Exemple de préautorisation utilisant le code de vérification d'authentification du titulaire de carte pour Apple Pay et Google PayMC

```
package Canada;
import JavaAPI.*;
public class TestCanadaCavvPreauth
{
    public static void main(String[] args)
    {
        String store_id = "store5";
```

```

String api_token = "yesguy";
java.util.Date createDate = new java.util.Date();
String order_id = "Test"+createDate.getTime();
String cust_id = "CUS887H67";
String amount = "10.42";
String pan = "4242424242424242";
String expdate = "1911"; //YYMM format
String cavv = "AAABBJg0VhI0VniQEjRWAAAAAAA=";
String dynamic_descriptor = "123456";
String processing_country_code = "CA";
String crypt_type = "5";
String ds_trans_id = "12345";
boolean status_check = false;
boolean foreign_indicator= true; //New Foreign Indicator field

CavvPreAuth cavvPrauth = new CavvPreAuth();
cavvPrauth.setOrderId(order_id);
cavvPrauth.setCustId(cust_id);
cavvPrauth.setAmount(amount);
cavvPrauth.setPan(pan);
cavvPrauth.setExpdate(expdate);
cavvPrauth.setCavv(cavv);
cavvPrauth.setCryptType(crypt_type); //Mandatory for AMEX only
cavvPrauth.setDynamicDescriptor(dynamic_descriptor);
//cavvPrauth.setWalletIndicator("APP"); //set only for wallet transactions. e.g APPLE PAY
//cavvPrauth.setCmId("8nAK8712sGaAkls56"); //set only for usage with Offlinx - Unique max
50 alphanumeric characters transaction id generated by merchant
cavvPrauth.setThreeDSVersion("2.2.0"); //Mandatory for financial transactions using 3rd
Party 3-D Secure services
cavvPrauth.setThreeDSServerTransId("e11d4985-8d25-40ed-99d6-c3803fe5e68f"); //Mandatory
for financial transactions using 3rd Party 3-D Secure services" - obtained from
MpICavvLookup or MpithreeDSAAuthentication
//cavvPrauth.setDsTransId("12345");//Optional - to be used only if you are using 3rd party
3ds service

// TrId and TokenCryptogram are optional, refer documentation for more details.
cavvPrauth.setTrId("50189815682");
cavvPrauth.setTokenCryptogram("APmbM/411e0uAAH+s6xMAAADFA==");

//optional - Credential on File details
CofInfo cof = new CofInfo();
cof.setPaymentIndicator("U");
cof.setPaymentInformation("2");
cof.setIssuerId("139X3130ASCXAS9");

cavvPrauth.setCofInfo(cof);
cavvPrauth.setForeignIndicator(foreign_indicator);

HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false for production transactions
mpgReq.setstoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(cavvPrauth);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = " + receipt.getBankTotals());
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("CavvResultCode = " + receipt.getCavvResultCode());
System.out.println("IssuerId = " + receipt.getIssuerId());
System.out.println("ThreeDSVersion = " + receipt.getThreeDSVersion());
System.out.println("SourcePanLast4 = " + receipt.getSourcePanLast4());
System.out.println("Advice Code= " + receipt.getAdviceCode());
}

```

```

        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}

```

## 10.7 Ajout d'un jeton temporaire de Google Pay - GooglePayTokenTempAdd

Ceci crée un nouveau profil de carte de crédit pour un jeton temporaire à partir des données utiles locales chiffrées Google Pay. Pendant la durée de vie de ce jeton temporaire, il peut être utilisé pour effectuer l'authentification 3DS et des transactions financières par l'entremise de la préautorisation du jeton Google Pay ou de l'achat du jeton Google Pay.

Le champ de réponse `GooglePaymentMethod` renvoyé par cette demande vous indiquera si la carte sous-jacente dans Google Pay est le numéro de carte de financement (« FPAN ») ou un numéro de carte d'appareil (« DPAN »). Si un `GoogleTokenTempAdd` renvoie un FPAN, vous pouvez effectuer une authentification 3DS avec celui-ci; s'il renvoie un DPAN, l'authentification 3DS n'est pas nécessaire.

Accédez aux portails des développeurs d'Apple ou de Google pour obtenir des détails sur l'intégration directe à leurs portefeuilles et recueillir les données utiles locales.

### Éléments dont il faut tenir compte :

Le jeton temporaire peut avoir une durée de vie maximale de 15 minutes.

### Définition de l'objet de transaction GooglePay Temporary Token Add

```
GooglePayTokenTempAdd googlePayTokenTempAdd = new GooglePayTokenTempAdd();
```

### Objet `HttpsPostRequest` pour les transactions d'ajout d'un jeton temporaire de GooglePay

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setTransaction(GooglePayTokenTempAdd);
```

### Principaux champs d'objet de connexion (toutes les transactions API)

Variable	Type et limites	Description
Code de magasin	<i>Chaîne</i> S.O.	<code>mpgReq.setstoreId(store_id);</code> <code>store_id</code> (ID de commerce)
Jeton API	<i>Chaîne</i> S.O.	<code>mpgReq.setApiToken(api_token);</code> <code>api_token</code> (jeton API)

### Champs de demande pour les transactions d'ajout d'un jeton temporaire de GooglePay (obligatoires)

Pour obtenir une description de toutes les valeurs obligatoires et facultatives, consultez l'Annexe A :Définition des champs de demande.

Variable	Type et limites	Méthode Set
Réseau	<p><i>Chaîne</i></p> <p>Alphabétique</p>	<pre>googlePayTokenTempAdd.setNetwork(network);</pre> <p>Nom de marque de carte</p> <p>Champ sensible à la casse</p> <p>Valeurs possibles :</p> <ul style="list-style-type: none"> <li>Visa</li> <li>Mastercard</li> <li>American Express</li> <li><i>Interac</i></li> <li>Discover</li> </ul>
Jeton de paiement	<p><i>Objet</i></p> <p>S.O.</p>	<pre>googlePayTokenTempAdd.setPaymentToken(signature, protocol_version, signed_message);</pre> <p>Détails du paiement renvoyés par Google dans son objet <code>PaymentData</code> pour les transactions Google Pay (voir la section Champs de demande pour l'objet Jeton de paiement Google Pay – Obligatoires ci-dessous pour les détails des champs)</p>

### Champs de demande pour l'objet Jeton de paiement Google Pay - Obligatoires

Variable	Type et limites	Méthode Set
Signature	<p><i>Chaîne</i></p> <p>S.O.</p>	<pre>String signature = "ME...gwole";</pre> <p>Vérifie que le message provient bien de Google, codé en base64 et créé avec ECDSA par la clé de signature intermédiaire et renvoyé par Google dans son objet <code>PaymentData</code> pour les transactions GooglePay.</p>
Version du protocole	<p><i>Chaîne</i></p> <p>S.O.</p>	<pre>String protocol_version = "ECv1";</pre> <p>Identifie le système de chiffrement ou de signature selon lequel le message est créé, permet au protocole d'évoluer au fil du temps, si nécessaire, et est renvoyé par Google</p>

Variable	Type et limites	Méthode Set
		dans son objet PaymentData pour les transactions Google Pay
Message signé	Chaîne S.O.	<pre>String signed_message = "\"encryptedMessage\..."";</pre> <p>Un objet JSON sérialisé sous la forme d'une chaîne HTML sécurisée qui contient le message codé, la clé publique éphémère et la balise, est sérialisé pour simplifier le processus de vérification de la signature et est renvoyé par Google dans son objet PaymentData pour les transactions Google Pay.</p>

### Exemple de transaction d'ajout d'un jeton temporaire de GooglePay

```
package Canada;
import JavaAPI.GooglePayPreauth;
import JavaAPI.GooglePayTokenTempAdd;
import JavaAPI.HttpsPostRequest;
import JavaAPI.Receipt;
public class TestCanadaGooglePayTokenTempAdd
{
    public static void main(String[] args)
    {
        java.util.Date createDate = new java.util.Date();
        String store_id = "intuit_sped";
        String api_token = "spedguy";
        String network = "MASTERCARD";
        String signature =
"MEYCIQdjfGZ1k/8h+eH9Ue5UxJsgDEFimp6YIrWhpt+eW3kAIhAOKikmz1B/C4WB5g3mXy139euOHnSQ7bQW12ch
gwole";
        String protocol_version = "ECv1";
        String signed_message =
"\"encryptedMessage\":\"nAEP5f0pzvU+cJHwxCwrCVPRrl96NugevgfrdidPOB5B+WG7+yrsYoUVA7HopRD5y5G
CldQwrKnP2h2w/Qc2HBfn+G/g2IXqPBzMjguhpGITr6lV0tRLaYimxrgrbh/Xn8DxfW++pTHHoo+0xJiON6o3JC4vM6w
mAuhjjweOgiDeKpgxJKE18NULR2RK10tvongkR80K8Et7CT+W01XoMCoYrH3tJDKMtovyFnfHPMAXLeV4NfVV+ZwhwD3
F+tGm7bQkPFMy2xUQxzdj7/H03vmyxwsblSKXhVG3hWKPmnY/+Gkb2K0pAicOHaB/SZuwaxHQll30jaNafUIm96R9T2Y
c3p5gmnGiR03R9H5R8JqLL9Wb7LncvfIwuQppgbAKa6HdbuSjbehNotW8S34VqxvpeSFQFUNYgkQ+fVEU/VaC1E17PyF
8AMZKN10ZIZ1jj7jntqoD\",\"ephemeralPublicKey\":\"BD5snQM3HF2gdCyERaF9XBPDGOXL8fNyTM9QY/xNTi9
VkJWTtq5sg7dYgPxLmQuwIhBN9OyLULAMsNcmsv2TT7k\\u003d\",\"tag\":\"hyG7Ty/qQAZel2INIMtDQPMAfod
VhUinW451hJrcP4\\u003d\"}";
        String processing_country_code = "CA";
        boolean status_check = false;

        GooglePayTokenTempAdd googlePayTokenTempAdd = new GooglePayTokenTempAdd();
        googlePayTokenTempAdd.setPaymentToken(signature, protocol_version, signed_message);
        HttpsPostRequest mpgReq = new HttpsPostRequest();
        mpgReq.setProcCountryCode(processing_country_code);
        mpgReq.setTestMode(true); //false or comment out this line for production transactions
        mpgReq.setstoreId(store_id);
        mpgReq.setApiToken(api_token);
        mpgReq.setTransaction(googlePayTokenTempAdd);
        mpgReq.setStatusCheck(status_check);

        //Optional - Proxy
        mpgReq.setProxy(false); //true to use proxy
        mpgReq.setProxyHost("proxyURL");
        mpgReq.setProxyPort("proxyPort");
        mpgReq.setProxyUser("proxyUser"); //optional - domainName\user
        mpgReq.setProxyPassword("proxyPassword"); //optional
        mpgReq.send();
```

## Exemple de transaction d'ajout d'un jeton temporaire de GooglePay

```
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("Message = " + receipt.getMessage());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("DataKey = " + receipt.getDataKey());
System.out.println("MaskedPan = " + receipt.getResMaskedPan());
System.out.println("Exp Date = " + receipt.getResExpdate());
System.out.println("Payment Type = " + receipt.getPaymentType());
System.out.println("GooglepayPaymentMethod = " + receipt.getGooglepayPaymentMethod());
}
catch (Exception e)
{
e.printStackTrace();
}
}
```

## 10.8 Transaction de préautorisation par jeton Google PayMD

La transaction de préautorisation par jeton Google Pay<sup>MD</sup> est utilisée après avoir transformé un compte Google Pay en jeton temporaire à l'aide de notre ajout temporaire d'un jeton Google Pay puis avoir effectué une authentification 3DS avec le jeton. Cette opération permet de vérifier les fonds déposés sur la carte du client et de les bloquer pendant une période spécifiée par l'émetteur de la carte.

Pour effectuer l'authentification 3-D Secure, le MPI de Moneris ou tout autre MPI tiers peut être utilisé.

Accédez aux portails des développeurs d'Apple ou de Google pour obtenir des détails sur l'intégration directe à leurs portefeuilles et recueillir les données utiles.

### Transaction de préautorisation par jeton Google Pay<sup>MD</sup> pour la définition de l'objet de la transaction

```
GooglePayTokenPreauth googlePayTokenPreauth = new GooglePayTokenPreauth();
```

### Objet HttpsPostRequest pour la transaction Google Pay™ Token Preauth

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setTransaction(googlePayTokenPreauth);
```

### Principaux champs d'objet de connexion (toutes les transactions API)

Variable	Type et limites	Description
Code de magasin	Chaîne S.O.	mpgReq.setstoreId(store_id); store_id (ID de commerce)
Jeton API	Chaîne S.O.	mpgReq.setApiToken(api_token); api_token Jeton API

## Champs de demande de transaction de préautorisation par jeton Google Pay<sup>MD</sup> - Obligatoires

Variable	Type et limites	Méthode Set
ID de commande	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p> <p>a-Z A-Z 0-9 _ - : @ espaces</p>	googlePayTokenPreauth.setOrderId(order_id);
Clé de donnée	<p><i>Chaîne</i></p> <p>25 caractères alphanumériques</p>	<p>googlePayTokenPreauth.setData(data_key);</p> <p>Le jeton temporaire renvoyé par une demande GooglePayTokenTempAdd</p>
Montant	<p><i>Chaîne</i></p> <p>Décimale à 10 caractères</p> <p>Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point</p> <p><b>EXAMPLE : 1234567.89</b></p>	googlePayTokenPreauth.setAmount(amount);
Code de vérification d'authentification du titulaire de carte (CAVV)	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p>	<p>googlePayTokenPreauth.setCavv(cavv);</p> <p>Le cryptogramme 3DS</p>
<b>REMARQUE:</b> Pour les transactions de préautorisation par jeton Google Pay <sup>MD</sup> , le champ CAVV contient le cryptogramme 3DS uniquement lorsque l'authentification3DS est utilisée au préalable. Si vous avez choisi d'ignorer l'authentification 3DS, vous pouvez omettre le champ CAVV.		<p>Envoyé dans toutes les transactions financières avec 3-D Secure, y compris Verified By Visa, Mastercard SecureCode et American Express SafeKey</p>
Indicateur de commerce électronique	<p><i>Chaîne</i></p> <p>1 caractère alphanumérique</p>	<p>googlePayTokenPreauth.setCryptType(crypt);</p> <p>Décrit la catégorie de transaction de commerce électronique en cours de traitement</p> <p>Les valeurs autorisées sont :</p>
<b>NOTE:</b> Pour les transactions d'achat de jetons Google Pay <sup>MD</sup> et de préautorisation de jetons utilisant l'authentification 3DS, utilisez l'indicateur de commerce électronique obtenu à partir de votre authentification 3DS.		<p>1 – Commande postale ou téléphonique – Unique</p>

Variable	Type et limites	Méthode Set
		<p>2 – Commande postale ou téléphonique – Périodique</p> <p>3 – Commande postale ou téléphonique – Versement</p> <p>4 – Commande postale ou téléphonique – Classification inconnue</p> <p>5 – Transaction électronique authentifiée (3-D Secure)</p> <p>6 – Transaction électronique non authentifiée (3-D Secure)</p> <p>7 – Commerçant prenant en charge le SSL</p> <p><b>Lors d'une transaction utilisant les renseignements d'identification au dossier pour laquelle le champ de demande d'indicateur de commerce électronique est également inclus, les valeurs permises de ce champ dépendent des valeurs ci-dessous envoyées pour l'indicateur de paiement.</b></p> <p>Si l'indicateur de paiement = R, les valeurs autorisées pour l'indicateur de commerce électronique sont 2, 5 ou 6.</p> <p>Si l'indicateur de paiement = V, les valeurs autorisées pour l'indicateur de commerce électronique sont 2, 5 ou 6.</p> <p>Si l'indicateur de paiement = C, les valeurs autorisées pour l'indicateur de commerce électronique sont 1, 5, 6 ou 7.</p> <p>Si l'indicateur de paiement = U, alors les valeurs autorisées pour l'indicateur de commerce électronique sont 1 ou 7.</p> <p>Si l'indicateur de paiement = Z, les valeurs autorisées pour l'indicateur de commerce électronique sont 1, 5, 6 ou 7.</p>
ID de transaction du serveur 3DS	<i>Chaîne</i> 36 caractères numériques	<pre>googlePayTokenPreauth.setThreeDServerTransId("ThreeDServerTransId");</pre> <p>Les données sont obtenues à partir d'une demande de recherche de CAVV ou d'une demande d'authentification MPI 3DS</p> <p><b>REMARQUE:</b> Obtenu à partir de la demande de recherche de CAVV ou de la demande d'authentification MPI 3DS. Pour les transactions d'achat de jetons Google Pay<sup>MD</sup> et de préautorisation de jetons qui n'utilisent pas l'authentification</p>

Variable	Type et limites	Méthode Set
3DS, vous pouvez omettre l'ID de transaction du serveur 3DS.		
Version de 3DS  <b>REMARQUE:</b> Cette variable est obligatoire pour les transactions financières utilisant les services 3-D Secure d'un tiers. Si vous avez choisi d'ignorer l'authentification 3DS, vous pouvez omettre le champ Version 3DS.	<i>Chaîne</i> 10 caractères numériques	<code>googlePayTokenPreauth.setThreeDSVersion("ThreeDSVersion");</code>  Valeurs acceptables :  2.0.0 = Protocole 3DS 2.0.0 2.1.0 = Protocole 3DS 2.1.0 2.2.0 = Protocole 3DS 2.2.0 2.3.0 = Protocole 3DS 2.3.0
Réseau	<i>Chaîne</i> Alphabétique	<code>googlePayTokenPreauth.setNetwork(network);</code>  Nom de marque de carte  Champ sensible à la casse  Valeurs possibles :  Visa  Mastercard  American Express  <i>Interac</i>  Discover

#### Champs de demande de transaction de préautorisation par jeton Google Pay<sup>MD</sup> - Facultatifs

Variable	Type et limites	Méthode Set
ID client	<i>Chaîne</i> 50 caractères alphanumériques	<code>googlePayTokenPreauth.setCustomerId(cust_id);</code>  <div style="border: 1px solid #ccc; padding: 5px; background-color: #e0f2e0;"><b>REMARQUE :</b> Les caractères spéciaux suivants ne sont pas autorisés : &lt;&gt;\$%=?^{}[]\</div>
Descripteur dynamique	<i>Chaîne</i>	<code>googlePayTokenPreauth.setDynamicDescriptor(dynamic_descriptor);</code>

Variable	Type et limites	Méthode Set
	<p>Maximum 20 caractères alphanumériques</p> <p>Total de 22 caractères incluant le nom du commerce et un séparateur</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <b>REMARQUE :</b> Les caractères spéciaux suivants ne sont pas autorisés :            &lt;&gt;\$%=?^{}[]\         </div>	

### Exemple d'une transaction de préautorisation par jeton Google Pay<sup>MD</sup>

```

package Canada;
import JavaAPI.GooglePayTokenPreauth;
import JavaAPI.HttpsPostRequest;
import JavaAPI.Receipt;
public class TestCanadaGooglePayTokenPreauth
{
  public static void main(String[] args)
  {
    java.util.Date createDate = new java.util.Date();
    String order_id = "Test"+createDate.getTime();
    String store_id = "intuit_sped";
    String api_token = "spedguy";
    String amount = "1.00";
    String cust_id = "nqa-cust_id";
    String network = "MASTERCARD";
    String data_key = "ot-1yAH5XwBu72JH19zU3ozQCmR1";
    String crypt_type = "2";
    String threeeds_server_trans_id = "de1b97ee-c610-4877-b53f-c1c5ecd99bf0";
    String threeeds_version = "2.2";
    String cavv = "kAABApFSYyd4l2eQQFJjAAAAAAA=";
    String dynamic_descriptor = "nqa-dd";
    String processing_country_code = "CA";
    boolean status_check = false;
    GooglePayTokenPreauth googlePayTokenPreauth = new GooglePayTokenPreauth();
    googlePayTokenPreauth.setOrderId(order_id);
    googlePayTokenPreauth.setCustomerId(cust_id);
    googlePayTokenPreauth.setAmount(amount);
    googlePayTokenPreauth.setNetwork(network);
    googlePayTokenPreauth.setDynamicDescriptor(dynamic_descriptor);
    googlePayTokenPreauth.setDataKey(data_key);
    googlePayTokenPreauth.setCryptType(crypt_type);
    googlePayTokenPreauth.setThreeDSServerTransId(threeeds_server_trans_id);
    googlePayTokenPreauth.setDSTransId(ds_trans_id);
    googlePayTokenPreauth.setThreeDSVersion(threeeds_version);
    googlePayTokenPreauth.setCavv(cavv);

    HttpsPostRequest mpgReq = new HttpsPostRequest();
    mpgReq.setProcCountryCode(processing_country_code);
    mpgReq.setTestMode(true); //false or comment out this line for production transactions
    mpgReq.setstoreId(store_id);
    mpgReq.setApiToken(api_token);
    mpgReq.setTransaction(googlePayTokenPreauth);
    mpgReq.setStatusCheck(status_check);
    //Optional - Proxy
    mpgReq.setProxy(false); //true to use proxy
    mpgReq.setProxyHost("proxyURL");
    mpgReq.setProxyPort("proxyPort");
    mpgReq.setProxyUser("proxyUser"); //optional - domainName\User
    mpgReq.setProxyPassword("proxyPassword"); //optional
    mpgReq.send();
    try
    {
      Receipt receipt = mpgReq.getReceipt();
      System.out.println("CardType = " + receipt.getCardType());
      System.out.println("TransAmount = " + receipt.getTransAmount());
    }
  }
}

```

```

System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = " + receipt.getBankTotals());
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("IsVisaDebit = " + receipt.getIsVisaDebit());
System.out.println("Payment Type = " + receipt.getPaymentType());
System.out.println("HostId = " + receipt.getHostId());
System.out.println("IssuerId = " + receipt.getIssuerId());
System.out.println("SourcePanLast4 = " + receipt.getSourcePanLast4());
System.out.println("ThreeDSVersion = " + receipt.getThreeDSVersion());
System.out.println("CavvResultCode = " + receipt.getCavvResultCode());
System.out.println("DataKey = " + receipt.getDataKey());
System.out.println("Par = " + receipt.getPar());
System.out.println("GooglepayPaymentMethod = " + receipt.getGooglepayPaymentMethod());
}
catch (Exception e)
{
e.printStackTrace();
}
}
}

```

## 10.9 Achat de jetons Google PayMD

La transaction d'achat par jeton Google Pay<sup>MD</sup> est utilisée après avoir transformé un compte Google Pay en jeton temporaire à l'aide de notre ajout temporaire de jeton Google Pay puis avoir effectué une authentification 3DS avec le jeton. Cette transaction vérifie que les fonds sont présents sur la carte du client, retire les fonds de la carte et prépare les fonds pour qu'ils soient déposés dans le compte de commerçant.

Pour effectuer l'authentification 3-D Secure, le MPI de Moneris ou tout autre MPI tiers peut être utilisé.

Accédez aux portails des développeurs d'Apple ou de Google pour obtenir des détails sur l'intégration directe à leurs portefeuilles et recueillir les données utiles.

### Achat de jetons Google Pay<sup>MD</sup> pour la définition de l'objet de la transaction

```
GooglePayTokenPurchase googlePayTokenPurchase = new GooglePayTokenPurchase();
```

### Objet HttpsPostRequest pour la transaction d'achat de jetons Google Pay<sup>MD</sup>

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setTransaction(googlePayTokenPurchase);
```

### Principaux champs d'objet de connexion (toutes les transactions API)

Variable	Type et limites	Description
Code de magasin	<i>Chaîne</i>  S.O.	mpgReq.setstoreId(store_id);  store_id (ID de commerce)
Jeton API	<i>Chaîne</i>	mpgReq.setApiToken(api_token);

Variable	Type et limites	Description
	S.O.	api_token (Jeton API)

### Champs de demande de transaction d'achat de jetons Google Pay<sup>MD</sup> - Obligatoires

Variable	Type et limites	Méthode Set
ID de commande	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p> <p>a-Z A-Z 0-9 _ - : @ espaces</p>	googlePayTokenPurchase.setOrderId(order_id);
Clé de donnée	<p><i>Chaîne</i></p> <p>25 caractères alphanumériques</p>	<p>googlePayTokenPurchase.setData(data_key);</p> <p>Le jeton temporaire renvoyé par une demande GooglePayTokenTempAdd</p>
Montant	<p><i>Chaîne</i></p> <p>Décimale à 10 caractères</p> <p>Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point</p> <p>EXEMPLE : 1234567.89</p>	googlePayTokenPurchase.setAmount(amount);
Code de vérification d'authentification du titulaire de carte (CAVV)	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p>	<p>googlePayTokenPurchase.setCavv(cavv);</p> <p>Le cryptogramme 3DS</p> <p>Envoyé dans toutes les transactions financières avec 3-D Secure, y compris Verified By Visa, Mastercard SecureCode et American Express SafeKey</p>
Indicateur de commerce électronique	<p><i>Chaîne</i></p> <p>1 caractère alphanumérique</p>	<p>googlePayTokenPurchase.setCryptType(crypt);</p> <p>Décrit la catégorie de transaction de commerce électronique en cours de traitement</p>
REMARQUE: Pour les transactions d'achat de jetons Google Pay <sup>MD</sup> , le champ CAVV contient le cryptogramme 3DS uniquement lorsque l'authentification 3DS est utilisée au préalable. Si vous avez choisi d'ignorer l'authentification 3DS, vous pouvez omettre le champ CAVV.		

Variable	Type et limites	Méthode Set
utilisant l'authentification 3DS, utilisez l'indicateur de commerce électronique obtenu à partir de votre authentification 3DS.		<p>Les valeurs autorisées sont :</p> <p>1 – Commande postale ou téléphonique – Unique</p> <p>2 – Commande postale ou téléphonique – Périodique</p> <p>3 – Commande postale ou téléphonique – Versement</p> <p>4 – Commande postale ou téléphonique – Classification inconnue</p> <p>5 – Transaction électronique authentifiée (3-D Secure)</p> <p>6 – Transaction électronique non authentifiée (3-D Secure)</p> <p>7 – Commerçant prenant en charge le SSL</p> <p><b>Lors d'une transaction utilisant les renseignements d'identification au dossier pour laquelle le champ de demande d'indicateur de commerce électronique est également inclus, les valeurs permises de ce champ dépendent des valeurs ci-dessous envoyées pour l'indicateur de paiement.</b></p> <p>Si l'indicateur de paiement = R, les valeurs autorisées pour l'indicateur de commerce électronique sont 2, 5 ou 6.</p> <p>Si l'indicateur de paiement = V, les valeurs autorisées pour l'indicateur de commerce électronique sont 2, 5 ou 6.</p> <p>Si l'indicateur de paiement = C, les valeurs autorisées pour l'indicateur de commerce électronique sont 1, 5, 6 ou 7.</p> <p>Si l'indicateur de paiement = U, alors les valeurs autorisées pour l'indicateur de commerce électronique sont 1 ou 7.</p> <p>Si l'indicateur de paiement = Z, les valeurs autorisées pour l'indicateur de commerce électronique sont 1, 5, 6 ou 7.</p>
ID de transaction du serveur 3DS	<i>Chaîne</i> 36 caractères numériques	<pre>googlePayTokenPurchase.setThreeDServerTransId("ThreeDServerTransId");</pre> <p>Les données sont obtenues à partir d'une demande de recherche de</p> <p><b>REMARQUE:</b> Cette variable est obtenue à partir de la demande de recherche de CAVV ou de la demande d'authentification MPI 3DS. Pour les transactions d'achat</p>

Variable	Type et limites	Méthode Set
de jetons Google Pay™ et de préautorisation par jeton qui n'utilisent pas l'authentification 3DS, vous pouvez omettre l'ID de transaction du serveur 3DS.		CAV Vou une demande d'authentification MPI 3DS.
Version de 3DS  <b>REMARQUE :</b> Si vous avez choisi d'ignorer l'authentification 3DS, vous pouvez omettre le champ Version 3DS.	<i>Chaîne</i>  10 caractères numériques	googlePayTokenPurchase.setThreeDSVersion("ThreeDSVersion");  Valeurs acceptables :  2.0.0 = Protocole 3DS 2.0.0  2.1.0 = Protocole 3DS 2.1.0  2.2.0 = Protocole 3DS 2.2.0  2.3.0 = Protocole 3DS 2.3.0
Réseau	<i>Chaîne</i>  Alphabétique	googlePayTokenPurchase.setNetwork(network);  Nom de marque de carte  Champ sensible à la casse  Valeurs possibles :  Visa  Mastercard  American Express  <i>Interac</i>  Discover

#### Champs de demande de transaction d'achat des jetons Google Pay<sup>MD</sup> - Facultatifs

Variable	Type et limites	Méthode Set
ID client	<i>Chaîne</i>  50 caractères alphanumériques  <div style="border: 1px solid #ccc; padding: 5px; background-color: #e0f2fd;"><b>REMARQUE :</b> Les caractères spéciaux suivants ne sont pas autorisés : &lt; &gt; \$ % = ? ^ { } [ ] \</div>	googlePayTokenPurchase.setCustomerId(cust_id);

Variable	Type et limites	Méthode Set
Descripteur dynamique	<p><i>Chaîne</i></p> <p>Maximum 20 caractères alphanumériques</p> <p>Total de 22 caractères incluant le nom du commerce et un séparateur</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <b>REMARQUE :</b> Les caractères spéciaux suivants ne sont pas autorisés :          &lt;&gt;\$%=?^{}[]\       </div>	googlePayTokenPurchase.setDynamicDescriptor(dynamic_descriptor);

## Exemple d'achat de jetons Google Pay<sup>MD</sup>

```

package Canada;
import JavaAPI.GooglePayTokenPurchase;
import JavaAPI.HttpsPostRequest;
import JavaAPI.Receipt;
public class TestCanadaGooglePayTokenPurchase
{
  public static void main(String[] args)
  {
    java.util.Date createDate = new java.util.Date();
    String order_id = "Test"+createDate.getTime();
    String store_id = "intuit_sped";
    String api_token = "spedguy";
    String amount = "1.00";
    String cust_id = "nqa-cust_id";
    String network = "MASTERCARD";
    String data_key = "ot-5N7NHu23o7w82xL8KLOXV2vC1";
    String crypt_type = "2";
    String threeDS_server_trans_id = "de1b97ee-c610-4877-b53f-c1c5ecd99bf0";
    String threeDS_version = "2.2";
    String cavv = "kAABApFSYyd412eQQFJjAAAAAAA=";
    String dynamic_descriptor = "nqa-dd";
    String processing_country_code = "CA";
    boolean status_check = false;

    GooglePayTokenPurchase googlePayTokenPurchase = new GooglePayTokenPurchase();
    googlePayTokenPurchase.setOrderId(order_id);
    googlePayTokenPurchase.setCustomerId(cust_id);
    googlePayTokenPurchase.setAmount(amount);
    googlePayTokenPurchase.setNetwork(network);
    googlePayTokenPurchase.setDataKey(data_key);
    googlePayTokenPurchase.setCryptType(crypt_type);
    googlePayTokenPurchase.setThreeDSserverTransId(threeDS_server_trans_id);
    googlePayTokenPurchase.setDSTransId(ds_trans_id);
    googlePayTokenPurchase.setThreeDSVersion(threeDS_version);
    googlePayTokenPurchase.setCavv(cavv);
    googlePayTokenPurchase.setDynamicDescriptor(dynamic_descriptor);

    HttpsPostRequest mpgReq = new HttpsPostRequest();
    mpgReq.setProcCountryCode(processing_country_code);
    mpgReq.setTestMode(true); //false or comment out this line for production transactions
    mpgReq.setstoreId(store_id);
    mpgReq.setApiToken(api_token);
    mpgReq.setTransaction(googlePayTokenPurchase);
    mpgReq.setStatusCheck(status_check);

    //Optional - Proxy
    mpgReq.setProxy(false); //true to use proxy
    mpgReq.setProxyHost("proxyURL");
    mpgReq.setProxyPort("proxyPort");
    mpgReq.setProxyUser("proxyUser"); //optional - domainName\user
    mpgReq.setProxyPassword("proxyPassword"); //optional
  }
}

```

```
mpgReq.send();

try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = " + receipt.getBankTotals());
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("IsVisaDebit = " + receipt.getIsVisaDebit());
System.out.println("Payment Type = " + receipt.getPaymentType());
System.out.println("HostId = " + receipt.getHostId());
System.out.println("IssuerId = " + receipt.getIssuerId());
System.out.println("SourcePanLast4 = " + receipt.getSourcePanLast4());
System.out.println("ThreeDSVersion = " + receipt.getThreeDSVersion());
System.out.println("CavvResultCode = " + receipt.getCavvResultCode());
System.out.println("DataKey = " + receipt.getDataKey());
System.out.println("Par = " + receipt.getPar());
System.out.println("GooglepayPaymentMethod = " + receipt.getGooglepayPaymentMethod());
}
catch (Exception e)
{
e.printStackTrace();
}
}
```

## 11 Offlinx<sup>MC</sup>

- Qu'est-ce qu'un pixel invisible?
- Offlinx<sup>MC</sup> et les transactions API

### 11.1 Qu'est-ce qu'un pixel invisible?

Un pixel invisible est une partie du code qui va sur une page Web et demande un fichier image (une minuscule image transparente ou pixel) lorsqu'il est chargé, qui, sans être visible par l'utilisateur, permet à Offlinx<sup>MC</sup> de recueillir des renseignements pertinents sur l'utilisateur.

Les données recueillies par notre pixel invisible sont :

- Anonymes (non identifiables individuellement) et conformes aux normes de confidentialité
- Sécurisées (La communication SSL est utilisée pour transmettre les données en toute sécurité.)
- Aucunement partagées avec qui que ce soit

### 11.2 Offlinx<sup>MC</sup> et les transactions API

La fonction Offlinx<sup>MC</sup> Card Match pour le pixel invisible peut être mise en œuvre grâce à l'API unifiée avec la variable Card Match ID, qui correspond à l'ID de la transaction (Transaction ID) dans Offlinx<sup>MC</sup>. La variable Card Match ID doit être une valeur unique pour chaque transaction.

Pour plus de renseignements sur la solution Offlinx<sup>MC</sup>, consultez le guide de configuration des pixels invisibles de Offlinx<sup>MC</sup>, disponible auprès de votre gestionnaire de compte ou de service.

Transactions API pour lesquelles cela s'applique :

- Achat
- Préautorisation
- Achat avec 3-D Secure (cavv\_purchase)
- Préautorisation avec 3-D Secure (cavv\_prauth)
- Achat utilisant le code de vérification d'authentification du titulaire de carte – Apple Pay et Google Pay<sup>MC</sup>
- Préautorisation utilisant le code de vérification d'authentification du titulaire de carte – Apple Pay et Google Pay<sup>MC</sup>

## 12 Frais de commodité

- 12.1 À propos des frais de commodité
- 12.3 Achat avec frais de commodité
- 12.4 Achat avec renseignements sur le client et frais commodité
- 12.5 Achat avec frais de commodité utilisant 3-D Secure

### 12.1 À propos des frais de commodité

Le programme de frais de commodité a été conçu pour permettre aux commerçants d'offrir au titulaire de carte d'utiliser un autre mode de paiement moyennant des frais. Cela ne s'applique que lorsque le commerçant offre une véritable « commodité » sous la forme d'un canal de paiement de substitution qui ne fait pas partie de ses canaux de paiement habituels en personne. Les frais de commodité seront facturés séparément en plus de ce que le consommateur paie pour les biens ou services qui lui ont été fournis, et ces frais apparaîtront sur une ligne distincte du relevé du consommateur.

Les transactions avec frais de commodité ne sont pas compatibles avec la TMD ou les portefeuilles électroniques.

**REMARQUE :** Le programme de frais de commodité n'est offert qu'à certains codes de catégorie de commerçant (CCC) pris en charge. Veuillez communiquer avec votre gestionnaire de compte afin d'obtenir de plus amples renseignements.

### 12.2 Objet Convenience Fee Information

Toute transaction prenant en charge les frais de commodité dispose d'une méthode set pour l'objet Convenience Fee Information.

L'objet Convenience Fee Information contient un champ de demande, **convenience fee amount**.

#### Définition de l'objet Convenience Fee Info

```
ConvFeeInfo convFeeInfo = new ConvFeeInfo();
```

#### Méthode set de l'objet Convenience Fee Info

```
<transaction>.setConvenienceFee(convFeeInfo);
```

#### Champs de demande de l'objet Convenience Fee Information

Variable	Type et limites	Description
Information sur les frais de commodité	<i>Objet</i> S. O.	Contient des champs liées à la fonction de frais de commodité

Variable	Type et limites	Description
Montant des frais de commodité	<i>Chaîne</i> 9 caractères décimaux	Montant en dollars facturé au client en tant que frais de commodité

## 12.3 Achat avec frais de commodité

Les renseignements ci-dessous décrivent une demande de transaction d'achat qui comprend également l'objet Convenience Fee Info.

### Définition de l'objet de transaction Purchase with Convenience Fee

```
Purchase purchase = new Purchase();
```

### Objet HttpsPostRequest pour les transactions d'achat avec frais de commodité

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
```

```
mpgReq.setTransaction(purchase);
```

### Champs de demande liés aux transactions d'achat avec frais de commodité (obligatoires)

Pour obtenir une description de toutes les valeurs obligatoires et facultatives, consultez l'annexe A Définition des champs de demande.

Variable	Type et limites	Méthode Set
Information sur les frais de commodité	<i>Objet</i> S. O.	ConvFeeInfo convFeeInfo = new ConvFeeInfo(); purchase.setConvenienceFee(convFeeInfo);
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques a-Z A-Z 0-9 _ - : . @ espaces	purchase.setOrderId(order_id);
Montant	<i>Chaîne</i> 10 caractères décimaux Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal	purchase.setAmount(amount);

Variable	Type et limites	Méthode Set
	<b>EXEMPLE :</b> 1 234 567,89	
Numéro de carte de crédit	<i>Chaîne</i> Maximum 20 caractères alphanumériques	<code>purchase.setPan(pan);</code>
Date d'expiration	<i>Chaîne</i> 4 caractères alphanumériques AAMM	<code>purchase.setExpDate(expiry_date);</code>
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	<code>purchase.setCryptType(crypt);</code>
<b>REMARQUE :</b> Lors d'une transaction utilisant les renseignements d'identification au dossier pour laquelle le champ de demande e-commerce indicator est également inclus, les valeurs acceptables de ce champ dépendent de la valeur envoyée pour l'indicateur de paiement. Pour plus d'information, consultez les définitions des champs de réponse.		
Montant des frais de commodité	<i>Chaîne</i> 9 caractères décimaux	<code>convFeeInfo.setConvenienceFee(convfee_amount);</code>

#### Champs de demande liés aux transactions de d'achat avec frais de commodité (facultatifs)

Variable	Type et limites	Méthode Set
ID de client	<i>Chaîne</i> 50 caractères alphanumériques <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <b>REMARQUE :</b> Certains caractères         </div>	<code>purchase.setCustId(cust_id);</code>

Variable	Type et limites	Méthode Set
	<p>spéciaux ne sont pas autorisés :</p> <p>&lt;&gt;\$%=?^{}[]\</p>	
Descripteur dynamique	<p><i>Chaîne</i></p> <p>20 caractères alphanumériques</p> <p>Total de 22 caractères incluant votre nom de commerçant et un séparateur</p> <p><b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés :</p> <p>&lt;&gt;\$%=?^{}[]\</p>	<pre>purchase.setDynamicDescriptor(dynamic_descriptor);</pre>
Renseignements du SVA	<p><i>Objet</i></p> <p>S. O.</p>	<pre>AvsInfo avsCheck = new AvsInfo(); purchase.setAvsInfo(avsCheck);</pre>
Renseignements du NVC	<p><i>Objet</i></p> <p>S. O.</p>	<pre>CvdInfo cvdCheck = new CvdInfo(); purchase.setCvdInfo(cvdCheck);</pre>

#### Exemple d'achat avec frais de commodité

```

package Canada;
import JavaAPI.*;
public class TestCanadaConvFeePurchase
{
public static void main(String args[])
{
String store_id = "monca00392";
String api_token = "qYdISUhHiOdfTr1CLNpN";
String processing_country_code = "CA";
java.util.Date createDate = new java.util.Date();
String order_id = "Test"+createDate.getTime();
String amount = "10.00";
String pan = "4242424242424242";
String expdate = "1911";
String crypt = "7";

ConvFeeInfo convFeeInfo = new ConvFeeInfo();
convFeeInfo.setConvenienceFee("1.00");

Purchase purchase = new Purchase();
purchase.setOrderId(order_id);
purchase.setAmount(amount);
purchase.setPan(pan);
purchase.setExpdate(expdate);
purchase.setCryptType(crypt);
purchase.setConvFeeInfo(convFeeInfo);
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false for production transactions
}
}
```

```

mpgReq.setstoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(purchase);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = " + receipt.getBankTotals());
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("CfSuccess = " + receipt.getCfSuccess());
System.out.println("CfStatus = " + receipt.getCfStatus());
System.out.println("FeeAmount = " + receipt.getFeeAmount());
System.out.println("FeeRate = " + receipt.getFeeRate());
System.out.println("FeeType = " + receipt.getFeeType());
}
catch (Exception e)
{
e.printStackTrace();
}
}
}

```

## 12.4 Achat avec renseignements sur le client et frais commodité

Les renseignements ci-dessous décrivent une demande de transaction d'achat qui comprend également les objets Convenience Fee Info et Customer Information.

### Définition de l'objet de transaction Purchase with Customer Info and Convenience Fee

```
Purchase purchase = new Purchase();
```

### Objet HttpsPostRequest pour les transactions d'achat avec renseignements sur le client et frais commodité

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
```

```
mpgReq.setTransaction(purchase);
```

## Champs de demande liés aux transactions d'achat avec renseignements sur le client et frais commodité (obligatoires)

Pour obtenir une description de toutes les valeurs obligatoires et facultatives, consultez l'annexe A Définition des champs de demande.

Variable	Type et limites	Méthode Set
Information sur les frais de commodité	<i>Objet</i> S. O.	<code>ConvFeeInfo convFeeInfo = new ConvFeeInfo(); purchase.setConvenienceFee(convFeeInfo);</code>
ID de commande	<i>Chaîne</i>  50 caractères alphanumériques  a-Z A-Z 0-9 _ - : . @ espaces	<code>purchase.setOrderId(order_id);</code>
Montant	<i>Chaîne</i>  10 caractères décimaux  Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal  <b>EXEMPLE : 1 234 567,89</b>	<code>purchase.setAmount(amount);</code>
Numéro de carte de crédit	<i>Chaîne</i>  Maximum 20 caractères alphanumériques	<code>purchase.setPan(pan);</code>
Date d'expiration	<i>Chaîne</i>  4 caractères alphanumériques  AAMM	<code>purchase.setExpDate(expiry_date);</code>
Indicateur de commerce électronique	<i>Chaîne</i>  1 caractère alphanumérique	<code>purchase.setCryptType(crypt);</code>
Montant des frais de commodité	<i>Chaîne</i>  9 caractères décimaux	<code>purchaseconvFeeInfo.setConvenienceFee(convfee_amount);</code>

**Champs de demande liés aux transactions d'achat avec renseignements sur le client et frais commodité (facultatifs)**

Variable	Type et limites	Méthode Set
ID de client	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p> <div style="background-color: #e0f2e0; padding: 5px;"> <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : &lt;&gt;\$%=?^{}[]\</div>	purchase.setCustId(cust_id);
Descripteur dynamique	<p><i>Chaîne</i></p> <p>20 caractères alphanumériques</p> <p>Total de 22 caractères incluant votre nom de commerçant et un séparateur</p> <div style="background-color: #e0f2e0; padding: 5px;"> <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : &lt;&gt;\$%=?^{}[]\</div>	purchase.setDynamicDescriptor(dynamic_descriptor);
Renseignements du client	<p><i>Objet</i></p> <p>S. O.</p>	purchase.setCustInfo(customer);
Renseignements du SVA	<p><i>Objet</i></p> <p>S. O.</p>	purchase.setAvsInfo(avsCheck);
Renseignements du NVC	<p><i>Objet</i></p> <p>S. O.</p> <div style="background-color: #e0f2e0; padding: 5px;"> <b>REMARQUE :</b> Lorsque les renseignements d'identification sont stockés lors de la première transaction, l'objet CVD (numéro de vérification de carte, ou NVC) doit être envoyé. Pour les transactions subséquentes utilisant les renseignements d'identification stockés, le NVC peut être envoyé avec les transactions entamées par le titulaire de carte seulement. Les commerçants ne doivent pas enregistrer le NVC.         </div>	purchase.setCvdInfo(cvdCheck);

**Exemple d'achat avec renseignements sur le client et frais commodité**

```

import java.util.*;
import JavaAPI.*;
public class TestCanadaConvFeePurchaseCustInfo
{
    public static void main(String[] args)
    {
        String store_id = "monca00392";
        String api_token = "qYdISUhHiOdfTr1CLNpN";
        java.util.Date createDate = new java.util.Date();
        String order_id = "Test"+createDate.getTime();
        String amount = "10.00";
        String pan = "4242424242424242";
        String expdate = "1901"; //YYMM format
        String crypt = "7";
        String processing_country_code = "CA";
        boolean status_check = false;
        /***** Billing/Shipping Variables *****/
        String first_name = "Bob";
        String last_name = "Smith";
        String company_name = "ProLine Inc.";
        String address = "623 Bears Ave";
        String city = "Chicago";
        String province = "Illinois";
        String postal_code = "M1M2M1";
        String country = "Canada";
        String phone = "777-999-7777";
        String fax = "777-999-7778";
        String tax1 = "10.00";
        String tax2 = "5.78";
        String tax3 = "4.56";
        String shipping_cost = "10.00";
        /***** Order Line Item Variables *****/
        String[] item_description = new String[] { "Chicago Bears Helmet", "Soldier Field Poster" };
        String[] item_quantity = new String[] { "1", "1" };
        String[] item_product_code = new String[] { "CB3450", "SF998S" };
        String[] item_extended_amount = new String[] { "150.00", "19.79" };
        /***** */
        /* Customer Information Option 1 */
        /* */
        /***** */
        /***** Customer Information Object *****/
        CustInfo customer = new CustInfo();
        /***** Set Customer Billing Information *****/
        customer.setBilling(first_name, last_name, company_name, address, city,
            province, postal_code, country, phone, fax, tax1, tax2,
            tax3, shipping_cost);
        /***** Set Customer Shipping Information *****/
        customer.setShipping(first_name, last_name, company_name, address, city,
            province, postal_code, country, phone, fax, tax1, tax2,
            tax3, shipping_cost);
        /***** Order Line Items *****/
        customer.setItem(item_description[0], item_quantity[0],
            item_product_code[0], item_extended_amount[0]);
        customer.setItem(item_description[1], item_quantity[1],
            item_product_code[1], item_extended_amount[1]);
        /***** */
        /* Customer Information Option 2 */
        /* */
        /***** */
        /***** Customer Information Object *****/
        CustInfo customer2 = new CustInfo();
        /***** Billing Hashtable *****/
        Hashtable<String, String> b = new Hashtable<String, String>(); //billing hashtable
        b.put("first_name", first_name);
        b.put("last_name", last_name);
        b.put("company_name", company_name);
        b.put("address", address);
        b.put("city", city);
        b.put("province", province);
    }
}

```

```

b.put("postal_code", postal_code);
b.put("country", country);
b.put("phone", phone);
b.put("fax", fax);
b.put("tax1", tax1); //federal tax
b.put("tax2", tax2); //prov tax
b.put("tax3", tax3); //luxury tax
b.put("shipping_cost", shipping_cost); //shipping cost
customer2.setBilling(b);
/********************* Shipping Hashtable *****/
Hashtable<String, String> s = new Hashtable<String, String>(); //shipping hashtable
s.put("first_name", first_name);
s.put("last_name", last_name);
s.put("company_name", company_name);
s.put("address", address);
s.put("city", city);
s.put("province", province);
s.put("postal_code", postal_code);
s.put("country", country);
s.put("phone", phone);
s.put("fax", fax);
s.put("tax1", tax1); //federal tax
s.put("tax2", tax2); //prov tax
s.put("tax3", tax3); //luxury tax
s.put("shipping_cost", shipping_cost); //shipping cost
customer2.setShipping(s);
/********************* Order Line Item1 Hashtable *****/
Hashtable<String, String> i1 = new Hashtable<String, String>(); //item hashtable #1
i1.put("name", item_description[0]);
i1.put("quantity", item_quantity[0]);
i1.put("product_code", item_product_code[0]);
i1.put("extended_amount", item_extended_amount[0]);
customer2.setItem(i1);
/********************* Order Line Item2 Hashtable *****/
Hashtable<String, String> i2 = new Hashtable<String, String>(); //item hashtable #2
i2.put("name", "item2's name");
i2.put("quantity", "7");
i2.put("product_code", "item2's product code");
i2.put("extended_amount", "5.01");
customer2.setItem(i2);
/********************* Miscellaneous Customer Information Methods *****/
customer.setEmail("nick@widget.com");
customer.setInstructions("Make it fast!");

/********************* Convenience Fee *****/
ConvFeeInfo convFeeInfo = new ConvFeeInfo();
convFeeInfo.setConvenienceFee("1.00");
/********************* Transactional Request Object *****/
Purchase purchase = new Purchase();
purchase.setOrderId(order_id);
purchase.setAmount(amount);
purchase.setPan(pan);
purchase.setExdate(expdate);
purchase.setCryptType(crypt);
purchase.setCustInfo(customer);
purchase.setConvFeeInfo(convFeeInfo);
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false for production transactions
mpgReq.setstoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(purchase);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = " + receipt.getBankTotals());
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
}

```

```
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("IsVisaDebit = " + receipt.getIsVisaDebit());

System.out.println("CfSuccess = " + receipt.getCfSuccess());
System.out.println("CfStatus = " + receipt.getCfStatus());
System.out.println("FeeAmount = " + receipt.getFeeAmount());
System.out.println("FeeRate = " + receipt.getFeeRate());
System.out.println("FeeType = " + receipt.getFeeType());
}
catch (Exception e)
{
e.printStackTrace();
}
}
```

## 12.5 Achat avec frais de commodité utilisant 3-D Secure

Les renseignements ci-dessous décrivent une demande de transaction d'achat avec 3-D Secure qui comprend également l'objet Convenience Fee Info.

### Définition de l'objet de transaction Convenience Fee Purchase with 3-D Secure

```
CavvPurchase cavvPurchase = new CavvPurchase();
```

### Objet HttpsPostRequest pour les transactions d'achat avec frais de commodité utilisant 3-D Secure

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
```

### Champs de demande liés aux transactions d'achat avec frais de commodité utilisant 3-D Secure (obligatoires)

Pour obtenir une description de toutes les valeurs obligatoires et facultatives, consultez l'annexe A Définition des champs de demande.

Variable	Type et limites	Méthode Set
Information sur les frais de commodité	<i>Objet</i> S. O.	cavvPurchase.setConvenienceFee( convFeeInfo);
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques a-Z A-Z 0-9 _ - : @ espaces	cavvPurchase.setOrderId(order_id);
Montant	<i>Chaîne</i> 10 caractères décimaux Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal  <b>EXEMPLE : 1 234 567,89</b>	cavvPurchase.setAmount(amount);
Numéro de carte de crédit	<i>Chaîne</i> Maximum 20 caractères alphanumériques	cavvPurchase.setPan(pan);

Variable	Type et limites	Méthode Set
Date d'expiration	<p><i>Chaîne</i></p> <p>4 caractères alphanumériques</p> <p>AAMM</p>	cavvPurchase.setExpDate(expiry_date);
Indicateur de commerce électronique	<p><i>Chaîne</i></p> <p>1 caractère alphanumérique</p>	cavvPurchase.setCryptType(crypt);
Code de vérification d'authentification du titulaire de carte	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p>	cavvPurchase.setCavv(cavv);
Montant des frais de commodité	<p><i>Chaîne</i></p> <p>9 caractères décimaux</p>	convFeeInfo.setConvenienceFee(convfee_amount);

### Champs de demande liés aux transactions d'achat avec frais de commodité utilisant 3- D Secure (facultatifs)

Variable	Type et limites	Méthode Set
Vérification d'état	<p><i>Valeur booléenne</i></p> <p>true/false</p>	mpgReq.setStatusCheck(status_check);
ID de client	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p> <p><b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés : &lt;&gt; \$ % = ? ^ { } [ ] \</p>	cavvPurchase.setCustId(cust_id);
Descripteur dynamique	<p><i>Chaîne</i></p> <p>20 caractères alphanumériques</p> <p>Total de 22 caractères incluant votre nom de</p>	cavvPurchase.setDynamicDescriptor(dynamic_descriptor);

Variable	Type et limites	Méthode Set
	<p>commerçant et un séparateur</p> <div style="background-color: #e0f2f1; padding: 10px;"> <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés :            &lt;&gt;\$%=?^{}[]\         </div>	
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	cavvPurchase.setCryptType(crypt);
Renseignements du client	<i>Objet</i> S. O.	cavvPurchase.setCustInfo(customer);
Renseignements du SVA	<i>Objet</i> S. O.	cavvPurchase.setAvsInfo(avsCheck);
Renseignements du NVC	<i>Objet</i> S. O.	cavvPurchase.setCvdInfo(cvdCheck);
<b>REMARQUE :</b> Lorsque les renseignements d'identification sont stockés lors de la première transaction, l'objet CVD (numéro de vérification de carte, ou NVC) doit être envoyé. Pour les transactions subséquentes utilisant les renseignements d'identification stockés, le NVC peut être envoyé avec les transactions entamées par le titulaire de carte seulement. <b>Les commerçants ne doivent pas enregistrer le NVC.</b>		

#### Exemple d'achat avec frais de commodité utilisant 3-D Secure

```

package Canada;
import JavaAPI.*;
public class TestCanadaConvFeeCavvPurchase
{
  public static void main(String[] args)
  {
    String store_id = "monca00392";
    String api_token = "qYdISUhHiOdfTr1CLNpN";
    java.util.Date createDate = new java.util.Date();
    String order_id = "Test"+createDate.getTime();
    String cust_id = "CUS887H67";
    String amount = "10.42";
    String pan = "4242424242424242";
    String expdate = "1901"; //YYMM
  }
}
  
```

```

String cavv = "AAABBJg0VhI0VniQEjRWAAAAAAA=";
String dynamic_descriptor = "123456";
String processing_country_code = "CA";
String crypt_type = "5";
boolean status_check = false;
/**************** Convenience Fee *****/
ConvFeeInfo convFeeInfo = new ConvFeeInfo();
convFeeInfo.setConvenienceFee("1.00");

CavvPurchase cavvPurchase = new CavvPurchase();
cavvPurchase.setOrderId(order_id);
cavvPurchase.setCustId(cust_id);
cavvPurchase.setAmount(amount);
cavvPurchase.setPan(pan);
cavvPurchase.setExpdate(expdate);
cavvPurchase.setCavv(cavv);
cavvPurchase.setCryptType(crypt_type); //Mandatory for AMEX only
cavvPurchase.setDynamicDescriptor(dynamic_descriptor);
cavvPurchase.setConvFeeInfo(convFeeInfo);
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(processing_country_code);
mpgReq.setTestMode(true); //false for production transactions
mpgReq.setstoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(cavvPurchase);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = " + receipt.getBankTotals());
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("CavvResultCode = " + receipt.getCavvResultCode());

System.out.println("CfSuccess = " + receipt.getCfSuccess());
System.out.println("CfStatus = " + receipt.getCfStatus());
System.out.println("FeeAmount = " + receipt.getFeeAmount());
System.out.println("FeeRate = " + receipt.getFeeRate());
System.out.println("FeeType = " + receipt.getFeeType());
}
catch (Exception e)
{
e.printStackTrace();
}
}
}
}

```

## 13 Facturation périodique

- 13.1 À propos de la facturation périodique
- 13.2 Achat avec la facturation périodique
- 1 Achat avec la facturation périodique
- 13.3 Mise à jour de la facturation périodique
- 1 Mise à jour de la facturation périodique
- 13.4 Codes et champs de réponse liés à la facturation périodique
- 13.5 Renseignements d'identification au dossier et facturation périodique

### 13.1 À propos de la facturation périodique

La facturation périodique vous permet de configurer des paiements que Moneris traite automatiquement et porte le montant de la transaction à la carte du commerçant en votre nom en fonction des renseignements du cycle de facturation que vous fournissez.

Les séries de facturation périodique sont créées en envoyant l'objet Recurring Billing durant ces transactions :

- Achat
- Achat avec la chambre forte
- Achat avec 3-D Secure (cavvPurchase)

Vous pouvez modifier une série de facturation périodique après l'avoir créée en effectuant une transaction administrative de mise à jour de la facturation périodique.

**REMARQUE :** Si vous préférez gérer vos séries de facturation périodique par l'entremise de votre propre système de commerçant, vous pouvez envoyer les paiements périodiques en tant que transactions d'achat de base en ajoutant la valeur 2 dans le champ d'indicateur de commerce électronique (`crypt_type`) ainsi qu'en incluant l'objet Credential on File Info.

### 13.2 Achat avec la facturation périodique

#### Définition de l'objet Recurring Billing Info

```
Recur recurring_cycle = new Recur(recur_unit, start_now, start_date,  
num_recur, period, recur_amount);  
  
Recur recurInfo = new Recur(recurUnit, startNow, recurStartDateStr, period,  
numRecurs, recurAmount);  
  
transaction.setRecur(recurInfo);
```

#### Méthode Set pour l'objet de transaction

```
<transaction>.setRecur(recurring_cycle);
```

## Champs de demande pour l'objet Recurring Billing Info

Variable	Type et limites	Description
Nombre d'occurrences	<i>Chaîne</i> Valeur numérique 1 ou 999	Il s'agit du nombre d'occurrences de la transaction.
Période	<i>Chaîne</i> Valeur numérique 1 ou 999	Il s'agit du nombre d'intervalles de la variable recur unit qui doivent s'écouler entre chaque facturation périodique.
Date de début	<i>Chaîne</i> Format AAAAMMMJJ	Il s'agit de la date de la première transaction périodique future (la date doit être future).  Si un montant additionnel est facturé immédiatement, la variable start now doit être réglée à true.
Commencer maintenant	<i>Chaîne</i> true/false	Réglez cette variable à true si un montant est porté immédiatement à la carte. Autrement, réglez la variable à false.  Lorsque la variable est réglée à false, effectuez une transaction de vérification de carte avant d'envoyer l'achat avec les objets Recurring Billing et Credential on File.  <b>REMARQUE :</b> Le montant à facturer immédiatement peut différer des montants subséquents.
Montant récurrent	<i>Chaîne</i> 10 caractères décimaux, minimum de 3 chiffres  Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal	Il s'agit du montant en dollars de la transaction périodique.  Il s'agit du montant facturé à la date de départ (start_date) et qui sera ensuite facturé à répétition en fonction de l'intervalle défini par les valeurs period et recur unit.
	<b>EXEMPLE : 1 234 567,89</b>	

Variable	Type et limites	Description
Unité répétée	<p><i>Chaîne</i></p> <p>Jour, semaine, mois ou fin du mois</p>	<p>Il s'agit de l'unité utilisée comme base pour l'intervalle.</p> <p>Elle fonctionne avec la variable period pour déterminer la fréquence de facturation.</p>

#### Exemple de transaction d'achat avec la facturation périodique

```

package Canada;

import java.util.*;
import JavaAPI.*;
public class TestCanadaPurchaseRecur
{
    public static void main(String[] args)
    {
        String store_id = "store5";
        String api_token = "yesguy";
        java.util.Date createDate = new java.util.Date();
        String order_id = "Test"+createDate.getTime();
        String amount = "10.00";
        String pan = "4242424242424242";
        String expiry_date = "1901"; //YYMM format
        String crypt = "7";
        /***** Recur Variables *****/
        String recur_unit = "month"; //eom = end of month
        String start_now = "true";
        String start_date = "2018/04/01";
        String num_recurr = "12";
        String period = "1";
        String recur_amount = "30.00";
        String processing_country_code = "CA";
        boolean status_check = false;
        /***** Recur Object Option1 *****/
        Recur recurring_cycle = new Recur(recur_unit, start_now, start_date,
        num_recurr, period, recur_amount);
        /***** Recur Object Option2 *****/
        Hashtable<String, String> recur_hash = new Hashtable<String, String>();
        recur_hash.put("recur_unit", recur_unit);
        recur_hash.put("start_now", start_now);
        recur_hash.put("start_date", start_date);
        recur_hash.put("num_recurr", num_recurr);
        recur_hash.put("period", period);
        recur_hash.put("recur_amount", recur_amount);
        /***** Transactional Object *****/
        Purchase purchase = new Purchase();
        purchase.setOrderId(order_id);
        purchase.setAmount(amount);
        purchase.setPan(pan);
        purchase.setExpdate(expiry_date);
        purchase.setCryptType(crypt);
        /***** Set Recur *****/
        purchase.setRecur(recurring_cycle);
        //Mandatory on Recurs - Credential on File details
        CofInfo cof = new CofInfo();
        cof.setPaymentIndicator("R");
        cof.setPaymentInformation("2");
        cof.setIssuerId("139X3130ASCXAS9");

        purchase.setCofInfo(cof);

        /***** Https Post Request *****/
        HttpsPostRequest mpgReq = new HttpsPostRequest();
        mpgReq.setProcCountryCode(processing_country_code);
        mpgReq.setTestMode(true); //false for production transactions
        mpgReq.setstoreId(store_id);
    }
}

```

```

mpgReq.setApiToken(api_token);
mpgReq.setTransaction(purchase);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
***** Receipt *****/
try
{
Receipt receipt = mpgReq.getReceipt();
System.out.println("CardType = " + receipt.getCardType());
System.out.println("TransAmount = " + receipt.getTransAmount());
System.out.println("TxnNumber = " + receipt.getTxnNumber());
System.out.println("ReceiptId = " + receipt.getReceiptId());
System.out.println("TransType = " + receipt.getTransType());
System.out.println("ReferenceNum = " + receipt.getReferenceNum());
System.out.println("ResponseCode = " + receipt.getResponseCode());
System.out.println("ISO = " + receipt.getISO());
System.out.println("BankTotals = " + receipt.getBankTotals());
System.out.println("Message = " + receipt.getMessage());
System.out.println("AuthCode = " + receipt.getAuthCode());
System.out.println("Complete = " + receipt.getComplete());
System.out.println("TransDate = " + receipt.getTransDate());
System.out.println("TransTime = " + receipt.getTransTime());
System.out.println("Ticket = " + receipt.getTicket());
System.out.println("TimedOut = " + receipt.getTimedOut());
System.out.println("Recur Success = " + receipt.getRecurSuccess());
System.out.println("IsVisaDebit = " + receipt.getIsVisaDebit());
System.out.println("IssuerId = " + receipt.getIssuerId());
}
catch (Exception e)
{
e.printStackTrace();
}
}
}

```

### 13.3 Mise à jour de la facturation périodique

Après avoir configuré une série de transactions de facturation périodique, vous pouvez modifier certains des renseignements de cette série tant et aussi longtemps que le nombre d'occurrences prédéfini n'est pas écoulé.

Avant d'effectuer une transaction de mise à jour de facturation périodique afin de mettre à jour le numéro de carte de crédit, vous devez effectuer une demande de vérification de carte. Cette exigence ne s'applique pas si vous modifiez seulement le calendrier de paiement ou le montant.

**Éléments dont il faut tenir compte :**

Lorsque vous mettez à jour une facturation périodique, gardez en tête que la date de fin ne peut pas être plus de 10 ans dans le futur et que vous ne pouvez pas modifier la transaction pour qu'elle prenne fin aujourd'hui ou à une date antérieure.

### Définition de l'objet de transaction Recurring Billing Update

```
RecurUpdate recurUpdate = new RecurUpdate();
```

### Objet HttpsPostRequest pour les transactions de mise à jour d'une facturation périodique

```
HttpsPostRequest mpgReq = new HttpsPostRequest();

mpgReq.setTransaction(recurUpdate);
```

### Valeurs des transactions de mise à jour d'une facturation périodique

**Tableau 1 Mise à jour d'une facturation périodique – Champs de base requis**

Variable	Type et limites	Méthode Set
No de commande order_id	<i>Chaîne</i> 50 caractères alphanumériques	recurUpdate.setOrderId(order_id);

**Tableau 2 Mise à jour d'une facturation périodique – Champs de base facultatifs**

Variable	Type et limites	Méthode Set
ID du client cust_id	<i>Chaîne</i> 50 caractères alphanumériques	recurUpdate.setCustId(cust_id);
Numéro de carte de crédit NIB	<i>Chaîne</i> 20 caractères alphanumériques	recurUpdate.setPan(pan);
Date d'expiration expiry_date	<i>Chaîne</i> AAMM	recurUpdate.setExdate(expiry_date);

**Tableau 3 Mise à jour d'une facturation périodique – Champs liés à la facturation périodique requis**

Variable	Type et limites	Méthode Set	Description
Méthode Set	<i>Chaîne</i>	recurUpdate.setRecurAmount(recur_amount);	Cette variable modifie le montant facturé à répétition.  Le changement entrera en vigueur lors de la prochaine facturation.

Variable	Type et limites	Méthode Set	Description
Ajout du nombre d'occurrences add_num	<i>Chaîne</i>  Valeur numérique, entre 1 et 999	<code>recurUpdate.setAddNumRecurs (add_num);</code>	Cette variable <b>ajoute</b> d'autres transactions au nombre actuel de transactions restantes.  Ceci est utile lorsqu'un client décide de prolonger un abonnement.  Cette variable ne peut pas être utilisée pour diminuer le nombre de transactions périodiques restantes. Vous devez plutôt utiliser la variable Changement du nombre d'occurrences.
Changement du nombre d'occurrences total_num	<i>Chaîne</i>  Valeur numérique, entre 1 et 999	<code>recurUpdate.setTotalNumRecurs (total_num);</code>	Cette variable <b>remplace</b> le nombre actuel de transactions périodiques restantes.
Suspendre la facturation périodique hold	<i>Chaîne</i>  true/false	<code>recurUpdate.setHold(hold);</code>	Cette variable met temporairement une transaction périodique en pause.  Lorsqu'une transaction est en pause, le montant récurrent n'est pas facturé. Cependant, le nombre de transactions restantes continue de diminuer durant cette période.
Arrêter la facturation récurrente terminate	<i>Chaîne</i>  true/false	<code>recurUpdate.setTerminate(terminate);</code>	Cette variable met fin à une transaction périodique.  <b>REMARQUE :</b> Lorsque vous mettez fin à une transaction périodique, celle-ci <b>ne peut pas</b> être réactivée. Une nouvelle transaction d'achat avec facturation périodique doit plutôt être effectuée.

### Exemple de transaction de mise à jour d'une facturation périodique

```

package Canada;
import JavaAPI.*;
public class TestCanadaRecurUpdate
{
    public static void main(String[] args)
    {
        String store_id = "store5";
        String api_token = "yesguy";
        String order_id = "Test155409282";
        String cust_id = "antonio";
        String recur_amount = "1.50";
        String pan = "4242424242424242";
        String expiry_date = "1902";
        //String add_num = "";
        //String total_num = "";
        //String hold = "";
        //String terminate = "";
        String processing_country_code = "CA";
        boolean status_check = false;
        //Credential on File details
        CofInfo cof = new CofInfo();
        cof.setIssuerId("139X3130ASCXAS9");

        RecurUpdate recurUpdate = new RecurUpdate();
        recurUpdate.setOrderId(order_id);
        recurUpdate.setCustId(cust_id);
        recurUpdate.setRecurAmount(recur_amount);
        recurUpdate.setPan(pan);
        recurUpdate.setExpydate(expiry_date);
        //recurUpdate.setAddNumRecurs(add_num);
        //recurUpdate.setTotalNumRecurs(total_num);
        //recurUpdate.setHold(hold);
        //recurUpdate.setTerminate(terminate);
        recurUpdate.setCofInfo(cof);
        HttpsPostRequest mpgReq = new HttpsPostRequest();
        mpgReq.setProcCountryCode(processing_country_code);
        mpgReq.setTestMode(true); //false for production transactions
        mpgReq.setstoreId(store_id);
        mpgReq.setApiToken(api_token);
        mpgReq.setTransaction(recurUpdate);
        mpgReq.setStatusCheck(status_check);
        mpgReq.send();
        try
        {
            Receipt receipt = mpgReq.getReceipt();
            System.out.println("ReceiptId = " + receipt.getReceiptId());
            System.out.println("ResponseCode = " + receipt.getResponseCode());
            System.out.println("Message = " + receipt.getMessage());
            System.out.println("Complete = " + receipt.getComplete());
            System.out.println("TransDate = " + receipt.getTransDate());
            System.out.println("TransTime = " + receipt.getTransTime());
            System.out.println("TimedOut = " + receipt.getTimedOut());
            System.out.println("RecurUpdateSuccess = " + receipt.getRecurUpdateSuccess());
            System.out.println("NextRecurDate = " + receipt.getNextRecurDate());
            System.out.println("RecurEndDate = " + receipt.getRecurEndDate());
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}

```

## 13.4 Codes et champs de réponse liés à la facturation périodique

Le tableau 19 décrit les champs de réponse liés à la facturation périodique. Certains champs apparaissent lorsque vous configurez une transaction périodique (notamment lors d'une transaction d'achat), alors que d'autres apparaissent seulement lorsque vous mettez à jour une transaction existante en y ajoutant la facturation périodique.

### Définition de l'objet Receipt

```
Receipt receipt = mpgReq.GetReceipt();
```

**Tableau 19 : Champs de réponse liés à la facturation périodique**

Valeur	Type	Limites		Méthode Get
		Description		
<b>Objet de transaction avec les champs de réponse liés à la facturation périodique</b>				
Code de réponse	Chaîne	3 caractère numérique	receipt.getResponseCode () ;	Consultez le tableau 20 pour obtenir une description des codes de réponse possibles.
	Indiquez le code de réponse de la transaction.			
Réussite de la répétition	Chaîne	À confirmer	receipt.getRecurSuccess () ;	Indique si la transaction a été correctement enregistrée.
	Indique si la transaction a été correctement enregistrée.			
<b>Champs de réponse liés à l'objet Recur update</b>				
Réussite de la mise à jour de la facturation périodique	Chaîne	true/false	receipt.getRecurUpdateSuccess () ;	Indique si la transaction a été correctement mise à jour.
	Indique si la transaction a été correctement mise à jour.			
Prochaine date de l'occurrence	Chaîne	Format aaaa-mm-jj	receipt.getNextRecurDate () ;	Indique lorsque la transaction sera de nouveau facturée.
	Indique lorsque la transaction sera de nouveau facturée.			
Date de fin de la répétition	Chaîne	Format aaaa-mm-jj	receipt.getRecurEndDate () ;	Indique lorsque la transaction de facturation périodique prendra fin.
	Indique lorsque la transaction de facturation périodique prendra fin.			

La réponse Recur Update est une valeur numérique à trois chiffres. La liste suivante comprend toutes les réponses possibles lorsqu'une transaction de mise à jour de la facturation périodique a été envoyée.

**Tableau 20 : Codes de réponse liés à la mise à jour d'une transaction périodique**

Valeur de la demande	Définition
000	La transaction a été correctement mise à jour.

**Tableau 20 : Codes de réponse liés à la mise à jour d'une transaction périodique**

Valeur de la demande	Définition
001	La transaction périodique a bien été mise à jour (facultatif : terminée)
983	Impossible de trouver la transaction précédente
984	Erreur de données : (facultatif : nom du champ)
985	Nombre d'occurrences non valide
986	Transaction incomplète : délai écoulé
null	Erreur : XML mal formé

## 13.5 Renseignements d'identification au dossier et facturation périodique

**REMARQUE :** La valeur du champ **payment indicator** doit être **R** lors de l'envoi de transactions de facturation périodique.

Pour les transactions de facturation périodique qui commencent **immédiatement** :

1. Envoyez une demande de transaction d'achat en incluant les objets Recurring Billing et Credential on File Info. Assurez-vous que le champ **start now** de l'objet Recurring Billing indique « true ».

Pour les transactions de facturation périodique qui commencent à une date **ultérieure** :

1. Envoyez une transaction de vérification de carte incluant l'objet Credential on File Info afin d'obtenir l'ID de l'émetteur (issuer ID).
2. Envoyez une demande de transaction d'achat en incluant les objets Recur et Credential on File Info.

Pour mettre à jour le numéro de carte de crédit d'une série de transactions de facturation périodique (ne s'applique pas si vous modifiez seulement le calendrier de paiement ou le montant de cette série de transactions) :

1. Envoyez une demande de transaction de vérification de carte incluant l'objet Credential on File Info afin d'obtenir l'ID de l'émetteur (issuer ID).
2. Envoyez une transaction de mise à jour de facturation périodique.

Pour plus de renseignements sur l'objet Recurring Billing, consultez la section Définition des champs de demande – Facturation périodique (recurring).

## 14 Renseignements du client

- 14.1 Utiliser l'objet Customer Information
- 14.2 Exemple d'une transaction incluant l'objet Customer Information

L'objet Customer Information permet d'inclure un certain nombre de champs lors d'une transaction financière, et ces renseignements sont enregistrés par Moneris. Ces renseignements peuvent être consultés ultérieurement dans le centre de ressources pour commerçants.

Les transactions suivantes peuvent inclure l'objet Customer Information

- Achat (de base , par Débit *Interac* et avec la chambre forte)
- Préautorisation (de base et avec la chambre forte)
- Réautorisation (de base)

L'objet Customer Information contient trois types de renseignements :

- des renseignements de facturation et d'expédition;
- des propriétés diverses au sujet du client;
- des renseignements sur les articles.

### Éléments dont il faut tenir compte :

- Si vous envoyez des caractères non inclus dans la liste de caractère autorisé, il est possible que ces renseignements supplémentaires ne soient pas enregistrés.
  - Tous les champs sont alphanumériques, et les caractères suivants sont autorisés : a-z A-Z 0-9 \_ - : . @ \$ = /
  - Tous les accents français doivent être codés en tant qu'entité HTML, comme &acute.
- Les données incluses dans les champs Billing et Shipping Address ne seront pas utilisées à des fins de vérification d'adresse.

### 14.1 Utiliser l'objet Customer Information

- 15.1.1 Objet Customer Info – Propriétés diverses
- 15.1.2 Objet Customer Info – Renseignements de facturation et d'expédition
- 15.1.3 Objet Customer Info – Information sur les articles

En plus d'instancier un objet de transaction et un objet de connexion (comme vous le feriez pour une transaction normale), vous devez instancier un objet CustInfo.

Toute transaction prenant en charge l'objet CustInfo a une méthode setCustInfo. Elle est utilisée pour écrire les renseignements sur le client dans l'objet de transaction avant d'écrire l'objet de transaction dans l'objet de connexion.

### Définition de l'objet CustInfo

```
CustInfo customer = new CustInfo();
```

### Méthode Set pour l'objet de transaction

```
<transaction>.setCustInfo(customer);
```

#### 14.1.1 Objet Customer Info – Propriétés diverses

Bien que la majorité des données concernant les renseignements du client est répartie en objet, certaines valeurs sont des propriétés de l'objet CustInfo lui-même. Elles sont expliquées dans le tableau ci-dessous.

Tableau 21 : Propriétés diverses de l'objet CustInfo

Valeur	Type	Limites	Méthode Set
Adresse courriel	Chaîne	60 caractères alphanumériques	customer.setEmail(email);
Instructions	Chaîne	100 caractères alphanumériques	customer.setInstructions(note);

#### 14.1.2 Objet Customer Info – Renseignements de facturation et d'expédition

Les renseignements de facturation et d'expédition sont enregistrés dans le cadre de l'objet Customer Information. Ils peuvent être ajouté à l'objet de deux façons :

- les méthodes Set;
- les tables de hachage.

Peu importe la méthode choisie, vous écrirez les renseignements inclus dans le tableau ci-dessous pour les renseignements de facturation et les renseignements d'expédition.

Toutes les valeurs sont des chaînes alphanumériques. Leur longueur maximale est indiquée dans la colonne Limite.

Tableau 22 : Valeurs liées aux renseignements de facturation et d'expédition

Valeur	Limite	Clé de la table de hachage
Prénom	30	"first_name"
Nom de famille	30	"last_name"
Nom de l'entreprise	En 50	"company_name"

Valeur	Limite	Clé de la table de hachage
Adresse	En 70	"address"
Ville	30	"city"
Province/État	30	"province"
Code postal ou ZIP	30	"postal_code"
Pays	30	"country"
Numéro de téléphone (vocal)	30	"phone"
Numéro de télécopieur	30	"fax"
Taxe fédérale	10	"tax1"
Taxe provinciale	10	"tax2"
Taxe locale ou spéciale	10	"tax3"
Frais d'expédition	10	"shipping_cost"

#### 14.1.2.1 Méthodes Set pour les renseignements d'expédition et de facturation

Les renseignements de facturation et d'expédition pour un objet CustInfo donné sont écrits au moyen des méthodes `customer.setBilling()` et `customer.setShipping()`, respectivement :

```
customer.setBilling(first_name, last_name, company_name, address, city,
province, postal_code, country, phone, fax, tax1, tax2, tax3, shipping_cost);

customer.setShipping(first_name, last_name, company_name, address, city,
province, postal_code, country, phone, fax, tax1, tax2, tax3, shipping_cost);
```

Ces deux méthodes ont le même ensemble d'arguments obligatoires. Elles sont décrites dans le tableau des valeurs de facturation et d'expédition à la section 14.1.2.1 Méthodes Set pour les renseignements d'expédition et de facturation.

Pour obtenir un exemple de code, consultez la section 14.2 Exemple d'une transaction incluant l'objet Customer Information.

### 14.1.2.2 Utiliser les tables de hachage pour les renseignements de facturation et d'expédition

Vous pouvez ajouter les renseignements de facturation ou d'expédition au moyen de tables de hachage, comme suit :

1. Instanciez un objet CustInfo.
2. Instanciez un objet de table de hachage. (L'exemple de code utilise une table de hachage différente pour la facturation et l'expédition à des fins de clarté. Cependant, un développeur habile pourra réutiliser la même.)
3. Créez la table de hachage au moyen de méthodes Put et des clés de table de hachage indiquées dans le tableau Valeurs liées aux renseignements de facturation et d'expédition qui se trouve dans la section 14.1.2 Objet Customer Info – Renseignements de facturation et d'expédition.
4. Appelez la méthode setBilling/setShipping de l'objet CustInfo afin de transférer les renseignements de la table de hachage à l'objet CustInfo.
5. Appelez la méthode setCustInfo de l'objet de transaction afin d'écrire l'objet CustInfo (en ajoutant les renseignements de facturation et d'expédition à l'objet de transaction).

Pour obtenir un exemple de code, consultez la section 14.2 Exemple d'une transaction incluant l'objet Customer Information.

### 14.1.3 Objet Customer Info – Information sur les articles

L'objet Customer Information peut contenir des renseignements au sujet de plusieurs articles. Les valeurs indiquées dans le tableau ci-dessous peuvent être ajoutées à chaque article.

Toutes les valeurs sont des chaînes, mais notez les directives précisées dans la colonne Limites.

**Tableau 23 : Valeurs liées aux renseignements sur les articles**

Valeur	Limites	Clé de la table de hachage
Nom de l'article	45 caractères alphanumériques	"name"
Nombre d'articles	5 caractère numérique	"quantity"
Code de produit de l'article	20 caractères alphanumériques	"product_code"
Montant final de l'article	9 caractères décimaux composée d'au moins 3 chiffres et de 2 cents  De 0.01 à 999 999,99	"extended_amount"

Une façon de représenter plusieurs articles est d'utiliser quatre tableaux. Il s'agit de la méthode utilisée dans l'exemple de code. Cependant, il existe deux façons d'ajouter les renseignements de l'article à l'objet CustInfo :

- la méthode Set;
- les tables de hachage.

#### 14.1.3.1 Méthodes Set pour les renseignements sur les articles

Tous les renseignements sur l'article qui se trouvent dans le tableau Valeurs liées aux renseignements sur les articles de la section 14.1.3 Objet Customer Info – Information sur les articles sont ajoutés à l'objet CustInfo en une seule instruction pour un article donné. Par exemple :

```
customer.setItem(item_description, item_quantity, item_product_code,
item_extended_amount);
```

Pour voir un exemple de code montrant comment utiliser des tableaux pour écrire des renseignements au sujet de deux articles, consultez la section 14.2 Exemple d'une transaction incluant l'objet Customer Information.

#### 14.1.3.2 Utiliser les tables de hachage pour les renseignements sur les articles

Vous pouvez ajouter les renseignements sur les articles au moyen de tables de hachage comme suit :

6. Instanciez un objet CustInfo.
7. Instanciez un objet de table de hachage. (L'exemple de code utilise une table de hachage différente pour chaque article à des fins de clarté. Cependant, un développeur habile pourra réutiliser la même.)
8. Créez la table de hachage au moyen de méthodesPut et des clés de table de hachage indiquées dans le tableau Valeurs liées aux renseignements sur les articles qui se trouve dans la section 14.1.3 Objet Customer Info – Information sur les articles.
9. Appelez la méthode setItem de l'objet CustInfo afin de transférer les renseignements de la table de hachage à l'objet CustInfo.
10. Appelez la méthode setCustInfo de l'objet de transaction afin d'écrire l'objet CustInfo (en ajoutant les renseignements sur les articles à l'objet de transaction).

Pour voir un exemple de code montrant comment utiliser des tableaux pour écrire des renseignements au sujet de deux articles, consultez la section 14.2 Exemple d'une transaction incluant l'objet Customer Information.

### 14.2 Exemple d'une transaction incluant l'objet Customer Information

Vous trouverez ci-dessous deux exemples d'une transaction d'achat de base incluant les renseignements sur le client. Les deux exemples commencent avec la même déclaration de variables.

Les valeurs qui ne sont pas incluses dans la fonction Customer Information ne sont pas affichées.

Veuillez noter que les deux articles commandés sont représentés par quatre tableaux, et les renseignements de facturation et d'expédition sont les mêmes.

#### Déclaration des variables (identique pour les deux méthodes)

```
***** Billing/Shipping Variables *****
String first_name = "Bob";
String last_name = "Smith";
String company_name = "ProLine Inc.";
String address = "623 Bears Ave";
String city = "Chicago";
String province = "Illinois";
String postal_code = "M1M2M1";
String country = "Canada";
String phone = "777-999-7777";
String fax = "777-999-7778";
```

```

String tax1 = "10.00";
String tax2 = "5.78";
String tax3 = "4.56";
String shipping_cost = "10.00";

***** Order Line Item Variables *****
String[] item_description = new String[] { "Chicago Bears Helmet", "Soldier Field Poster" };
String[] item_quantity = new String[] { "1", "1" };
String[] item_product_code = new String[] { "CB3450", "SF998S" };
String[] item_extended_amount = new String[] { "150.00", "19.79" };
*****
```

### Exemple d'une transaction d'achat incluant les renseignements du client – Méthode Set

```

CustInfo customer = new CustInfo();

***** Miscellaneous Customer Information Methods *****
customer.setEmail("nick@widget.com");
customer.setInstructions("Make it fast!");

***** Set Customer Billing Information *****
customer.setBilling(first_name, last_name, company_name, address, city, province, postal_code,
country, phone, fax, tax1, tax2, tax3, shipping_cost);

***** Set Customer Shipping Information *****
customer.setShipping(first_name, last_name, company_name, address, city, province,
postal_code, country, phone, fax, tax1, tax2, tax3, shipping_cost);

***** Order Line Items *****
customer.setItem(item_description[0], item_quantity[0], item_product_code[0],
item_extended_amount[0]);
customer.setItem(item_description[1], item_quantity[1], item_product_code[1],
item_extended_amount[1]);

Purchase purchase = new Purchase();
purchase.setCustInfo(customer);

HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setTransaction(purchase);
mpgReq.send();
```

```

CustInfo customer = new CustInfo();

***** Miscellaneous Customer Information Methods *****
customer.setEmail("nick@widget.com");
customer.setInstructions("Make it fast!");

***** Set Customer Billing Information *****
customer.setBilling(first_name, last_name, company_name, address, city, province, postal_code,
country, phone, fax, tax1, tax2, tax3, shipping_cost);

***** Set Customer Shipping Information *****
customer.setShipping(first_name, last_name, company_name, address, city, province,
postal_code, country, phone, fax, tax1, tax2, tax3, shipping_cost);

***** Order Line Items *****
customer.setItem(item_description[0], item_quantity[0], item_product_code[0],
item_extended_amount[0]);
customer.setItem(item_description[1], item_quantity[1], item_product_code[1],
item_extended_amount[1]);

Purchase purchase = new Purchase();
purchase.setCustInfo(customer);

HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setTransaction(purchase);
mpgReq.send();
```

## Exemple d'une transaction d'achat incluant les renseignements du client – Table de hachage

```
CustInfo customer2 = new CustInfo();
/****************** Miscellaneous Customer Information Methods *****/
customer.setEmail("nick@widget.com");
customer.setInstructions("Make it fast!");
/****************** Billing Hashtable *****/
Hashtable<String, String> b = new Hashtable<String, String>(); //billing hashtable
b.put("first_name", first_name);
b.put("last_name", last_name);
b.put("company_name", company_name);
b.put("address", address);
b.put("city", city);
b.put("province", province);
b.put("postal_code", postal_code);
b.put("country", country);
b.put("phone", phone);
b.put("fax", fax);
b.put("tax1", tax1); //federal tax
b.put("tax2", tax2); //prov tax
b.put("tax3", tax3); //luxury tax
b.put("shipping_cost", shipping_cost); //shipping cost
customer2.setBilling(b);
/****************** Shipping Hashtable *****/
Hashtable<String, String> s = new Hashtable<String, String>(); //shipping hashtable
s.put("first_name", first_name);
s.put("last_name", last_name);
s.put("company_name", company_name);
s.put("address", address);
s.put("city", city);
s.put("province", province);
s.put("postal_code", postal_code);
s.put("country", country);
s.put("phone", phone);
s.put("fax", fax);
s.put("tax1", tax1); //federal tax
s.put("tax2", tax2); //prov tax
s.put("tax3", tax3); //luxury tax
s.put("shipping_cost", shipping_cost); //shipping cost
customer2.setShipping(s);
/****************** Order Line Item1 Hashtable *****/
Hashtable<String, String> i1 = new Hashtable<String, String>(); //item hashtable #1
i1.put("name", item_description[0]);
i1.put("quantity", item_quantity[0]);
i1.put("product_code", item_product_code[0]);
i1.put("extended_amount", item_extended_amount[0]);
customer2.setItem(i1);
/****************** Order Line Item2 Hashtable *****/
Hashtable<String, String> i2 = new Hashtable<String, String>(); //item hashtable #2
i2.put("name", "item2's name");
i2.put("quantity", "7");
i2.put("product_code", "item2's product code");
i2.put("extended_amount", "5.01");
customer2.setItem(i2);

Purchase purchase = new Purchase();
purchase.setCustInfo(customer);
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setTransaction(purchase);
mpgReq.send();
```

## 15 Vérification d'état

- 15.1 À propos de la vérification d'état
- 15.2 Utiliser les champs de réponse liés à la vérification d'état
- 15.3 Exemple d'achat avec la vérification d'état

### 15.1 À propos de la vérification d'état

La valeur d'objet de connexion Status Check (vérification d'état) permet aux commerçants de vérifier si une transaction passée a été traitée correctement.

Pour procéder à une vérification d'état, renvoyez la transaction d'origine avec les mêmes paramètres, mais réglez la valeur status check à `true` ou `false`.

Lorsque la valeur est réglée à « `true` », la passerelle vérifiera l'état d'une transaction dont l'ID de commande (`order_id`) correspond à celui de la transaction actuelle.

- Si une transaction est trouvée, la passerelle renverra les renseignements sur cette transaction.
- Si aucune transaction n'est trouvée, la passerelle répondra avec un message indiquant qu'aucune transaction n'a été trouvée.

Lorsque la valeur est réglée à « `false` », la transaction sera traitée comme une nouvelle transaction.

Par exemple, si vous envoyez une transaction d'achat en demandant une vérification d'état, incluez les mêmes valeurs que celles de la transaction d'origine, y compris l'ID de commande et le montant.

Pour pouvoir utiliser cette fonction, celle-ci doit être activée dans votre profil de commerçant. Pour ce faire, communiquez avec Moneris.

#### Éléments dont il faut tenir compte :

- La demande de vérification d'état doit être utilisée une seule fois et immédiatement (dans les deux minutes) après l'échec d'une transaction.  
Cette demande ne doit pas être utilisée pour vérifier les demandes `openTotals` et `batchClose`.
- Ne renvoyez pas de demande de vérification d'état si le délai s'est écoulé. Une enquête supplémentaire est requise dans cette situation.

### 15.2 Utiliser les champs de réponse liés à la vérification d'état

Après avoir utilisé l'objet de connexion pour envoyer une demande de vérification d'état, vous pouvez utiliser l'objet Receipt pour obtenir l'information souhaitée au sujet de la réussite de la transaction d'origine.

Les champs de réponse concernant la vérification d'état sont `Status Code` et `Status Message`.

Valeurs de réponse Status Code possibles :

- 0-49 = transaction réussie
- 50-999 = transaction échouée

Valeurs de réponse Status Message possibles :

- Trouvé : le code est 0-49
- Non trouvé ou nul : le code est 50-999

Si le message d'état est `Found`, tous les autres champs de réponse sont identiques à ceux de la transaction d'origine.

Si le message d'état est `Not found`, tous les autres champs de réponse seront `Null`.

### 15.3 Exemple d'achat avec la vérification d'état

#### Exemple d'une transaction d'achat avec une vérification d'état

```
package Canada;
import JavaAPI.*;
public class TestCanadaPurchase
{
    public static void main(String[] args)
    {
        Purchase purchase = new Purchase();
        purchase.setOrderId("order");
        purchase.setAmount("1.00");
        purchase.setPan("4242424242424242");
        purchase.setExdate("2202");
        purchase.setCryptType("1");

        HttpsPostRequest mpgReq = new HttpsPostRequest();
        mpgReq.setProcCountryCode("CA");
        mpgReq.setTestMode(true); //false for production transactions
        mpgReq.setstoreId("store1");
        mpgReq.setApiToken("yesguy");
        mpgReq.setTransaction(purchase);
        boolean status_check = true;
        mpgReq.setStatusCheck(status_check);
        mpgReq.send();

        try
        {
            Receipt receipt = mpgReq.getReceipt();
            System.out.println("StatusCode = " + receipt.getStatusCode());
            System.out.println("StatusMessage = " + receipt.getStatusMessage());
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

## 16 Mise à l'essai d'une solution

- 16.1 À propos du centre de ressources pour commerçants
- 16.2 Se connecter à l'environnement d'assurance de la qualité du centre de ressources pour commerçants
- 16.3 Renseignements d'identification test du centre de ressources pour commerçants
- 16.4 Obtenir un ID de commerce et un jeton API uniques pour l'environnement de test
- 16.5 Traiter une transaction
- 16.7 Mise à l'essai d'un module d'extension pour les commerçants
- 16.8 Mise à l'essai de Visa Checkout
- 16.9 Cartes de test
- 16.10 Simulation du serveur de traitement

### 16.1 À propos du centre de ressources pour commerçants

Le centre de ressources pour commerçants est l'interface utilisateur des services de Passerelle Moneris. Il existe également une version d'assurance de la qualité du centre de ressources pour commerçants conçue spécialement pour que vous et d'autres développeurs puissiez tester vos intégrations API avec la passerelle.

Vous pouvez accéder à l'environnement de tests du centre de ressources pour commerçants à l'adresse suivante :

<https://esqa.moneris.com/mpg> (Canada)

L'environnement de test est généralement accessible en tout temps, mais nous ne garantissons pas une disponibilité à 100 %. De plus, veuillez noter que d'autres commerçants utilisent l'environnement de tests du centre de ressources pour commerçants. Par conséquent, il est possible que vous voyez des transactions et des ID d'utilisateur que vous n'aurez pas créés. Afin de respecter les autres commerçants utilisant l'environnement de test, nous vous demandons de travailler uniquement avec les transactions et les utilisateurs que vous avez créés. Ceci s'applique pour le traitement des transactions de remboursement, la modification des mots de passe ou l'essai d'autres fonctions.

### 16.2 Se connecter à l'environnement d'assurance de la qualité du centre de ressources pour commerçants

Pour vous connecter à l'environnement d'assurance de la qualité du centre de ressources pour commerçants afin de procéder à des tests :

1. Accédez à l'environnement d'assurance de la qualité du centre de ressources pour commerçants (<https://esqa.moneris.com/mpg>).
2. Entrez votre nom d'utilisateur et votre mot de passe, qui sont identiques à vos identifiants du portail pour développeurs.
3. Entrez votre ID de commerce, que vous pouvez obtenir en suivant les instructions décrites dans la section 16.3 Renseignements d'identification test du centre de ressources pour commerçants.

## 16.3 Renseignements d'identification test du centre de ressources pour commerçants

À des fins de test, vous pouvez soit utiliser les commerces tests existants du centre de ressources pour commerçants, soit créer votre propre commerce test où vous ne verrez que vos propres transactions. Si vous préférez utiliser les commerces déjà configurés, utilisez les identifiants tests indiqués dans les tableaux suivants ainsi que les lignes de code correspondantes, comme montré dans les exemples ci-dessous.

### Exemple de code correspondant pour le Canada

```
String processing_country_code = "CA";  
  
mpgReq.setTestMode(true);  
  
String store_id = "store5";  
  
String api_token = "yesguy";
```

**Tableau 24 : Identifiants du serveur de test – Canada**

store_id	api_token	Username	Password	Autre information
store1	yesguy	demouser	password	
store2	yesguy	demouser	password	
store3	yesguy	demouser	password	
store4	yesguy	demouser	password	
store5	yesguy	demouser	password	
monca00392	yesguy	demouser	password	Utilisez ce commerce pour tester des transactions liées à des frais de commodité.
moncaqagt1	mgtokenguy1	demouser	password	Utilisez ce commerce pour tester le partage de jetons.
moncaqagt2	mgtokenguy2	demouser	password	Utilisez ce commerce pour tester le partage de jetons.
moncaqagt3	mgtokenguy3	demouser	password	Utilisez ce commerce pour tester le partage de jetons.

Store_id	api_token	Username	Password	Autre information
monca01428	mcmpguy	demouser	password	Utilisez ce magasin pour tester la carte MasterCard MasterPass

Vous pouvez également créer et utiliser un commerce test où vous ne verrez que vos propres transactions. Pour en savoir plus à ce sujet, consultez la section Obtenir un ID de commerce et un jeton API uniques pour l'environnement de test (page 494)

## 16.4 Obtenir un ID de commerce et un jeton API uniques pour l'environnement de test

Pour envoyer des demandes de transaction par l'entremise de l'API, vous devez avoir un ID de commerce ainsi qu'un jeton API correspondant. Pour vos tests, vous pouvez utiliser les commerces test déjà configurés dans le centre de ressources pour commerçants ou vous pouvez créer votre propre commerce test, dans lequel vous verrez uniquement vos propres transactions.

Pour obtenir votre ID de commerce et votre jeton API uniques :

1. Connectez-vous au portail pour développeurs (<https://developer.moneris.com>).
2. Dans la boîte de dialogue My Profile, cliquez sur le bouton Full Profile.
3. Dans la section My Testing Credentials, sélectionnez Request Testing Credentials.
4. Entrez votre mot de passe du portail pour développeurs et sélectionnez votre pays.
5. Notez l'ID de commerce et le jeton API que vous recevez, car vous en aurez besoin pour vous connecter au centre de ressources pour commerçants (ID de commerce) et pour les demandes d'API (jeton API).

FULL PROFILE

Vous pouvez également utiliser les commerces test déjà configurés dans le centre de ressources pour commerçants, comme décrit dans la section Renseignements d'identification test du centre de ressources pour commerçants (page 493).

## 16.5 Traiter une transaction

- 1.1 Sommaire
- 1.2 Objet HttpsPostRequest
- 1.3 Objet Receipt

### 16.5.1 Sommaire

Il existe des étapes communes pour chaque transaction traitée.

1. Instanciez l'objet de transaction (par exemple, Achat) et mettez-le à jour en fonction des définitions d'objet faisant référence à la transaction individuelle.
2. Instanciez l'objet de connexion HttpsPostRequest et mettez-le à jour en fonction des renseignements de connexion, des renseignements sur le serveur et l'objet de transaction que vous avez créé à l'étape 16.5.

La section 16.5 (page 495) fournit la définition de l'objet de connexion HttpsPostRequest. Cet objet et ses variables s'appliquent à **chaque** demande de transaction.

3. Invoquez la méthode `send()` de l'objet HttpsPostRequest.
4. Instanciez l'objet Receipt, en invoquant la méthode `get Receipt` de l'objet HttpsPostRequest. Utilisez cet objet pour récupérer les détails de réponse applicables.

Certaines transactions peuvent nécessiter des étapes supplémentaires à celles énumérées ici. Vous trouverez ci-dessous un exemple de transaction d'achat avec une description de chaque étape importante. Pour des exemples de code détaillés d'autres types de transactions, consultez le fichier ZIP de l'API .NET.

**REMARQUE :** À des fins d'illustration, l'ordre dans lequel les lignes de code apparaissent ci-dessous peut différer légèrement du même exemple de code présenté ailleurs dans ce document.

```
import java.io.*;
import java.util.*;
import java.net.*;
import JavaAPI.*;
```

Inclure toutes les classes nécessaires

```
public class TestPurchase
{
public static void main(String args[])
{
    java.util.Date createDate = new
    java.util.Date();
```

```
String order_id =
"Test"+createDate.getTime();
String amount = "5.00";
String pan = "4242424242424242";
String expdate = "1901"; //YYMM format
String crypt = "7";
String processing_country_code = "CA";
```

Définir toutes les valeurs obligatoires pour les propriétés de l'objet de transaction

```
String store_id = "store5";
String api_token = "yesguy";
```

Définir toutes les valeurs obligatoires pour les propriétés de l'objet de connexion

<pre>Purchase purchase = new Purchase(); purchase.setOrderId(order_id); purchase.setAmount(amount); purchase.setPan(pan); purchase.setExpdate(expdate); purchase.setCryptType(crypt); purchase.setDynamicDescriptor("2134565");</pre>	<p>Instancier l'objet de transaction et attribuer des valeurs aux propriétés</p>
<pre>HttpsPostRequest mpgReq = new HttpsPostRequest();  mpgReq.setProcCountryCode(processing     _country_code); mpgReq.setTestMode(true); mpgReq.setstoreId(store_id); mpgReq.setApiToken(api_token); mpgReq.setTransaction(purchase); mpgReq.setStatusCheck(status_check);</pre>	<p>Instancier l'objet de connexion et attribuer des valeurs aux propriétés, y compris à l'objet de transaction que vous venez de créer</p>
<pre>mpgReq.send();  try {     Receipt receipt = mpgReq.getReceipt();</pre>	<p>Invoquer la méthode <code>send()</code> de l'objet de connexion</p>
<pre>System.out.println("CardType = " + receipt.getCardType()); System.out.println("TransAmount = " + receipt.getTransAmount()); System.out.println("TxnNumber = " + receipt.getTxnNumber()); System.out.println("ReceiptId = " + receipt.getReceiptId()); System.out.println("TransType = " + receipt.getTransType()); System.out.println("ReferenceNum = " + receipt.getReferenceNum()); System.out.println("ResponseCode = " + receipt.getResponseCode()); System.out.println("ISO = " + receipt.getISO()); System.out.println("BankTotals = " + receipt.getBankTotals()); System.out.println("Message = " + receipt.getMessage()); System.out.println("AuthCode = " + receipt.getAuthCode()); System.out.println("Complete = " + receipt.getComplete()); System.out.println("TransDate = " + receipt.getTransDate()); System.out.println("TransTime = " + receipt.getTransTime()); System.out.println("Ticket = " + receipt.getTicket()); System.out.println("TimedOut = " + receipt.getTimedOut()); System.out.println("IsVisaDebit = " + receipt.getIsVisaDebit()); }</pre>	<p>Instancier l'objet <code>Receipt</code> et utiliser ses méthodes <code>get</code> pour récupérer les données de réponse souhaitées</p>

## 16.5.2 Objet HttpsPostRequest

L'objet de transaction que vous instanciez devient une propriété de cet objet lorsque vous appelez sa méthode set de transaction.

### Définition de l'objet HttpsPostRequest

```
HttpsPostRequest mpgReq = new HttpsPostRequest();
```

Après avoir instancié l'objet HttpsPostRequest, mettez à jour ses valeurs obligatoires et facultatives, comme indiqué dans les tableaux de valeurs suivants.

**Tableau 25 : Valeurs obligatoires de l'objet HttpsPostRequest**

Valeur	Type	Limites	Méthode Set
	Description		
Code du pays de traitement	Chaîne	2 caractères alphabétiques	mpgReq.setProcCountryCode(processинг_country_code);
CA pour Canada, US pour États-Unis			
Mode de mise à 'essai (test mode)	Valeur booléenne	true/false	mpgReq.setTestMode(true);
Régler à true en mode de mise à l'essai Régler à false (ou couper la ligne entière) en mode production			
ID du commerce	Chaîne	10 caractères alphanumériques	mpgReq.setstoreId(store_id);
Identifiant unique fourni par Moneris lors de la création du compte de commerçant.			
Voir la section 16.1 À propos du centre de ressources pour commerçants pour obtenir plus de détails sur l'environnement de mise à l'essai			
Jeton API	Chaîne	20 caractères alphanumériques	mpgReq.setApiToken(api_token);
Chaîne de caractères alphanumériques unique attribuée au commerçant lors de l'activation de son compte Pour localiser votre jeton API de production, consultez les paramètres administratifs du centre de ressources pour commerçants.			
Consultez la section 16.3 Renseignements d'identification test du centre de ressources pour commerçants pour obtenir des détails sur l'environnement de mise en essai.			
Transaction	Objet	Non applicable	mpgReq.setTransaction(transaction);
Cet argument est l'un des nombreux types de transaction abordés dans le reste de ce manuel, comme les transactions d'achat, de remboursement, etc. Cet objet est instancié à l'étape 1 ci-dessus.			

**Tableau 1 Valeurs facultatives de l'objet HttpsPostRequest**

Valeur	Type	Limites	Méthode Set
			Description
Vérification d'état	Valeur booléenne	true/false	mpgReq.setStatusCheck(status_check);
	Consultez l'annexe A Définition des champs de demande.		
	<b>REMARQUE :</b> Bien que cette valeur appartienne à l'objet HttpsPostRequest, elle n'est prise en charge que par certaines transactions. Vérifiez la définition de chaque transaction pour savoir si la vérification de l'état peut être utilisée.		

### 16.5.3 Objet Receipt

Après avoir envoyé une transaction à l'aide de la méthode send de l'objet HttpsPostRequest, vous pouvez instancier un objet Receipt.

#### Définition de l'objet Receipt

```
Receipt receipt = mpgReq.getReceipt();
```

Pour obtenir une explication approfondie des méthodes et des propriétés de l'objet Receipt, consultez l'Annexe B Définitions des champs de réponse.

## 16.6 Mise à l'essai d'un module d'extension pour les commerçants

Pour tester votre mise en œuvre du module d'extension pour les commerçants de Moneris, vous pouvez utiliser l'environnement PIT (production integration testing) de Visa, MasterCard et Amex. Le processus de test est légèrement différent de celui d'un environnement de production, car lorsque la fenêtre en ligne est générée, elle ne contient aucune zone de saisie. Au lieu de cela, elle contient une fenêtre de données et un bouton **Submit**. En cliquant sur **Submit**, la réponse est chargée dans la fenêtre de test. La réponse ne sera pas affichée en production.

**REMARQUE :** Les solutions SecureCode de Mastercard et Amex SafeKey ne peuvent pas être testées directement dans notre environnement de test actuel. Cependant, le processus et le comportement testés avec les cartes de test Visa seront les mêmes que pour SecureCode de Mastercard et Amex SafeKey.

Lors du test, vous pouvez utiliser les numéros de carte de test suivants, avec n'importe quelle date d'expiration future. Utilisez les renseignements appropriés sur la carte test figurant dans les tableaux ci-dessous : Visa et Mastercard utilisent les mêmes renseignements de carte test, tandis qu'Amex utilise des renseignements uniques.

**Tableau 26 : Numéros de carte de test des modules d'extension pour les commerçants (Visa et Mastercard uniquement)**

Numéro de la carte	VERes	PARes	Action
4012001037141112 4242424242424242	Y	true	TXN = Fonction d'appel pour créer une fenêtre en ligne (inLine) ACS = Envoi du code de vérification d'authentification du titulaire de carte à Passerelle Moneris en se servant soit de la transaction CAVV Purchase, soit la transaction CAVV Pre-Authorization
4012001038488884	U	S. O.	Envoi de la transaction à Passerelle Moneris en se servant soit de la transaction d'achat ou de préautorisation de base Définir la valeur crypt_type = 7.
4012001038443335	N	S. O.	Envoi de la transaction à Passerelle Moneris en se servant soit de la transaction d'achat ou de préautorisation de base Définir la valeur crypt_type = 6.
4012001037461114	Y	false	La carte a échoué l'authentification. Le commerçant peut choisir d'envoyer la transaction ou de la refuser. Si la transaction est envoyée, utilisez la valeur crypt type = 7.

**Tableau 27 : Numéros de carte de test des modules d'extension pour les commerçants (Amex uniquement)**

Numéro de la carte	VERes	Mot de passe requis?	PARes	Action
375987000000062	U	Facultatif	S. O.	TXN = Fonction d'appel pour créer une fenêtre en ligne (inLine) ACS = Envoi du code de vérification d'authentification du titulaire de carte à Passerelle Moneris en se servant soit de la transaction CAVV Purchase ou CAVV Pre-Authorization. Définir la valeur crypt_type = 7.
375987000000021	Y	Oui : test13fail	false	La carte a échoué l'authentification. Le commerçant peut choisir d'envoyer la transaction ou de la refuser. Si la transaction est envoyée, utilisez la valeur crypt_type = 7.
En 375987000000013	N	Facultatif	S. O.	Envoi de la transaction à Passerelle Moneris en se servant soit de la transaction d'achat ou de préautorisation de base. Définir la valeur crypt_type = 6.
En 374500261001009	Y	Oui : test09	true	La carte a échoué l'authentification. Le commerçant peut choisir d'envoyer la transaction ou de la refuser. Définir la valeur crypt_type = 5.

**VERes**

Le résultat U, Y ou N est obtenu en utilisant getMessage().

**PARes**

Le résultat « true » ou « false » est obtenu en utilisant getSuccess().

Pour accéder au centre de ressources pour commerçants dans l'environnement de test, consultez la page <https://esqa.monteris.com/mpg>.

Les transactions de l'environnement de test ne doivent pas être supérieures à 11,00\$.

## 16.7 Numéro de carte de test

Pour des raisons de sécurité et de conformité, il est strictement interdit d'utiliser des numéros de carte de crédit et de débit réels lors des essais. Seuls les numéros de cartes de crédit et de débit de test doivent être utilisés.

Pour tester les transactions générales, utilisez les numéros de carte de test suivants :

Programme de cartes	Numéro de carte de test
Mastercard	5454545454545454
Visa	4242424242424242
Amex	373599005095005
JCB	3566007770015365
Diners	36462462742008
Track2	5258968987035454=06061015454001060101?
Discover	6011000992927602
UnionPay	6250944000000771

### 16.7.1 Numéro de carte de test pour les transactions de niveaux 2 et 3

Lors des tests de transactions de niveaux 2 et 3, utilisez les numéros de carte ci-dessous.

Marque de carte	Numéro de carte de test
Mastercard	En 5454545442424242
Visa	En 4242424254545454
Amex	En 373269005095005

### 16.7.2 Cartes de Test pour Visa Checkout

Programme de cartes	Numéro de carte de test
Visa	4005520201264821 (sans image de carte)
Visa	4242424242424242 (avec image de carte)

Mastercard	5500005555555559
American Express	340353278080900
Discover	6011003179988686

## 16.8 Simulation du serveur de traitement

L'environnement de tests a été conçu pour reproduire l'environnement de production le plus fidèlement possible. Une différence majeure est que Moneris ne peut pas envoyer de transactions test sur le réseau d'autorisation de production. Par conséquent, les réponses des émetteurs de carte sont simulées. De plus, afin d'émuler des situations d'approbation, de refus et d'erreur, certaines variables de transaction doivent déclencher diverses situations de réponse et d'erreur.

L'environnement de test approuve et refuse les transactions selon la valeur des cents du montant envoyé. Par exemple, une transaction effectuée pour un montant de 9,00 \$ ou 1,00 \$ sera approuvée en raison de la valeur des cents de ,00.

Les transactions effectuées dans l'environnement de test ne doivent pas dépasser 11,00 \$.

Pour consulter une liste de toutes les réponses actuelles de l'environnement de tests pour diverses valeurs de cents, consultez le tableau Test Environment Penny Response à <https://developer.monteris.com>.

**REMARQUE :** Ces réponses peuvent changer sans préavis. Consultez le portail pour développeurs de Moneris (<https://developer.monteris.com>, en anglais seulement) pour accéder aux plus récents documents et fichiers à télécharger.

## 17 Passage à l'environnement de production

- 17.1 Activation d'un compte de magasin dans l'environnement de production
- 17.2 Configuration d'un commerce pour l'environnement de production
- 17.3 Exigences de reçu
- 1 Obténir de l'aide

### 17.1 Activation d'un compte de magasin dans l'environnement de production

Les étapes ci-dessous indiquent comment activer votre compte de production afin de traiter les transactions dans l'environnement de production.

1. Obtenez votre lettre ou télécopie d'activation de Moneris.
2. Accédez à la page <https://www.moneris.com/activate>.
3. Entrez l'ID de votre commerce et l'ID commerçant figurant dans la lettre ou télécopie et cliquez sur **Activate**.
4. Suivez les instructions à l'écran pour créer un compte administrateur. Ce compte vous permettra d'accéder au centre de ressources pour commerçants.
5. Connectez-vous au centre de ressources pour commerçants à la page <https://www3.moneris.com/mpg> en utilisant les identifiants créés à l'étape 19.1.
6. Cliquez sur **ADMIN**, puis sur **STORE SETTINGS**.
7. Repérez le jeton API au haut de la page. Vous utiliserez ce jeton API ainsi que l'ID du commerce que vous avez reçu dans votre lettre ou télécopie pour transférer toutes les transactions de production par l'entremise de l'API.

Une fois votre commerce de production activé, vous devez le configurer pour qu'il pointe vers le serveur de production. Pour savoir comment procéder, consultez la section Configuration d'un commerce pour l'environnement de production (page 503)

**REMARQUE :** Pour plus de renseignements sur la façon d'utiliser le centre de ressources pour commerçants, consultez le guide de l'utilisateur du centre de ressources pour commerçants de Passerelle Moneris, qui se trouve sur le portail pour développeurs (<https://developer.moneris.com>).

### 17.2 Configuration d'un commerce pour l'environnement de production

Après avoir terminé votre mise à l'essai et avoir activé votre commerce de production, vous êtes prêt à connecter votre commerce au serveur de production.

Configurer votre commerce pour l'environnement de production :

1. Changez le mode de mise à l'essai de la méthode Set de `true` à `false`.

2. Changez l'ID de commerce afin de refléter l'ID de commerce de production que vous avez reçu lorsque vous avez activé votre commerce de production. Pour passer en revue les étapes d'activation d'un commerce de production, consultez la section Activation d'un compte de magasin dans l'environnement de production(page 503)[Activation d'un compte de magasin dans l'environnement de](#).
3. Remplacez le jeton API par celui utilisé pour la production, que vous avez reçu durant l'activation
4. Si vous ne l'avez pas encore fait, changez le code pour refléter le bon pays de traitement (Canada, pour la majorité des commerçants). Pour en savoir plus à ce sujet

Le tableau ci-dessous illustre les étapes ci-dessus en utilisant les codes pertinents (et où **x** est un caractère alphanumérique).

Étape	Code de test	Changements dans l'environnement de production
1	Aucun changement apporté à une chaîne pour cet article, seule la méthode Set est modifiée : <code>mpgReq.setTestMode(true);</code>	Méthode Set pour l'environnement de production <code>mpgReq.setTestMode(<b>false</b>);</code>
2	Chaîne :  <code>String store_id = "store5";</code>  Méthode Set associée :  <code>mpgReq.setstoreId(store_id);</code>	Chaîne pour l'environnement de production :  <code>String store_id = "<b>monXXXXXXXXX</b>";</code>
3	Chaîne :  <code>String api_token = "yesguy";</code>  Méthode Set associée :  <code>mpgReq.setApiToken(api_token);</code>	Chaîne pour l'environnement de production :  <code>String api_token = "<b>xxxx</b>";</code>

### 17.3 Exigences relatives aux reçus

Visa et Mastercard s'attendent à ce que certains détails soient fournis au titulaire de la carte et figurent sur le reçu lorsqu'une transaction est approuvée.

Les reçus doivent être conformes aux normes décrites dans les exigences relatives à l'intégration des reçus. Pour connaître toutes les exigences relatives aux reçus couvrant tous les scénarios de transaction, consultez le portail pour développeurs de Moneris à la page <https://developer.moneris.com>.

La production du reçu doit commencer lorsque la réponse appropriée à la demande de transaction est reçue par l'application. La transaction peut être l'une des suivantes :

- **Vente** (Achat)

- **Autorisation** (Préautorisation)
- **Conclusion d'autorisation** (Conclusion)
- **Vente hors ligne** (Transaction forcée)
- **Annulation de vente** (Correction d'achat, annulation)
- **Remboursement**.

Les termes en caractères gras énumérés ci-dessus sont les noms des transactions tels qu'ils doivent être affichés sur les reçus. Les autres termes utilisés pour la transaction sont indiqués entre parenthèses.

### 17.3.1 Exigences de certification

Les reçus de transactions avec carte présente sont nécessaires pour compléter la certification.

#### Intégration des transactions avec carte absente

La certification est facultative, mais fortement recommandée.

#### Intégration des transactions avec carte présente

Après avoir effectué le développement et les essais, votre application doit suivre un processus de certification au cours duquel tous les types de transactions applicables doivent être démontrés et les reçus correspondants doivent être générés.

Communiquez avec notre centre d'assistance du soutien à l'intégration afin d'obtenir la liste de vérification des essais pour la certification qui doit être remplie et renvoyée aux fins de vérification. (Consultez la section Obtenir de l'aide à la page 14 pour connaître les coordonnées des personnes à contacter.) Assurez-vous d'inclure la version de l'application de votre produit. Toute modification apportée au produit après la certification nécessite une nouvelle certification.

Une fois que les exigences de certification sont satisfaites, Moneris vous remettra une lettre de certification officielle.

## Annexe A Définition des champs de demande

Cette section définit les variables de demande de transaction. Certains champs ne sont pas obligatoires, consultez les rubriques relatives aux différents types de transactions pour savoir si un champ est obligatoire ou facultatif.

- A.1 Définition des champs de demande – Champs de connexion
- A.2 Définition des champs de demande – Champs de base
- A.3 Définition des champs de demande – Renseignements d'identification au dossier
- A.4 Définition des champs de demande – Chambre forte
- A.5 Définition des champs de demande – Transactions de niveaux 2 et 3 de Visa
- A.6 Définition des champs de demande – Transactions de niveaux 2 et 3 de Mastercard
- A.7 Définition des champs de demande – Transactions de niveaux 2 et 3 d'Amex
- Annexe A Définition des champs de réponse – Modules d'extension pour les commerçants
- A.9 Définition des champs de demande – TMD
- A.10 Définition des champs de demande – Offlinx<sup>MC</sup>
- A.11 Définition des champs de demande – Frais de commodité
- A.12 Définition des champs de demande – Transaction périodiques

### A.1 Définition des champs de demande – Champs de connexion

**Principaux champs d'objet de connexion (toutes les transactions API)**

Variable	Type et limites	Description
ID de commerce store_id	<i>Chaîne</i> S. O.	Identifiant unique fourni par Moneris lors de la configuration du compte de commerçant
Jeton API api_token	<i>Chaîne</i> S. O.	Chaîne de caractères alphanumériques unique créée par Moneris lors de l'activation du compte de commerçant  Pour obtenir votre jeton API, accédez aux paramètres Admin de l'environnement de test ou de production de votre magasin dans le Centre de ressources pour commerçants, qui se trouve aux liens URL qui suivent :  Test : <a href="https://esqa.moneris.com/mpg/?chlan">https://esqa.moneris.com/mpg/?chlan</a>

Variable	Type et limites	Description
		<p><a href="#">g=fr</a></p> <p>Production :  <a href="https://www3.moneris.com/mpg/?chl=ang=fr">https://www3.moneris.com/mpg/?chl=ang=fr</a></p>

## A.2 Définition des champs de demande – Champs de base

Variable	Type et limites	Description
Montant amount	<p><i>Chaîne</i></p> <p>10 caractères décimaux</p> <p>Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal</p>	<p>Montant de la transaction en dollars</p> <p>Doit comporter au moins trois chiffres, dont deux décimales</p> <p>Valeur minimale acceptée : 0,01 \$, valeur maximale acceptée : 9 999 999,99 \$</p>
	<b>EXEMPLE : 1 234 567,89</b>	
Code d'autorisation auth_code	<p><i>Chaîne</i></p> <p>8 caractères alphanumériques</p>	Code d'autorisation requis pour effectuer une transaction forcée; fourni dans la réponse de la banque émettrice
Montant de conclusion comp_amount	<p><i>Chaîne</i></p> <p>10 caractères décimaux</p> <p>Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal</p>	<p>Montant en dollars d'une transaction de conclusion de préautorisation, qui peut être différent du montant initial de la préautorisation</p>
	<b>EXEMPLE : 1 234 567,89</b>	
Numéro de carte de crédit pan	<p><i>Chaîne</i></p> <p>Maximum 20 caractères alphanumériques</p>	Numéro de carte de crédit, qui comporte généralement 16 chiffres – le champ peut comporter un maximum de 20 chiffres en vue de l'expansion future des numéros de carte

Variable	Type et limites	Description
		Contient le jeton requis pour les transactions de transformation en jetons
Descripteur dynamique dynamic_descriptor	<p><i>Chaîne</i></p> <p>20 caractères alphanumériques</p> <p>Total de 22 caractères incluant votre nom de commerçant et un séparateur</p> <div style="background-color: #e0f2fd; padding: 10px;"> <b>REMARQUE :</b> Certains caractères spéciaux ne sont pas autorisés :            &lt;&gt;\$%=?^{}[]\'         </div>	<p>Description définie par le commerçant, envoyée pour chaque transaction, qui apparaîtra sur le relevé de carte de crédit, ajoutée au nom de l'entreprise du commerçant</p> <p>Selon l'émetteur de la carte, le relevé comportera généralement le descripteur dynamique ajouté au nom commercial existant du commerçant, séparé par le caractère « / »; les caractères supplémentaires seront coupés.</p> <div style="background-color: #e0f2fd; padding: 10px;"> <b>REMARQUE :</b> La limite maximale de 22 caractères doit inclure le « / » comme l'un des caractères.         </div>
Indicateur de commerce électronique crypt_type	<p><i>Chaîne</i></p> <p>1 caractère alphanumérique</p>	<p>Décrit la catégorie de transaction de commerce électronique en cours de traitement Les valeurs autorisées sont :</p> <p>1 = Commande postale/téléphonique – Unique</p> <p>2 = Commande postale/téléphonique – Périodique</p> <p>3 = Commande postale/téléphonique – Versement</p> <p>4 = Commande postale/téléphonique – Classification inconnue</p> <p>5 = Transaction électronique authentifiée (3-D Secure)</p> <p>6 = Transaction électronique non authentifiée (3-D Secure)</p> <p>7 = Commerçant prenant en charge le SSL</p> <p><b>Lors d'une transaction utilisant les renseignements d'identification au dossier pour laquelle le champ de demande e-commerce indicator est</b></p>

Variable	Type et limites	Description
		<p><b>également inclus</b>, les valeurs permises de ce champ dépendent de la valeur envoyée pour la variable <b>payment indicator</b>. Pour plus d'information, consultez les définitions des champs de réponse.</p> <p>Si l'indicateur de paiement = R, alors les valeurs autorisées pour l'indicateur de commerce électronique sont les suivantes : 2, 5 ou 6</p> <p>Si l'indicateur de paiement = Z, alors les valeurs autorisées pour l'indicateur de commerce électronique sont les suivantes : 2, 5 ou 6</p> <p>Si l'indicateur de paiement = C, alors les valeurs autorisées pour l'indicateur de commerce électronique sont les suivantes : 1, 5, 6 ou 7</p> <p>Si l'indicateur de paiement = U, alors les valeurs autorisées pour l'indicateur de commerce électronique sont les suivantes : 1 ou 7</p> <p>Si l'indicateur de paiement = Z, alors les valeurs autorisées pour l'indicateur de commerce électronique sont les suivantes : 1, 5, 6 ou 7</p> <p><b>Pour les transactions Apple Pay ou Google Pay<sup>MC</sup> où vous effectuez un déchiffrement</b> : envoyez la valeur du champ <b>ecilIndicator</b> ou <b>3dsEcilIndicator</b> renvoyée dans les données.</p> <p>Si la valeur n'est pas présente dans les données, envoyez la valeur 5; si vous obtenez une valeur à deux caractères (p. ex. 05 ou 07), supprimez le 0 initial et envoyez-nous simplement le deuxième caractère.</p> <p>Les valeurs autorisées pour Apple Pay et Google Pay<sup>MC</sup> sont :</p> <ul style="list-style-type: none"> <li>5 : Transaction de commerce électronique authentifiée</li> <li>7 : Commerçant supportant le SSL</li> </ul>
Date d'expiration expdate	<i>Chaîne</i> 4 caractères	Date d'expiration de la carte de crédit, au format AAMM

Variable	Type et limites	Description
	alphanumériques AAMM	<b>REMARQUE :</b> Il s'agit de l'inverse du format de date MMAA qui est affichée sur la carte.
is incremental  is_incremental	<i>Valeur booléenne</i>  true/false	<pre>preauth.setIsIncremental(is_incremental);</pre> <p>Indique si cette préautorisation utilise un montant estimé. Les estimations permettent d'incrémenter le montant retenu au moyen de demandes subséquentes d'autorisation incrémentale. La valeur par défaut est « false ».</p> <p>{b}REMARQUE : {/b}veuillez noter que si ce champ contient la valeur « true », la préautorisation n'est valable que pour une seule conclusion de préautorisation. Toute conclusion soumise à titre de conclusion partielle est traitée comme une conclusion complète (ship_indicator= P est traité comme ship_indicator= F lorsque, dans la préautorisation originale, le paramètre is_incremental=true).</p>
Indicateur étranger	<i>Valeur booléenne</i>  true/false	<pre>purchase.setForeignIndicator(foreign_indicator);</pre> <p>Utilisé pour identifier les transactions nationales traitées par une commerçante ou un commerçant qui se trouve dans un autre pays.</p>

ID de commande order_id	<i>Chaîne</i>  50 caractères alphanumériques  a-Z A-Z 0-9 _ - : . @ espaces	Identifiant de transaction déterminé par le commerçant et unique pour chaque transaction d'achat, de préautorisation et de remboursement indépendant Il ne peut y avoir deux de ces types transactions avec le même ID de commande.  Pour les transactions de remboursement, de conclusion ou de correction d'achat, l'ID de la commande doit correspondre à celui de la transaction originale.
ID de la commande d'origine orig_order_id	<i>Chaîne</i>	ID de commande de la transaction originale de préautorisation, utilisé comme référence pour retrouver les détails de paiement originaux
Indicateur d'expédition ship_indicator	<i>Chaîne</i>  1 caractère alphanumérique	Sert à identifier les transactions de conclusion qui nécessitent plusieurs envois, également appelées conclusions multiples  Par défaut, si l'indicateur d'expédition n'est pas envoyé, la transaction de préautorisation est considérée comme finale.  Pour indiquer que la conclusion de la préautorisation doit être laissée

Variable	Type et limites	Description
		<p>ouverte par l'émetteur, puisque des expéditions supplémentaires ou des conclusions sont en attente, il faut soumettre un indicateur d'expédition ayant une valeur de P.</p> <p>Valeurs possibles :</p> <p>P = Partiel</p> <p>F = Final</p>
Numéro de transaction txn_number	<p><i>Chaîne</i></p> <p>255 caractères (alphanumériques, trait d'union ou trait de soulignement)</p> <p>Longueur variable</p>	<p>Fait référence à la transaction originale lors de l'exécution d'une transaction subséquente (c.-à-d. conclusion d'une préautorisation, correction d'un achat ou remboursement)</p> <p>Cette valeur est renvoyée dans la réponse de la transaction originale.</p> <p>Conclusion de préautorisation : fait référence à une préautorisation</p> <p>Remboursement ou correction d'achat : fait référence à un achat ou à la conclusion d'une préautorisation</p>
Indicateur de portefeuille électronique wallet_indicator	<p><i>Chaîne</i></p> <p>3 caractères alphanumériques</p>	<p>Indique qu'un numéro de carte a été obtenu par l'entremise d'un portefeuille électronique, comme Apple Pay, Google Pay<sup>MC</sup>, Visa Checkout et Mastercard MasterPass, ou par l'entremise d'un jeton provenant de la marque de carte</p> <p>Requis pour les transactions Apple Pay et Google Pay<sup>MC</sup> pour lesquelles vous utilisez votre propre API pour déchiffrer les données</p> <p>Valeurs possibles :</p> <p>APP = Apple Pay intégré à une application</p>

Variable	Type et limites	Description
		<p>APW = Apple Pay intégrée à un site Web</p> <p>GPP = Google Pay intégré à une application</p> <p>GPW = Google Pay intégré à un site Web</p> <p>VCO = Visa Checkout</p> <p>MMP = Mastercard Masterpass</p> <p>MVN = Transformation en jetons de Moneris du réseau Visa</p> <p>MMN = Transformation en jetons de Moneris du réseau Mastercard</p> <p>MAN = Transformation en jetons de Moneris du réseau Amex</p> <p>TVN = Transformation en jetons tierce du réseau Visa</p> <p>TMN = Transformation en jetons tierce du réseau Mastercard</p> <p>TAN = Transformation en jetons tierce du réseau Amex</p> <p><b>REMARQUE :</b> Veuillez noter que si ce champ est inclus pour indiquer Apple Pay ou Google Pay<sup>MC</sup>, alors les frais de commodité ne sont pas pris en charge.</p> <p><b>REMARQUE :</b> Les indiscteurs de portefeuille électronique liés à la transformation en jetons du réseau ne sont pas dans l'appel API; ils se trouvent dans le centre de ressources pour commerçants.</p>

### A.3 Définition des champs de demande – Renseignements d'identification au dossier

Variable	Type et limites	Description
ID de l'émetteur	<p><i>Chaîne</i></p> <p>15 caractères alphanumériques</p> <p><b>REMARQUE :</b> Cette variable est requise pour chaque transaction entamée par le commerçant à la</p>	<p>Identifiant unique pour les renseignements d'identification stockés du titulaire de la carte</p> <p>Renvoyé dans la réponse de la marque</p>

Variable	Type et limites	Description
<p>suite de la première transaction. Après l'envoi de la première transaction, l'ID d'émetteur (issuer ID) est reçu dans la réponse de transaction, puis utilisé dans les demandes de transaction subséquentes.</p>	<p>Longueur variable</p>	<p>de la carte lors du traitement d'une transaction de renseignements d'identification au dossier</p> <p>Si les renseignements d'identification du titulaire de carte sont stockés pour la première fois et que l'ID de l'émetteur a été retourné dans la réponse, vous devez enregistrer l'ID de l'émetteur dans votre système afin de l'utiliser dans les transactions ultérieures de renseignements d'identification au dossier (s'applique uniquement aux transactions initiées par le commerçant).</p> <p>L'ID de l'émetteur doit être enregistré dans vos systèmes lorsqu'il est renvoyé par Passerelle Moneris dans les données de réponse, que la valeur ait été reçue ou non.</p> <p>En tant que meilleure pratique, si l'ID de l'émetteur n'est pas renvoyé et que vous avez reçu une valeur NULL à la place, stockez cette valeur et envoyez-la dans la transaction suivante.</p>
<p>Indicateur de paiement</p>	<p><i>Chaîne</i> 1 caractère alphabétique</p>	<p>Indique l'utilisation actuelle ou prévue des renseignements d'identification</p> <p>Valeurs possibles pour les premières transactions :</p> <p>C = Transaction non planifiée liée aux renseignements d'identification au dossier (premières transactions seulement)</p> <p>R = Transaction périodique</p> <p>V = Transaction périodique à paiement variable</p> <p>Valeurs possibles pour les transactions ultérieures :</p> <p>R = Transaction périodique</p> <p>V = Transaction périodique à paiement variable</p> <p>U = Transaction non planifiée entamée par le</p>

Variable	Type et limites	Description
		<p>commerçant</p> <p>Z = Transaction non planifiée entamée par le client</p> <p>Lors d'une transaction utilisant les renseignements d'identification au dossier pour laquelle le champ de demande e-commerce indicator est également inclus, les valeurs acceptables de ce champ dépendent de la valeur envoyée pour l'indicateur de paiement, comme suit :</p> <p>Si l'indicateur de paiement = R, alors les valeurs autorisées pour l'indicateur de commerce électronique sont les suivantes : 2, 5 ou 6</p> <p>Si l'indicateur de paiement = Z, alors les valeurs autorisées pour l'indicateur de commerce électronique sont les suivantes : 2, 5 ou 6</p> <p>Si l'indicateur de paiement = C, alors les valeurs autorisées pour l'indicateur de commerce électronique sont les suivantes : 1, 5, 6 ou 7</p> <p>Si l'indicateur de paiement = U, alors les valeurs autorisées pour l'indicateur de commerce électronique sont les suivantes : 1 ou 7</p> <p>Si l'indicateur de paiement = Z, alors les valeurs autorisées pour l'indicateur de commerce électronique sont les suivantes : 1, 5, 6 ou 7</p>
Information de paiement	<p><i>Chaîne</i></p> <p>1 caractère numérique</p>	<p>Indique si la transaction est la première ou une transaction ultérieure dans la série</p> <p>Valeurs possibles :</p> <p>0 = Première transaction d'une série (stockage des données de paiement fournies par le titulaire de la carte)</p> <p>2 = Transactions ultérieures (utilisant les données de paiement précédemment stockées)</p>

## A.4 Définition des champs de demande - Ajout du jeton temporaire de Google Pay

Variable	Type et limites	Description
Jeton de paiement <code>&lt;payment_token&gt;</code>	<i>Objet S.O.</i>	Détails du paiement renvoyés par Google dans son objet <code>PaymentData</code> pour les transactions Google Pay (voir la section <code>Definition of Request Fields - GooglePay Token Temp Add</code> ci-dessous pour les détails des champs).
Signature <code>&lt;signature&gt;</code>	<i>Chaîne</i>	Vérifie que le message provient bien de Google, est codé en base64 et créé avec ECDSA par la clé de signature intermédiaire et est renvoyé par Google dans son objet <code>PaymentData</code> pour les transactions Google Pay
Version du protocole <code>&lt;protocol_version&gt;</code>	<i>Chaîne</i>	Identifie le système de chiffrement ou de signature selon lequel le message est créé, permet au protocole d'évoluer au fil du temps, si nécessaire, et est renvoyé par Google dans son objet <code>PaymentData</code> pour les transactions Google Pay
Message signé <code>&lt;signed_message&gt;</code>	<i>Chaîne</i>	Un objet JSON sérialisé sous la forme d'une chaîne HTML sécurisée qui contient le message chiffré, la clé publique éphémère et la balise, est sérialisé pour simplifier le processus de vérification de la signature et est renvoyé par Google dans son objet <code>PaymentData</code> pour les transactions Google Pay

## A.5 Définition des champs de demande – Chambre forte

Variable	Type et limites	Description
Clé de données	<p><i>Chaîne</i></p> <p>25 caractères alphanumériques</p>	<p>Identifiant unique pour un profil de chambre forte, utilisé dans les futures transactions financières de la chambre forte pour associer une transaction à ce profil.</p> <p>La clé de données est générée par Moneris et vous est retournée dans l'objet Receipt lors du premier enregistrement du profil.</p>
Format de la clé de données	<p><i>Chaîne</i></p> <p>2 caractères alphanumériques</p>	<p>Précise le format de la clé de données renvoyée</p> <p>Si ce champ est laissé vide, le format de la clé de données sera de 25 caractères alphanumériques par défaut.</p> <p>Valeurs possibles :</p> <p>0 = Clé de données alphanumériques de 25 caractères</p> <p>OU = Clé de données alphanumériques unique à 25 caractères</p>
Durée	<p><i>Chaîne</i></p> <p>3 caractère numérique</p> <p>Maximum de 900 secondes</p>	Durée pendant laquelle le jeton temporaire doit être disponible
Adresse courriel	<p><i>Chaîne</i></p> <p>30 caractères alphanumériques</p>	<p>Adresse courriel du client</p> <p>Peut être envoyée lors de la création ou de la mise à jour d'un profil de chambre forte</p>
Remarque	<p><i>Chaîne</i></p> <p>30 caractères alphanumériques</p>	<p>Utilisé pour tout renseignement supplémentaire relatif au client</p> <p>Peut être envoyée lors de la création ou de la mise à jour d'un profil de chambre forte</p>

Numéro de téléphone	<i>Chaîne</i> 30 caractères alphanumériques	Numéro de téléphone du client  Peut être envoyée lors de la création ou de la mise à jour d'un profil de chambre forte
return issuer ID	<i>Valeur booléenne</i> true/false	Lorsque la valeur est « True », la passerelle renvoie l'identifiant de la banque émettrice. La valeur par défaut est « False ».

## A.6 Définition des champs de demande – Transactions de niveaux 2 et 3 de Visa

Tableau 1 Visa – Données communes des carte d'entreprise – Champs de demande de niveau 2

Requis*	Nom du champ	Limites	Méthode Set	Description
Y	Taxe nationale	12 caractères décimaux	TRANSACTIONNAME.setNationalTax(national_tax);	<p>Doit être identique à la taxe nationale (TPS ou TVH) apparaissant sur la facture</p> <p>Valeur minimale de 0,01 et maximale de 999 999,99 Doit comporter 2 décimales.</p>
Y	Numéro de TVA du commerçant/Référence d'entreprise unique	20 caractères alphanumériques	TRANSACTIONNAME.setMerchantVatNo(merchant_vat_no);	<p>Numéro d'enregistrement des taxes du client</p> <p>Doit être fourni si des taxes sont incluses dans la facture</p> <p><b>REMARQUE :</b> Il ne doit pas y avoir que des espaces ou que des zéros.</p>

Requis*	Nom du champ	Limites	Méthode Set	Description
C	Taxe locale	12 caractères décimaux	TRANSACTIONNAME .setLocalTax(local_tax);	<p>Doit être identique à la taxe locale (TVP ou TVQ) apparaissant sur la facture</p> <p>Si la taxe locale est inclue, elle ne doit pas être composée uniquement d'espaces ou de zéros. Elle doit être fournie si la taxe locale (TVP ou TVQ) s'applique.</p> <p>Valeur minimale de 0,01</p> <p>Valeur maximale de 999 999,99</p> <p>Doit comporter 2 décimales</p>
C	Numéro d'enregistrement des taxes locales (TVP ou TVQ) du commerçant	15 caractères alphanumériques	TRANSACTIONNAME .setLocalTaxNo(local_tax_no);	<p>Numéro d'enregistrement des taxes locales (TVP ou TVQ) du commerçant</p> <p>Il doit être fourni si des taxes sont incluses dans la facture et si la taxe locale est inclue , il ne doit pas être composé uniquement d'espaces ou zéros.</p> <p>Doit être fourni si la taxe locale</p>

Requis*	Nom du champ	Limites	Méthode Set	Description
				(TVP ou TVQ) s'applique
C	Numéro de TVA du client	13 caractères alphanumériques	TRANSACTIONNAME .setCustomerVatNo(customer_vat_no);	Si le numéro d'enregistrement des taxes du client figure sur la facture pour justifier des transactions exemptées de taxes, il doit être indiqué ici.
C	Code de client/Code de référence du client	16 caractères alphanumériques	TRANSACTIONNAME .setCri(cri);	Valeur que le client peut fournir au fournisseur au point de vente (il doit être entré si le client le fournit)
N	Code de client	17 caractères alphanumériques	TRANSACTIONNAME .setCustomerCode(customer_code);	Champ facultatif qui ne sera pas envoyé à Visa, mais qui se trouvera dans les rapports de Moneris
N	Numéro de facture	17 caractères alphanumériques	TRANSACTIONNAME .setInvoiceNumber(invoice_number);	Champ facultatif qui ne sera pas envoyé à Visa, mais qui se trouvera dans les rapports de Moneris

\* Y = Obligatoire, N = Facultatif, C = Conditionnel

Tableau 2 Visa – Données communes de carte d'entreprise – Champs de demande de niveau 2 (VSPurcha

Requis	Variable	Nom du champ	Taille/Type	Description
C*	Nom de l'acheteur	buyer_name	30 caractères	Nom de l'acheteur ou

Requis	Variable	Nom du champ	Taille/Type	Description
			alphanumériques	du destinataire  * Requis par l'ARC uniquement si la transaction est supérieure à 150 \$
C*	Taux de taxe locale	local_tax_rate	4 caractères décimaux	Indique le taux de taxe détaillé appliqué en fonction du montant de taxe locale  <b>EXEMPLE :</b> Une TVP de 8 % devrait être 8,0.  Valeur maximale de 99,99  * Doit être fourni si la taxe locale (TVP ou TVQ) s'applique.
N	Droits de douane	duty_amount	9 caractères décimaux	Montant de douane de l'achat total  Un montant avec un symbole négatif signifie que le montant est un crédit, un montant avec un symbole positif ou sans symbole signifie que le montant est un débit.  Valeur maximale de 999 999,99 (sans symbole)
N	Traitemen t des rabais sur la facture	discount_treatment	1 caractère numérique	Indique la façon dont le commerçant gère les rabais  Doit être l'une des

Requis	Variable	Nom du champ	Taille/Type	Description
				<p>valeurs suivantes :</p> <p>0 = Aucun rabais ne s'applique au niveau de la facture</p> <p>1 = La taxe est calculée sur les totaux après l'application des rabais</p> <p>2 = La taxe est calculée sur les totaux avant l'application des rabais</p>
N	Montant du rabais au niveau de la facture	discount_amt	9 caractères décimaux	<p>Montant du rabais (s'il est fourni au niveau de la facture selon le traitement des rabais sur la facture)</p> <p>Ne doit pas être zéro si la valeur de traitement des rabais sur la facture est 1 ou 2</p> <p>Valeur minimale de 0,00 et maximale de 999 999,99</p>
C*	Code postal d'expédition	ship_to_pos_code	10 caractères alphanumériques	<p>Le code postal ou le code ZIP auquel la marchandise sera expédiée.</p> <p>* Requis s'il y a une expédition.</p> <p>Code postal alphanumérique complet – Format ANA&lt;space&gt;NAN requis si expédié à une adresse canadienne</p>
C	Code postal	ship_from_pos_code	10 caractères	Code postal ou code

Requis	Variable	Nom du champ	Taille/Type	Description
	d'origine		alphanumériques	<p>ZIP d'origine de la marchandise</p> <p>Pour les adresses canadiennes, un code postal alphanumérique complet au format ANA&lt;espace&gt;NAN valide est requis pour le commerçant.</p>
C*	Code du pays de destination	des_cou_code	2 caractères alphanumériques	<p>Code du pays où la marchandise achetée sera expédiée</p> <p>Utiliser le format ISO 3166-1 alpha-2</p> <p><b>REMARQUE :</b> Requis s'il apparaît sur la facture d'une transaction internationale.</p>
Y	Numéro de référence unique de la facture d'une TVA	vat_ref_num	25 caractères alphanumériques	<p>Numéro de référence unique de la facture d'une TVA</p> <p>Doit être rempli avec le numéro de facture, qui ne peut pas être composé uniquement d'espaces ou de zéros</p>
Y	Traitements fiscaux	tax_treatment	1 caractère numérique	<p>Doit être l'une des valeurs suivantes :</p> <p>0 = Prix nets avec taxe calculée au niveau de chaque ligne</p> <p>1 = Prix nets avec taxe calculée au niveau de la facture</p> <p>2 = Prix bruts fournis avec les renseignements sur les</p>

Requis	Variable	Nom du champ	Taille/Type	Description
				<p>taxes à chaque ligne</p> <p>3 = Prix bruts fournis avec les renseignements sur la taxe au niveau de la facture</p> <p>4 = Aucune taxe ne s'applique (petit commerçant) sur la facture de la transaction</p>
N	Montant des frais de transport/expédition	freight_amount	9 caractères décimaux	<p>Frais de transport de l'achat total</p> <p>Si les frais d'expédition ne sont pas inclus dans une ligne d'article, ils doivent être indiqués ici, le cas échéant.</p> <p>Montant en numéraire signé : un montant avec un symbole négatif signifie que le montant est un crédit, alors qu'un montant avec un symbole positif ou sans symbole signifie que le montant est un débit.</p> <p>Valeur maximale de 999 999,99 (sans symbole)</p>
C	Taux des frais de transport TPS ou TVH	gst_hst_freight_rate	4 caractères décimaux	<p>Taux de la TPS (à l'exclusion de la TVP) ou de la TVH appliqué au montant de l'expédition (conformément au traitement fiscal)</p> <p>Si le montant de transport ou</p>

Requis	Variable	Nom du champ	Taille/Type	Description
				<p>d'expédition est fourni, alors ce taux de taxe (TPS nationale ou TVH) doit être fourni.</p> <p>Montant en numéraire, valeur maximale de 99,99 Par exemple, une TVH de 13 % équivaut à 13,00</p>
C	Montant des frais de transport TPS ou TVH	gst_hst_freight_amount	9 caractères décimaux	<p>Montant de la TPS (excluant la TVP) ou de la TVH appliquée au montant de l'expédition</p> <p>Si le montant de transport ou d'expédition est fourni, alors ce taux de taxe (TPS nationale ou TVH) doit être fourni si la valeur de traitement fiscal est de 0 ou 2.</p> <p>Montant en numéraire signé : valeur maximale sans symbole de 999 999,99</p>

Tableau 3 Visa – Détails de ligne d'article – Champs de demande de niveau 3 (VSPurchl)

Requis	Variable	Nom du champ	Taille/Type	Description
C	Code de commodité de l'article	item_com_code	12 caractères alphanumériques	Code de commodité du produit de la ligne d'article (si ce champ n'est pas envoyé, le champ productCode doit

Requis	Variable	Nom du champ	Taille/Type	Description
				l'être)
Y	Code du produit	product_code	12 caractères alphanumériques	<p>Code de produit pour cette ligne d'article – code de produit du commerçant, code de produit du fabricant ou code de produit de l'acheteur</p> <p>Il s'agit généralement de l'UGS ou de l'identifiant utilisé par le commerçant pour faire le suivi de l'article ou du service et en fixer le prix.</p> <p>Il devrait toujours être fourni pour chaque ligne d'article.</p>
Y	Description de l'article	item_description	35 caractères alphanumériques	Description de la ligne d'article
Y	Nombre d'article	item_quantity	12 caractères décimaux	<p>Quantité facturée pour cette ligne d'article</p> <p>Jusqu'à quatre décimales sont supportés, les chiffres entiers sont acceptés</p> <p>Valeur minimale de 0,0001</p> <p>Valeur maximale de 999 999 999 999</p>

Requis	Variable	Nom du champ	Taille/Type	Description
Y	Unité de mesure de l'article	item_uom	2 caractères alphanumériques	Unité de mesure Utilisez les unités de mesure et codes permis par la norme ANSI X-12 EDI.
Y	Coût unitaire de l'article	unit_cost	12 caractères décimaux	Coût unitaire de chaque article De 2 à 4 décimales sont acceptées Valeur minimale de 0,0001 Valeur maximale de 999 999,9999
N	Montant de la TVA	vat_tax_amt	12 caractères décimaux	Tout montant de taxe sur la valeur ajoutée ou autre taxe de vente Doit comporter 2 décimales Valeur minimale de 0,01 Valeur maximale de 999 999,99
N	Taux de la TVA	vat_tax_rate	4 caractères décimaux	Taux de la taxe de vente  <b>EXEMPLE :</b> Une TVP de 8 % devrait être 8,0.  Valeur maximale de 99,99
Y	TraITEMENT des rabais	discount_treatmentL	1 caractère numérique	Doit être l'une des valeurs suivantes :

Requis	Variable	Nom du champ	Taille/Type	Description
				<p>0 = Aucun rabais ne s'applique au niveau de la facture</p> <p>1 = La taxe est calculée sur les totaux après l'application des rabais</p> <p>2 = La taxe est calculée sur les totaux avant l'application des rabais</p>
C	Montant du rabais	discount_amtL	12 caractères décimaux	<p>Montant du rabais, s'il est prévu pour cette ligne d'article selon la ligne d'article du montant du rabais (Discount Treatment)</p> <p>Ne doit pas être zéro si la valeur de la ligne d'article du montant du rabais est de 1 ou 2</p> <p>Doit comporter 2 décimales</p> <p>Valeur minimale de 0,01</p> <p>Valeur maximale de 999 999,99</p>

## A.7 Définition des champs de demande – Transactions de niveaux 2 et 3 de Mastercard

Objets Table1 – Niveaux 2 et 3 Mastercard

Objets MCCorpais	Description
MCCorpac	Données communes de carte d'entreprise
MCCorpal	Détails de la ligne d'article.

**Tableau 2 Mastercard – Données communes de carte d’entreprise (MCCorpac) – Champs de demande de niveau 2**

Requis	Variable	Nom du champ	Taille/Type	Description
N	AustinTetraNumber	Numéro Austin-Tetra	15 caractères alphanumériques	Numéro Austin-Tetra du commerçant
N	NaicsCode	Code SCIAN (NAICS code)	15 caractères alphanumériques	Code du système de classification des industries de l’Amérique du Nord (SCIAN) attribué au commerçant
N	CustomerCode	Code de client	25 caractères alphanumériques	Numéro de contrôle, tel qu’un numéro de bon de commande, un numéro de projet, un numéro de répartition de service ou un nom donné au commerçant par le fournisseur Justifié à gauche et peut comporter des espaces
N	UniqueInvoiceNumber	Numéro de facture unique	17 caractères alphanumériques	Numéro unique associé à la transaction individuelle fourni par le commerçant
N	CommodityCode	Code de marchandise	15 caractères alphanumériques	Code attribué par le commerçant qui catégorise le mieux l’article acheté
N	OrderDate	Date de la commande	6 caractère numérique	Date d’achat de l’article Si présent, doit contenir une date valide au format AAMMJJ
N	CorporationVatNumber	Numéro TVA d’entreprise	20 caractères alphanumériques	Contient le numéro de taxe sur la valeur ajoutée (TVA) d’une entreprise
N	CustomerVatNumber	Numéro de TVA	20 caractères	Contient le numéro de

Requis	Variable	Nom du champ	Taille/Type	Description
		du client	alphanumériques	TVA du client ou du titulaire de carte qui est utilisé pour identifier le client lors de l'achat de biens et de services vendus par le commerçant
N	FreightAmount	Montant des frais de transport	12 caractères décimaux	Frais d'expédition sur l'achat total Doit contenir 2 décimales
N	DutyAmount	Droits de douane	12 caractères décimaux	Droits de douane sur le total de l'achat, doivent avoir 2 décimales
N	DestinationProvinceCode	Code de l'État ou de la province de destination	3 caractères alphanumériques	État ou province du pays où les marchandises seront expédiées Justifié à gauche avec des espace de fin. P. ex. ONT – Ontario
N	DestinationCountryCode	Code du pays de destination	3 caractères alphanumériques	Code du pays où la marchandise sera expédiée Justifié à gauche avec des espace de fin. P. ex. CAN – Canada
N	ShipFromPosCode	Code postal d'origine	10 caractères alphanumériques	Code postal ou code ZIP d'origine de la marchandise
N	ShipToPosCode	Code postal de destination	10 caractères alphanumériques	Code postal ou code ZIP auquel la marchandise sera expédiée
N	AuthorizedContactName	Nom de la personne-ressource autorisée	36 caractères alphanumériques	Nom d'une personne ou d'une société qui agit à titre de personne-ressource pour les achats

Requis	Variable	Nom du champ	Taille/Type	Description
				autorisés par l'entreprise
N	AuthorizedContactPhone	Numéro de téléphone de la personne-ressource autorisée	17 caractères alphanumériques	Numéro de téléphone d'une personne ou d'une société avec laquelle il faut communiquer pour les achats autorisés par l'entreprise
N	AdditionalCardAcceptordata	Données supplémentaires de l'accepteur de carte	40 caractères alphanumériques	Renseignements supplémentaires sur l'accepteur de cartes
N	CardAcceptorType	Type d'accepteur de cartes	8 caractères alphanumériques	Différentes classifications des caractéristiques de propriété des entreprises  Ce champ prend 8 caractères. Chaque caractère représente un composant différent, soit :  Le premier caractère représente le type d'entreprise et contient un code permettant d'identifier la classification ou le type d'entreprise :  Société par actions  Inconnu  Individuel ou propriétaire unique  Partenariat  Association, état ou fiducie  Organisations exonérées d'impôts (501C)

Requis	Variable	Nom du champ	Taille/Type	Description
				<p>Organisation internationale</p> <p>Société à responsabilité limitée (SARL)</p> <p>Agence gouvernementale</p> <p>Le deuxième caractère représente le type de propriétaire d'entreprise. Il contient un code permettant d'identifier les caractéristiques propres au propriétaire de l'entreprise.</p> <p>1 = Aucune classification d'application</p> <p>2 = Propriétaire d'entreprise femme</p> <p>3 = Propriétaire d'entreprise femme avec un handicap physique</p> <p>4 = Propriétaire d'entreprise homme avec un handicap physique</p> <p>0 = Inconnu</p> <p>Le troisième caractère représente le type de certification d'entreprise. Il contient un code permettant d'identifier les caractéristiques relatives au type de certification de l'entreprise, par exemple une certification de petite entreprise, d'entreprise défavorisée ou autre type de certification :</p> <p>1 = Non certifiée</p> <p>2 = Certification de petite entreprise par le Small Business Administration (SBA)</p> <p>3 = Certification SBA de petite entreprise</p>

Requis	Variable	Nom du champ	Taille/Type	Description
				<p>défavorisée</p> <p>4 = Autre certification reconnue par un gouvernement ou une agence (comme le Minority Supplier Development Council)</p> <p>5 = Petite entreprise auto-certifiée</p> <p>6 = Certification de la SBA en tant que petite entreprise et autre certification reconnue par le gouvernement ou une agence</p> <p>7 = Certification de la SBA en tant que petite entreprise défavorisée et autre certification reconnue par le gouvernement ou une agence</p> <p>8 = Autre certification reconnue par un gouvernement ou une agence et certification en tant que petite entreprise auto-certifiée</p> <p>A = Certification de la SBA comme 8(a)</p> <p>B = Petite entreprise défavorisée auto-certifiée (SDB)</p> <p>C = Certification de la SBA comme HUBZone</p> <p>O = Inconnu</p> <p>Le quatrième caractère représente le type racial ou ethnique de l'entreprise. Il contient un code identifiant la race ou l'ethnicité du propriétaire majoritaire de l'entreprise.</p> <p>1 = Afro-américain</p> <p>2 = Américain d'origine</p>

Requis	Variable	Nom du champ	Taille/Type	Description
				<p>asiatique et pacifique</p> <p>3 = Américain d'origine asiatique subcontinentale</p> <p>4 = Américain d'origine hispanique</p> <p>5 = Autochtone de l'Amérique du Nord</p> <p>6 = Autochtone hawaïen</p> <p>7 = Autochtone d'Alaska</p> <p>8 = Caucasiens</p> <p>9 = Autre</p> <p>0 = Inconnu</p> <p>Le cinquième caractère indique si le code du type d'entreprise a été fourni.</p> <p>Y = Le type d'entreprise est fourni</p> <p>N = Le type d'entreprise n'a pas été fourni</p> <p>R = L'accepteur de la carte a refusé de fournir le type d'entreprise</p> <p>Le sixième caractère indique si le code du type de propriétaire de l'entreprise a été fourni.</p> <p>Y = Le type de propriétaire de l'entreprise est fourni</p> <p>N = Le type de propriétaires d'entreprise n'a pas été fourni</p> <p>R = L'accepteur de la carte a refusé de fournir le type d'entreprise</p> <p>Le septième caractère indique si le code du type de certification d'entreprise a été fourni.</p> <p>Y = Le type de certification</p>

Requis	Variable	Nom du champ	Taille/Type	Description
				<p>de l'entreprise est fourni</p> <p>N = Le type de certification de l'entreprise n'a pas été fourni</p> <p>R = L'accepteur de la carte a refusé de fournir le type d'entreprise</p> <p>Le huitième caractère indique si le type racial ou ethnique de l'entreprise a été fourni.</p> <p>Y = Le type racial ou ethnique de l'entreprise est fourni</p> <p>N = Le type racial ou ethnique de l'entreprise n'a pas été fourni</p> <p>R = L'accepteur de la carte a refusé de fournir le type racial ou ethnique de l'entreprise</p>
N	CardAcceptorTaxId	Numéro de taxe de l'accepteur de carte	20 caractères alphanumériques	Numéro de taxe fédérale des États-Unis pour le numéro de TVA
N	CardAcceptorReferenceNumber	Numéro de référence de l'accepteur de carte	25 caractères alphanumériques	Code qui facilite la communication et la tenue des registres de l'accepteur de cartes ou de l'entreprise
N	CardAcceptorVatNumber	Numéro de TVA de l'accepteur de carte	20 caractères alphanumériques	Numéro de taxe sur la valeur ajoutée (TVA) pour l'emplacement de l'accepteur de carte utilisé pour identifier l'accepteur de la carte lors de la collecte et de la déclaration des taxes
C*	Tax	Taxes	Jusqu'à 6 tableaux	Il peut y avoir jusqu'à 6 tableaux contenant des détails de taxes

Requis	Variable	Nom du champ	Taille/Type	Description
				<p>différents. Consultez le tableau des taxes ci-dessous connaître pour la description de chaque champ.</p> <p>* Ce champ est obligatoire sous certaines conditions. Si vous utilisez ce tableau, vous devez remplir tous les champs du tableau des taxes, comme indiqué dans les champs de demande du tableau des taxes ci-dessous.</p>

**Tableau 3 Mastercard – Détails de ligne d'article (MCCorpal) – Champs de demande de niveau 3**

Requis	Variable	Nom du champ	Taille/Type	Description
N	CustomerCode	Code de client	25 caractères alphanumériques	Numéro de contrôle, tel qu'un numéro de bon de commande, un numéro de projet, un numéro de répartition de service ou un nom donné au commerçant par le fournisseur Justifié à gauche et peut comporter des espaces
N	LineItemDate	Date de la ligne d'article	6 caractère numérique	Date d'achat de l'article mentionnée dans les détails de la ligne d'article de la carte d'entreprise Format AAMMJJ
N	ShipDate	Date d'expédition	6 caractère	Date à laquelle la

Requis	Variable	Nom du champ	Taille/Type	Description
			numérique	Marchandise a été expédiée à la destination Format AAMMJJ
N	OrderDate	Date de la commande	6 caractère numérique	Date d'achat de l'article Format AAMMJJ
Y	ProductCode	Code du produit	12 caractères alphanumériques	Code du produit de la ligne d'article (si ce champ n'est pas envoyé, le champ itemComCode doit l'être).  Si une ligne d'article « Transport/Expédition » apparaît dans la commande, la valeur productCode doit être « Freight/Shipping » (Transport/Expédition) . Si une ligne d'article « Rabais » apparaît dans la commande, la valeur productCode doit être « Discount » (Rabais).
Y	ItemDescription	Description de l'article	35 caractères alphanumériques	Description de la ligne d'article
Y	ItemQuantity	Nombre d'article	12 caractères alphanumériques	Quantité d'article acheté
Y	UnitCost	Coût unitaire	12 caractères décimaux	Indique le coût unitaire de chaque article  Doit contenir un minimum de 2 décimales (maximum

Requis	Variable	Nom du champ	Taille/Type	Description
				de 5 décimales)  Valeur minimale de 0,00001 et maximale de 999 999,99999
Y	ItemUnitMeasure	Unité de mesure de l'article	12 caractères alphanumériques	Code de l'unité de mesure de la ligne d'article
Y	ExtItemAmount	Montant prolongé de l'article	9 caractères décimaux	Indique la valeur de l'article individuel qui est normalement calculée en multipliant le prix par la quantité  Doit comporter 2 décimales  Valeur minimale de 0,00 et maximale de 999 999,99
N	DiscountAmount	Montant du rabais	9 caractères décimaux	Indique le montant du rabais de l'article  Doit comporter 2 décimales  Valeur minimale de 0,00 et maximale de 999 999,99
N	CommodityCode	Code de marchandise	15 caractères alphanumériques	Code assigné par le commerçant qui catégorise le mieux les articles achetés
C*	Tax	Taxes	Jusqu'à 6 tableaux	Il peut y avoir jusqu'à 6 tableaux contenant des détails de taxes différents. Consultez le tableau des taxes ci-dessous connaître pour la description de

Requis	Variable	Nom du champ	Taille/Type	Description
				<p>chaque champ.</p> <p>* Ce champ est obligatoire sous certaines conditions. Si vous utilisez ce tableau, vous devez remplir tous les champs du tableau des taxes, comme indiqué dans les champs de demande du tableau des taxes ci-dessous.</p>

Tableau 4 Champs de demande de tableau des taxes – Transactions Mastercard de niveaux 2 et 3

Requis	Variable	Nom du champ	Taille/Type	Description
M	tax_amount	Montant des taxes	12 caractères décimaux	<p>Indique le montant des taxes pour l'achat de biens ou de services</p> <p>Doit comporter 2 décimales</p> <p>Valeur maximale de 999 999,99</p>
M	tax_rate	Taux de taxe	5 caractères décimaux	<p>Indique le taux de taxe détaillé qui est appliqué en fonction de la taxe</p> <p><b>EXEMPLE :</b> Une TVP de 5 % devrait être « 5,0 », alors qu'une TVP de ou 9,975 % devrait être « 9,975 »</p> <p>Peut contenir jusqu'à 3 décimales, avec une valeur minimale de 0,001 ainsi qu'une valeur maximale de 9 999,9</p>

Requis	Variable	Nom du champ	Taille/Type	Description
M	tax_type	Type de taxe	4 caractères alphanumériques	Indique le types de taxe, par exemple TVP, TVQ, TPS, TVH
M	tax_id	Numéro de taxe	20 caractères alphanumériques	Fournit un numéro d'identification utilisé par l'accepteur de carte avec l'autorité fiscale selon un montant de taxe précis tel qu'un numéro de TVP ou de TVH
M	tax_included_in_sales	Taxe incluse dans l'indicateur de vente	1 caractère alphanumérique	<p>Il s'agit de l'indicateur utilisé pour la saisie et la déclaration de taxes supplémentaires.</p> <p>Les valeurs valides sont :</p> <p>Y = Taxe incluse dans le montant total de l'achat</p> <p>N = Taxe non incluse dans le montant total de l'achat</p>

## A.8 Définition des champs de demande – Transactions de niveaux 2 et 3 de Amex

Tableau 1 Champs de demande de niveaux 2 et 3 de Amex --Table1 – Champs d'en-tête

Requis	Variable	Nom du champ	Taille/Type	Description
C	big04	Numéro du bon de commande	22 caractères alphanumériques	<p>Numéro du bon de commande fourni par le titulaire de la carte, qui est saisi par le commerçant au point de vente</p> <p>Cette entrée est utilisée dans le processus de déclaration et de production de rapports et</p>

Requis	Variable	Nom du champ	Taille/Type	Description										
				<p>elle peut inclure des renseignements comptables propres au client.</p> <p>Obligatoire si le client du commerçant fournit un numéro de bon de commande</p>										
N	big05	Numéro de libération (release number)	30 caractères alphanumériques	Numéro qui identifie la libération d'un bon de commande qui a déjà été passé par les parties concernées par la transaction										
N	big10	Numéro de facture	8 caractères alphanumériques	Inclut le numéro de facture ou de référence Amex										
Y	n101	Code d'identification d'entité	2 caractères alphanumériques	<p>Valeurs acceptées :</p> <p>R6 = Demandeur (obligatoire)</p> <p>BG = Groupe d'achat (facultatif)</p> <p>SF = Expéditeur (facultatif)</p> <p>ST = Destinataire (facultatif)</p> <p>40 = Récepteur (facultatif)</p>										
Y	n102	Nom	40 caractères alphanumériques	<p><b>Code n101 Signification n102</b></p> <table border="1"> <tr> <td>R6</td><td>Nom du demandeur</td></tr> <tr> <td>BG</td><td>Nom du groupe acheteur</td></tr> <tr> <td>SF</td><td>Nom de l'expéditeur</td></tr> <tr> <td>ST</td><td>Nom du destinataire</td></tr> <tr> <td>En 40</td><td>Nom du récepteur</td></tr> </table>	R6	Nom du demandeur	BG	Nom du groupe acheteur	SF	Nom de l'expéditeur	ST	Nom du destinataire	En 40	Nom du récepteur
R6	Nom du demandeur													
BG	Nom du groupe acheteur													
SF	Nom de l'expéditeur													
ST	Nom du destinataire													
En 40	Nom du récepteur													

Requis	Variable	Nom du champ	Taille/Type	Description												
N	n301	Adresse	40 caractères alphanumériques	Adresse												
N	n401	Ville	30 caractères alphanumériques	Ville												
N	n402	État ou province	2 caractères alphanumériques	État ou province												
N	n403	Code postal	15 caractères alphanumériques	Code postal												
Y	ref01	Élément d'identification de la référence	2 caractères alphanumériques	<p>Cet élément peut contenir les valeurs suivantes pour les occurrences correspondantes de l'objet N1Loop :</p> <table> <thead> <tr> <th>Valeur n101</th> <th>Dénotation ref01</th> </tr> </thead> <tbody> <tr> <td>R6</td> <td>Valeurs acceptées : 4C = Code de destination de la marchandise (obligatoire) CR = Numéro de référence du client (conditionnel)</td> </tr> <tr> <td>BG</td> <td>S. O.</td> </tr> <tr> <td>SF</td> <td>S. O.</td> </tr> <tr> <td>ST</td> <td>S. O.</td> </tr> <tr> <td>En 40</td> <td>S. O.</td> </tr> </tbody> </table>	Valeur n101	Dénotation ref01	R6	Valeurs acceptées : 4C = Code de destination de la marchandise (obligatoire) CR = Numéro de référence du client (conditionnel)	BG	S. O.	SF	S. O.	ST	S. O.	En 40	S. O.
Valeur n101	Dénotation ref01															
R6	Valeurs acceptées : 4C = Code de destination de la marchandise (obligatoire) CR = Numéro de référence du client (conditionnel)															
BG	S. O.															
SF	S. O.															
ST	S. O.															
En 40	S. O.															
Y	ref02	Identification de la référence	15 caractères alphanumériques	<p>VR est l'ID du fournisseur, les autres codes décrivent les éléments suivants :</p> <table> <thead> <tr> <th>Code ref01</th> <th>Dénotation ref02</th> </tr> </thead> <tbody> <tr> <td>4C</td> <td>Code postal canadien ou code ZIP de l'adresse du</td> </tr> </tbody> </table>	Code ref01	Dénotation ref02	4C	Code postal canadien ou code ZIP de l'adresse du								
Code ref01	Dénotation ref02															
4C	Code postal canadien ou code ZIP de l'adresse du															

Requis	Variable	Nom du champ	Taille/Type	Description
				destinataire (obligatoire)
				CR Numéro de référence du titulaire de la carte (facultatif)

Tableau 2 Champs de demande de niveaux 2 et 3 de Amex – Table2 – Champs de détail

Requis	Variable	Nom du champ	Taille/Type	Description
Y	it102	Quantité facturée pour la ligne d'article	10 caractères décimaux	Quantité d'article Jusqu'à 2 décimales sont prises en charge. Valeur minimale de 0,0 et maximale de 9 999 999 999
Y	it103	Code de l'unité ou de la base de mesure	2 caractères alphanumériques	Code de l'unité de mesure de la ligne d'article Doit contenir un code qui indique l'unité de mesure de la valeur ou la manière dont une mesure est prise  <b>EXEMPLE :</b> EA = chaque, E5 = pouces  Consultez le site ANSI X-12 EDI Allowable Units of Measure and Codes pour la liste des codes.
Y	it104	Prix unitaire	15 caractères décimaux	Coût unitaire de chaque article

Requis	Variable	Nom du champ	Taille/Type	Description
				Doit comporter 2 décimales  Valeur minimale de 0,00 et maximale de 999 999,99
N	it105	Code tarifaire de la base ou de l'unité	2 caractères alphanumériques	Code identifiant le type de prix unitaire d'un article
				<b>EXEMPLE :</b> DR = vendeur (dealer), AP = prix conseillé (advise price)
				Consultez le site ASC X12 004010 Element 639 pour la liste des codes.
N	it10618	Élément d'identification du produit ou du service	2 caractères alphanumériques	Valeurs acceptées :  MG = Numéro de pièce du fabricant  VC = Numéro de catalogue du fournisseur  SK = Numéro de référence du fournisseur  UP = Code universel du produit  VP = Numéro de pièce du fournisseur  PO = Numéro du bon de commande  AN = Code du bien défini par le client
N	it10719	Numéro de produit ou de service	<b>it10618</b> Taille ou type <b>it10719</b> VC 20 caractères alphanumérique	Le numéro du produit ou du service correspond au qualificateur

Requis	Variable	Nom du champ	Taille/Type	Description						
			<table border="1"> <tr> <td></td><td>s</td></tr> <tr> <td>PO</td><td>22 caractères alphanumérique s</td></tr> <tr> <td>Autre</td><td>30 caractères alphanumérique s</td></tr> </table>		s	PO	22 caractères alphanumérique s	Autre	30 caractères alphanumérique s	<p>précédent défini dans la variable it10618.</p> <p>La longueur maximale dépend du qualificateur défini dans la variable it10618.</p>
	s									
PO	22 caractères alphanumérique s									
Autre	30 caractères alphanumérique s									
C	txi01	Code du type de taxe	2 caractères alphanumériques	<p>Valeurs acceptées :</p> <p>CA = Taxe municipale (facultatif)</p> <p>CT = Taxe de comté (facultatif)</p> <p>EV = Taxe environnementale (facultatif)</p> <p>GS = Taxe sur les biens et services (TPS) (facultatif)</p> <p>LS = Taxe de vente d'État et locale (facultatif)</p> <p>LT = Taxe de vente locale (facultatif)</p> <p>PG = Taxe de vente provinciale (TVP) (facultatif)</p> <p>SP = Taxe d'État ou provinciale également appelée taxe de vente du Québec (TVQ) (facultatif)</p> <p>ST = Taxe de vente d'État (facultatif)</p> <p>TX = Toutes les taxes (obligatoire)</p> <p>VA = Taxe sur la valeur ajoutée aussi appelée taxe de vente harmonisée du Canada (TVH) (facultatif)</p>						

Requis	Variable	Nom du champ	Taille/Type	Description
C	txi02	Montant en numéraire	16 caractères décimaux	<p>Cet élément peut contenir le montant de la taxe en numéraire qui correspond au code de type de taxe du champ txi01.</p> <div style="border: 1px solid black; padding: 5px;"> <p><b>REMARQUE :</b> Si le champ txi02 est utilisé dans l'occurrence obligatoire de txi01=TX, txi02 doit contenir le montant total de la taxe applicable à la totalité de la facture (transaction).</p> <p>Si les taxes ne s'appliquent pas à l'ensemble de la facture (transaction), txi02 doit être égal à 0,00.</p> </div> <p>La valeur maximale qui peut être entrée dans ce champ est « 9 999,99 », soit 9 999,99 \$ CA.</p> <p>Un débit est entré comme suit : En 9999,99</p> <p>Un crédit est entré comme suit : En - 9999,99</p>
C	txi03	Pourcentage	10 caractères décimaux	Indique le pourcentage de taxe (sous forme décimale) qui correspond au code de type de taxe défini dans le

Requis	Variable	Nom du champ	Taille/Type	Description
				<p>champ txi01</p> <p>Jusqu'à 2 décimales sont supportées</p>
C	txi06	Code d'exonération fiscale	1 caractère alphanumérique	<p>Cet élément peut contenir le code d'exonération fiscale qui identifie l'état d'exonération des ventes ainsi que la taxe correspondant au code de type de taxe indiqué dans le champ txi01.</p> <p>Valeurs acceptées :</p> <ul style="list-style-type: none"> <li>1 = Oui (exonéré d'impôt)</li> <li>2 = Non (non exonéré d'impôt)</li> <li>4 = Non exonéré ou pour la revente</li> <li>A = Main d'œuvre imposable, matériel exonéré</li> <li>B = Matériaux taxables, main-d'œuvre exonérée</li> <li>C = Non imposable</li> <li>F = Exonéré (taxe sur les produits et services)</li> <li>G = Exonéré (taxe de vente provinciale)</li> <li>L = Service local exonéré</li> <li>R = Exonération périodique</li> <li>U = Utilisation exonérée</li> </ul>
Y	pam05	Montant final de l'article	10 caractères décimaux	Indique la valeur de l'article

Requis	Variable	Nom du champ	Taille/Type	Description
				<p>individuel qui est normalement calculée en multipliant le prix par la quantité</p> <p>Doit comporter 2 décimales</p> <p>Valeur minimale de 0,00 et maximale de 99 999,99</p>
Y	pid05	Description de la ligne d'article	80 caractères alphanumériques	<p>Description de la ligne d'article</p> <p>Décrit l'article individuel acheté</p> <p>Ce champ concerne chaque ligne de la transaction.</p>

Tableau 3 Champs de demande de niveaux 2 et 3 de Amex – Table3 – Champs de sommaire

Requis	Variable	Nom du champ	Taille/Type	Description
C	txi01	Code du type de taxe	2 caractères alphanumériques	<p>Valeurs acceptées :</p> <p>CA = Taxe municipale (facultatif)</p> <p>CT = Taxe de comté (facultatif)</p> <p>EV = Taxe environnementale (facultatif)</p> <p>GS = Taxe sur les biens et services (TPS) (facultatif)</p> <p>LS = Taxe de vente d'État et locale (facultatif)</p> <p>LT = Taxe de vente</p>

Requis	Variable	Nom du champ	Taille/Type	Description
				<p>locale (facultatif)</p> <p>PG = Taxe de vente provinciale (TVP) (facultatif)</p> <p>SP = Taxe d'État ou provinciale également appelée taxe de vente du Québec (TVQ) (facultatif)</p> <p>ST = Taxe de vente d'État (facultatif)</p> <p>TX = Toutes les taxes (obligatoire)</p> <p>VA = Taxe sur la valeur ajoutée aussi appelée taxe de vente harmonisée du Canada (TVH) (facultatif)</p>
C	txi02	Montant en numéraire	16 caractères décimaux	<p>Cet élément peut contenir le montant de la taxe en numéraire qui correspond au code de type de taxe du champ txi01.</p> <div style="border: 1px solid black; padding: 5px;"> <p><b>REMARQUE :</b> Si le champ txi02 est utilisé dans l'occurrence obligatoire de txi01=TX, txi02 doit contenir le montant total de la taxe applicable à la totalité de la facture (transaction).</p> <p>Si les taxes ne s'appliquent pas à l'ensemble de la facture (transaction), txi02 doit être égal à 0,00.</p> </div> <p>La valeur maximale qui peut être entrée dans</p>

Requis	Variable	Nom du champ	Taille/Type	Description
				<p>ce champ est « 9 999.99 », soit 9 999,99 \$ CA.</p> <p>Un débit est entré comme suit : En 9999,99</p> <p>Un crédit est entré comme suit : En - 9999,99</p>
C	txi03	Pourcentage	10 caractères décimaux	<p>Indique le pourcentage de taxe (sous forme décimale) qui correspond au code de type de taxe défini dans le champ txi01</p> <p>Jusqu'à 2 décimales sont supportées</p>
C	txi06	Code d'exonération fiscale	1 caractère alphanumérique	<p>Valeurs acceptées :</p> <p>1 = Oui (exonéré d'impôt)</p> <p>2 = Non (non exonéré d'impôt)</p> <p>4 = Non exonéré ou pour la revente</p> <p>A = Main d'œuvre imposable, matériel exonéré</p> <p>B = Matériaux taxables, main-d'œuvre exonérée</p> <p>C = Non imposable</p> <p>F = Exonéré (taxe sur les produits et services)</p>

Requis	Variable	Nom du champ	Taille/Type	Description
				<p>G = Exonéré (taxe de vente provinciale)</p> <p>L = Service local exonéré</p> <p>R = Exonération périodique</p> <p>U = Utilisation exonérée</p>

## A.9 Définition des champs de demande – 3-D Secure 2.2

Variable	Type et limites	Description
Adresse de facturation billAddress1	<i>Chaîne</i> 50 caractères alphanumériques	Adresse de facturation du titulaire de carte
Ville de facturation billCity	<i>Chaîne</i> 50 caractères alphanumériques	Ville de facturation du titulaire de carte
Pays de facturation billCountry	<i>Chaîne</i> 3 caractères alphanumériques	Correspond à un code de pays de 3 chiffres de la norme ISO 3166-1.
Code postal de facturation billPostalCode	<i>Chaîne</i> 16 caractères alphanumériques	Code postal de facturation du titulaire de la carte
Province de facturation billProvince	<i>Chaîne</i> 3 caractères alphanumériques	Défini dans la sous-division du pays de la norme ISO 3166-2
Java activé dans le navigateur	<i>Chaîne</i>	Indique si Java est activé dans le navigateur

Variable	Type et limites	Description
browserJavaEnabled	1 caractère alphabétique	Valeurs acceptées : T = Vrai F = Faux
Langue du navigateur browserLanguage	<i>Chaîne</i> 8 caractères alphanumériques	Comme défini dans IETF BCP47
Hauteur de la fenêtre du navigateur browserScreenHeight	<i>Chaîne</i> 6 caractère numérique	Hauteur en pixels de l'écran du titulaire de carte
Largeur de la fenêtre du navigateur browserScreenWidth	<i>Chaîne</i> 6 caractère numérique	Largeur en pixels de l'écran du titulaire de carte
Agent utilisateur du navigateur browserUserAgent	<i>Chaîne</i> 2048 caractères alphanumériques	Agent utilisateur du navigateur
Nom du titulaire de carte cardholderName	<i>Chaîne</i> 45 caractères alphanumériques  <b>REMARQUE :</b> Les caractères accentués ne sont pas autorisés.	Nom du titulaire de carte
Taille de la fenêtre challengeWindowsize	<i>Chaîne</i> 2 caractères alphanumériques	Concerne le rendu de la contestation du serveur de contrôle d'accès (ACS) dans le navigateur.  Valeurs acceptées : 01 = 250 x 400 02 = 390 x 400

Variable	Type et limites	Description
		03 = 500 x 600 04 = 600 x 400 05 = Écran complet
cres	<i>Chaîne</i>	Données de réponse de la contestation
cres	200 caractères alphanumériques	
Devise currency	<i>Chaîne</i> 3 caractère numérique	Code de devise à 3 chiffres ISO 4217 (CAD = 124, USD = 840)
<b>REMARQUE :</b> Ce champ ne devrait pas être envoyé, à moins que la tarification multidevise soit activée dans votre compte de commerçant.		
ID de transaction DS dsTransId	<i>Chaîne</i> 36 caractères alphanumériques	Identifiant de transaction universellement unique attribué par le DS pour identifier une transaction unique
<b>REMARQUE:</b> Uniquement utilisé dans les transactions financières utilisant les services 3-D Secure d'un tiers		
Courriel email	<i>Chaîne</i> 254 caractères alphanumériques	Adresse électronique du titulaire de la carte
URL de notification notificationUrl	<i>Chaîne</i> 256 caractères alphanumériques	URL du site Web qui recevra la réponse du serveur de contrôle d'accès (ACS) concernant la conclusion de la méthode 3DS
Demande de contestation requestChallenge	<i>Chaîne</i> 1 caractère alphabétique	Indique si une contestation est demandée pour cette transaction  Valeurs acceptées :  Y = Oui  N = Non

Variable	Type et limites	Description
Type de demande requestType	<i>Chaîne</i>  2 caractères alphanumériques	Valeurs acceptées :  01 = Paiement entamé par le titulaire de la carte 02 = Périodique  03 = transaction effectuée par versements  04 = ajout d'une carte  05 = gestion de la carte  06 = vérification du titulaire de la carte dans le cadre de l'identification et de la vérification du jeton EMV
Adresse d'expédition shipAddress1	<i>Chaîne</i>  50 caractères alphanumériques	Adresse d'expédition
Ville d'expédition shipCity	<i>Chaîne</i>  50 caractères alphanumériques	Ville d'expédition
Pays d'expédition shipCountry	<i>Chaîne</i>  3 caractères alphanumériques	Pays de destination  Correspond à un code de pays de 3 chiffres ISO 3166-1.
Code postal d'expédition shipPostalCode	<i>Chaîne</i>  16 caractères alphanumériques	Code postal d'expédition
Province d'expédition shipProvince	<i>Chaîne</i>  3 caractères alphanumériques	Province d'expédition  Défini dans la sous-division du pays de la norme ISO 3166-2

Indicateur de conclusion 3DS threedsCompletionInd	<p><i>Chaîne</i></p> <p>1 caractère alphanumérique</p>	<p>Indique si le processus de MpICardLookup de la méthode 3ds s'est déroulée avec succès</p> <p>Valeurs acceptées :</p> <p>Y = Conclu avec succès</p> <p>N = N'a pas été conclu avec succès</p> <p>U = Non disponible</p>
browser IP Address  <browser_ip>	<p><i>Chaîne</i></p> <p>Permet l'utilisation des caractères « . » et « : ».</p> <p>45 caractères alphanumériques</p>	<p>Adresse IP du navigateur telle qu'elle est renvoyée par les en-têtes HTTP au demandeur 3DS.</p> <p>{b}REMARQUE : {/b}ce champ n'est pas obligatoire, mais vous devez tout de même le définir. Il est fortement recommandé de fournir cette information. Le fait de ne pas fournir la valeur de ce champ peut augmenter le risque de refus.</p>
cardholder work phone number  <work_phone>	<p><i>Objet</i></p> <p>S.O.</p>	<p>Numéro de téléphone au travail du titulaire de la carte</p> <p>{b}REMARQUE : {/b}ce champ n'est pas obligatoire, mais vous devez tout de même le définir. Il est fortement recommandé de fournir au moins un des numéros de téléphone du titulaire de la carte. Le fait de ne pas fournir au moins un des numéros de téléphone du titulaire de la carte peut augmenter le risque de refus.</p> <p>{b}REMARQUE : {/b}il s'agit d'un objet imbriqué dans la transaction. Pour plus d'information sur les champs de l'objet Cardholder Phone Number Info, consultez la section Objet Cardholder Phone Number Info et variables.</p>
cardholder home phone number  <home_phone>	<p><i>Objet</i></p> <p>S.O.</p>	<p>Numéro de téléphone à domicile du titulaire de la carte</p> <p>{b}REMARQUE : {/b}ce champ n'est pas obligatoire, mais vous devez tout de même le définir. Il est fortement recommandé de fournir au moins un des numéros de téléphone du titulaire de la carte. Le fait de ne pas fournir au moins un des numéros de téléphone du titulaire de la carte peut augmenter le risque de refus.</p> <p>{b}REMARQUE : {/b}il s'agit d'un objet imbriqué dans la transaction. Pour plus</p>

		d'information sur les champs de l'objet Cardholder Phone Number Info, consultez la section Objet Cardholder Phone Number Info et variables.
cardholder mobile phone number  <mobile_phone>	<i>Objet</i>  S.O.	<p>Numéro de téléphone cellulaire du titulaire de la carte</p> <p>{b}REMARQUE : {/b}ce champ n'est pas obligatoire, mais vous devez tout de même le définir. Il est fortement recommandé de fournir au moins un des numéros de téléphone du titulaire de la carte. Le fait de ne pas fournir au moins un des numéros de téléphone du titulaire de la carte peut augmenter le risque de refus.</p> <p>{b}REMARQUE : {/b}il s'agit d'un objet imbriqué dans la transaction. Pour plus d'information sur les champs de l'objet Cardholder Phone Number Info, consultez la section Objet Cardholder Phone Number Info et variables.</p>

#### Numéro de téléphone du titulaire de la carte 3DS pour les modules d'extension des commerçants

Variable	Type et limites	Description
country code  <country_code>	<i>Chaîne</i>  3 caractères numériques	Code de pays du numéro de téléphone fourni par le titulaire de la carte.
phone number  <phone_number>	<i>Chaîne</i>  15 caractères numériques	Le numéro de téléphone fourni par le titulaire de la carte.

## A.10 Définition des champs de demande – TMD

Variable	Type et limites	Description
Numéro de la version de la TMD	<p><i>Chaîne</i></p> <p>Valeur numérique</p> <p>La version actuelle est 1.0</p>	Numéro de version de la TMD
Montant du titulaire de carte	<p><i>Chaîne</i></p> <p>12 caractère numérique</p> <p>La plus petite unité discrète de devise étrangère</p>	Montant, en unités de devise étrangère, qui sera facturé au titulaire de la carte pour la transaction
Code de devise du titulaire de carte	<p><i>Chaîne</i></p> <p>3 caractère numérique</p>	Code ISO représentant la devise étrangère du titulaire de la carte

### Champs facultatifs de la TMD

Variable	Type et limites	Description
Jeton du taux de la TMD	<p><i>Chaîne</i></p> <p>S. O.</p>	Jeton représentant un taux de change temporairement fixé, obtenu en réponse à la transaction de taux d'obtention de la TMD et utilisé dans les demandes ultérieures de transactions financières de la TMD dans le but de réclamer ce taux.

### Champs de la demande de la transaction du taux d'obtention de la TMD

Variable	Type et limites	Description
Numéro de la version de la TMD	<p><i>Chaîne</i></p> <p>Valeur numérique</p> <p>La version actuelle est 1.0</p>	Numéro de version de la TMD
Type de transaction de taux	<i>Chaîne</i>	Valeur représentant le type de demande de transaction ultérieure

Variable	Type et limites	Description
	1 caractère alphabétique	<p>pour laquelle le jeton de taux sera utilisé</p> <p>Valeurs acceptées :</p> <p>P = Achat</p> <p>R = Remboursement</p>
Renseignements sur le taux de la TMD	<i>Objet</i>  S. O.	Objet imbriqué dans la transaction du taux d'obtention de la TMD contenant les champs <b>add cardholder amount</b> et <b>add merchant settlement</b>

#### Champs de demande de l'objet MCP Rate Info

Au moins une des variables suivantes doit être envoyée :

Variable	Type et limites	Description
Ajouter le montant du titulaire de carte	<i>Tableau chaîne</i>  12 caractères numériques, 3 caractères numériques  (la plus petite unité discrète de devise étrangère, code de devise)	Un tableau de chaînes de caractères représentant :  Le montant, en unités de devise étrangère, qui sera facturé au titulaire de la carte Le code de devise ISO correspondant à la devise étrangère du titulaire de la carte
Ajouter le montant de règlement du commerçant	<i>Tableau chaîne</i>  12 caractères numériques, 3 caractères numériques  (montant en centimes de dollars canadiens, code de devise)	Un tableau de chaînes de caractères représentant :  Le montant que le commerçant recevra dans la transaction, en dollars canadiens Le code de devise ISO correspondant à la devise étrangère du titulaire de la carte

## A.11 Définition des champs de demande – Offlinx<sup>MC</sup>

S'applique uniquement à l'intégration d'Offlinx<sup>MC</sup>

Variable	Type et limites	Description
ID de correspondance de carte	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p>	<p>Correspond à l'ID de transaction utilisé pour la fonction Offlinx<sup>MC</sup> Match pour le pixel invisible; il s'agit d'un identifiant unique créé par le commerçant.</p> <p>Doit être unique pour chaque transaction</p>

## A.12 Définition des champs de demande – Frais de commodité

Variable	Type et limites	Description
Information sur les frais de commodité	<p><i>Objet</i></p> <p>S. O.</p>	Contient des champs liés à la fonction de frais de commodité
Montant des frais de commodité	<p><i>Chaîne</i></p> <p>9 caractères décimaux</p>	Montant en dollars facturé au client en tant que frais de commodité

## A.13 Définition des champs de demande – Transaction périodiques

### Champs de demande pour l'objet Recurring Billing Info

Variable	Type et limites	Description
Nombre d'occurrences	<i>Chaîne</i> Valeur numérique 1 ou 999	Il s'agit du nombre d'occurrences de la transaction.
Période	<i>Chaîne</i> Valeur numérique 1 ou 999	Il s'agit du nombre d'intervalles de la variable recur unit qui doivent s'écouler entre chaque facturation périodique.
Date de début	<i>Chaîne</i> Format AAAAMMMJJ	Il s'agit de la date de la première transaction périodique future (la date doit être future).  Si un montant additionnel est facturé immédiatement, la variable start now doit être réglée à true.
Commencer maintenant	<i>Chaîne</i> true/false	Réglez cette variable à true si un montant est porté immédiatement à la carte. Autrement, réglez la variable à false.  Lorsque la variable est réglée à false, effectuez une transaction de vérification de carte avant d'envoyer l'achat avec les objets Recurring Billing et Credential on File.
		<b>REMARQUE :</b> Le montant à facturer immédiatement peut différer des montants subséquents.
Montant récurrent	<i>Chaîne</i> 10 caractères décimaux, minimum de 3 chiffres Jusqu'à 7 chiffres (dollars) +	Il s'agit du montant en dollars de la transaction périodique.  Il s'agit du montant facturé à la date de départ (start_date) et qui sera ensuite facturé à répétition en

<p>point décimal (.) + 2 chiffres (sous) après le point décimal</p> <p><b>EXEMPLE : 1 234 567,89</b></p>	<p>fonction de l'intervalle défini par les valeurs period et recur unit.</p>
--	--

<p>Unité répétée</p>	<p><i>Chaîne</i></p> <p>Jour, semaine, mois ou fin du mois</p>	<p>Il s'agit de l'unité utilisée comme base pour l'intervalle.</p> <p>Elle fonctionne avec la variable period pour déterminer la fréquence de facturation.</p>
----------------------	--	--

## A.14 Définition des champs de demande – Renseignements d'identification au dossier

Variable	Type et limites	Description
Information sur le versement	<i>Objet</i> S. O.	Contient les champs de demande liés aux versements
ID du plan de versements	<i>Chaîne</i>  36 caractères alphanumériques  Longueur fixe	Identifiant généré par la marque de carte pour un plan de versements
Référence du plan de versements	<i>Chaîne</i>  10 caractères alphanumériques  Longueur fixe	Nom unique et humain du plan de versements
Version des modalités	<i>Chaîne</i>  10 caractères alphanumériques  Longueur variable (de 1 à 10 caractères)	Version des modalités du plan de versements accepté par le titulaire de carte  Cette version augmente automatiquement chaque fois que le plan est mis à jour par l'émetteur.

## A.15 Définition des champs de la demande – Objet Account Name Verification

Champs de demande dans l'objet Account Name Verification. L'objet ne peut être inclus que dans les transactions de type Card Verification. La vérification du nom du compte ne s'applique qu'aux cartes de crédit Visa.

Variable	Type et limites	Description
First Name <first_name>	<i>Chaîne</i> 32 caractères alphanumériques	Prénom du titulaire de la carte
Middle Name <middle_name>	<i>Chaîne</i> 32 caractères alphanumériques	Deuxième prénom du titulaire de la carte
Last Name <last_name>	<i>Chaîne</i> 32 caractères alphanumériques	Nom de famille du titulaire de la carte

## Annexe B Définition des champs de réponse

Tableau 28 : Valeurs de réponse de l'objet Receipt

	Type	Limites	Méthode Get
	Description		
<b>Champs de réponse généraux</b>			
Type de carte	Chaîne	2 caractères alphabétique (1 caractère minimum)	<code>receipt.getCardType();</code>
Représente le type de carte utilisée dans la transaction, par exemple, Visa ou Mastercard			
Valeurs possibles :			
V = Visa			
M = Mastercard			
AX = American Express			
DC = Diner's Card			
NO = Novus ou Discover			
SE = Sears			
D = Débit			
C1 = JCB			
Montant de la transaction	Chaîne	10 caractères décimaux	<code>receipt.getTransAmount();</code>
		Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal	
		EXEMPLE : 1 234 567,89	
Montant de la transaction qui a été traitée			
Numéro de transaction	Chaîne	255 caractères alphanumériques	<code>receipt.getTxnNumber();</code>
		Identificateur de transaction de Passerelle Moneris souvent nécessaire pour les transactions de suivi (telles que le remboursement et la correction d'achat) afin de renvoyer à la transaction originale	
ID de reçu	Chaîne	50 caractères alphanumériques	<code>receipt.getReceiptId();</code>
		Numéro de commande qui a été précisé dans la demande de transaction	
Type de transaction	Chaîne	2 caractères alphanumériques	<code>receipt.getTransType();</code>

	Type	Limites	Méthode Get
	Description		
	0 = Achat 1 = Préautorisation 2 = Conclusion 4 = Remboursement 11 = Nul		
Numéro de référence	Chaîne	18 caractère numérique	<code>receipt.getReferenceNum();</code>
		Terminal utilisé pour traiter la transaction ainsi que numéro du quart, du lot et de la séquence. Ces données sont généralement référence aux transactions sur les systèmes hôtes, et doivent être affichées sur tous les reçus présentés au client.  Ces informations doivent être enregistrées par le commerçant. Par exemple : En 660123450010690030 66012345 : ID du terminal 001 : Numéro du quart 069 : Numéro de lot 003 : Numéro de la transaction dans le lot	
Code de réponse	Chaîne	3 caractère numérique	<code>receipt.getResponseCode();</code>
		<ul style="list-style-type: none"> <li>• Inférieur à 50 : Transaction approuvée</li> <li>• Supérieur ou égal à 50 : Transaction refusée</li> <li>• Null : Transaction incomplète</li> </ul> <p>Pour plus de détails sur les codes de réponse, consultez le document sur les codes de réponse qui se trouve dans le portail pou (<a href="https://developer.moneris.com">https://developer.moneris.com</a>).</p>	
ISO	Chaîne	2 caractère numérique	<code>receipt.getISO();</code>
	Code de réponse ISO		
Totaux de banques	Objet		<code>receipt.getBankTotals();</code>
		Données de réponse reçues dans une demande de fermeture de lot et de totaux des lots ouverts Consultez l'Annexe B Définition des champs de réponse à la page 564.	
Message	Chaîne	100 caractères alphanumériques	<code>receipt.getMessage();</code>
		Description de la réponse renvoyée par l'émetteur  Le message renvoyé par l'émetteur est destiné à informer le commerçant uniquement, il <b>ne doit pas</b> être affiché sur les reçus	
Code d'autorisation	Chaîne	8 caractères alphanumériques	<code>receipt.getAuthCode();</code>
		Code d'autorisation reçu de l'institution émettrice	
Complete	Chaîne	true/false	<code>receipt.getComplete();</code>

	Type	Limites	Méthode Get
	Description		
La transaction a été envoyée au serveur d'autorisation, et une réponse a été reçue.			
Date de transaction	Chaîne	Format : aaaa-mm-jj	receipt.getTransDate();
Marquage de la date du serveur de traitement			
Heure de la transaction (transaction time)	Chaîne	Format : ##:##:##	receipt.getTransTime();
Horodateur du serveur de traitement			
Billet	Chaîne	S. O.	receipt.getTicket();
Champ réservé			
Délai écoulé	Chaîne	true/false	receipt.getTimedOut();
La transaction a échoué en raison d'un délai trop long.			
Visa débit	Chaîne	true/false	receipt.getIsVisaDebit();
Indique si la carte traitée est une carte Visa Débit.			

PBBLifeCycleTraceID	Chaîne	15 caractères alphanumériques	
Identifiant unique d'une transaction provenant des systèmes Interac Direct. S'applique uniquement aux transactions Interac Direct et est utilisé pour établir un lien avec les transactions suivantes.			
Account Name Verification Result	Chaîne	10 caractères alphanumériques	receipt.getAccountNameResult();

<pre>&lt;AccountNameVerificationResult&gt;</pre>	<p>Code indiquant les résultats de la vérification du nom du compte Visa.</p> <p>Positions 1 et 2 : état général de la demande.</p> <p>Positions 3 et 4 : état de concordance du nom complet</p> <p>Positions 5 et 6 : état de concordance du nom de famille</p> <p>Positions 7 et 8 : état de concordance du deuxième prénom</p> <p>Positions 9 et 10 : état de concordance du prénom</p> <p>Valeurs de l'état de la demande :</p> <p>00 = vérification de la concordance du nom effectuée</p> <p>01 = vérification de la concordance du nom non effectuée</p> <p>02 = vérification de la concordance du nom non prise en charge</p> <p>Valeurs pour la concordance du nom complet et la concordance du nom de famille, du deuxième prénom et du prénom :</p> <p>01 = Concordance complète</p> <p>50 = Concordance partielle</p> <p>99 = Aucune concordance</p>
<b>Batch Close/Open Totals response fields</b>	

### Champs de réponse de fermeture ou d'ouverture de lot

Types de cartes traitées	Tableau chaînes	S. O.	<code>receipt.getCreditCards(ecr_no);</code>
Renvoie tous les types de cartes traités dans le lot actuel pour l'ID du terminal et le numéro CEE de la demande			
ID du terminal	Chaîne	8 caractères alphanumériques	Code à venir  <code>receipt.getTerminalIDs();</code>
Retourne l'ID du terminal et le numéro CEE de la demande			
Nombre d'achats	Chaîne	4 caractères numériques	<code>receipt.getPurchaseCount(ecr, cardType);</code>
Indique le nombre de transactions d'achat, de conclusion de préautorisation ou les transactions forcées effectuées.			
Si aucun élément n'a été traité dans le lot, la valeur renvoyée sera 0000.			
Montant de l'achat	Chaîne	11 caractères alphanumériques	<code>receipt.getPurchaseAmount(ecr, cardType);</code>

---

Indique le montant en dollars traité pour les transactions d'achat, de conclusion de préautorisation ou les transactions forcées commence par un + et est suivi de 10 chiffres, les 8 premiers indiquent le montant et les 2 derniers la valeur en centièmes.

**EXEMPLE :** +0000000000 = 0,00 et +0000041625 = 416,25

---

	Type	Limites	Méthode Get
	Description		
Nombre de remboursements	Chaîne	4 caractères numériques	<code>receipt.getRefundCount(ecr, cardType);</code>
Indique le nombre de transactions de remboursement ou de remboursement indépendant traitées. Si aucun élément n'a été traité dans le lot, la valeur renvoyée sera 0000.			
Montant du remboursement	Chaîne	11 caractères alphanumériques	<code>receipt.getRefundAmount(ecr, cardType);</code>
Indique le montant en dollars du traitement des transactions de remboursement, de remboursement indépendant ou de crédit compensation automatisée Ce champ commence par un + et est suivi de 10 chiffres, les 8 premiers indiquent le montant et les en centièmes.  Exemple : +0000000000 = 0,00 et +0000041625 = 416,25			
Nombre de corrections d'achat	Chaîne	4 caractères numériques	<code>receipt.getCorrectionCount(ecr, cardType);</code>
Indique le nombre de transactions de correction d'achat traitées Si aucun élément n'a été traité dans le lot, la valeur renvoyée sera 0000.			
Montant de la correction d'achat (correction amount)	Chaîne	11 caractères alphanumériques	<code>receipt.getCorrectionAmount(ecr, cardType);</code>
Indique le montant en dollars des transactions de correction d'achat traitées Ce champ commence par un + et est suivi de 10 chiffres, les 8 premiers indiquent le montant et les 2 derniers la valeur en centièmes.  EXEMPLE : +0000000000 = 0,00 et +0000041625 = 416,25			

**Champs de réponse pour la facturation périodique (consulter l'annexe A, à la page 1)**

Réussite de la répétition	Chaîne	true/false	<code>receipt.getRecurSuccess();</code>
Indique si la transaction de facturation périodique a été configurée avec succès pour une facturation future			
Réussite de la mise à jour de la facturation périodique	Chaîne	true/false	<code>receipt.getRecurUpdateSuccess();</code>
Indique la réussite de la mise à jour de la facturation périodique			
Prochaine date de l'occurrence	Chaîne	aaaa-mm-jj	<code>receipt.getNextRecurDate();</code>
Indique la prochaine date de facturation périodique			
Date de fin de la répétition	Chaîne	aaaa-mm-jj	<code>receipt.getRecurEndDate();</code>
Indique la dernière date de facturation périodique			

**Champs de réponse pour la vérification de l'état (consulter )**

Code d'état	Chaîne	3 caractères alphanumériques	<code>receipt.getStatusCode();</code>
-------------	--------	------------------------------	---------------------------------------

	Type	Limites	Méthode Get
	Description		
Inférieur à 50 : Transaction trouvée et réussie Supérieur ou égal à 50 : Transaction non trouvée et échouée			
<b>REMARQUE :</b> Le code d'état n'est généré que si la valeur <b>true</b> est attribuée à la propriété Status Check de l'objet de connexion.			
Message d'état	Chaîne	found (trouvé) ou not found (non trouvé)	receipt.getStatusMessage () ;
Trouvé : Le code d'état se trouve entre 0 et 49, inclusivement. Non trouvé ou nul : Le code d'état se trouve entre 50 et 999, inclusivement.			
<b>REMARQUE :</b> Le message d'état n'est généré que si la valeur <b>true</b> est attribuée à la propriété Status Check de l'objet de connexion.			
Champs de réponse du SVA (consulter la section 9.1, à la page 388)			
Code de réponse du SVA	Chaîne	1 caractère alphanumérique	receipt.getAvsResultCode () ;
Indique le résultat de la vérification de l'adresse Pour une liste complète des codes de réponse possibles, consultez l'Annexe B			
Champs de réponse du NVC (consulter )			
Code de résultat du code de vérification d'authentification du titulaire de carte	Chaîne	2 caractères alphanumériques	receipt.getCvdresultCode () ;
Indique le résultat de la validation du NVC Le premier octet est l'indicateur numérique du NVC envoyé avec la demande, et le deuxième octet est le code de réponse. Les codes de réponse possibles sont indiqués à l'annexe B			
Champs de réponse du jeton GooglePay			
Mode de paiement GooglePay	String	4 caractères alphanumériques	receipt.getGooglepayPaymentMethod () ;
Ceci indique si la carte sous-jacente utilisée dans le portefeuille électronique Google Pay est le FPAN ou un DPAN. Si un GoogleTokenTempAdd renvoie un FPAN, vous pouvez effectuer une authentification 3DS avec celui-ci; s'il renvoie un DPAN, l'authentification 3DS n'est pas nécessaire.			
Champs de réponse des modules d'extension pour les commerçants (consulter la section« MPI » à la page 1)			
Type	Chaîne	99 caractères alphanumériques	
Le message VERes, PARes ou erreur définit le type de réponse que vous recevez.			
Réussite	Valeur booléenne	true/false	receipt.getMpiSuccess () ;
True si la tentative a été réussie, false si la tentative a échoué			
Message	Chaîne	100 caractère alphabétique	receipt.getMpiMessage () ;

	Type	Limites	Méthode Get
	Description		
		<p>Les transactions MPI TXN peuvent produire les valeurs suivantes :</p> <p>Y : Crée une fenêtre contextuelle du formulaire de vérification Vérifié par Visa</p> <p>N : Envoie l'achat ou la préautorisation avec la valeur crypt type = 6</p> <p>U : Envoie l'achat ou la préautorisation avec la valeur crypt type = 7</p> <p>Les transactions MPI ACS peuvent produire les valeurs suivantes :</p> <p>Y ou A : (Également receipt.getMpiSuccess () =true) Continuer avec la transaction d'achat ou de préautorisation utilisant la vérification d'authentification du titulaire de carte (CAVV).</p> <p>N : Échec de l'authentification ou transaction à haut risque Il est recommandé de ne pas poursuivre la transaction.</p> <p>Selon la tolérance au risque du commerçant et les résultats d'autres méthodes de détection de la fraude, la transaction peut être effectuée avec une valeur crypt type = 7.</p> <p>U ou time out : Envoie l'achat ou la préautorisation avec la valeur crypt type = 7.</p>	
Term URL	Chaîne	255 caractères alphanumériques	
	URL à laquelle la valeur PArEs est renvoyée		
MD	Chaîne	1024 caractères alphanumériques	
	Données définies par le commerçant qui ont été renvoyées		
ACS URL	Chaîne	255 caractères alphanumériques	
	URL utilisée pour la fenêtre contextuelle générée		
Code de vérification d'authentification du titulaire de carte	Chaîne	28 caractères alphanumériques	receipt.getMpiCavv () ;
	Données d'authentification Vérifié par Visa, Mastercard SecureCode et American Express SafeKey		
Indicateur de module d'extension pour les commerçants de commerce électronique	Chaîne	1 caractère alphanumérique	receipt.getMPIEci () ;
Codes de résultat liés au code de vérification d'authentification du titulaire de carte (CAVV)	Chaîne	1 caractère alphanumérique	receipt.getCavvresultCode () ;

Type	Limites	Méthode Get
Description		
Indique le résultat du code de vérification d'authentification du titulaire de carte (CAVV) Visa Pour plus de renseignements, voir Codes de réponse du code de vérification d'authentification du titulaire de carte pour Vérifié par Visa.		
0 = Résultats de l'authentification du CAVV non valides 1 = Échec de la validation du CAVV; authentification 2 = Réussite de la validation du CAVV; authentification 3= Réussite de la validation du CAVV; tentative 4 = Échec de la validation du CAVV; tentative 7 = Échec de la validation du CAVV; tentative (cartes émises aux États-Unis uniquement) 8 = Réussite de la validation du CAVV; tentative (cartes émises aux États-Unis uniquement)		
Le code de résultat du CAVV indique le résultat de la validation du CAVV.		
Formulaire du module d'extension pour les commerçants en ligne		receipt.getMpiInLineForm();
Champs de réponse de la chambre forte (consulter la section 4.1, à la page 64)		
Clé de données	Chaîne	28 caractères alphanumériques
Le champ de réponse data key (clé de données) est rempli lorsque vous envoyez les transactions suivantes : Ajout d'une carte de crédit à la chambre forte (ResAddCC) (page 68), Ajout d'une carte de crédit chiffrée à la chambre forte – EncResAddCC (page 69), Transformation en jeton d'une carte de crédit dans la chambre forte (ResTokenizeCC)(page 96), Ajout d'un jeton temporaire à la chambre forte (ResTempAdd)(page 75), Ajout d'un jeton à la chambre forte (ResAddToken) (page 93). La clé de données est l'identifiant du profil que toutes les transactions finan utilisant la chambre forte utiliseront pour accéder aux renseignements enregistrés.		
Type de paiement associé à la chambre forte	Chaîne	cc
Indique le type de paiement associé à un profil de chambre forte		
Type de paiement pour la carte arrivant à expiration	Chaîne	cc
Indique le type de paiement associé à un profil de chambre forte. Applicable au type de transaction d'obtention des cartes expirées dans la chambre forte		
Numéro de carte de crédit masqué par la chambre forte	Chaîne	20 caractère numérique
Renvoie les 4 premiers ou les 4 derniers chiffres du numéro de carte enregistré dans le profil		
Numéro de carte de crédit masqué de la carte arrivant à expiration	Chaîne	20 caractère numérique
receipt.getExpMaskedPan();		

Renvoie les 4 premiers ou les 4 derniers chiffres du numéro de carte enregistré dans le profil Applicable au type de transaction d'obtention des cartes expirées dans la chambre forte

---

	Type	Limites	Méthode Get
	Description		
Réussite de la chambre forte	Chaîne	true/false	<code>receipt.getResSuccess();</code>
Indique si la transaction de la chambre forte a été réussie			
ID de client dans la chambre forte	Chaîne	30 caractères alphanumériques	<code>receipt.getResCustId();</code>
Renvoie l'ID de client enregistré dans le profil			
ID de client lié à la carte arrivant à expiration	Chaîne	30 caractères alphanumériques	<code>receipt.getExpCustId();</code>
Renvoie l'ID de client enregistré dans le profil. Applicable au type de transaction d'obtention des cartes expirées dans la chambre forte			
Numéro de téléphone dans la chambre forte	Chaîne	30 caractères alphanumériques	<code>receipt.getResPhone();</code>
Renvoie le numéro de téléphone enregistré dans le profil			
Numéro de téléphone de la carte arrivant à expiration	Chaîne	30 caractères alphanumériques	<code>receipt.getExpPhone();</code>
Renvoie le numéro de téléphone enregistré dans le profil. Applicable au type de transaction d'obtention des cartes expirées dans la chambre forte			
Adresse courriel dans la chambre forte	Chaîne	30 caractères alphanumériques	<code>receipt.getResEmail();</code>
Renvoie l'adresse courriel enregistrée dans le profil			
Adresse courriel de la carte arrivant à expiration	Chaîne	30 caractères alphanumériques	<code>receipt.getExpEmail();</code>
Renvoie l'adresse courriel enregistrée dans le profil. Applicable au type de transaction d'obtention des cartes expirées dans la chambre forte			
Note dans la chambre forte	Chaîne	30 caractères alphanumériques	<code>receipt.getResNote();</code>
Renvoie la note enregistrée dans le profil			
Note de la carte arrivant à expiration	Chaîne	30 caractères alphanumériques	<code>receipt.getExpNote();</code>
Renvoie la note enregistrée dans le profil. Applicable au type de transaction d'obtention des cartes expirées dans la chambre forte			
Date d'expiration dans la chambre forte	Chaîne	4 caractères numériques	<code>receipt.getResExdate();</code>
Renvoie la date d'expiration du numéro de carte enregistré dans le profil Format AAMM			
Date d'expiration de la carte arrivant à expiration	Chaîne	4 caractères numériques	<code>receipt.getExpExdate();</code>

	Type	Limites	Méthode Get
	Description		
		Renvoie la date d'expiration du numéro de carte enregistré dans le profil Format AAMM Applicable au type de transaction d'obtention des cartes expirées dans la chambre forte	
Indicateur de commerce électronique dans la chambre forte	Chaîne	1 caractère numérique	receipt.getResCryptType();
		Renvoie l'indicateur de commerce électronique enregistré dans le profil	
Indicateur de commerce électronique de la carte arrivant à expiration	Chaîne	1 caractère numérique	receipt.getExpCryptType();
		Renvoie l'indicateur de commerce électronique enregistré dans le profil. Applicable au type de transaction d'obtention des cartes expirées dans la chambre forte	
Numéro d'immeuble pour le SVA dans la chambre forte	Chaîne	19 caractères alphanumériques	receipt.getResAvsStreetNumber();
		Renvoie le numéro d'immeuble pour le SVA enregistré dans le profil. Si aucun autre numéro d'immeuble pour le SVA n'est transmis dans la demande de transaction, cette valeur sera soumise à l'émetteur avec la transaction financière.	
Numéro d'immeuble pour le SVA de la carte arrivant à expiration	Chaîne	19 caractères alphanumériques	receipt.getExpAvsStreetNumber();
		Renvoie le numéro d'immeuble pour le SVA enregistré dans le profil. Si aucun autre numéro d'immeuble pour le SVA n'est transmis dans la demande de transaction, cette valeur sera soumise à l'émetteur avec la transaction financière. Applicable au type de transaction d'obtention des cartes expirées dans la chambre forte	
Nom de rue pour le SVA dans la chambre forte	Chaîne	19 caractères alphanumériques	receipt.getResAvsStreetName();
		Renvoie le nom de rue pour le SVA enregistré dans le profil. Si aucun autre numéro d'immeuble pour le SVA n'est transmis dans la demande de transaction, cette valeur sera soumise à l'émetteur avec la transaction financière.	
Nom de rue pour le SVA de la carte arrivant à expiration	Chaîne	19 caractères alphanumériques	receipt.getExpAvsStreetName();
		Renvoie le nom de rue pour le SVA enregistré dans le profil. Si aucun autre numéro d'immeuble pour le SVA n'est transmis dans la demande de transaction, cette valeur sera soumise à l'émetteur avec la transaction financière. Applicable au type de transaction d'obtention des cartes expirées dans la chambre forte	
Code postal pour le SVA dans la chambre forte	Chaîne	9 caractères alphanumériques	receipt.getResAvsZipcode();
		Renvoie le code postal ou le code ZIP pour le SVA enregistré dans le profil. Si aucun autre numéro d'immeuble pour le SVA n'est transmis dans la demande de transaction, cette valeur sera soumise à l'émetteur avec la transaction financière.	
Code postal pour le SVA de la carte arrivant à expiration	Chaîne	9 caractères alphanumériques	receipt.getExpAvsZipcode();
		Renvoie le code postal ou le code ZIP pour le SVA enregistré dans le profil. Si aucun autre numéro d'immeuble pour le SVA n'est transmis dans la demande de transaction, cette valeur sera soumise à l'émetteur avec la transaction financière. Applicable au type de transaction d'obtention des cartes expirées dans la chambre forte	

	Type	Limites	Méthode Get
	Description		
Numéro de carte de crédit dans la chambre forte	Chaîne	20 caractère numérique	<code>receipt.getResPan();</code>
Renvoie le numéro complet de la carte de crédit enregistré dans le profil de la chambre forte. S'applique aux transactions de recherche d'un numéro complet dans la chambre forte uniquement			
Carte d'entreprise	Chaîne	true/false	<code>receipt.getCorporateCard();</code>
Indique si la carte associée au profil de la chambre forte est une carte d'entreprise			
<b>Champs de réponse pour la bande magnétique chiffrée (consulter la Section 1, à la page 1)</b>			
Numéro de carte de crédit masqué	Chaîne	20 caractères alphanumériques	<code>receipt.getMaskedPan();</code>
<b>Champs de réponse pour les frais de commodité (consulter l'annexe A, à la page 1)</b>			
Réussite des frais de commodité	Chaîne	true/false	<code>receipt.getCfSuccess();</code>
Indique si la transaction de frais de commodité a été traitée avec succès			
État des frais de commodité	Chaîne	3 caractères alphanumériques	<code>receipt.getCfStatus();</code>
Indique l'état des transactions du commerçant et des frais de commodité. Le champ CfStatus fournit des détails sur le comportement des transactions et doit être cité lorsque vous contactez l'équipe du service la clientèle de Moneris.			
Valeurs possibles :			
1 ou 1F = Première transaction d'achat conclue			
2 ou 2F = Deuxième transaction d'achat conclue			
3 = Transaction d'annulation conclue			
4A ou 4D = Transaction de remboursement conclue			
7 ou 7F = Transaction de remboursement indépendant du commerçant conclue			
8 ou 8F = Transaction de remboursement du commerçant conclue			
9 ou 9F = Première transaction d'annulation conclue			
10 ou 10F = Deuxième transaction d'annulation conclue			
11A ou 11D - Transaction de remboursement conclue			
Montant des frais de commodité	Chaîne	9 caractères décimaux	<code>receipt.getFeeAmount();</code>
Montant attendu des frais de commodité. Ce champ renvoie le montant soumis par le commerçant pour une transaction réussie. Pour une transaction échouée, il renverra le montant attendu des frais de commodité.			
Taux des frais de commodité	Chaîne	9 caractères décimaux	<code>receipt.getFeeRate();</code>

	Type	Limites	Méthode Get
	Description		
Type de frais de commodité	Taux des frais de commodité défini dans le profil du commerçant. Par exemple : 1,00 = Un montant fixe 10,0 = Un montant en pourcentage		
	Chaîne	AMT ou PCT	receipt.getFeeType();
<b>Champ de réponse du code d'avis pour les commerçants</b>			
Code d'avis	Chaîne	2 caractères alphanumériques	receipt.getAdviceCode();
	Le message renvoyé par l'émetteur est destiné à informer le commerçant uniquement, et n'est pas destiné aux reçus des clients.  Pour plus de détails sur les codes de réponse qui sont renvoyés, consultez le document sur les codes d'avis à la page <a href="https://developer.moneris.com">https://developer.moneris.com</a> .		

**Tableau 29 : Codes de réponse de transaction financière**

Code	Description
Code de réponse inférieur à 50	Transaction approuvée
Code de réponse supérieur ou égal à 50	Transaction refusée
Code de réponse NULL	Transaction non envoyée pour autorisation

Pour plus de détails sur les codes de réponse qui sont renvoyés, consultez le document sur les codes de réponse à la page <https://developer.moneris.com>

**Tableau 30 : Réponses d'administration de la chambre forte**

Code	Description
001	Données de la carte de crédit enregistrées avec succès Données de la carte de crédit mises à jour avec succès Données de la carte de crédit supprimées avec succès Données de la carte de crédit localisées avec succès Numéros des cartes expirant localisés avec succès (REMARQUE : No = le nombre de cartes localisées)
983	Impossible de trouver la transaction précédente
986	Transaction incomplète : délai écoulé
987	Transaction non valide
988	Cartes expirées introuvables
Null	Erreur : XML mal formé

## B.1 Définition des champs de réponse – 3-D Secure

Les champs de réponse suivants sont particuliers aux transactions 3-D Secure

Variable	Type et limites	Description
Code de vérification d'authentification du titulaire de carte	<i>Chaîne</i> 50 caractères alphanumériques	<code>receipt.getMPICavv()</code>  Code de vérification d'authentification du titulaire de carte, qui doit être fourni dans la transaction financière
Indicateur de conclusion de contestation	<i>Chaîne</i> Y ou N	<code>receipt.getMPIChallengeCompletionIndicator()</code>  Indique le résultat de la demande de contestation
URL de contestation	<i>Chaîne</i> 2048 Caractères alphanumériques	<code>receipt.getMPIChallengeURL()</code>  Si la valeur de transStatus est « C », ce champ sera rempli avec l'URL pour forcer la transaction challengeData afin de créer l'écran de contestation du titulaire de la carte.
Type de message	<i>Chaîne</i> 4 Caractères alphanumériques	<code>receipt.getMPIMessageType()</code>  Indique le message de demande  Nomenclature EMV « ARES »
URL de la méthode 3DS	<i>Chaîne</i> Caractères alphanumériques	<code>receipt.getMPIThreeDSMethodURL()</code>  Terminal d'empreinte de l'appareil
Données de la méthode 3DS	<i>Chaîne</i> Caractères alphanumériques	<code>receipt.getMPIThreeDSMethodData()</code>  Données qui doivent être affichées sur l'URL de la méthode 3DS
ID de transaction du serveur 3DS	<i>Chaîne</i> 36 Caractères alphanumériques	<code>receipt.getMPIThreeDSServerTransId()</code>  Identifiant de transaction unique de 3-D Secure

Version de 3DS	<i>Chaîne</i> 10 caractères numériques	receipt.getThreeDSVersion() ;  Le numéro de la version 3DS
État de la transaction	<i>Chaîne</i> 1 caractère alphanumérique  Y ou N	receipt.getMpiTransStatus()  Indique le résultat du serveur de contrôle d'accès (ACS)
Données de la contestation	Caractères alphanumériques	Si le transStatus est "C", ce champ sera rempli avec les données permettant d'afficher le challengeURL. Les données doivent être enregistrées dans un champ portant le nom de "creq".
ICE	1 caractère alphabétique	Type de chiffrement qui doit être fourni dans la demande financière  Si l'état de la transaction est "Y" ou "A", cette valeur sera renvoyée dans la réponse. L'ICE est utilisé pour les transactions financières ultérieures.
DS trans ID	36 caractères alphanumériques	Identifiant de transaction universellement unique attribué par le serveur de répertoires (DS) pour identifier une transaction unique
Raison de l'état de la transaction	2 caractères numériques	Fournit des informations supplémentaires sur la raison pour laquelle le champ TransStatus a la valeur spécifiée  Valeurs possibles :  01 = Échec de l'authentification de la carte 02 = Appareil inconnu 03 = Appareil non pris en charge 04 = Limite de fréquence d'authentification dépassée 05 = Carte expirée 06 = Numéro de carte non valide 07 = Transaction non valide 08 = Pas d'enregistrement de carte 09 = Défaut de sécurité 10 = Carte volée 11 = Soupçon de fraude 12 = Transaction refusée non permise au titulaire de carte 13 = Le titulaire de la carte n'est pas inscrit au service

		<p>14 = Transaction interrompue au niveau de l'ACS</p> <p>15 = Confiance faible</p> <p>16 = Confiance moyenne</p>
Renseignements du titulaire de la carte	128 caractères alphanumériques	Un texte est fourni par l'ACS ou l'émetteur au titulaire de la carte lors d'une transaction sans friction ou découpée. Il fournit des informations au titulaire de la carte. Par exemple, « Une authentification supplémentaire est nécessaire pour cette transaction, veuillez contacter (nom de l'émetteur) au xxx-xxx-xxxx ».
Type d'authentification	2 caractères numériques	<p>Méthode d'authentification 3-D Secure que l'émetteur utilisera pour contester le titulaire de carte</p> <p>01 = Statique</p> <p>02 = Dynamique</p> <p>03 = Hors limites</p> <p>04 = Dissociée</p>
ID de transaction 3DS de l'ACS	36 caractères alphanumériques	Identifiant de transaction universellement unique attribué par le serveur de contrôle d'accès (ACS) de l'émetteur pour identifier une transaction unique
Horodatage de l'authentification 3DS	12 caractères numériques	<p>Date et heure au fuseau horaire UTC auxquelles a eu lieu l'authentification 3DS du titulaire de carte</p> <p>Format de la date = AAAAMMJJHHMM</p>

## B.2 Définition des champs de réponse – TMD

### Champs de réponse liés à la TMD

Variable	Type et limites	Méthode Get et description
Taux de la TMD	<p><i>Chaîne</i></p> <p>9 caractères décimaux</p> <p>Longueur variable</p>	<pre>receipt.getMCPRate();</pre> <p>Taux de change (devise étrangère en CAD) qui sera utilisé pour la transaction</p> <p>Si une variable <b>MCP rate token</b> a été utilisée, elle reflétera le taux obtenu par la transaction d'obtention du taux de la TMD; si aucun jeton n'a été utilisé, le taux est le taux de change actuel récupéré par Passerelle Moneris.</p>
Devise de règlement du commerçant	<p><i>Chaîne</i></p> <p>3 caractère numérique</p>	<pre>receipt.getMerchantSettlementCurrency();</pre> <p>Devise dans laquelle le commerçant effectue les règlements</p>
Montant de règlement du commerçant	<p><i>Chaîne</i></p> <p>10 caractères décimaux</p> <p>Jusqu'à 7 chiffres (dollars) + point décimal (.) + 2 chiffres (sous) après le point décimal</p> <p><b>EXEMPLE : 1 234 567,89</b></p>	<pre>receipt.getMerchantSettlementAmount();</pre> <p>Montant qui sera versé au commerçant, en dollars canadiens</p>
Code de devise du titulaire de carte	<p><i>Chaîne</i></p> <p>3 caractère numérique</p>	<pre>receipt.getCardholderCurrencyCode();</pre> <p>Code ISO pour la devise étrangère que le titulaire de la carte utilise pour payer</p>
Montant du titulaire de carte	<p><i>Chaîne</i></p> <p>12 caractère numérique</p> <p>Longueur variable</p>	<pre>receipt.getCardholderAmount();</pre> <p>Montant, en unités de devise étrangère, que le titulaire de la carte paiera lors de la transaction</p>

Variable	Type et limites	Méthode Get et description
Code d'état d'erreur de la TMD (MCP error status code)	<i>Chaîne</i> 4 caractères numériques Longueur variable	<pre>receipt.getMCPErrorStatusCode();</pre> Chiffre représentant une réponse de code d'erreur de la TMD
Message d'erreur de la TMD	<i>Chaîne</i> 250 caractères alphanumériques Longueur variable	<pre>receipt.getMCPErrormessage();</pre> Message correspondant à un code d'erreur de la TMD
ID du serveur	<i>Chaîne</i> 15 caractères alphanumériques	<pre>receipt.getHostId();</pre> Identifiant unique utilisé sur la plateforme Moneris

#### Champs de réponse spécifiques à la transaction d'obtention du taux de la TMD

Variable	Type et limites	Méthode Get et description
Type de transaction de taux	<i>Chaîne</i> Maximum 8 caractères alphabétiques PURCHASE (achat) ou REFUND (remboursement)	<pre>receipt.getRateTxnType();</pre> Correspond au type de transaction envoyé dans la demande
Jeton du taux de la TMD	<i>Chaîne</i> 17 caractères alphanumériques	<pre>receipt.GetMCPRateToken();</pre> Jeton à durée limitée représentant un taux de change temporairement fixé pour une utilisation lors de transactions financières  Ce champ est renvoyé dans la réponse à une demande d'obtention du taux de la TMD.
Heure de début de la demande de taux	<i>Chaîne</i> 24 caractères alphanumériques	<pre>receipt.getRateInqStartTime();</pre> Heure locale (ISO 8601) à laquelle le taux est demandé

Variable	Type et limites	Méthode Get et description
Heure de fin de la demande de taux	<i>Chaîne</i> 24 caractères alphanumériques	<code>receipt.getRateInqStartTime();</code> Heure locale (ISO 8601) à laquelle le taux est renvoyé
Heure de début de la période de validité du taux	<i>Chaîne</i> 10 caractère numérique	<code>receipt.getRateValidityStartTime();</code> Heure (unix UTC) à partir de laquelle le taux est valide
Heure de fin de la période de validité du taux	<i>Chaîne</i> 10 caractère numérique	<code>receipt.getRateValidityEndTime();</code> Heure (unix UTC) jusqu'à laquelle le taux est valide
Période de validité du taux	<i>Chaîne</i> 3 caractère numérique Longueur variable	<code>receipt.getRateValidityPeriod();</code> Période en minutes pendant laquelle ce taux est valide

### B.3 Définition des champs de réponse – Versements Visa

Champs de réponse apparaissant dans la transaction de recherche de plan de versements

Variable	Type et limites	Description
Plans de versement admissibles	<i>Objet</i> S. O.	Contient les champs liés au plan de versements
Nombre de plans	<i>Chaîne</i> Valeur numérique	Nombre total de plans de versement pouvant être offerts aux titulaires de carte
Détails du plan	<i>Objet tableau</i> S. O.	Contient les champs liés à un plan de versements en particulier  Chaque plan de versements pouvant être offert aux titulaires de carte est

Variable	Type et limites	Description
		représenté par un objet Plan Details précis.
Taux annuel en pourcentage	<p><i>Chaîne</i></p> <p>Valeur numérique</p>	<p>Taux annuel en pourcentage lié aux paiements du plan de versements; utilisé à des fins d'exposition seulement, ne peut pas être utilisé pour les calculs</p> <p>Valeurs acceptées : de 0 à 10000</p> <p>Le taux en pourcentage est affiché avec deux décimales implicites.</p> <p><b>EXEMPLE : 320 = 3,2 %</b></p>
Fréquence des versements	<p><i>Chaîne</i></p> <p>Maximum 10 caractères alphabétiques</p>	<p>Fréquence des versements du plan</p> <p>Valeurs possibles :</p> <p>WEEKLY (hebdomadaire)</p> <p>BIWEEKLY (toutes les deux semaines)</p> <p>MONTHLY (mensuel)</p> <p>BIMONTHLY (tous les deux mois)</p>
ID du plan de versements	<p><i>Chaîne</i></p> <p>36 caractères alphanumériques</p> <p>Longueur fixe</p>	<p>Identifiant généré par la marque de carte pour un plan de versements</p> <p>Utilisé comme champ de demande dans l'objet Installment Info</p>
Nom du plan de versements	<p><i>Chaîne</i></p> <p>Maximum 255 caractères alphanumériques</p>	<p>Nom du plan de versements, peut ne pas être unique</p>
Référence du plan de versements	<p><i>Chaîne</i></p> <p>10 caractères alphanumériques</p>	<p>Nom unique et humain du plan de versements</p> <p>Utilisé comme champ de demande dans l'objet Installment Info</p>

Variable	Type et limites	Description
	Longueur fixe	
Type de plan de versements	<p><i>Chaîne</i></p> <p>Maximum 20 caractères alphanumériques</p>	<p>Type de plan de versements</p> <p>Valeurs possibles :</p> <ul style="list-style-type: none"> <li>ISSUER_PROMOTION</li> <li>BI_LATERAL</li> <li>ISSUER_DEFAULT</li> <li>MARKET</li> </ul>
Nombre de versements	<p><i>Chaîne</i></p> <p>4 caractères numériques</p> <p>Minimum de 1, maximum de 1 000</p>	Nombre de versements maximal du plan
Premier versement	<p><i>Objet</i></p> <p>S. O.</p>	Contient les détails lié au coût du premier versement
Montant du premier versement	<p><i>Chaîne</i></p> <p>Maximum 9 caractères numériques</p>	<p>Montant du paiement du premier versement</p> <p>Les deux derniers chiffres représentent les sous.</p> <p><b>EXEMPLE : 123112 = 1 231,12 \$</b></p>
Frais du premier versement	<p><i>Chaîne</i></p> <p>Maximum 9 caractères numériques</p>	<p>Frais facturés lors du premier versement</p> <p>Les deux derniers chiffres représentent les sous.</p> <p><b>EXEMPLE : 123112 = 1 231,12 \$</b></p>
Frais initiaux	<i>Chaîne</i>	Frais initiaux facturés au titulaire de carte pour le plan de versements; facturés uniquement lors du premier

Variable	Type et limites	Description
	Valeur numérique	versement
Dernier versement	<i>Objet</i> S. O.	Contient les détails lié au coût du dernier versement
Montant du dernier versement	<i>Chaîne</i> Maximum 9 caractères numériques	Montant du paiement du versement final  Les deux derniers chiffres représentent les sous.  <b>EXEMPLE : 123112 = 1 231,12 \$</b>
Frais du dernier versement	<i>Chaîne</i> Maximum 9 caractères numériques	Frais facturés lors du dernier versement  Les deux derniers chiffres représentent les sous.  <b>EXEMPLE : 123112 = 1 231,12 \$</b>
Information liée à la promotion	<i>Objet</i> S. O.	Contient les renseignements liée à la promotion transmis entre l'émetteur et le commerçant
Code promotionnel	<i>Chaîne</i> 2 caractères alphanumériques	Identifiant externe du plan fourni par l'émetteur
ID de la promotion	<i>Chaîne</i> Maximum 8 caractères alphanumériques	Identifiant externe fournit par l'émetteur qui identifie un programme ou une promotion
Modalités	<i>Objet tableau</i> S. O.	Contient les champs liés aux modalités présentées aux titulaires de carte

Variable	Type et limites	Description
Nombre d'occurrences des modalités	<i>Chaîne</i> Valeur numérique	Nombre d'occurrences de l'ensemble de modalités liées à un plan de versements en particulier; représente le nombre de langues dans lesquelles les modalités sont offertes
Détails des modalités	<i>Objet</i> S. O.	Contient les détails liés à un ensemble de modalité en particulier (anglais, français, etc.)  Chaque langue a son propre objet.
Code de langage	<i>Chaîne</i> 3 caractères alphanumériques	Code de langage pour le texte des modalités
Texte	<i>Chaîne</i> Maximum 2000 caractères alphanumériques	Texte des modalités du plan de versements
URL des modalités	<i>Chaîne</i> Maximum 1 000 caractères alphanumériques	URL HTTPS des modalités hébergée par l'émetteur pour afficher les modalités aux titulaires de carte
Version des modalités	<i>Chaîne</i> 10 caractères alphanumériques  Longueur variable (de 1 à 10 caractères)	Version des modalités du plan de versements accepté par le titulaire de carte  Cette version augmente automatiquement chaque fois que le plan est mis à jour par l'émetteur.
Frais totaux	<i>Chaîne</i> Maximum 9 caractères numériques	Frais totaux liés au plan  Les deux derniers chiffres représentent les sous.

Variable	Type et limites	Description
		<p style="background-color: #e0f2e0; padding: 5px;"><b>EXEMPLE :</b> 123112 = 1 231,12 \$</p>
Coût total du plan	<i>Chaîne</i> Valeur numérique	Représente le coût total du plan de versements sélectionné  Les chiffres les plus à droites représentent les unités mineures (p. ex. les sous en CAD); aucune fraction d'unité mineure

#### Champs de réponse apparaissant dans les transactions financières

Variable	Type et limites	Description
Résultats du versement	<i>Objet</i> S. O.	Contient les champs liés au plan de versement dans les transactions financières
ID du plan de versements	<i>Chaîne</i> 36 caractères alphanumériques Longueur fixe	Identifiant généré par la marque de carte pour un plan de versements
Référence du plan de versements	<i>Chaîne</i> 10 caractères alphanumériques Longueur fixe	Nom unique et humain du plan de versements
Version des modalités	<i>Chaîne</i> 10 caractères alphanumériques Longueur variable (de 1 à	Version des modalités du plan de versement accepté par le titulaire de carte  Cette version augmente automatiquement chaque fois que le

Variable	Type et limites	Description
	10 caractères)	plan est mis à jour par l'émetteur.
ID d'acceptation du plan	<p><i>Chaîne</i></p> <p>36 caractères alphanumériques</p> <p>Longueur fixe</p>	Nom humain court et unique créé par Visa en caractère alphanumérique du plan de versements
État du plan de versements	<p><i>Chaîne</i></p> <p>1 caractère alphabétique</p> <p>Longueur fixe</p>	<p>Valeurs possibles :</p> <p>N = Nouveau plan, pas encore accepté</p> <p>A = Plan accepté</p> <p>C = Plan annulé</p>
Réponse du plan	<p><i>Chaîne</i></p> <p>Maximum 50 caractères numériques</p>	<p>Code de réponse du plan de versements</p> <p>Valeurs possibles :</p> <p>00 = Plan traité et accepté</p> <p>Si la réponse n'est pas 00, cela signifie qu'un problème est survenu avec le plan de versements. Un message d'erreur verbeux indiquant que la demande a été reçue par Visa est retourné.</p>

## Annexe C Codes de réponse

### Codes de réponse d'approbation

Code de réponse	Messages
000	Approuvée, soldes des comptes Inclus (demande de solde), aucune raison de refuser Approuvée (soldes) Fichier traité et transaction réussie avec erreur
001	Approuvée, soldes de comptes non inclus Approuvée , aucun solde et approuvé ou conclu avec succès VIP Approuvé (aucun solde) et accusé de réception de l'avis – Responsabilité financière acceptée
002	Approuvée, country club
003	Approuvée, possibilité de plus d'un ID
004	Approuvée, en attente de l'ID (signature du brouillon papier)
005	Approuvée, à l'aveugle
006	Approuvée, VIP
En 007	Approuvée, transaction administrative
008	Approuvé, fichier NEG national accepté
009	Approuvée, transaction commerciale
010	Approuvée pour un montant partiel
023	Amex – approbation de crédit

Code de réponse	Messages
024	Amex 77 – approbation de crédit
027	Transaction déjà annulée
028	Crédit VIP approuvé
029	Confirmation de la réponse de crédit
900	Erreur générale
901	URL non valide
902	XML mal formé

**Codes de réponse de refus**

Code de réponse	Messages
050	Ne pas honorer Refuser Consulter l'émetteur de carte Échec de la certification de l'ID Refuser – Ne pas honorer Carte non initialisée Demande refusée : Refuser – Frais inacceptables Impossible de localiser la transaction originale Fraude soupçonnée Refuser – L'accepteur de carte doit appeler le service de sécurité de l'acquéreur Montant non rapproché – Totaux fournis Impossible de trouver le numéro du terminal du GAB ou du PDV Échec du CAC

Code de réponse	Messages
	<p>Demande refusée :</p> <p>Échec du CAC</p> <p>Réservé</p> <p>Échec du processus de sécurité</p> <p>Pas d'arriérés (le reçu de transaction n'est pas imprimé)</p> <p>Type de fichier non valide</p> <p>Pas de fichier de ce type</p> <p>Fichier verrouillé</p> <p>Échoué</p> <p>Longueur de fichier incorrecte</p> <p>Erreur de décompression de fichier</p> <p>Erreur de nom de fichier</p> <p>Le fichier ne peut pas être reçu</p> <p>Refuser – Ne pas honorer</p>
051	Carte expirée
052	<p>Nombre de tentatives de saisie du code NIP dépassé</p> <p>Limite de tentatives de code NIP dépassée</p> <p>Nombre autorisé de tentatives de code NIP dépassé</p>
053	Pas de partage
054	Aucun module de sécurité
055	Transaction non valide
056	<p>Pas de support ou transaction non autorisée à l'acquéreur</p> <p>Transaction non supportée par l'institution financière ou non supportée par l'acquéreur</p>

Code de réponse	Messages
057	Carte perdue ou volée
058	État non valide
059	Refuser (garder la carte) – Carte restreinte Carte restreinte
060	Pas de compte de chèques Pas de compte d'épargne
061	Pas de PBF
062	Erreur de mise à jour de la valeur PBF
063	Type d'autorisation invalide
064	Mauvaise piste 2
065	Ajustement non autorisé
066	Incrément d'avance sur carte de crédit non valide
067	Date de transaction non valide
068	Erreur PTLF
069	Mauvais message d'erreur ou aucun résultat de méthode de vérification du titulaire de carte Mauvais message – erreur de modification ou erreur de format
070	Aucun IDF Émetteur non valide Émetteur non valide ou émetteur non valide ou banque introuvable

Code de réponse	Messages
071	<p>Autorisation d'acheminement non valide</p> <p>Impossible à acheminer ou Institution financière ou installation de réseau intermédiaire introuvable pour l'acheminement</p> <p>Acheminement non valide vers l'authentification ou Numéro d'identification d'émetteur non valide</p>
072	Carte sur le fichier national NEG
073	Service d'acheminement non valide (destination)
074	<p>Impossible d'autoriser</p> <p>Entrer de nouveau la transaction</p> <p>La transaction ne peut être conclue</p> <p>Refuser – infraction à la sécurité – infraction à la loi</p> <p>Problème de système – demander au titulaire d'insérer sa carte dans le lecteur de cartes à puce</p> <p>Merchant Link n'est pas connecté (Connexion à la gestion du réseau requise)</p>
075	Longueur du numéro de carte de crédit non valide
076	Fonds faibles
077	Préautorisation complète
078	<p>Transaction reproduite</p> <p>Transaction reproduite ou demande en cours</p>
079	Remboursement maximal en ligne atteint
080	Remboursement maximum hors ligne atteint
081	Crédit maximal par remboursement atteint

Code de réponse	Messages
082	Nombre d'utilisations dépassé
083	Crédit de remboursement maximal atteint
084	Transaction dupliquée – le numéro d'autorisation a déjà été corrigé par le serveur
085	Enquête non autorisée
086	Limite de plancher dépassée
087	Quantité maximale de crédit de remboursement par le détaillant
088	Appeler
089	État du FAC inactif ou fermé
090	Dossier de référence plein
091	Problème de fichier NEG
092	Avance inférieure au minimum
093	Retard de paiement
094	Montant supérieur au maximum
095	Montant supérieur au maximum Montant supérieur à la limite ou limite du montant de la transaction dépassée
096	NIP requis
097	Échec de la vérification Mod 10

Code de réponse	Messages
098	Transaction forcée
099	Mauvais PBF

#### Codes de réponse de référence

Code de réponse	Messages
100	Impossible de traiter la transaction  Demande non valide, Contactez l'équipe de la certification des PDV des clients de Moneris pour les refus répétés.  Réseau indisponible  Défaillance du système
101	Appeler
102	Référer – Appel  Carte expirée  Personne ressource de l'accepteur de carte  Appeler la sécurité de l'acquéreur de l'accepteur de carte
103	Problème de fichier NEG
104	Problème de FAC
105	Carte non acceptée
106	Montant supérieur au maximum
107	Limite quotidienne dépassée
108	Problème de FAC
109	Avance inférieure au minimum

Code de réponse	Messages
110	Nombre d'utilisations dépassé
111	Retard de paiement
112	Montant supérieur au maximum
113	Inactivité
115	Erreur PTLF
121	Problème de fichier administratif
122	Impossible de valider le NIP, le module de sécurité est en panne

**Codes de réponse d'erreur de système**

Code de réponse	Messages
150	Code de service ou de commerçant non valide Commerçant non inscrit au dossier Commerçant non inscrit au dossier ou commerçant non valide
200	Compte non valide Numéro de carte non valide Compte non valide ou Refuser – Aucun type de compte demandé
201	NIP incorrect NIP non valide ou Numéro d'identification personnel incorrect Erreur de blocage du NIP
202	Avance inférieure au minimum
203	Carte administrative nécessaire

Code de réponse	Messages
204	Montant supérieur au maximum
205	Montant de l'avance non valide Montant original incorrect Mauvais message ou Montant non valide Erreur du montant original de la transaction
206	FAC introuvable Compte de « destination » non valide Compte d'« origine » non valide Compte non valide
En 207	Date de transaction non valide
208	Date d'expiration non valide
209	Code de transaction non valide
210	Erreur de synchronisation de la clé NIP
212	Destination indisponible
251	Erreur sur le montant en argent comptant
252	Débit non pris en charge

#### **Codes de réponse American Express (refus)**

Code de réponse	Messages
426	AMEX – Refus 12
427	AMEX – Commerçant non valide

Code de réponse	Messages
429	AMEX – Erreur de compte
430	AMEX – Carte expirée
431	AMEX – Appeler Amex
434	AMEX – Appeler 03 Remarque : NVC non valide (CID)
435	AMEX – Système en panne
436	AMEX – Appeler 05
437	AMEX – Refusé
438	AMEX – Refusé
439	AMEX – Erreur de service
440	AMEX – Appeler Amex
441	AMEX – Erreur de montant

**Codes de réponse des cartes de crédit (refus)**

Code de réponse	Messages
408	CARTE DE CRÉDIT – Utilisation limitée de la carte – Se référer à la succursale
475	CARTE DE CRÉDIT – Date d'expiration non valide
476	CARTE DE CRÉDIT – Transaction non valide, rejetée Pas de compte de crédit Transaction non valide ou transactions connexes non valides

Code de réponse	Messages
	<p>Impossible à traiter ou défaillance soupçonnée, erreur de transaction connexe</p> <p>Impossible d'autoriser :</p> <p>La coupure est en cours.</p> <p>L'émetteur n'est pas en mesure de traiter la transaction</p> <p>Défaillance du système de commutation</p> <p>Réponse de l'émetteur non reçue par CUPS</p> <p>Impossible d'autoriser ou état illégal de l'acquéreur</p>
477	<p>CARTE DE CRÉDIT – Appel de référence ou numéro de carte non valide</p> <p>Numéro de carte non valide (le compte n'existe pas)</p> <p>Refus – Carte introuvable</p> <p>Articles ne figurant pas sur le livret bancaire au-delà de la limite, refus ou numéro de carte non valide</p>
478	CARTE DE CRÉDIT – Refuser, récupérer la carte, appeler
479	CARTE DE CRÉDIT – Refuser, récupérer la carte
480	CARTE DE CRÉDIT – Refuser, récupérer la carte
481	<p>CARTE DE CRÉDIT – Refuser</p> <p>Transaction interdite par le titulaire de la carte</p> <p>Fonds insuffisants ou solde inadéquat</p> <p>Transaction non valide</p> <p>Transaction interdite par le commerçant</p>
482	CARTE DE CRÉDIT – Carte expirée
483	<p>CARTE DE CRÉDIT – Référez-vous à l'émetteur.</p> <p>Refus – L'accepteur de carte doit communiquer avec l'acquéreur.</p>

Code de réponse	Messages
484	CARTE DE CRÉDIT – Carte expirée – référer
485	CARTE DE CRÉDIT – Non autorisée
486	CARTE DE CRÉDIT – Erreur cryptographique du CVC
487	CARTE DE CRÉDIT – CVC non valide
489	CARTE DE CRÉDIT – CVC non valide
490	CARTE DE CRÉDIT – CVC non valide
492	Problème de système – demander au titulaire d'insérer sa carte dans le lecteur de cartes à puce  Le nombre de retraits est dépassé

**Codes de réponse de refus du système**

Code de réponse	Messages
800	Mauvais format
801	Données erronées
802	L'ID du commis est non valide.
809	Mauvaise fermeture
810	Expiration du système
811	Erreur système
821	Mauvaise longueur de réponse
842	Échec de la recherche du plan de versements

Code de réponse	Messages
877	Bloc NIP non valide
878	Erreur de longueur du NIP
880	Dernier envoi d'une transaction à plusieurs envois
881	Envoi intermédiaire d'une transaction à plusieurs envois
889	Erreur de synchronisation de la clé CAC
898	Mauvaise valeur CAC
899	Mauvais numéro de séquence – renvoyer la transaction
900	Conclusion – Tentatives de NIP dépassées
901	Conclusion – Carte expirée
902	Conclusion – Conclusion NEG
903	Conclusion – État FAC à 3
904	Conclusion – Avance < Minimum
905	Conclusion – Nombre de fois utilisé
906	Conclusion – Retard
907	Conclusion – Montant supérieur au maximum
908	Conclusion – Montant dépasse le maximum Conclusion – Conclusion Prendre la carte

Code de réponse	Messages
	Fraude soupçonnée Conclusion définitive Refuser – Garder la carte : Conditions spéciales Carte expirée Équipe de la fraude L'accepteur de carte doit appeler l'acquéreur. Ne pas honorer
950	Carte administrative désactivée dans le compte du commerçant

**Autres codes de réponse**

Code de réponse	Message
599	Refuser

**Codes de réponse administratifs**

Code de réponse	Messages
960	Échec de l'initialisation – Pas de correspondance avec l'ID du commerçant
961	Échec de l'initialisation – Pas de correspondance avec l'ID du PED
962	Échec de l'initialisation – Pas de correspondance avec l'ID de l'imprimante
963	Pas de correspondance pour le code de sondage
964	Échec de l'initialisation – Pas de correspondance avec l'ID du concentrateur
965	Numéro de version du logiciel non valide

Code de réponse	Messages
966	Nom du terminal en double
En 970	Table du terminal ou du commis pleine
983	Totaux de commis non disponibles : certains ID de commis n'existent pas ou affichent des totaux de zéro.
989	Erreur de CAC sur la transaction 95 (Initialisation et échange), ce qui indique souvent que des clés erronées ont été introduites dans un appareil ou qu'une erreur de synchronisation KCAC est survenue.

#### Codes de demande d'annulation EMV

Code de réponse	Messages
990	La carte à puce refuse une transaction approuvée par le serveur
991	Carte à puce retirée avant la fin des communications ICC

## Annexe D Messages d'erreur

### Messages d'erreurs apparaissant lorsque la passerelle est inaccessible

#### Global Error Receipt

Vous n'êtes pas connecté à nos serveurs. Ce problème peut être causé par un pare-feu ou par votre connexion Internet.

#### Response Code = NULL

Le code de réponse obtenu peut être « null » pour diverses raisons. La plupart du temps, l'explication est incluse dans le champ Message.

Lorsque vous obtenez une réponse « NULL », cela peut signifier que l'émetteur, le serveur de carte de crédit ou la passerelle sont inaccessibles. Ce problème peut survenir lorsque ceux-ci sont hors ligne ou parce que vous n'êtes pas en mesure de vous connecter à Internet.

Une réponse « NULL » peut également être obtenue lorsqu'un message de transaction n'est pas formaté correctement.

#### Messages d'erreur obtenus dans le champ Message de la réponse :

##### XML Parse Error in Request: <System specific detail>

L'API a envoyé un mauvais document XML au miniserveur.

##### XML Parse Error in Response: <System specific detail>

Le miniserveur a renvoyé un mauvais document XML.

##### Transaction Not Completed Timed Out

Le délai de transaction s'est écoulé avant que le serveur réponde à la passerelle.

##### Request was not allowed at this time

Le serveur n'est pas connecté.

##### Could not establish connection with the gateway: <System specific detail>

La passerelle n'accepte pas les transactions ou le serveur n'a pas un bon accès à Internet.

##### Input/Output Error: <System specific detail>

Le miniserveur n'est pas fonctionnel.

##### The transaction was not sent to the host because of a duplicate order id

Un ID de commande existant a été utilisé.

##### The transaction was not sent to the host because of a duplicate order id

La date d'expiration envoyée n'était pas au bon format.

### Messages d'erreur de la chambre forte

#### Can not find previous

La clé de donnée fournie n'a pas été trouvée dans nos dossiers, ou le profil n'est plus actif.

#### Invalid Transaction

La transaction ne peut pas être traitée, car des données incorrectes ont été envoyées.

#### OU

Un champ obligatoire n'a pas été rempli ou un code SEC invalide a été envoyé.

#### Malformed XML

Erreur d'analyse.

**Incomplete**

Délai écoulé.

**OU**

Les cartes expirées sont introuvables.

## Mention des droits d'auteur

Droits d'auteur © Novembre 2022 Solutions Moneris, 3300, rue Bloor Ouest, Toronto (Ontario), M8X 2X2.

Tous droits réservés. Il est interdit de reproduire ou de diffuser le présent document, que ce soit en partie ou en totalité, sous quelque forme ou par quelque moyen que ce soit, électronique ou mécanique, y compris par photocopie, sans l'autorisation de Solutions Moneris.

Ce document a été produit comme guide de référence pour aider les clients de Moneris, ci-après dénommés commerçants. Tous les efforts ont été faits pour rendre les renseignements contenus dans ce guide de référence aussi précis que possible. Les auteurs de Solutions Moneris n'assument aucune responsabilité envers toute personne ou entité en ce qui concerne toute perte ou tout dommage lié aux renseignements contenus dans le présent guide de référence ou en découlant.

## Marques de commerce

Moneris et le logo Solutions Moneris sont des marques de commerce déposées de Corporation Solutions Moneris.

Tous les logiciels, matériels et produits technologiques cités dans ce document sont revendiqués comme des marques de commerce ou des marques de commerce déposées de leurs entreprises respectives.

Imprimé au Canada

10 9 8 7 6 5 4 3 2 1

