



API de Passerelle Moneris – Guide d'intégration – Java

Version : 1.5.0

Copyright © Moneris Solutions, 2023

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 v1.5.0

- 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

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
Changements apportés à la v1.3.1
- Modification du code de réponse 959 dans la section « Autres codes de réponse »
Changements apportés à la v1.3.0
- 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^{MC} et retrait des mentions d'Android Pay
- Séparation des flux de traitement des transactions en deux sujets distincts pour Apple Pay et Google Pay^{MC}, soit Sommaire du processus de transaction avec Apple Pay et Sommaire du processus de transaction avec Google Pay^{MC}

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

Changements apportés à la v1.2.3

Ajout de renseignements concernant le transfert de données Offlin^{MC} pour le pixel invisible Card Match des transactions traitées par l'API unifiée

Table des matières

Table des matières

Sécurité et conformité	2
Confidentialité	2
Changements apportés à la v1.5.0	3
Obtenir de l'aide	14
1. À propos du présent document.....	15
1.1 Objectif	15
1.2 À qui ce guide s'adresse-t-il?.....	15
2. Ensemble de transactions de base	16
2.1 Achat.....	17
2.2 Préautorisation.....	23
2.3 Conclusion de préautorisation	29
2.4 Transaction forcée.....	33
2.5 Correction d'achat.....	36
2.6 Remboursement.....	39
2.7 Remboursement indépendant	42
2.8 Vérification de carte avec le SVA et le NVC.....	45
2.9 Fermeture de lot.....	49
2.10 Ouverture des totaux	51
3 Renseignements d'identification au dossier	54
3.1 À propos des renseignements d'identification au dossier	54
3.2 Objet Credential on File Info et variables.....	54
3.3 Types de transactions nécessitant des renseignements d'identification au dossier	55
3.4 Transactions nécessitant des renseignements d'identification au dossier entamées par le commerçant et entamées par le titulaire de carte	55
3.5 Transactions initiales nécessitant des renseignements d'identification au dossier	56
3.6 Renseignements d'identification enregistrés au dossier avec des renseignements d'identification précédemment stockés.....	56
3.7 Valeurs des indicateurs de paiement (payment indicator) et de commerce électronique (e-commerce indicator)	57
3.8 Transformation en jetons des cartes de crédit et des renseignements d'identification au dossier	57
3.9 Renseignements d'identification au dossier et conversion de jetons temporaires	58
3.10 Transactions de vérification de carte et de renseignements d'identification au dossier	58
3.10.1 Quand effectuer une vérification de carte avec les renseignements d'identification au dossier	59
3.10.2 Renseignements d'identification au dossier et ajout de jeton à la chambre forte	59
3.10.3 Renseignements d'identification au dossier et mise à jour d'une carte de crédit dans la chambre forte.....	59
3.10.4 Renseignements d'identification au dossier et ajout d'une carte de crédit à la chambre forte	60

3.10.5 Renseignements d'identification au dossier et facturation périodique ...	60
4 Chambre forte.....	61
4.1 À propos des transactions utilisant la chambre forte	61
4.2 Types de transactions utilisant la chambre forte	61
4.2.1 Types de transactions administratives utilisant la chambre forte.....	61
4.2.2 Types de transactions financières utilisant la chambre forte.....	63
4.3 Transactions utilisant la chambre forte qui prennent en charge les jetons temporaires	63
4.4 Chambre forte et versements	64
4.5 Chambre forte et transformation en jetons du réseau.....	64
4.6 Transactions administratives utilisant la chambre forte.....	64
4.6.1 Ajout d'une carte de crédit à la chambre forte (ResAddCC).....	64
4.6.2 Ajout d'un jeton temporaire à la chambre forte (ResTempAdd)	71
4.6.3 Mise à jour d'une carte de crédit dans la chambre forte (ResUpdateCC) ..	74
4.6.4 Suppression d'un profil de la chambre forte (ResDelete)	81
4.6.5 Recherche d'un numéro complet dans la chambre forte (ResLookupFull)	83
4.6.6 Recherche d'un numéro masqué dans la chambre forte (ResLookupMasked).....	84
4.6.7 Obtention des cartes expirées dans la chambre forte (ResGetExpiring)....	86
4.6.8 Détection de carte d'entreprise dans la chambre forte (ResIsCorporateCard)	88
4.6.9 Ajout d'un jeton à la chambre forte (ResAddToken)	89
4.6.10 Transformation en jeton d'une carte de crédit dans la chambre forte (ResTokenizeCC)	93
4.7 Transactions financières utilisant la chambre forte	97
4.7.1 Changements apportés à l'ID du client.....	97
4.7.2 Achat avec la chambre forte (ResPurchaseCC)	98
4.7.3 Préautorisation avec la chambre forte (ResPreauthCC)	103
4.7.4 Transaction de remboursement indépendant avec la chambre forte (ResIndRefundCC)	109
4.7.5 Transaction forcée avec la chambre forte (ResForcePostCC).....	112
4.7.6 Vérification de la carte avec la chambre forte (ResCardVerificationCC) ..	115
4.8 Transformation en jetons hébergée.....	119
5 Paiements INTERAC^{MD} en ligne	120
5.1 À propos des transactions de paiement INTERAC ^{MD} en ligne	120
5.2 Autres documents et références	120
5.3 Exigences du site Web et de certification	121
5.3.1 Éléments à fournir à Moneris	121
5.3.2 Processus de certification	121
5.3.3 Exigences du client.....	122
5.3.4 Délais.....	122
5.4 Flux de transaction pour les paiements INTERAC ^{MD} en ligne	123
5.5 Envoi d'une transaction d'achat avec INTERAC ^{MD} en ligne	124
5.5.1 Demande de garantie de fonds	124
5.5.2 Réponse du compte bancaire en ligne et demande de confirmation des fonds.....	125
5.6 Achat avec INTERAC ^{MD} en ligne	125
5.7 Remboursement avec INTERAC ^{MD} en ligne	128
5.8 Définitions des champs liés aux paiements INTERAC ^{MD} en ligne	130
6 Les transactions de niveaux 2 et 3.....	133
6.1 À propos des transactions de niveaux 2 et 3.....	133
6.2 Transactions Visa de niveaux 2 et 3	133

6.2 Types de transactions de niveaux 2 et 3 par carte Visa	133
6.2 Flux de transaction de niveaux 2 et 3 par carte Visa	135
6.2.3 Conclusion par carte Visa.....	136
6.2.4 Correction d'achat par carte Visa	141
6.2.5 Transaction forcée par carte Visa	143
6.2.6 Remboursement par carte Visa	148
6.2.7 Remboursement indépendant par carte Visa.....	153
6.2.8 Transaction VS Corpais	158
6.3 Transactions de niveau 2 et 3 de Mastercard	168
6.3.1 Types de transaction de niveaux 2 et 3 par carte Mastercard	168
6.3.2 Flux de transaction de niveaux 2 et 3 par carte Mastercard	170
6.3.3 Conclusion par carte Mastercard.....	171
6.3.4 Transaction forcée par carte Mastercard	173
6.3.5 Correction d'achat par carte Mastercard	176
6.3.6 Remboursement par carte Mastercard	178
6.3.7 Remboursement indépendant par carte Mastercard.....	180
6.3.8 MC Corpais – Données communes de carte d'entreprise avec les détails de la ligne d'article.....	182
6.4 Transactions American Express de niveaux 2 et 3	199
6.4.1 Types de transaction de niveaux 2 et 3 par carte Amex.....	199
6.4.2 Flux de transaction de niveaux 2 et 3 par carte Amex.....	201
6.4.3 Objets de données de niveaux 2 et 3 pour les transactions par carte Amex	201
6.4.4 Conclusion par carte Amex.....	221
6.4.5 Transaction forcée par carte Amex.....	225
6.4.6 Correction d'achat par carte Amex.....	229
6.4.7 Remboursement par carte Amex.....	230
6.4.8 Remboursement indépendant par carte Amex	234
7 3-D Secure 2.0	239
7.1 À propos de la solution 3-D Secure 2.0	239
7.1.1 Mises en place de l'outil 3-D Secure.....	239
7.1.2 Hors de portée ou non pris en charge	240
7.1.3 Compatibilité des versions.....	240
7.1.4 Passage de l'outil 3-D Secure 1.0 à 2.0	240
7.2 Créer votre intégration 3-D Secure 2.0	240
7.2.1 Activation de la fonction 3-D Secure	240
7.2.2 Flux des transactions avec la solution 3-D Secure 2.0	241
7.3 Mise en œuvre de la demande de recherche de carte (CardLookup).....	241
7.3.1 Demande de recherche de carte (Card Lookup).....	242
7.4 Mise en œuvre de la demande d'authentification 3DS pour les modules d'extension des commerçants	244
7.4.1 Mise en œuvre de la demande d'authentification 3DS pour les modules d'extension des commerçants	244
7.5 Traitement du flux de contestation.....	251
7.5.1 Demande de recherche de code de vérification d'authentification du titulaire de carte (CAVV) – mpiCavvLookup.....	252
7.6 Effectuer l'autorisation.....	254
7.6.1 Achat avec la solution 3-D Secure (cavv_Purchase)	254
7.6.2 Préautorisation avec la solution 3-D Secure (cavv_Preauth)	263
7.6.3 Achat avec la chambre forte et la solution 3-D Secure	271
7.6.4 Préautorisation avec la chambre forte et la solution 3-D Secure.....	275

7.6.5 Achat avec la solution 3-D Secure et facturation périodique	279
7.7 Tester votre intégration 3-D Secure 2.0	281
7.8 Passage à la phase de production avec 3-D Secure 2.0.....	281
7.9 Codes TransStatus de la solution 3-D Secure 2.0	281
7.10 Codes de résultat liés au code de vérification d'authentification du titulaire de carte (CAVV)	282
7.10.1 Codes de résultat liés au code de vérification d'authentification du titulaire de carte (CAVV) de Visa.....	283
7.10.2 Codes de résultat liés au code de vérification d'authentification du titulaire de carte (CAVV) de Mastercard.....	284
7.10.3 Codes de résultat liés au code de vérification d'authentification du titulaire de carte (CAVV) de American Express.....	285
8 Tarification multidevise (TMD)	286
8.1 À propos de la tarification multidevise (TMD)	286
8.2 Méthode de traitement des transactions avec la TMD.....	286
8.3 Obtention du taux de la TMD	287
8.4 Achat utilisant la TMD	289
8.5 Achat utilisant la TMD avec 3-D Secure.....	294
8.6 Achat utilisant la TMD avec 3-D Secure et la chambre forte	299
8.7 Préautorisation utilisant la TMD	305
8.8 Préautorisation utilisant la TMD avec 3-D Secure.....	310
8.9 Préautorisation utilisant la TMD avec 3-D Secure et la chambre forte.....	315
8.10 Conclusion de préautorisation utilisant la TMD	321
8.11 Correction d'achat utilisant la TMD	325
8.12 Remboursement utilisant la TMD	328
8.13 Remboursement indépendant utilisant la TMD.....	331
8.14 Achat utilisant la TMD avec la chambre forte	336
8.15 Préautorisation utilisant la TMD avec la chambre forte	340
8.16 Remboursement indépendant utilisant la TMD avec la chambre forte.....	344
8.17 Codes de devise de la TMD	348
8.18 Codes d'erreur de la TMD	356
9 Versements Visa.....	357
9.1 À propos de la solution Versements Visa.....	357
9.2 Types de transaction Versements Visa.....	357
9.3 Envoi de transactions avec la solution Versements Visa.....	357
9.4 Recherche de plan de versements	358
9.5 Recherche de plan de versements dans la chambre forte.....	361
9.6 Objet Installment Info	364
10. Outils de protection contre la fraude en ligne.....	366
10.1 Service de vérification d'adresse.....	367
10.1.1 À propos du service de vérification d'adresse (SVA)	367
10.1.2 Objet AVS Info.....	368
10.1.3 Codes de réponse du SVA	368
10.1.4 Exemple d'une transaction utilisant le SVA	373
10.2 Numéro de vérification de carte (NVC)	374
10.2.1 À propos du numéro de vérification de carte (NVC)	374
10.2.2 Transactions nécessitant le NCV.....	374
10.2.3 Objet CVD Information	375
10.2.4 Codes de résultat liés au NVC	376
10.2.5 Exemple d'une transaction d'achat avec l'objet CVD Info.....	377
10.3 Outil de gestion des risques transactionnels	378

10.3.1 À propos de l'outil de gestion des risques transactionnels	378
10.3.2 Introduction aux demandes d'information (queries)	378
10.3.3 Demande d'information sur la session (Session Query).....	379
10.3.4 Demande d'information sur les attributs (Attribute Query)	386
10.3.5 Gérer les réponses	390
10.3.6 Ajout de balises de profilage à votre site Web	404
10.4 Intégration de tous les outils de protection contre la fraude offerts	406
10.4.1 Options de mise en œuvre de l'outil de gestion des risques transactionnels (TRMT).....	406
10.4.2 Liste de vérification de mise en œuvre.....	407
10.4.3 Prise de décision	408
11 Intégration d'Apple Pay et de Google Pay^{MC}	409
11.1 À propos de l'intégration d'Apple Pay et de Google Pay ^{MC}	409
11.2 Sommaire du processus de transaction Apple Pay	409
11.3 Sommaire du processus de transaction Google Pay ^{MC}	410
11.4 À propos de l'intégration de l'API d'Apple Pay et de Google Pay ^{MC}	411
11.4.1 Types de transaction utilisées pour Apple Pay et Google Pay ^{MC}	411
11.5 Achat utilisant le code de vérification d'authentification du titulaire de carte – Apple Pay et Google Pay ^{MC}	412
11.6 Préautorisation utilisant le code de vérification d'authentification du titulaire de carte – Apple Pay et Google Pay ^{MC}	417
12 Offlinx^{MC}	423
12.1 Qu'est-ce qu'un pixel invisible?.....	423
12.2 Offlinx ^{MC} et les transactions API	423
13 Frais de commodité	424
13.1 À propos des frais de commodité.....	424
13.2 Objet Convenience Fee Information	424
13.3 Achat avec frais de commodité	425
13.4 Achat avec renseignements sur le client et frais commodité	428
13.5 Achat avec frais de commodité utilisant 3-D Secure.....	433
14 Facturation périodique	438
14.1 À propos de la facturation périodique	438
14.2 Achat avec la facturation périodique	438
14.3 Mise à jour de la facturation périodique.....	441
14.4 Codes et champs de réponse liés à la facturation périodique	445
14.5 Renseignements d'identification au dossier et facturation périodique.....	446
15. Renseignements du client.....	448
15.1 Utiliser l'objet Customer Information	448
15.1.1 Objet Customer Info – Propriétés diverses.....	449
15.1.2 Objet Customer Info – Renseignements de facturation et d'expédition	449
15.1.3 Objet Customer Info – Information sur les articles	451
15.2 Exemple d'une transaction incluant l'objet Customer Information.....	452
16 Vérification d'état	455
16.1 À propos de la vérification d'état	455
16.2 Utiliser les champs de réponse liés à la vérification d'état	455
16.3 Exemple d'achat avec la vérification d'état	456
17 Visa Checkout.....	457
17.1 À propos de Visa Checkout.....	457
17.2 Intégration de Visa Checkout Lightbox.....	457

17.3 Flux de transactions avec Visa Checkout.....	458
17.4 Achat avec Visa Checkout.....	458
17.5 Préautorisation avec Visa Checkout – VdotMePreAuth.....	461
17.6 Conclusion avec Visa Checkout	463
17.7 Correction d'achat avec Visa Checkout	466
17.8 Remboursement avec Visa Checkout.....	468
17.9 Renseignements avec Visa Checkout	471
18 Mise à l'essai d'une solution	474
18.1 À propos du centre de ressources pour commerçants	474
18.2 Se connecter à l'environnement d'assurance de la qualité du centre de ressources pour commerçants	474
18.3 Renseignements d'identification test du centre de ressources pour commerçants.....	475
18.4 Obtenir un ID de commerce et un jeton API uniques pour l'environnement de test.....	476
18.5 Traiter une transaction.....	477
18.5.1 Sommaire	477
18.5.2 Objet HttpsPostRequest	479
18.5.3 Objet Receipt	480
18.6 Mise à l'essai d'une solution de paiement INTERAC ^{MD} en ligne.....	480
18.7 Mise à l'essai d'un module d'extension pour les commerçants	482
18.8 Mise à l'essai de Visa Checkout.....	483
18.8.1 Création d'une configuration Visa Checkout pour la mise en essai.....	484
18.9 Numéro de carte de test	484
18.9.1 Numéro de carte de test pour les transactions de niveaux 2 et 3.....	485
18.9.2 Cartes de Test pour Visa Checkout	485
18.10 Simulation du serveur de traitement	485
19 Passage à l'environnement de production.....	487
19.1 Activation d'un compte de magasin dans l'environnement de production....	487
19.2 Configuration d'un commerce pour l'environnement de production	487
19.2.1 Configurer un commerce de paiement en ligne d'INTERAC ^{MD} pour l'environnement de production.....	488
19.3 Exigences relatives aux reçus	490
19.3.1 Exigences de certification	490
Annexe A Définition des champs de demande	491
A.1 Définition des champs de demande – Champs de connection.....	491
A.2 Définition des champs de demande – Champs de base	492
A.3 Définition des champs de demande – Renseignements d'identification au dossier	497
A.4 Définition des champs de demande – Chambre forte	499
A.5 Définition des champs de demande – Transactions de niveaux 2 et 3 de Visa ..	501
A.6 Définition des champs de demande – Transactions de niveaux 2 et 3 de Mastercard	511
A.7 Définition des champs de demande – Transactions de niveaux 2 et 3 de Amex523	
A.8 Définition des champs de demande – 3-D Secure 2.0	534
A.9 Définition des champs de demande – TMD.....	537
A.10 Définition des champs de demande – Offlinx ^{MC}	539
A.11 Définition des champs de demande – Frais de commodité.....	540
A.12 Définition des champs de demande – Transaction périodiques.....	541
A.13 Définition des champs de demande – Renseignements d'identification au dossier	542

Annexe B Définition des champs de réponse.....	544
B.1 Définition des champs de réponse – 3-D Secure	557
B.2 Définition des champs de réponse – TMD	559
B.3 Définition des champs de réponse – Versements Visa	561
Annexe C Codes de réponse	568
Annexe D Messages d'erreur	583
Annexe E Listes de contrôle pour les commerçants concernant la certification des paiements en INTERAC^{MD} ligne	585
Annexe F Listes de contrôle des fournisseurs de services tiers pour les tests de certification des paiements INTERAC^{MD} en ligne.....	589
Annexe G Listes de contrôle pour les commerçants concernant la certification des paiements INTERAC^{MD} en ligne	594
Annexe H Détail du cas de test pour la certification des paiements INTERAC^{MD} en ligne.....	597
H.1 Validations communes.....	597
H.2 Cas de test.....	597
H.3 Valeurs des cas de tests frontaux de commerçants	601
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	Pendant le développement	En production
<p>Communiquez avec nos spécialistes de l'intégration des clients :</p> <p>clientintegrations@moneris.com</p>	<p>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 électroniques :</p> <p>1 866 319-7450</p> <p>eproducts@moneris.com</p>	<p>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 :</p> <p>onlinepayments@moneris.com</p> <p>1 866 319-7450</p> <p>Vous pouvez appeler en tout temps.</p>

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

<http://community.monерis.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 Java 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 Java.

Configuration requise

- Java
- 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) + signe décimal (.) + 2 chiffres (sous) après le point décimal	purchase.setAmount(amount);
	EXEMPLE : 1234567.89	
Numéro de carte de crédit	<i>Chaîne</i>	purchase.setPan(pan);

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

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

Variable	Type et limites	
ID de client	<i>Chaîne</i> 50 caractères alphanumériques	<code>purchase.setCustId(cust_id);</code>
ID de correspondance de carte	<i>Chaîne</i> 50 caractères alphanumériques	<code>purchase.setCmId(transaction_id);</code>
REMARQUE : Applicables à Offlinx ^{MC} seulement, chaque transaction doit avoir une valeur unique		
Renseignements du client	<i>Objet</i> S. O.	<code>purchase.setCustInfo(customer);</code>
Renseignements du SVA	<i>Objet</i> S. O.	<code>purchase.setAvsInfo(avsCheck);</code>
Renseignements du NVC	<i>Objet</i> S. O.	<code>purchase.setCvdInfo(cvdCheck);</code>
REMARQUE : 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. Les commerçants ne doivent pas enregistrer le NVC.		
Information sur les frais de commodité REMARQUE : 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.setRecurInfo(recurInfo);</code>
Descripteur dynamique	<i>Chaîne</i> 20 caractères alphanumériques	<code>purchase.setDynamicDescriptor(dynamic_descriptor);</code>
Indicateur de portefeuille électronique REMARQUE : 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.	<i>Chaîne</i> 3 caractères alphanumériques	<code>purchase.setWalletIndicator(wallet_indicator);</code>

Variable	Type et limites	
Renseignements d'identification au dossier cof	<i>Objet</i> S. O.	<code>purchase.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.		
Information sur le versement Pour les champs liés à cet objet, consultez la section 9.6 Objet Information sur le versement	<i>Objet</i> S. O.	<code>purchase.setInstallmentInfo(installmentInfo);</code>
REMARQUE : 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 Longueur variable	<code>cof.setIssuerId("VALUE_FOR_ISSUER_ID");</code>
REMARQUE : 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.		REMARQUE : 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)

Variable	Type et limites	
Indicateur de paiement	<p><i>Chaîne</i></p> <p>1 caractère alphabétique</p>	<pre>cof.setPaymentIndicator("PAYMENT_INDICATOR_VALUE");</pre> <p>REMARQUE : 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 sur les paiements	<p><i>Chaîne</i></p> <p>1 caractère numérique</p>	<pre>cof.setPaymentInformation("PAYMENT_INFO_VALUE");</pre> <p>REMARQUE : 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 = "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;

        InstallmentInfo installmentInfo = new InstallmentInfo();
        installmentInfo.setPlanId("130d93f9-8a5d-a78c-4741-14905057ce01");
        installmentInfo.setPlanIdRef("0000000063");
        installmentInfo.setTacVersion("1");

        Purchase purchase = new Purchase();
        purchase.setOrderId(order_id);
        purchase.setAmount(amount);
        purchase.setPan(pan);
        purchase.setExpdate(expdate);
        purchase.setCryptType(crypt);
        purchase.setDynamicDescriptor("123456");
        purchase.setInstallmentInfo(installmentInfo);
        //purchase.setWalletIndicator(""); //Refer documentation for possible values
        purchase.setCmId("8nAK8712sGaKls56"); //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");

purchase.setCofInfo(cof);

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(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("MCPAmount = " + receipt.getMCPAmount());
System.out.println("MCPCurrencyCode = " + receipt.getMCPCurrencyCode());
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();
}
}
}

```

2.2 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 2.3 Conclusion de préautorisation pour en savoir plus.

- On ne peut conclure une transaction de préautorisation qu'une seule fois.

```
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	preauth.setorderId(order_id);

Variable	Type et limites	
	alphanumériques a-Z A-Z 0-9 _ - : . @ espaces	
Montant	<p><i>Chaîne</i></p> <p>10 caractères décimaux</p> <p>Jusqu'à 7 chiffres (dollars) + signe décimal (.) + 2 chiffres (sous) après le point décimal</p> <p>EXEMPLE : 1234567.89</p>	<pre>preauth.setAmount(amount);</pre>
Numéro de carte de crédit	<p><i>Chaîne</i></p> <p>Maximum 20 caractères alphanumériques</p>	<pre>preauth.setPan(pan);</pre>
Date d'expiration	<p><i>Chaîne</i></p> <p>4 caractères alphanumériques</p> <p>AAMM</p>	<pre>preauth.setExpDate(expiry_date);</pre>
Indicateur de commerce électronique	<p><i>Chaîne</i></p> <p>1 caractère alphanumérique</p>	<pre>preauth.setCryptType(crypt);</pre>

Champs de demande pour les transactions de préautorisation (facultatifs)

Variable	Type et limites	
Descripteur dynamique	<i>Chaîne</i>	<pre>preauth.setDynamicDescriptor(dynamic_descriptor);</pre>
Pour les transactions de préautorisation REMARQUE : 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.	<p>20 caractères alphanumériques</p> <p>Total de 22 caractères incluant votre nom de commerçant et un</p>	

Variable	Type et limites	
Autrement, la valeur du champ doit être renvoyée durant la conclusion de préautorisation.	séparateur REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <> \$ % = ? ^ { } [] \	
ID de correspondance de carte REMARQUE : Applicables à Offlinx ^{MC} seulement, chaque transaction doit avoir une valeur unique	<i>Chaîne</i> 50 caractères alphanumériques	<code>preauth.setCmId(transaction_id);</code>
Renseignements du client	<i>Objet</i> S. O.	<code>preauth.setCustInfo(customer);</code>
Renseignements du SVA	<i>Objet</i> S. O.	<code>preauth.setAvsInfo(avcCheck);</code>
Renseignements du NVC 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.	<i>Objet</i> S. O.	<code>preauth.setCvdInfo(cvdCheck);</code>
ID de client	<i>Chaîne</i> 50 caractères alphanumériques REMARQUE : Certains caractères spéciaux ne sont pas autorisés :	<code>preauth.setCustId(cust_id);</code>

Variable	Type et limites	
	<>\$ % = ?^{}[]\	
Indicateur de portefeuille électronique	<p><i>Chaîne</i></p> <p>3 caractères alphanumériques</p> <p>REMARQUE : 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.</p>	<pre>preauth.setWalletIndicator(wallet_indicator);</pre>
Autorisation finale	<p><i>Chaîne</i></p> <p>true/false</p> <p>REMARQUE : Applicable uniquement aux transactions par carte Mastercard</p>	<pre>preauth.setFinalAuth("true");</pre>
Renseignements d'identification au dossier 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.	<i>Objet</i> S. O.	<pre>cof.setCofInfo(cof);</pre>
Information sur le versement Pour les champs liés à cet objet, consultez la section 9.6 Objet	<i>Objet</i> S. O.	<pre>preauth.setInstallmentInfo(installmentInfo);</pre>

Variable	Type et limites
Information sur le versement	
REMARQUE : 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 Longueur variable	<code>cof.setIssuerId("VALUE_FOR_ISSUER_ID");</code> REMARQUE : 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	<code>cof.setPaymentIndicator("PAYMENT_INDICATOR_VALUE");</code> REMARQUE : 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 sur les paiements	<i>Chaîne</i> 1 caractère numérique	<code>cof.setPaymentInformation("PAYMENT_INFO_VALUE");</code> REMARQUE : 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
	d'identification au dossier (cof)

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 status_check = false;

PreAuth preauth = new PreAuth();
preauth.setOrderId(order_id);
preauth.setAmount(amount);
preauth.setPan(pan);
preauth.setExpdate(expdate);
preauth.setCryptType(crypt);
preauth.setInstallmentInfo(installmentInfo);
//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");
//optional - Credential on File details

InstallmentInfo installmentInfo = new InstallmentInfo();
installmentInfo.setPlanId("130d93f9-8a5d-a78c-4741-14905057ce01");
installmentInfo.setPlanIdRef("0000000063");
installmentInfo.setTacVersion("1");

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 or comment out this line 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();
}
}
}

```

2.3 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
----------	-----------------

Variable	Type et limites	
ID de commerce	<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	<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 : https://esqa.moneris.com/mpg/?chlang=fr</p> <p>Production : https://www3.moneris.com/mpg/?chlang=fr</p>

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

Variable	Type et limites	
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques a-Z A-Z 0-9 _ - : . @ espaces	<pre>completion.setorderId(order_id);</pre>
Montant de conclusion	<i>Chaîne</i> 10 caractères décimaux Jusqu'à 7 chiffres (dollars) +	<pre>completion.setCompAmount(comp_amount);</pre>

Variable	Type et limites	
	signe décimal (.) + 2 chiffres (sous) après le point décimal EXEMPLE : 1234567.89	
Numéro de transaction	<i>Chaîne</i> 255 caractères (alphanumériques, trait d'union ou trait de soulignement) Longueur variable	<code>completion.setTxnNumber(txn_number);</code>
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	<code>completion.setCryptType(crypt);</code>

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 <div style="background-color: #e0f2e0; padding: 5px;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\ </div>	<code>completion.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;"> REMARQUE : Certains caractères </div>	<code>completion.setDynamicDescriptor(dynamic_descriptor);</code>

Variable	Type et limites	Description
de préautorisation.	spéciaux ne sont pas autorisés : <>\$%=?^{}[]\	
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 or comment out this line 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();
}
}
}

```

2.4 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 a-Z A-Z 0-9 _ - : @ espaces	forcepost.setOrderId(order_id);
Montant	<i>Chaîne</i> 10 caractères décimaux Jusqu'à 7 chiffres (dollars) + signe décimal (.) + 2 chiffres (sous) après le point décimal EXEMPLE : 1234567.89	forcepost.setAmount(amount);
Numéro de carte de crédit	<i>Chaîne</i> Maximum 20 caractères alphanumériques	forcepost.setPan(pan);
Date d'expiration	<i>Chaîne</i> 4 caractères alphanumériques AAMM	forcepost.setExpDate(expiry_date);

Variable	Type et limites	
Code d'autorisation	<i>Chaîne</i> 8 caractères alphanumériques	forcepost.setAuthCode(auth_code);
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	forcepost.setCryptType(crypt);

Champs de demande pour les transactions forcées (facultatifs)

Variable	Type et limites	
ID de client	<i>Chaîne</i> 50 caractères alphanumériques <div style="background-color: #e0f2fd; padding: 5px; border: 1px solid #d9ead3;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\ </div>	forcepost.setCustId(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; border: 1px solid #d9ead3;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\ </div>	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.setCustId(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 or comment out this line 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();
}
}
}

```

2.5 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	<i>Chaîne</i> 50 caractères alphanumériques a-Z A-Z 0-9 _ - : . @ espaces	purchasecorrection.setorderId(order_id);
Numéro de transaction	<i>Chaîne</i> 255 caractères (alphanumériques, trait d'union ou trait de soulignement) Longueur variable	purchasecorrection.setTxnNumber(txn_number);

Variable	Type et limites	
Indicateur de commerce électronique	<p><i>Chaîne</i></p> <p>1 caractère alphanumérique</p>	<pre>purchasecorrection.setCryptType(crypt);</pre>

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> <div style="background-color: #e0f2f1; padding: 5px;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\ </div>	<pre>purchasecorrection.setCustomerId(cust_id);</pre>
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;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\ </div>	<pre>purchasecorrection.setDynamicDescriptor(dynamic_descriptor);</pre>

Exemple de transaction de correction d'achat

```
package Canada;
import JavaAPI.*;
public class TestCanadaPurchaseCorrection
{
  public static void main(String[] args)
  {
    String store_id = "store5";
    String api_token = "yesguy";
    String order_id = "Test1432065003686";
    String txn_number = "42014-0_10";
    String crypt = "7";
    String dynamic_descriptor = "123456";
    String processing_country_code = "CA";
    boolean status_check = false;
    PurchaseCorrection purchasecorrection = new PurchaseCorrection();
    purchasecorrection.setOrderId(order_id);
    purchasecorrection.setTxnNumber(txn_number);
```

```

purchaseCorrection.setCryptType(crypt);
purchaseCorrection.setDynamicDescriptor(dynamic_descriptor);
purchaseCorrection.setCustId("my customer id");
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(purchaseCorrection);
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());

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();
}
}
}

```

2.6 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> 10 caractères décimaux Jusqu'à 7 chiffres (dollars) + signe décimal (.) + 2 chiffres (sous) après le point décimal	refund.setAmount(amount);
Numéro de transaction	<i>Chaîne</i> 255 caractères	refund.setTxnNumber(txn_number);

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

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 or comment out this line 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 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
----------	-----------------

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 (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) + signe décimal (.) + 2 chiffres (sous) après le point décimal EXEMPLE : 1234567.89	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);

Variable	Type et limites	
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	<i>Chaîne</i> 50 caractères alphanumériques <div style="background-color: #e0f2fd; padding: 5px;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\ </div>	indrefund.setCustId(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;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\ </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.setCustId(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 or comment out this line 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();
}
}
}

```

2.8 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	cardVerification.setExpDate(expiry_date);

Variable	Type et limites	
	AAMM	
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	<code>cardVerification.setCryptType(crypt);</code>
Renseignements du SVA	<i>Objet</i> S. O.	<code>cardVerification.setAvsInfo(avsCheck);</code>
Renseignements du NVC 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.	<i>Objet</i> S. O.	<code>cardVerification.setCvdInfo(cvdCheck);</code>

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_ISSUER_ID");</code>
REMARQUE : 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.	Longueur variable	REMARQUE : 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("PAYME</code>

Variable	Type et limites	
REMARQUE : 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.	1 caractère alphabétique	<p>NT_INDICATOR_VALUE") ;</p> <p>REMARQUE : 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 sur les paiements	<i>Chaîne</i> 1 caractère numérique	<pre>cof.setPaymentInformation("PAYMENT_INFO_VALUE");</pre> <p>REMARQUE : 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 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);

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

```

cof.setPaymentInformation("2");
cof.setIssuerId("139X3130ASCXAS9");

cardVerification.setCofInfo(cof);
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(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());
}
catch (Exception e)
{
e.printStackTrace();
}
}
}
}

```

2.9 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	Chaîne	mpgReq.setstoreId(store_id);

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

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)	<pre>batchclose.setEcrno(ecr_no);</pre>

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();
    batchclose.setEcrno(ecr_no);
    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(batchclose);
    mpgReq.setStatusCheck(status_check);
    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 = " + 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 = 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();
}
}
}
```

2.10 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>	mpgReq.setstoreId(store_id);

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

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)	<pre>opentotals.setEcrno(ecr_no); ;</pre>

Exemple de 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 or comment out this line 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).

Pour plus de renseignements, consultez la section 1 Définition des champs 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 (ResPreauthCC)
- 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 4.6.10 Transformation en jeton d'une carte de crédit dans la chambre forte (ResTokenizeCC).

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 1 Ajout de jeton temporaire à la chambre forte.

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 10.2 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 2.8 Vérification de la carte avec le SVA et le NVC. Voir la section 1 Vérification de carte.

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 4.6.9 Ajout de 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 4.6.3 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 4.6.1 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 `getMpITxnForm()`.

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 9 Versements Visa.

4.5 Chambre forte et transformation en jetons du réseau

La transformation en jetons du réseau correspond à la transformation en jetons d'une carte de paiement. L'approvisionnement des paiements par l'entremise de la transformation en jetons du réseau minimise le risque d'utilisation frauduleuse des données en cas de compromission du terminal ou du compte du client.

Si la transformation en jetons du réseau est activée, Moneris approvisionne le jeton du réseau pour les cartes ajoutées de façon permanente à la chambre forte de Moneris. Lors de transactions financières subséquentes traitées avec le jeton de Moneris, Moneris utilisera le jeton du réseau correspondant, le cas échéant.

Pour en savoir plus sur la transformation en jetons du réseau, consultez la section 1 À propos de la transformation en jetons du réseau.

4.6 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.

4.6.1 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	<i>Chaîne</i> Maximum 20 caractères alphanumériques	resaddcc.setPan(pan);
Date d'expiration	<i>Chaîne</i> 4 caractères alphanumériques AAMM	resaddcc.setExpDate(expiry_date);
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	resaddcc.setCryptType(crypt);
Renseignements d'identification au dossier cof	<i>Objet</i> S. O.	resaddcc.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 les transactions d'ajout de carte de crédit à la chambre forte (facultatifs)

Variable	Type et limites	Méthode Set
ID de client	<i>Chaîne</i> 50 caractères alphanumériques <div style="background-color: #e0f2fd; padding: 5px; margin-top: 10px;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <> \$ % = ? ^ { } [] \ </div>	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);
Note	<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	<i>Chaîne</i> 15 caractères	cof.setIssuerId("VALUE_FOR_ISSUER_ID");
REMARQUE : Cette variable est		

Variable	Type et limites	Méthode Set
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.	alphanumériques Longueur variable	REMARQUE : 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 or comment out this line 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());
        }
    }
}

```

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

```

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());
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 A Définition des champs de réponse (page 544).

4.6.1.1 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 de carte de crédit à la chambre forte – ResAddCC (page 64), Ajout d'une carte de crédit chiffrée à la chambre forte – EncResAddCC (page 68), Transformation en jeton d'une carte de crédit dans la chambre forte – ResTokenizeCC (page 93), Ajout d'un jeton temporaire à la chambre forte – ResTempAdd (page 71) ou Ajout d'un jeton à la chambre forte – ResAddToken (page 89). 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.

4.6.1.2 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 2 chiffrée	<i>Chaîne</i> 40 caractères numériques	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="background-color: #e0f2e0; padding: 5px;">REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\</div>	encresaddcc.setCustId(cust_id);
Renseignements du SVA	<i>Objet</i> S. O.	encresaddcc.setAvsInfo(avscCheck);

Variable	Type et limites	Méthode Set
Adresse courriel	<i>Chaîne</i> 30 caractères alphanumériques	encresaddcc.setEmail(email);
Numéro de téléphone	<i>Chaîne</i> 30 caractères alphanumériques	encresaddcc.setPhone(phone);
Note	<i>Chaîne</i> 30 caractères alphanumériques	encresaddcc.setNote(note);
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);
    }
}

```

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

```

mpgReq.setTestMode(true); //false or comment out this line 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 A Définition des champs de réponse (page 544).

4.6.2 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 caractères numériques 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	resTempAdd.setDataKeyFormat (da ta_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("0U");

    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(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 A Définition des champs de réponse (page 544).

4.6.3 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
----------	-----------------	-------------

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

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	<p><i>Chaîne</i></p> <p>20 caractères alphanumériques</p>	resUpdateCC.setPan(pan);
Date d'expiration	<p><i>Chaîne</i></p> <p>4 caractères alphanumériques</p> <p>AAMM</p>	resUpdateCC.setExpDate(expiry_date);
Indicateur de commerce électronique	<p><i>Chaîne</i></p> <p>1 caractère alphanumérique</p>	resUpdateCC.setCryptType(crypt);
ID de client	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p> <div style="background-color: #e0f2e0; padding: 5px;"> <p>REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\</p> </div>	resUpdateCC.setCustId(cust_id);
Adresse courriel	<p><i>Chaîne</i></p> <p>30 caractères</p>	resUpdateCC.setEmail(email);

Variable	Type et limites	Méthode Set
	alphanumériques	
Numéro de téléphone	<i>Chaîne</i> 30 caractères alphanumériques	resUpdateCC.setPhone(phone);
Note	<i>Chaîne</i> 30 caractères alphanumériques	resUpdateCC.setNote(note);
Renseignements du SVA	<i>Objet</i> S. O.	resUpdateCC.setAvsInfo(avsCheck);
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	cof.setIssuerId("VALUE_FOR_ISSUER_ID");
REMARQUE : 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	Longueur variable	REMARQUE : 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)

Variable	Type et limites	Méthode Set
demandes de transaction subséquentes.		

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 = "vthBJyN1BicbRkdWFZ9flyDB2";
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 or comment out this line 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());
}
}

```

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

```

System.out.println("Email = " + receipt.getResEmail());
System.out.println("Note = " + receipt.getResNote());
System.out.println("MaskedPan = " + 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();
}
}
}

```

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 A Définition des champs de réponse (page 544).

4.6.3.1 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 2 chiffrée	<i>Chaîne</i> 40 caractères numériques	enc_res_update_cc.setEncTrack2(enc_track2);

Variable	Type et limites	Méthode Set
Type d'appareil	<p><i>Chaîne</i></p> <p>30 caractères alphanumériques</p> <p>Sensible à la casse</p>	<code>enc_res_update_cc.setDeviceType(device_type);</code>

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.

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

Variable	Type et limites	Méthode Set
Indicateur de commerce électronique	<p><i>Chaîne</i></p> <p>1 caractère alphanumérique</p>	<code>enc_res_update_cc.setCryptType(crypt);</code>
ID de client	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p> <div style="background-color: #e0f2e0; padding: 10px; margin-top: 10px;"> <p>REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <code><>\$%=?^{}[]\</code></p> </div>	<code>enc_res_update_cc.setCustomerId(cust_id);</code>
Indicateur de commerce électronique	<p><i>Chaîne</i></p> <p>1 caractère alphanumérique</p>	<code>enc_res_update_cc.setCryptType(crypt);</code>

Variable	Type et limites	Méthode Set
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);
Note	<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 = "PHTM1pun7VOaSCFM2xdeP2Sim";
        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.setCustomerId(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 or comment out this line for production transactions
        mpgReq.setstoreId(store_id);
        mpgReq.setApiToken(api_token);
        mpgReq.setTransaction(enc_res_update_cc);
        mpgReq.send();
        try
        {
    
```

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

```

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 A Définition des champs de réponse (page 544).

4.6.4 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.

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	<code>resDelete.setData(data_key);</code>

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 or comment out this line 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();
    }
  }
}

```

Exemple de transaction de suppression de profil de la 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 A Définition des champs de réponse (page 544).

4.6.5 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";
```

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

```

String data_key = "pi3ZMZoTTM8pLM9wuwzs2KBxw";
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 or comment out this line 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 A Définition des champs de réponse (page 544).

4.6.6 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 = "pi3ZMZOttM8pLM9wuwuws2KBxw";
    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 or comment out this line 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());
    }
  }
}
```

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

```

System.out.println("Email = " + receipt.getResEmail());
System.out.println("Note = " + receipt.getResNote());
System.out.println("MaskedPan = " + 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();
}
}
}

```

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 A Définition des champs de réponse (page 544).

4.6.7 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 or comment out this line 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 A Définition des champs de réponse (page 544).

4.6.8 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 TestCanadaResIsCorporatecard
{
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;
ResIsCorporatecard resIsCorporatecard = new ResIsCorporatecard();
resIsCorporatecard.setData(data_key);
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(resIsCorporatecard);
mpgReq.setStatusCheck(status_check);
```

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

```

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 A Définition des champs de réponse (page 544).

4.6.9 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	<code>resAddToken.setData(data_key);</code>
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	<code>resAddToken.setCryptType(crypt);</code>
Renseignements d'identification au dossier cof	<i>Objet</i> S. O.	<code>cof.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.

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	<i>Chaîne</i> 50 caractères alphanumériques	<code>resAddToken.setCustId(cust_id);</code>
	REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <> \$ % = ? ^ { } [] \	
Renseignements du SVA	<i>Objet</i> S. O.	<code>resAddToken.setAvsInfo(avscCheck);</code>
Adresse courriel	<i>Chaîne</i>	<code>resAddToken.setEmail(email);</code>

Variable	Type et limites	Méthode Set
	30 caractères alphanumériques	
Numéro de téléphone	<i>Chaîne</i> 30 caractères alphanumériques	resAddToken.setPhone(phone);
Note	<i>Chaîne</i> 30 caractères alphanumériques	resAddToken.setNote(note);
Format de la clé de données	<i>Chaîne</i> 2 caractères alphanumériques	resAddToken.setDataKeyFormat(data_key_format);

Champs de demande pour l'objet Credential on File Info

Variable	Type et limites	Méthode Set
ID de l'émetteur	<i>Chaîne</i>	cof.setIssuerId("VALUE_FOR_ISSUER_ID");
REMARQUE : 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.	15 caractères alphanumériques Longueur variable	REMARQUE : 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";
```

Exemple de transaction d'ajout d'un jeton à la 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 A Définition des champs de réponse (page 544).

4.6.10 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.

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);

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: 5px; border: 1px solid black; margin-top: 10px;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <> \$ % = ? ^ { } [] \ </div>	resTokenizeCC.setCustId(cust_id);
Adresse courriel	<i>Chaîne</i> 30 caractères alphanumériques	resTokenizeCC.setEmail(email);
Numéro de téléphone	<i>Chaîne</i> 30 caractères alphanumériques	resTokenizeCC.setPhone(phone);
Note	<i>Chaîne</i>	resTokenizeCC.setNote(note);

Variable	Type et limites	Méthode Set
	30 caractères alphanumériques	
Renseignements du SVA	<i>Objet</i> S. O.	resTokenizeCC.setAvsInfo(avscCheck);
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);
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	cof.setIssuerId("VALUE_FOR_ISSUER_ID");
REMARQUE : 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.	Longueur variable	REMARQUE : 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 = "hurgle";
        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 or comment out this line 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());
        }
    }
}

```

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

```

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 A Définition des champs de réponse (page 544).

4.7 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.

4.7.1 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

4.7.2 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> 10 caractères décimaux Jusqu'à 7 chiffres (dollars) + signe décimal (.) + 2 chiffres (sous) après le point décimal	resPurchaseCC.setAmount(amount);
EXEMPLE : 1234567.89		
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	resPurchaseCC.setCryptType(crypt);

Variable	Type et limites	Méthode Set
Renseignements d'identification au dossier cof	<i>Objet</i> S. O.	<code>cof.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.

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	<code>mpgReq.setStatusCheck(status_check);</code>
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)	<code>resPurchaseCC.setExpDate(expiry_date);</code>
ID de client	<i>Chaîne</i> 50 caractères alphanumériques REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <code><> \$ % = ? ^ { } [] \</code>	<code>resPurchaseCC.setCustId(cust_id);</code>
Descripteur dynamique	<i>Chaîne</i>	<code>resPurchaseCC.setDynamicDescript</code>

Variable	Type et limites	Méthode Set
	<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;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\\ </div>	<code>or (dynamic_descriptor) ;</code>
Renseignements du client	<i>Objet</i> S. O.	<code>resPurchaseCC.setCustInfo(customer);</code>
Renseignements du SVA	<i>Objet</i> S. O.	<code>resPurchaseCC.setAvsInfo(avsCheck);</code>
Renseignements du NVC	<i>Objet</i> S. O.	<code>resPurchaseCC.setCvdInfo(cvdCheck);</code>
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.		
Facturation périodique	<i>Objet</i> S. O.	<code>resPurchaseCC.setRecurInfo(recurInfo);</code>
Information sur le versement	<i>Objet</i> S. O.	<code>resPurchaseCC.setInstallmentInfo(installmentInfo);</code>
Pour les champs liés à cet objet, consultez la section 9.6 Objet Information sur le versement		

Variable	Type et limites	Méthode Set
REMARQUE : 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 Longueur variable	<code>cof.setIssuerId("VALUE_FOR_ISSUER_ID");</code> REMARQUE : 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	<code>cof.setPaymentIndicator("PAYMENT_INDICATOR_VALUE");</code> REMARQUE : 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 sur les paiements	<i>Chaîne</i> 1 caractère numérique	<code>cof.setPaymentInformation("PAYMENT_INFO_VALUE");</code> REMARQUE : 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;
        InstallmentInfo installmentInfo = new InstallmentInfo();
        installmentInfo.setPlanId("130d93f9-8a5d-a78c-4741-14905057ce01");
        installmentInfo.setPlanIdRef("000000063");
        installmentInfo.setTacVersion("1");

        ResPurchaseCC resPurchaseCC = new ResPurchaseCC();
        resPurchaseCC.setDataKey(data_key);
        resPurchaseCC.setOrderId(order_id);
        resPurchaseCC.setCustId(cust_id);
        resPurchaseCC.setAmount(amount);
        resPurchaseCC.setCryptType(crypt_type);
        resPurchaseCC.setInstallmentInfo(installmentInfo);
        //resPurchaseCC.setDynamicDescriptor(descriptor);
        //resPurchaseCC.setExpDate(expdate); //Temp Tokens only
        //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 or comment out this line 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());
        }
    }
}

```

Exemple de transaction d'achat avec la chambre forte

```

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());

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 A Définition des champs de réponse (page 544).

4.7.3 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
----------	-----------------	-------------

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) + signe décimal (.) + 2 chiffres (sous) après le point décimal EXAMPLE : 1234567.89	<code>resPreauthCC.setAmount(amount);</code>
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>
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 les transactions de préautorisation avec la chambre forte (facultatifs)

Valeur	Limites	Méthode Set
Vérification d'état	Valeur booléenne true/false	mpgReq.setStatusCheck (status_check) ;
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 #ccc; padding: 5px; margin-top: 10px;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\ </div>	resPreauthCC.setDynamicDescriptor (dynamic_descriptor) ;
Date d'expiration	<p><i>Chaîne</i></p> <p>4 caractères alphanumériques</p> <p>AAMM</p>	resPreauthCC.setExpDate (expiry_date) ;
ID de client	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\ </div>	resPreauthCC.setCustomerId (cust_id) ;
Autorisation finale	<p><i>Chaîne</i></p> <p>true/false</p>	resPreauthCC.setFinalAuth ("true") ;
REMARQUE : Applicable uniquement aux transactions par carte Mastercard		

Valeur	Limites	Méthode Set
Renseignements du client	<i>Objet</i> S. O.	<code>resPreauthCC.setCustInfo(customer);</code>
Renseignements du SVA	<i>Objet</i> S. O.	<code>resPreauthCC.setAvsInfo(avsCheck);</code>
Renseignements du NVC	<i>Objet</i> S. O.	<code>resPreauthCC.setCvdInfo(cvdCheck);</code>
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.		
Information sur le versement Pour les champs liés à cet objet, consultez la section 9.6 Objet Information sur le versement	<i>Objet</i> S. O.	<code>resPreauthCC.setInstallmentInfo(installmentInfo);</code>
REMARQUE : 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>	<code>cof.setIssuerId("VALUE_FOR_ISSUER_ID");</code>
REMARQUE : Cette variable est requise pour chaque transaction entamée par le commerçant à la suite de la première transaction. Après l'envoi de la première	15 caractères alphanumériques Longueur variable	REMARQUE : 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	<i>Chaîne</i> 1 caractère alphabétique	<pre>cof.setPaymentIndicator("PAYMENT_INDICATOR_VALUE");</pre> <p>REMARQUE : 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 sur les paiements	<i>Chaîne</i> 1 caractère numérique	<pre>cof.setPaymentInformation("PAYMENT_INFO_VALUE");</pre> <p>REMARQUE : 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;
        InstallmentInfo installmentInfo = new InstallmentInfo();
        installmentInfo.setPlanId("130d93f9-8a5d-a78c-4741-14905057ce01");
```

Exemple de transaction de préautorisation avec la chambre forte

```

installmentInfo.setPlanIdRef("0000000063");
installmentInfo.setTacVersion("1");

ResPrauthCC resPrauthCC = new ResPrauthCC();
resPrauthCC.setDataKey(data_key);
resPrauthCC.setOrderId(order_id);
resPrauthCC.setCustId(cust_id);
resPrauthCC.setAmount(amount);
resPrauthCC.setCryptType(crypt_type);
resPrauthCC.setInstallmentInfo(installmentInfo);
//resPrauthCC.setDynamicDescriptor(dynamic_descriptor);
//resPrauthCC.setExpDate(expdate); //Temp Tokens only

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

resPrauthCC.setCofInfo(cof);
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(resPrauthCC);
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());

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)

```

Exemple de transaction de préautorisation avec la chambre forte

```

{
    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 A Définition des champs de réponse (page 544).

4.7.4 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	resIndRefundCC.setOrderId(order_id);

Variable	Type et limites	Méthode Set
	a-Z A-Z 0-9 _ - : . @ espaces	
Montant	<p><i>Chaîne</i></p> <p>10 caractères décimaux</p> <p>Jusqu'à 7 chiffres (dollars) + signe décimal (.) + 2 chiffres (sous) après le point décimal</p> <p>EXEMPLE : 1234567.89</p>	<pre>resIndRefundCC.setAmount(amount) ;</pre>
Indicateur de commerce électronique	<p><i>Chaîne</i></p> <p>1 caractère alphanumérique</p>	<pre>resIndRefundCC.setCryptType(crypt) ;</pre>

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>REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <> \$ % = ? ^ { } [] \</p>	<pre>resIndRefundCC.setCustId(cust_id) ;</pre>
Date d'expiration	<p><i>Chaîne</i></p> <p>4 caractères alphanumériques</p> <p>AAMM</p>	<pre>resIndRefundCC.setExpDate(expiry_date) ;</pre>
Vérification d'état	<p><i>Valeur booléenne</i></p> <p>true/false</p>	<pre>mpgReq.setStatusCheck(status_check) ;</pre>
Descripteur dynamique	<p><i>Chaîne</i></p> <p>20 caractères</p>	<pre>resIndRefundCC.setDynamicDescriptor(dynamic_descriptor) ;</pre>

Valeur	Limites	Méthode Set
	alphanumériques Total de 22 caractères incluant votre nom de commerçant et un séparateur <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\ </div>	

Exemple de transaction de remboursement indépendant avec la chambre forte
<pre> 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 = "eRNR6lU1RD6jmgS9OPqmmmbVrk"; 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); 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(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()); } </pre>

Exemple de transaction de remboursement indépendant avec la chambre forte

```

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());
}
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 A Définition des champs de réponse (page 544).

4.7.5 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) + signe décimal (.) + 2 chiffres	resForcePostCC.setAmount(amount);

Variable	Type et limites	Méthode Set
	(sous) après le point décimal EXEMPLE : 1234567.89	
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="background-color: #e0f2fd; padding: 5px;">REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\</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="background-color: #e0f2fd; padding: 5px;">REMARQUE : Certains caractères spéciaux ne sont pas autorisés :</div>	<code>resForcePostCC.setDynamicDescriptor(dynamic_descriptor);</code>

Variable	Type et limites	Méthode Set
	<>\$%=?^{}[]\	
Vérification d'état	Valeur booléenne 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.setCustomerId(cust_id); resForcePostCC.setAmount(amount); resForcePostCC.setDataKey(data_key); resForcePostCC.setAuthCode(auth_code); resForcePostCC.setCryptType(crypt_type); resForcePostCC.setDynamicDescriptor(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(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>

Exemple de transaction forcée avec la chambre forte

```

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

```

4.7.6 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>
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.		
Renseignements d'identification au dossier cof	<i>Objet</i> S. O.	<code>rescardverify.setCofInfo(cof);</code>
REMARQUE : 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 REMARQUE : 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"); REMARQUE : 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 REMARQUE : 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"); REMARQUE : 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 sur les paiements	<i>Chaîne</i> 1 caractère numérique	cof.setPaymentInformation("PAYMENT_INFO_VALUE"); REMARQUE : 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	
		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);
        //resCardVerificationCC.setExpdate("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 or comment out this line 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());
        }
    }
}

```

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

```

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());
//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());
}
catch (Exception e)
{
e.printStackTrace();
}
}
} // end TestResCardVerificationCC

```

4.8 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.monteris.com.

5 Paiements INTERAC^{MD} en ligne

- 5.1 À propos des transactions de paiement INTERAC^{MD} en ligne
- 5.2 Autres documents et références
- 5.3 Exigences du site Web et de certification
- 5.4 Flux de transaction pour les paiements INTERAC^{MD} en ligne
- 5.5 Envoi d'une transaction d'achat avec INTERAC^{MD} en ligne
- 5.6 Achat avec INTERAC^{MD} en ligne
- 5.7 Remboursement avec INTERAC^{MD} en ligne
- 5.8 Définitions des champs liés aux paiements INTERAC^{MD} en ligne

5.1 À propos des transactions de paiement INTERAC^{MD} en ligne

Le mode de paiement INTERAC^{MD} en ligne permet aux titulaires de carte de payer avec leur compte bancaire en ligne. Ce mode de paiement peut être combiné avec la solution API .NET de Passerelle Moneris pour accepter les paiements en ligne par cartes de crédit et de débit.

Les transactions INTERAC^{MD} en ligne traitées par l'API .NET se déroulent en deux étapes :

1. Le titulaire de carte garantit la disponibilité des fonds pour l'achat en utilisant son processus bancaire en ligne.
2. Le commerçant confirme le paiement en envoyant une demande d'achat INTERAC^{MD} en ligne à Moneris par l'entremise de l'API .NET.

Tous les objets de transaction définis dans la présente section peuvent être envoyés à l'objet de connexion `HttpsPostRequest` défini dans la section 18.5 Traitement d'une transaction.

Les transactions de paiement INTERAC^{MD} en ligne sont uniquement offertes aux **intégrations canadiennes**.

5.2 Autres documents et références

INTERAC^{MD} en ligne est offert par Acxsys Corporation, un utilisateur autorisé du logo *Interac*. Consultez la documentation et les sites Web suivants pour obtenir de plus amples renseignements.

Directives des commerçants pour les paiements INTERAC^{MD} en ligne

Consultez le portail pour développeurs de Moneris (<https://developer.moneris.com>, en anglais seulement) pour accéder aux plus récents documents et fichiers à télécharger.

Vous y trouverez les exigences pour chaque page visitées par les consommateurs dans un site Web de commerçant typique doté des paiements INTERAC^{MD} en ligne. Vous y trouverez également les exigences liées à n'importe quelle page (qui ne sont pas associées à une page en particulier).

Logos

Consultez le portail pour développeurs de Moneris (<https://developer.moneris.com>, en anglais seulement) pour accéder aux logos et aux fichiers à télécharger.

5.3 Exigences du site Web et de certification

5.3.1 Éléments à fournir à Moneris

Reportez-vous aux directives du commerçant indiquées dans la section 5.2 pour savoir comment utiliser adéquatement les logos et le terme « Paiements INTERAC^{MD} en ligne ». Vous devez fournir à Moneris les données d'inscription suivantes :

- Logo du commerçant à afficher sur la passerelle de paiement INTERAC^{MD} en ligne
- En français et en anglais
- 120 x 30 pixels
- Format PNG (seul format pris en charge)

- Nom de l'entreprise du commerçant
- En anglais et en français
- Maximum de 30 caractères

- Liste de toutes les URL de renvoi Soit les URL qui peuvent rediriger le client sur la passerelle de paiement INTERAC^{MD} en ligne
- Liste de toutes les URL pouvant apparaître dans le champ IDEBIT_FUNDEDURL du formulaire https POST envoyé à la passerelle de paiement INTERAC^{MD} en ligne
- Liste de toutes les URL pouvant apparaître dans le champ IDEBIT_NOTFUNDEDURL du formulaire https POST envoyé à la passerelle de paiement INTERAC^{MD} en ligne

Veuillez noter que si vos environnements de production et de test sont distincts, vous devez fournir les renseignements susmentionnés pour chaque environnement.

5.3.2 Processus de certification

Cas de test

Tous les commerçants indépendants, fournisseurs de service tiers et fournisseurs de panier d'achat doivent réussir le processus de certification en procédant à tous les cas de test décrits dans l'Annexe E (page 585) ainsi que dans l'Annexe F Listes de vérification concernant les tests de la certification d'INTERAC^{MD} en ligne pour les fournisseurs de service tiers à la page 589, respectivement. Ceci est requis lorsque vous avez effectué tous les tests.

Si vous apportez des changements importants concernant la fonction de paiement INTERAC^{MD} en ligne à votre site Web après avoir obtenu la certification, vous devez obtenir une nouvelle certification pour votre site Web en effectuant de nouveaux les cas de test.

Les cas de test pour la certification se trouvent à l'Annexe H (page 597), et vous y trouverez de plus amples renseignements sur chaque cas de test ainsi que toutes les exigences pour chacun de ces cas.

Captures d'écran

Vous devez fournir à Moneris des captures d'écran de votre processus de paiement montrant des exemples de transactions approuvées et refusées traitées par l'entremise du service de paiement INTERAC^{MD} en ligne.

Listes de vérification

Pour présenter en tout temps le service INTERAC en ligne comme une option de paiement sécuritaire, vous devez cocher chaque élément de la liste de vérification des exigences du commerçant, qui se trouve à l'Annexe E (page 585) ou à l'Annexe F (page 589), respectivement. Une description détaillée de chacune des exigences de ces listes de vérification se trouve dans le document Directives du commerçant concernant les paiements INTERAC^{MD} en ligne mentionné dans la section 5.2 (page 120). Si un élément ne s'applique pas à votre situation, indiquez qu'il est S. O.

Lorsque vous aurez terminé, envoyez les résultats par télécopieur ou par courriel au centre d'assistance à l'intégration de Moneris à des fins d'évaluation avant d'intégrer les changements à votre environnement de production.

5.3.3 Exigences du client

Listes de vérification

En tant que commerçant utilisant une solution de paiement INTERAC^{MD} en ligne tierce certifiée, vos clients doivent remplir le formulaire Listes de vérification des commerçants pour la certification d'INTERAC^{MD} en ligne (Annexe G, page 594). Ils **n'auront pas** besoin d'effectuer les cas de test.

Vos clients doivent également remplir la liste de vérification des exigences du commerçant (Annexe G, page 594). Assurez-vous d'indiquer clairement dans la documentation sur votre produit que vos clients doivent envoyer leurs résultats par télécopieur ou par courriel au centre d'assistance à l'intégration de Moneris à des fins d'inscription.

Captures d'écran

Vos clients doivent fournir à Moneris des captures d'écran de leur processus de paiement montrant des exemples de transactions approuvées et refusées traitées par l'entremise du service de paiement INTERAC^{MD} en ligne.

5.3.4 Délais

Veuillez noter que les commerçants faisant partie des codes de catégorie indiqués dans le Tableau 3 peuvent connaître des délais pouvant atteindre 7 jours lors de leur processus de certification ou d'inscription.

Tableau 3 : Codes de catégorie pouvant entraîner des délais de certification ou d'inscription

Code de catégorie	Type de commerçant
4812	Équipement de télécommunication, y compris la vente de téléphone
4829	Transferts d'argent – Commerçant
5045	Ordinateurs, périphériques d'ordinateur, logiciel
5732	Ventes d'électronique
6012	Institutions financières – Marchandise et services
6051	Quasi-espèces – Commerçant
6530	Chargement de valeur stockée à distance – Commerçant

Code de catégorie	Type de commerçant
6531	Fournisseur de service de paiement – Transfert d'argent pour un achat
6533	Fournisseur de service de paiement – Commerçant – Transaction de paiement

5.4 Flux de transaction pour les paiements INTERAC^{MD} en ligne

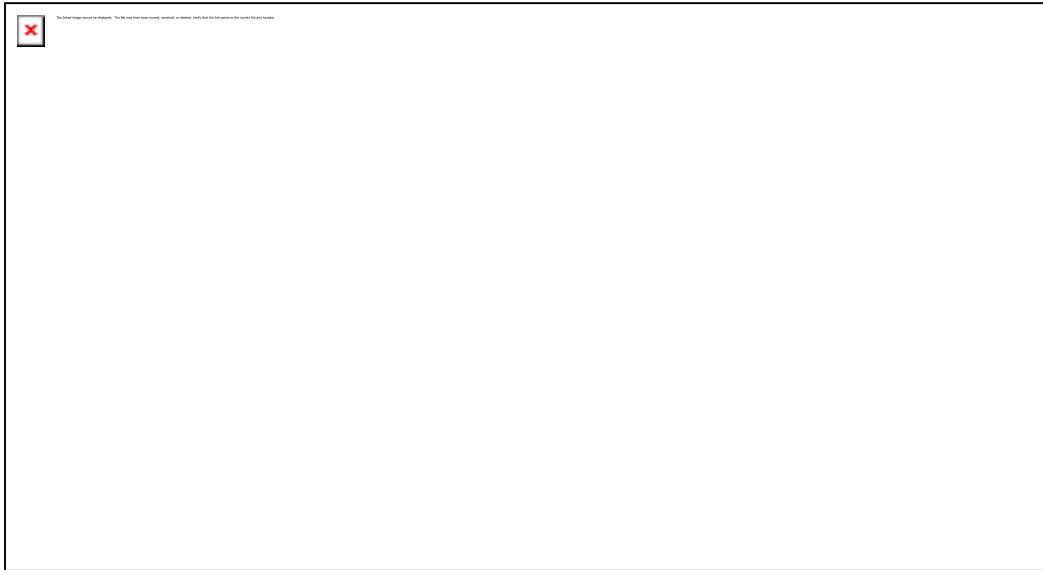


Image 2 : Flux de transaction pour les paiements INTERAC^{MD} en ligne

1. Le client sélectionne l'option Paiement INTERAC^{MD} en ligne sur le site Web du commerçant.
2. Le commerçant redirige le client à la passerelle IOP afin qu'il sélectionne l'institution financière (émetteur) de son choix. Cette étape comprend l'envoi des variables requises suivantes par formulaire via le protocole HTTPS :

- IDEBIT_MERCHNUM
- IDEBIT_AMOUNT
- IDEBIT_CURRENCY
- IDEBIT_FUNDEDURL
- IDEBIT_NOTFUNDEDURL
- IDEBIT_MERCHLANG
- IDEBIT_VERSIONIDEBIT_TERMID (facultatif)
- IDEBIT_INVOICE (facultatif)
- IDEBIT_MERCHDATA (facultatif)

3. Le client sélectionne un émetteur, puis est redirigé vers le site bancaire en ligne. Le client complète le processus bancaire en ligne et garantit que les fonds sont disponibles pour l'achat.

¹Cette valeur est affichée en sous. Par conséquent, 100 représente 1 \$.

4. Selon les résultats de l'étape 5.4, l'émetteur redirige le client vers l'URL pour les achats sans fonds suffisants (4a) ou avec fonds suffisants (4b) du commerçant par l'entremise de la passerelle IOP. Les deux URL peuvent apparaître sur la même page. Chaque URL doit valider les variables renvoyées en fonction de l'étape 5.8 (page 130) avant de continuer.

L'étape 5.4 montre les variables renvoyées lors de la redirection.

Si le client est redirigé vers l'URL pour les achats sans fonds suffisants, retournez à l'étape 5.4 et demandez un autre mode de paiement.

Si le client est redirigé vers l'URL pour les achats avec fonds suffisants, poursuivez à l'étape suivante.

5. Le commerçant envoie une demande d'achat INTERAC^{MD} en ligne à Passerelle Moneris. Pendant ce temps, le message « Veuillez patienter. » s'affiche sur l'écran du client. Cette étape devrait être effectuée dans les 30 minutes suivant la réception de la réponse à l'étape 5.4.

6. Le serveur de traitement de Moneris envoie une demande de confirmation de paiement à l'émetteur.

7. L'émetteur envoie une réponse (autorisation ou refus) au serveur de Moneris.

8. Passerelle Moneris transmet la réponse au commerçant. Si le paiement a été approuvé, le commerçant traite la commande.

Tableau 4 : Variables des URL avec ou sans fonds satisfaisants

À l'URL avec fonds satisfaisants seulement	Aux deux URL
IDEBIT_TRACK2	IDEBIT_VERSION
IDEBIT_ISSCONF	IDEBIT_ISSLANG
IDEBIT_ISSNAME	IDEBIT_TERMID (facultatif)
	IDEBIT_INVOICE (facultatif)
	IDEBIT_MERCHDATA (facultatif)

5.5 Envoi d'une transaction d'achat avec INTERAC^{MD} en ligne

5.5.1 Demande de garantie de fonds

Après avoir choisi de payer par INTERAC^{MD} en ligne, le client est redirigé par un formulaire HTML vers la passerelle de paiement INTERAC^{MD} en ligne. Vous trouverez ci-dessous un exemple de code utilisé pour envoyer la demande à la passerelle.

```
<form action='from Section 9' method='post'>
<input type='text' name='IDEBIT_INVOICE' value='your unique invoice number'>
    <input type='text' name='IDEBIT_AMOUNT' value='100'> <!-- ($1.00) use cent values instead
        of dollar.cent format -->
<input type='text' name='IDEBIT_MERCHNUM' value='from Moneris Solutions'>
<input type='text' name='IDEBIT_CURRENCY' value='CA'>
<input type='text' name='IDEBIT_FUNDEDURL' value='your funded url'>
<input type='text' name='IDEBIT_NOTFUNDEDURL' value='your not funded url'>
<input type='text' name='IDEBIT_ISSLANG' value='en'>
<input type='text' name='IDEBIT_VERSION' value='1'>
<input type="submit" name="Submit" value="Submit to Gateway">
</form>
```

5.5.2 Réponse du compte bancaire en ligne et demande de confirmation des fonds

Les variables de réponse sont renvoyées dans un formulaire HTML à l'URL pour les achats avec ou sans fonds suffisant fournie à INTERAC^{MD}.

Les variables suivantes doivent être validées (5.8, page 130) :

- IDEBIT_TRACK2
- IDEBIT_ISSCONF
- IDEBIT_ISSNAME
- IDEBIT_VERSION
- IDEBIT_ISSLANG
- IDEBIT_INVOICE

Veuillez noter que les variables IDEBIT_ISSCONF et IDEBIT_ISSNAME doivent être affichées sur le reçu du client produit par le commerçant.

Après la validation, la variable IDEBIT_TRACK2 est utilisée pour créer une transaction IDdebitPurchase, qui est envoyée à Passerelle Moneris afin de confirmer la disponibilité des fonds.

Si la validation échoue, redirigez le client vers la page principale et demandez-lui un autre mode de paiement.

Si la validation est réussie, une transaction IDdebitPurchase peut être envoyée à Passerelle Moneris.

5.6 Achat avec INTERAC^{MD} en ligne

Définition de l'objet de transaction INTERAC® Online Payment Purchase

```
IDebitPurchase IOP_Txn = new IDebitPurchase();
```

Objet HttpsPostRequest pour les transactions d'achat avec INTERAC^{MD} en ligne

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

Valeurs des transactions d'achat avec INTERAC^{MD} en ligne

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

Tableau 5 : Valeurs obligatoires de l'objet de transaction d'achat avec INTERAC^{MD} en ligne

Valeur	Type	Limites	Méthode Set
ID de commande	Chaîne	50 caractères alphanumériques	IOP_Txn.setOrderId(order_id);

Valeur	Type	Limites	Méthode Set
Montant	Chaîne	10 caractères décimaux Jusqu'à 7 chiffres (dollars) + signe décimal (.) + 2 chiffres (sous) après le point décimal EXEMPLE : 1234567.89	IOP_Txn.setAmount(amount);
Piste de données Track2	Chaîne	40 caractères alphanumériques	IOP_Txn.setTrack2(track2);

Tableau 6 : Valeurs facultatives des transactions d'achat avec INTERAC^{MD} en ligne

Valeur	Type	Limites	Méthode Set
ID du client	Chaîne	50 caractères alphanumériques	IOP_Txn.setCustId(cust_id);
Description dynamique	Chaîne	20 caractères alphanumériques	IOP_Txn.setDynamicDescriptor(dynamic_descriptor);
Renseignements du client	Objet	Non applicable. Cliquez ici . Consultez la section 15 (page 448).	IOP_Txn.setCustInfo(customer);

Exemple de transaction d'achat avec INTERAC^{MD} en ligne

```
package Canada;
import JavaAPI.*;
public class TestCanadaIDebitPurchase
{
    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 = "Lance_Briggs_55";
        String amount = "5.00";
        String track2 = "5268051119993326=0609AAAAAAAAAAAA000";
        String processing_country_code = "CA";
        boolean status_check = false;
        //***** Billing/Shipping Variables *****/
        String first_name = "Bob";
        String last_name = "Smith";
```

Exemple de transaction d'achat avec INTERAC^{MD} en ligne

```

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 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]);
/********************* Request *****/
IDebitPurchase IOP_Txn = new IDebitPurchase();
IOP_Txn.setOrderId(order_id);
IOP_Txn.setCustId(cust_id);
IOP_Txn.setAmount(amount);
IOP_Txn.setIdebitTrack2(track2);
IOP_Txn.setCustInfo(customer);
//IOP_Txn.setDynamicDescriptor("dynamicdescriptor1");
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(IOP_Txn);
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());
}
catch (Exception e)
{
}

```

Exemple de transaction d'achat avec INTERAC^{MD} en ligne

```
e.printStackTrace();
}
}
}
```

5.7 Remboursement avec INTERAC^{MD} en ligne

Pour traiter cette transaction, vous avez besoin de l'ID de commande et du numéro de transaction de la transaction d'achat avec INTERAC^{MD} en ligne d'origine.

Définition de l'objet de transaction INTERAC® Online Payment Refund

```
IDebitRefund refund = new IDebitRefund();
```

Objet HttpsPostRequest pour les transactions de remboursement avec INTERAC^{MD} en ligne

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

```
mpgReq.setTransaction(refund);
```

Valeurs facultatives de l'objet de transactions de remboursement avec INTERAC^{MD} en ligne

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

Tableau 7 : Valeurs obligatoires de l'objet de transaction de remboursement avec INTERAC^{MD} en ligne

Valeur	Type	Limites	Méthode Set
ID de commande	Chaîne	50 caractères alphanumériques	refund.setOrderId(order_id);
Montant	Chaîne	10 caractères décimaux Jusqu'à 7 chiffres (dollars) + signe décimal (.) + 2 chiffres (sous) après le point décimal EXEMPLE : 1234567.89	refund.setAmount(amount);
Numéro de transaction	Chaîne	255 caractères alphanumériques	refund.setTxnNumber(txn_number);

Tableau 8 : Valeurs facultatives de l'objet de transaction de remboursement avec INTERAC^{MD} en ligne

Valeur	Type	Limites	Méthode Set
ID du client	Chaîne	50 caractères alphanumériques	refund.setCustId(cust_id);
Vérification d'état	Valeur booléenne	true/false	mpgReq.setStatusCheck(status_check);

Modèles de programmation

Exemple de transaction de remboursement avec INTERAC^{MD} en ligne

```

package Canada;
import JavaAPI.*;
public class TestCanadaIDebitRefund
{
    public static void main(String[] args)
    {
        String store_id = "store5";
        String api_token = "yesguy";
        String order_id = "Test1435508096214";
        String amount = "5.00";
        String txn_number = "116181-0_10";
        String processing_country_code = "CA";
        String cust_id = "my customer id";
        boolean status_check = false;
        IDebitRefund refund = new IDebitRefund();
        refund.setOrderId(order_id);
        refund.setAmount(amount);
        refund.setTxnNumber(txn_number);
        refund.setCustId(cust_id);
        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(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());
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}

```

Exemple de transaction de remboursement avec INTERAC ^{MD} en ligne
}

5.8 Définitions des champs liés aux paiements INTERAC^{MD} en ligne

Tableau 9 : Définitions des champs

Valeur	Caractères		Limites	Description
IDEBIT_MERCHNU M	De 5 à 14	Chiffres et lettres majuscules		
		Ce champ est fourni par Moneris. Par exemple, 0003MONMPGXXXX.		
IDEBIT_TERMID	8	Chiffres et lettres majuscules		
		Champ facultatif		
IDEBIT_AMOUNT	De 1 à 12	Chiffres		
		Montant affiché en sous (par exemple, 1245 correspond à 12,45 \$) à porter à la carte		
IDEBIT_CURRENC Y	3	« CAD » ou « USD »		
		Devise nationale de la transaction		
IDEBIT_INVOICE	De 1 à 20	Caractères ISO-8859-1 codés limités à : Majuscules et minuscules Chiffres À Á Â Ä È É Ë Ì Ò Ù Ú Ü Ç à á â ä è é ê ë ì ò ù ú ü ý ç Espaces # \$. , - / = ? @ '		
		Champ facultatif Peut correspondre à l'ID de commande utilisé avec Passerelle Moneris pour les transactions de confirmation des fonds		
IDEBIT_MERCHDA TA	1024	Caractères ISO-8859-1 codés et restreints à un seul octet, caractère hexadécimal 20 à 7E (conformément aux ensembles de caractères US-ASCII et ISO-8859-1 Latin-1) Veuillez noter que les combinaisons de caractères suivantes pourraient ne pas être acceptées dans le champ IDEBIT_MERCHDATA : "/.", "%2E.", ".%2E", "%2E%2E", "\%2E%2E", "\\%2E.", "\\.%2E", "\\%2E%2E", "&#", "<", "%3C", ">", "%3E"		

Valeur	Caractères		Limites
	Description		
	<p>Il s'agit des données libres fournies par le commerçant qui seront transférées, inchangées, au commerçant une fois le paiement garanti dans la banque en ligne.</p> <p>Cette valeur peut servir à identifier le client, la séance ou les deux.</p>		
IDEBIT_FUNDEDURL	1024	<p>Caractères ISO-8859-1 codés et restreints à un octet ainsi qu'aux limites suivantes :</p> <ul style="list-style-type: none"> • Lettres majuscules et minuscules • Chiffres • ; / ? : @ & = + \$, - _ . ! ~ * ' () % 	
	Adresse Https à laquelle l'émetteur renvoi les titulaires de carte une fois les fonds garantis par la banque en ligne		
IDEBIT_NOTFUNDURL	1024	<p>Caractères ISO-8859-1 codés et restreints à un octet ainsi qu'aux limites suivantes :</p> <ul style="list-style-type: none"> • Lettres majuscules et minuscules • Chiffres • ; / ? : @ & = + \$, - _ . ! ~ * ' () % 	
	Adresse Https à laquelle l'émetteur renvoi les titulaires de carte lorsque le processus bancaire en ligne échoue ou est annulé		
IDEBIT_MERCHLANG	2	« en » ou « fr »	
	Langue actuelle du client sur le site du commerçant		
IDEBIT_VERSION	3	Chiffres	
	Initialement, la valeur est de 1		
IDEBIT_ISSLANG	2	« en » ou « fr »	
	Langue actuelle du client sur le site de l'émetteur		
IDEBIT_TRACK2	37	<p>Caractères ISO-8859-1 (codés et restreints à un seul octet), caractère hexadécimal 20 à 7E (conformément aux ensembles de caractères US-ASCII et ISO-8859-1 Latin-1)</p> <p>Valeur renournée par l'émetteur Valeur renournée par l'émetteur, qui inclut le PAN, la date d'expiration et l'ID de la transaction</p>	

Valeur	Caractères	Limites
		Description
IDEBIT_ISSCONF	15	<p>Caractères ISO-8859-1 codés limités à :</p> <ul style="list-style-type: none"> • Lettres majuscules et minuscules • Chiffres • À Á Â Ã È É Ê Ë Ò Ú Ü Ç à á â ã è é ê ë ò ù ü ý ç • Espaces • # \$. , - / = ? @ ' <p>Numéro de confirmation renvoyé par l'émetteur, qui doit être affiché sur la page de confirmation du commerçant et sur le reçu</p>
IDEBIT_ISSNAME	30	<p>Caractères ISO-8859-1 codés limités à :</p> <ul style="list-style-type: none"> • Lettres majuscules et minuscules • Chiffres • À Á Â Ã È É Ê Ë Ò Ú Ü Ç à á â ã è é ê ë ò ù ü ý ç • Espaces • # \$. , - / = ? @ • ' <p>Nom de l'émetteur à afficher sur la page de confirmation du commerçant et sur le reçu</p>

6 Les transactions de niveaux 2 et 3

- 6.1 À propos des transactions de niveaux 2 et 3
- 6.2 Les transactions Visa de niveaux 2 et 3
- 6.3 Les transactions Mastercard de niveaux 2 et 3
- 6.4 Les transactions American Express de niveaux 2 et 3

6.1 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 23).

6.2 Transactions Visa de niveaux 2 et 3

- 6.2.1 Types de transactions de niveaux 2 et 3 par carte Visa
- 6.2.2 Flux de transaction de niveaux 2 et 3 par carte Visa
- 6.2.3 Conclusion par carte Visa
- 6.2.5 Transaction forcée par carte Visa
- 6.2.4 Correction d'achat par carte Visa
- 6.2.6 Remboursement par carte Visa
- 6.2.7 Remboursement indépendant par carte Visa
- 6.2.8 Transaction VS Corpais
- 1 Facture VS Corpais
- 1 Facture VS Corpais – Itinéraire de passager

6.2 Types de transactions de niveaux 2 et 3 par carte 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 niveaux 2 et 3, il faut donc utiliser des transactions autres que celles de niveaux 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 les 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 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.

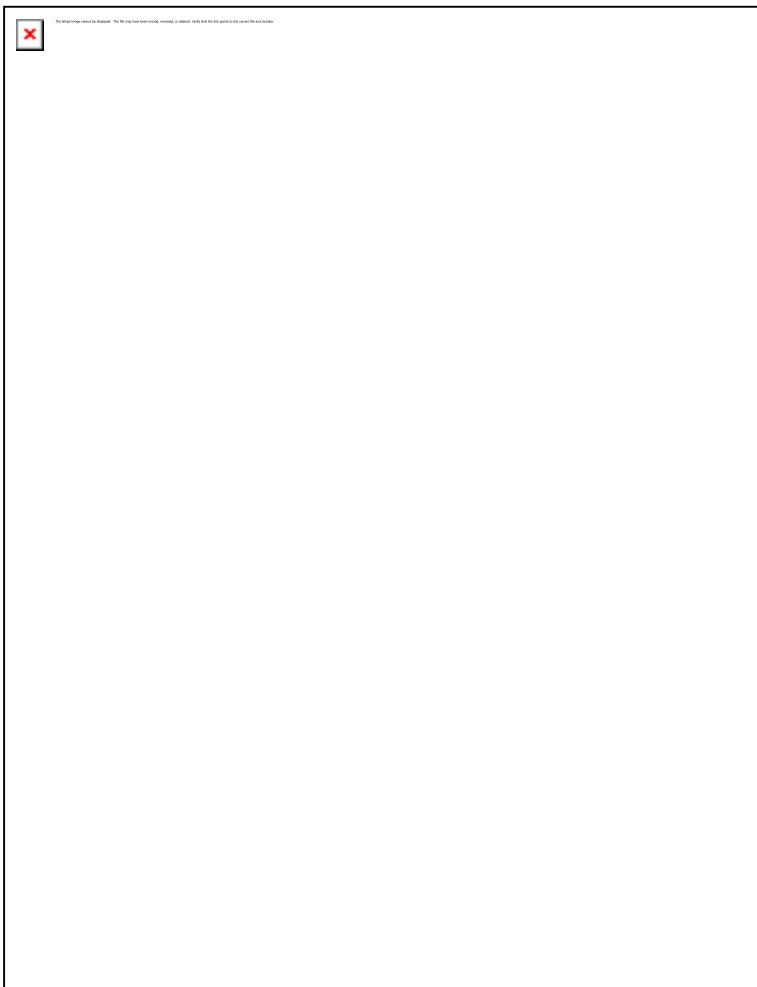
Transactions VS Corpais – (Données de niveaux 2 et 3)

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.

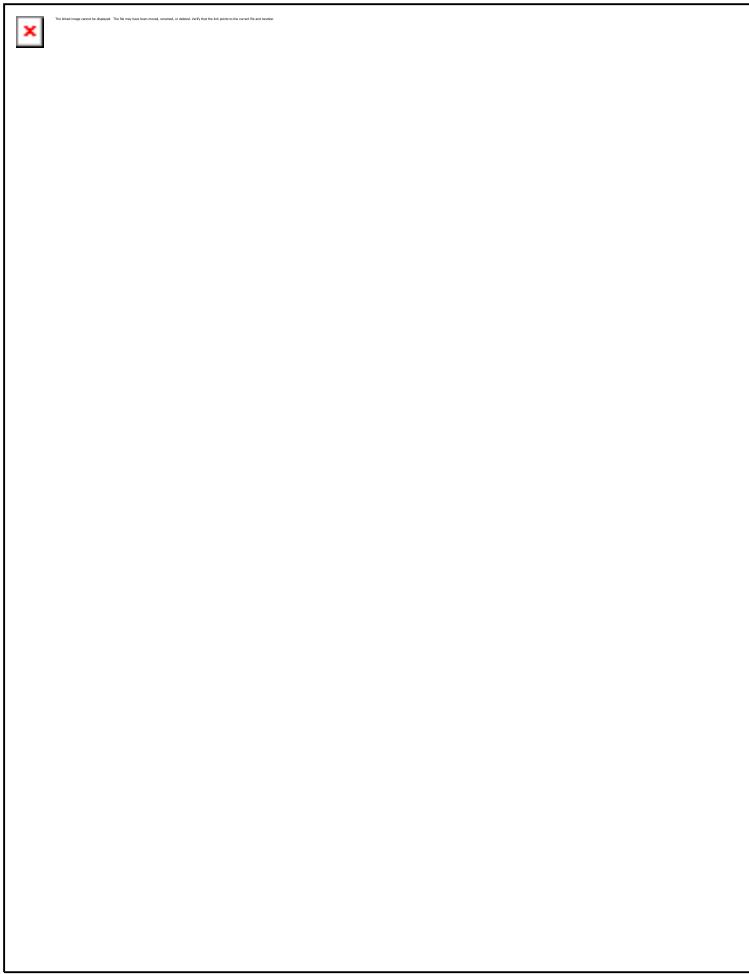
* Une correction d'achat par carte Visa peut être effectuée pour annuler une transaction tant que le lot qui contient la transaction originale reste ouvert. Lorsque la fonction de fermeture automatique est utilisée, le lot est fermé tous les jours entre 22 h et 23 h HNE.

6.2 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



6.2.3 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	<code>vsCompletion.setOrderId(order_id);</code>
Montant de la conclusion	<i>Chaîne</i> 10 caractères décimaux Jusqu'à 7 chiffres (dollars) + signe décimal (.) + 2 chiffres (sous) après le point décimal EXEMPLE : 1234567.89	<code>vsCompletion.setCompAmount(comp_amount);</code>
Numéro de transaction	<i>Chaîne</i> 255 caractères alphanumériques	<code>vsCompletion.setTxnNumber(txn_number);</code>
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	<code>vsCompletion.setCryptType(crypt);</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>vsCompletion.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	20 caractères	<code>vsCompletion</code>	Numéro

Requis*	Valeur	Limites	Méthode Set	Description
	commerçant/Référence d'entreprise unique	alphanumériques	.setMerchantVatNo (merchant_vat_no);	<p>d'enregistrement des taxes du client</p> <p>Doit être fourni si des taxes sont incluses dans la facture</p> <p>REMARQUE : Il ne doit pas y avoir que des espaces ou que des zéros.</p>
C	Taxe locale	12 caractères décimaux	vsCompletion .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	15 caractères alphanumériques	vsCompletion .setLocalTaxNo(loc	Numéro d'enregistrement

Requis*	Valeur	Limites	Méthode Set	Description
	taxes locales (TVP ou TVQ) du commerçant		al_tax_no);	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	vsCompletion .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	vsCompletion .setCri(crit);	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	vsCompletion .setCustomerCode (customer_code);	Champ facultatif qui ne sera pas envoyé à Visa, mais qui se trouvera dans les rapports de Moneris

Requis*	Valeur	Limites	Méthode Set	Description
N	Numéro de facture	17 caractères alphanumériques	vsCompletion .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 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 or comment out this line 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());
        }
    }
}

```

Exemple de conclusion par carte Visa

```

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);
}
}
}
}

```

6.2.4 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
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques	<code>vsPurchaseCorrection.setOrderId(order_id);</code>
Numéro de transaction	<i>Chaîne</i> 255 caractères alphanumériques	<code>vsPurchaseCorrection.setTxnNumber(txn_number);</code>

Variable	Type et limites	Méthode Set
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	<code>vsPurchaseCorrection.setCryptType(crypt);</code>

Exemple de transaction de correction d'achat par carte Visa

```

package Level23;
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 or comment out this line 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);
    }
  }
}

```

6.2.5 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
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques	vsForcePost.setOrderId(order_id);
Montant	<i>Chaîne</i> 10 caractères décimaux Jusqu'à 7 chiffres (dollars) + signe décimal (.) + 2 chiffres (sous) après le point décimal EXEMPLE : 1234567.89	vsForcePost.setAmount(amount);
Numéro de carte de crédit	<i>Chaîne</i> 20 caractères numériques	vsForcePost.setPan(pan);
Date d'expiration	<i>Chaîne</i>	vsForcePost.setExpDate(expiry_date);

Variable	Type et limites	Méthode Set
	4 caractères numériques Format AAMM	
Code d'autorisation	<i>Chaîne</i> 8 caractères alphanumériques	<code>vsForcePost.setAuthCode(auth_code);</code>
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	<code>vsForcePost.setCryptType(crypt);</code>

Champs de demande pour les transactions forcées par carte Visa (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

Requis*	Valeur	Limites	Méthode Set	Description
				<p>client</p> <p>Doit être fourni si des taxes sont incluses dans la facture</p> <p>REMARQUE : Il ne doit pas y avoir que des espaces ou que des zéros.</p>
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>	Numéro d'enregistrement des taxes locales (TVP ou TVQ) du

Requis*	Valeur	Limites	Méthode Set	Description
				<p>commerçant</p> <p>Il doit être fourni si des taxes sont incluses dans la facture et si la taxe locale est incluse , 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	vsForcePost .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	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 (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	vsForcePost	Champ facultatif

Requis*	Valeur	Limites	Méthode Set	Description
		alphanumériques	.setInvoiceNumber(invoice_number);	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 Level123;
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 or comment out this line for production transactions
mpgReq.setstoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(vsForcePost);
mpgReq.setStatusCheck(status_check);
}

```

Exemple de transaction forcée par carte Visa

```

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);
}
}
}
}

```

6.2.6 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
----------	-----------------	-------------

Variable	Type et limites	Méthode Set
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques	<code>vsRefund.setOrderId(order_id);</code>
Numéro de transaction	<i>Chaîne</i> 255 caractères alphanumériques	<code>vsRefund.setTxnNumber(txn_number);</code>
Montant	<i>Chaîne</i> 10 caractères décimaux Jusqu'à 7 chiffres (dollars) + signe décimal (.) + 2 chiffres (sous) après le point décimal EXEMPLE : 1234567.89	<code>vsRefund.setAmount(amount);</code>
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	<code>vsRefund.setCryptType(crypt);</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>vsRefund.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	20 caractères	<code>vsRefund</code>	Numéro

Requis*	Valeur	Limites	Méthode Set	Description
	commerçant/Référence d'entreprise unique	alphanumériques	.setMerchantVatNo(merchant_vat_no);	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	vsRefund.setLocalTax(local_tax);	Doit être identique à la taxe locale (TVP ou TVQ) apparaissant sur la facture 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. Valeur minimale de 0,01 Valeur maximale de 999 999,99 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

Requis*	Valeur	Limites	Méthode Set	Description
				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(crit);	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 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 or comment out this line 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);
        }
    }
}

```

Exemple de remboursement par carte Visa
}

6.2.7 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
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques	vsIndependentRefund.setOrderId(order_id);
Montant	<i>Chaîne</i> 10 caractères décimaux Jusqu'à 7 chiffres (dollars) + signe décimal (.) + 2 chiffres (sous) après le point décimal	vsIndependentRefund.setAmount(amount);
	EXEMPLE : 1234567.89	

Variable	Type et limites	Méthode Set
Numéro de carte de crédit	<i>Chaîne</i> 20 caractères numériques	<code>vsIndependentRefund.setPan (pan) ;</code>
Date d'expiration	<i>Chaîne</i> 4 caractères numériques Format AAMM	<code>vsIndependentRefund.setExpDate (expiry_date) ;</code>
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	<code>vsIndependentRefund.setCryptType (crypt) ;</code>

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	<code>vsIndependentRefund.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>vsIndependentRefund.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>vsIndependentRefund.setMerchantVatNo (merchant_vat_no) ;</code>	Numéro d’enregistrement des taxes du

Requis*	Valeur	Limites	Méthode Set	Description
				<p>client</p> <p>Doit être fourni si des taxes sont incluses dans la facture</p> <p>REMARQUE : Il ne doit pas y avoir que des espaces ou que des zéros.</p>
C	Taxe locale	12 caractères décimaux	vsIndependentRefund.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	vsIndependentRefund.setLocalTaxNo(local_tax_no);	Numéro d'enregistrement des taxes locales (TVP ou TVQ) du

Requis*	Valeur	Limites	Méthode Set	Description
				<p>commerçant</p> <p>Il doit être fourni si des taxes sont incluses dans la facture et si la taxe locale est incluse , 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);	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	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	vsIndependentRefund.set	Champ facultatif

Requis*	Valeur	Limites	Méthode Set	Description
		alphanumériques	tInvoiceNumber(invoice_number);	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 Level123;
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 or comment out this line for production transactions
mpgReq.setStoreId(store_id);
mpgReq.setApiToken(api_token);
mpgReq.setTransaction(vsIndependentRefund);
mpgReq.setStatusCheck(status_check);
mpgReq.send();
try
{
}

```

Exemple de transaction de remboursement indépendant par carte Visa

```

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);
}
}
}

```

6.2.8 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
----------	-----------------	-------------

Variable	Type et limites	Méthode Set
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques	<code>vsCorpais.setOrderId(order_id);</code>
Numéro de transaction	<i>Chaîne</i> 255 caractères alphanumériques	<code>vsCorpais.setTxnNumber(txn_number);</code>
vsPurcha Pour obtenir une liste des variables apparaissant dans cet objet, veuillez consulter le tableau ci-dessous.	<i>Chaîne</i> S. O.	<code>VsPurcha vsPurcha = new VsPurcha();</code> <code>vsCorpais.setVsPurch(vsPurcha, vsPurchl);</code>
vsPurchl Pour obtenir une liste des variables apparaissant dans cet objet, veuillez consulter le tableau ci-dessous.	<i>Chaîne</i> S. O.	<code>VsPurchl vsPurchl = new VsPurchl();</code> <code>vsCorpais.setVsPurch(vsPurcha, vsPurchl);</code>

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

6.2.8.1 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	<i>Chaîne</i> 30 caractères alphanumériques	Nom de l'acheteur ou du destinataire REMARQUE : Le nom est requis par l'ARC pour les transaction de plus de 150 \$.
Taux de taxe locale	<i>Chaîne</i> 4 caractères décimaux	Indique le taux de taxe détaillé appliquée en fonction du montant de taxe locale

Variable	Type et limites	Description
		<p>EXEMPLE : Une TVP de 8 % devrait être 8,0.</p>
		<p>Valeur minimale de 0,01</p> <p>Valeur maximale de 99,99</p>
		<p>REMARQUE : * Doit être fourni si la taxe locale (TVP ou TVQ) s'applique.</p>
Droits de douane	<i>Chaîne</i> 9 caractères décimaux	<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>
Traitemen t des rabais sur la facture	<i>Chaîne</i> 1 caractère numérique	<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> 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>

Variable	Type et limites	Description
Code postal d'expédition	<p><i>Chaîne</i></p> <p>10 caractères alphanumériques</p>	<p>Code postal ou code ZIP auquel la marchandise sera expédiée</p> <p>REMARQUE : * Requis s'il y a une expédition.</p> <p>Code postal alphanumérique complet – Format ANA<space>NAN requis si expédié à une adresse canadienne</p>
Code postal d'origine	<p><i>Chaîne</i></p> <p>10 caractères alphanumériques</p>	<p>Code postal ou code ZIP d'origine de la marchandise</p> <p>Pour les adresses canadiennes, un code postal alphanumérique complet au format ANA<espace>NAN valide est requis pour le commerçant.</p>
Code du pays de destination	<p>2 caractères alphanumériques</p>	<p>Code du pays où la marchandise achetée sera expédiée</p> <p>Utiliser le format ISO 3166-1 alpha-2</p> <p>REMARQUE : Requis s'il apparaît sur la facture d'une transaction internationale.</p>
Numéro de référence unique de la facture d'une TVA	<p><i>Chaîne</i></p> <p>25 caractères alphanumériques</p>	<p>Numéro de référence unique d'une facture incluant la TVA</p> <p>Doit être rempli avec le numéro de facture, qui ne peut pas être composé uniquement d'espaces ou de zéros</p>
Traitements fiscaux	<p><i>Chaîne</i></p> <p>1 caractère alphanumérique</p>	<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 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)</p>

Variable	Type et limites	Description
		commerçant) sur la facture de la transaction
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 débit.</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	<p>Montant de la TPS (excluant la TVP) ou de la TVH appliqué 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 de 999 999,99 (sans symbole)</p>

6.2.8.2 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 l'article	12 caractères alphanumériques	item_com_code	Code de commodité du produit de la ligne d'article (si ce champ n'est pas rempli, le code du produit (Product Code) doit l'être)
Y	Code du produit	12 caractères alphanumériques	product_code	Code de produit pour cette ligne d'article – code de produit du commerçant, code de produit du fabricant ou code

Requis *	Valeur	Limites	Variable/Champ	Description
				<p>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 0,0001</p> <p>Valeur maximale de 9999999999</p>
Y	Unité de mesure de l'article	2 caractères alphanumériques	item_uom	<p>Unité de mesure</p> <p>Utilisez les unités de mesure et codes permis par la norme ANSI X-12 EDI.</p>
Y	Coût unitaire de	12 caractères	unit_cost	Coût unitaire de

Requis *	Valeur	Limites	Variable/Champ	Description
	l'article	décimaux		<p>chaque article</p> <p>De 2 à 4 décimales sont acceptées</p> <p>Valeur minimale de 0,0001</p> <p>Valeur maximale de 999999,9999</p>
N	Montant de la TVA	12 caractères décimaux	vat_tax_amt	<p>Tout montant de taxe sur la valeur ajoutée ou autre taxe de vente</p> <p>Doit comporter 2 décimales</p> <p>Valeur minimale de 0,01</p> <p>Valeur maximale de 999 999,99</p>
N	Taux de la TVA	4 caractères décimaux	vat_tax_rate	<p>Taux de la taxe de vente</p> <p>EXEMPLE : Une TVP de 8 % devrait être 8,0.</p> <p>Valeur maximale de 99,99</p>
Y	Traitemet 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</p>

Requis *	Valeur	Limites	Variable/Champ	Description
				l'application des rabais
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>

6.2.8.3 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";

```

Exemple de transaction VS Corpais

```

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_amtl);
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 or comment out this line 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());
}

```

Exemple de transaction VS Corpais

```

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 Transactions de niveau 2 et 3 de Mastercard

- 6.3.1 Types de transaction de niveaux 2 et 3 par carte Mastercard
- 6.3.2 Flux de transaction de niveaux 2 et 3 par carte Mastercard
- 6.3.3 Conclusion par carte Mastercard
- 6.3.4 Transaction forcée par carte Mastercard
- 6.3.5 Correction d'achat par carte Mastercard
- 6.3.6 Remboursement par carte Mastercard
- 6.3.7 Remboursement indépendant par carte Mastercard
- 1 Transaction MC Corpais – Transactions de niveaux 2 et 3

6.3.1 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 niveaux 2 et 3, il faut donc utiliser des transactions autres que celles de niveaux 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)

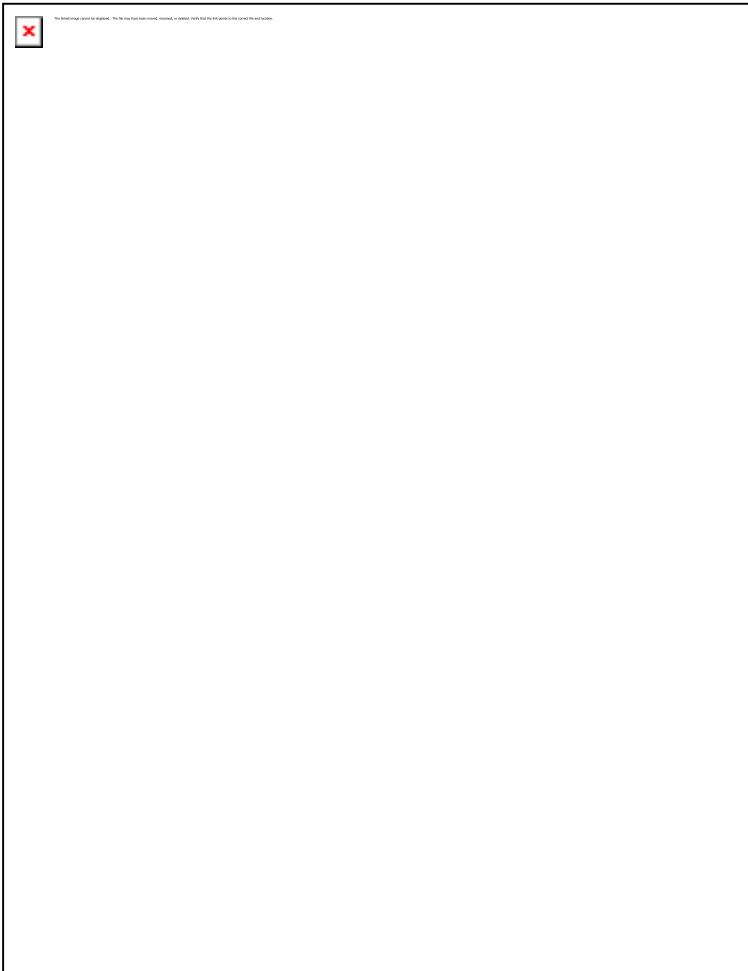
Les éléments communs de la ligne d'article MC Corpais contiendront tous les champs de données obligatoires et facultatifs pour les données de niveaux 2 et 3. Les données des éléments communs de la ligne d'article MC Corpais peuvent être envoyées lorsque la carte a été identifiée comme étant une carte 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 :

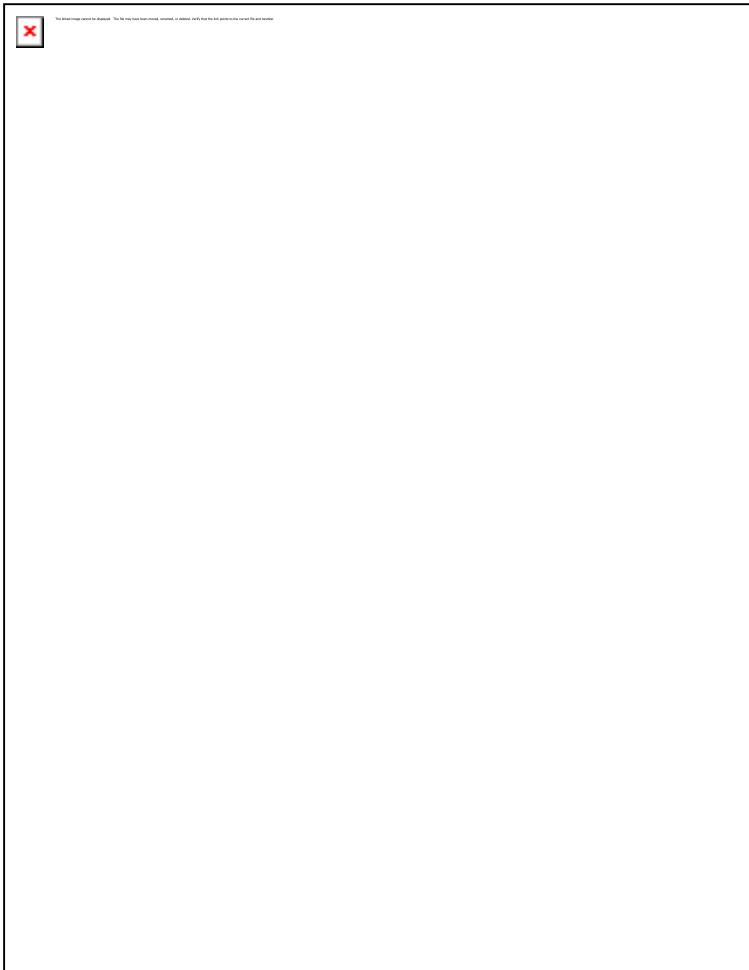
- Données communes de carte d'entreprise avec les détails de l'élément de la ligne d'article

6.3.2 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



6.3.3 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
ID 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) + signe décimal (.) + 2 chiffres (sous) après le point décimal	mcCompletion.setCompAmount(comp_amount);
	EXAMPLE : 1234567.89	
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 Level123;
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;
```

Exemple de conclusion par carte Mastercard

```

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 or comment out this line 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);
}
}
}
}

```

6.3.4 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
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques	mcforcepost.setOrderId(order_id);
Montant	<i>Chaîne</i> 10 caractères décimaux Jusqu'à 7 chiffres (dollars) + signe décimal (.) + 2 chiffres (sous) après le point décimal	mcforcepost.setAmount(amount);
	EXEMPLE : 1234567.89	
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	mcforcepost.setMerchantRefNo(merchant_ref_no);

Variable	Type et limites	Méthode Set
	alphanumériques	

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

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

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 or comment out this line 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());
        }
    }
}

```

Exemple de transaction forcée par carte Mastercard

```

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);
}
}
}
}

```

6.3.5 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
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques	mcpurchasecorrection.setOrderId(order_id);
Numéro de transaction	<i>Chaîne</i> 255 caractères	mcpurchasecorrection.setTxnNumber(txn_number);

Variable	Type et limites	Méthode Set
	alphanumériques	
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	<code>mcpurchasecorrection.setCryptType(crypt);</code>

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 or comment out this line 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);
        }
    }
}

```

Exemple de transaction de correction d'achat par carte Mastercard
}

6.3.6 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
ID de commande	50 caractères alphanumériques	mcRefund.setOrderId(order_id);
Montant	10 caractères décimaux Jusqu'à 7 chiffres (dollars) + signe décimal (.) + 2 chiffres (sous) après le point décimal EXAMPLE : 1234567.89	mcRefund.setAmount(amount);
Numéro de transaction	255 caractères alphanumériques	mcRefund.setTxnNumber(txn_number);
Indicateur de commerce électronique	1 caractère alphanumérique	mcRefund.setCryptType(crypt);
Numéro de référence du commerçant	19 caractères alphanumériques	mcRefund.setMerchantRefNo(merchant_ref_no);

Variable	Type et limites	Méthode Set

Exemple de remboursement par carte Mastercard

```

package Level23;
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 or comment out this line 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);
        }
    }
}

```

6.3.7 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 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 Independant 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
ID de commande	50 caractères alphanumériques	mcindrefund.setOrderId(order_id);
Montant	10 caractères décimaux Jusqu'à 7 chiffres (dollars) + signe décimal (.) + 2 chiffres (sous) après le point décimal	mcindrefund.setAmount(amount);
EXEMPLE : 1234567.89		

Variable	Type et limites	Méthode Set
Indicateur de commerce électronique	1 caractère alphanumérique	mcindrefund.setCryptType(crypt) ;
Numéro de carte de crédit	20 caractères numériques	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	<i>Chaîne</i> 50 caractères alphanumériques	mcindrefund.setCustId(cust_id) ;

Exemple d'une transaction de remboursement indépendant par carte Mastercard

```

package Level23;
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.setCustId(cust_id);
    mcindrefund.setAmount(amount);
    mcindrefund.setPan(pan);
    mcindrefund.setExpDate(expiry_date);
    mcindrefund.setCryptType(crypt);
  }
}
  
```

Exemple d'une transaction de remboursement indépendant par carte Mastercard

```

mcindrefund.setMerchantRefNo(merchant_ref_no);
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(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);
}
}
}

```

6.3.8 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 511).

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
ID 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);
MC Corpal	Objet	S. O.	mcCorpais.setMcCorpal(mcCorpal);

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

6.3.8.1 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 NAICS	15 caractères alphanumériques	mcCorpac.setNaicsCode(naics_code);	Code du système de classification des industries de l'Amérique du Nord (SCIANS) attribué à

Requis*	Valeur	Limites	Méthode Set	Description
				l'accepteur de la carte
N	Code de client	25 caractères alphanumériques	mcCorpac.setCustomerCode1 (customer_code1_c) ;	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
N	Date de la commande	6 caractères numériques Format AAMMJJ	mcCorpac.setOrderDate (order_date_c) ;	Date d'achat de l'article
				REMARQUE : Si présent, doit être une date valide.
N	Numéro de 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

Requis*	Valeur	Limites	Méthode Set	Description
				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 l'État ou de la province de destination	3 caractères alphanumériques	mcCorpac.setDestinationProvinceCode (destination_province_code) ;	Pays, province ou État où la marchandise sera expédiée Justifié à gauche avec des espaces de fin EXEMPLE : ONT = Ontario
N	Code du pays de destination	3 caractères alphanumériques Format ISO 3166-1 alpha-3	mcCorpac.setDestinationCountryCode (destination_country_code) ;	Code du pays où la marchandise sera expédiée Justifié à gauche avec des espaces de fin Format ISO 3166-1 alpha-3

Requis*	Valeur	Limites	Méthode Set	Description
				EXAMPLE : CAN = Canada
N	Code postal d'origine	10 caractères alphanumériques Format ANA NAN	mcCorpac.setShipFromPosCode (ship_from_pos_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	mcCorpac.setShipToPosCode (ship_to_pos_code_c) ;	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 personne-ressource autorisée	36 caractères alphanumériques	mcCorpac.setAuthorizedContactName (authorized_contact_name_c) ;	Nom d'une personne 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	40 caractères alphanumériques	mcCorpac.setAdditionalCardAcceptorData (additional_card_acceptor_data) ;	Renseignements supplémentaires sur l'accepteur de cartes

Requis*	Valeur	Limites	Méthode Set	Description
	de carte			
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"> Société par actions Inconnu Individuel ou propriétaire unique Partenariat Association, état ou fiducie Organisations exonérées d'impôts (501C) Organisation internationale Société à responsabilité limitée (SARL) Agence gouvernementale <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>

Requ is*	Valeur	Limites	Méthode Set	Description
				<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 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>

Requ is*	Valeur	Limites	Méthode Set	Description
				<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>0 = 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 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>

Requ is*	Valeur	Limites	Méthode Set	Description
				<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 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</p>

Requ is*	Valeur	Limites	Méthode Set	Description
				pas été fourni R = L'accepteur de la carte a refusé de fournir le type racial ou ethnique de l'entreprise
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) ;	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	Documents fiscaux	Jusqu'à 6 tableaux	mcCorpac.setTax (tax_c) ;	Jusqu'à 6 tableaux peuvent contenir des détails de taxes différents
				REMARQUE : Si vous utilisez cette variable, vous devez remplir tous les champs du tableau sur les taxes qui figure ci-dessous.

6.3.8.2 Transaction MC Corpal – Détails de ligne d'article

Transaction MC Corpal – Détails de ligne d'article

```
mcCorpal.setMcCorpal(customer_code1_1[0], line_item_date_1[0], ship_date_1[0],
order_date1_1[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_1[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]);
```

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_I	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	6 caractères numériques Format AAMMJJ	line_item_date_I	Date d'achat de l'article mentionnée dans les détails de la ligne d'article de la carte d'entreprise Numéro à longueur fixe de 6 chiffres, au format AAMMJJ
N	Date d'expédition	6 caractères numériques 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

Requis *	Valeur	Limites	Variable	Description
N	Date de la commande	6 caractères numériques 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 de la ligne d'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 alphanumériques	item_quantity_I	Quantité d'article acheté Jusqu'à 5 décimales sont supportées Valeur minimale de 0,0 et maximale de 9999999,99999
Y	Coût unitaire	12 caractères décimaux	unit_cost_I	Indique le coût unitaire de chaque article Doit contenir un minimum de 2 décimales (maximum de 5 décimales)

Requis *	Valeur	Limites	Variable	Description
				Valeur minimale de 0,00001 et maximale de 999999,99999
Y	Unité de mesure de l'article	12 caractères alphanumériques	item_unit_measure_I	Code de l'unité de mesure de la ligne d'article Unités de mesure et codes permis par la norme ANSI X-12 EDI
Y	Montant prolongé de l'article	9 caractères décimaux	ext_item_amount_I	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	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 attribué par le commerçant qui catégorise le mieux les articles achetés

Requis *	Valeur	Limites	Variable	Description
C	Documents fiscaux	Jusqu'à 6 tableaux	tax_l	<p>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</p> <p>REMARQUE : Si vous utilisez cette variable, vous devez remplir tous les champs du tableau sur les taxes qui figure ci-dessous.</p>

6.3.8.3 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 EXEMPLE : Une TVP 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

Requis *	Valeur	Limites	Variable	Description
Y	Numéro de taxe	20 caractères alphanumériques	tax_id_c/tax_id_l	Fournit un numéro d'identification utilisé par l'accepteur de carte auprès de 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>

6.3.8.4 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";
}

```

Exemple de transaction MC Corpais – Données communes de carte d'entreprise avec les détails de la ligne d'article

```

//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_l = new McTax[2];
tax_l[0] = new McTax();
tax_l[0].setTax(tax_amount_l[0], tax_rate_l[0], tax_type_l[0], tax_id_l[0],
tax_included_in_sales_l[0]);
tax_l[1] = new McTax();
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]);
//Create and set McCorpal for each item
McCorpal mcCorpal = new McCorpal();
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]);
mcCorpal.setMcCorpal(customer_code1_l[1], line_item_date_l[1], ship_date_l[1],
order_date1_l[1], medical_services_ship_to_health_industry_number_l[1], contract_number_l[1],

```

Exemple de transaction MC Corpais – Données communes de carte d'entreprise avec les détails de la ligne d'article

```

medical_services_adjustment_1[1], medical_services_product_number_qualifier_1[1],
product_code1_1[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 or comment out this line 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);
}
}
}
}

```

6.4 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

6.4.1 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 niveaux 2 et 3, il faut donc utiliser des transactions autres que celles de niveaux 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 Visa – (Transaction de conclusion 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. 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)

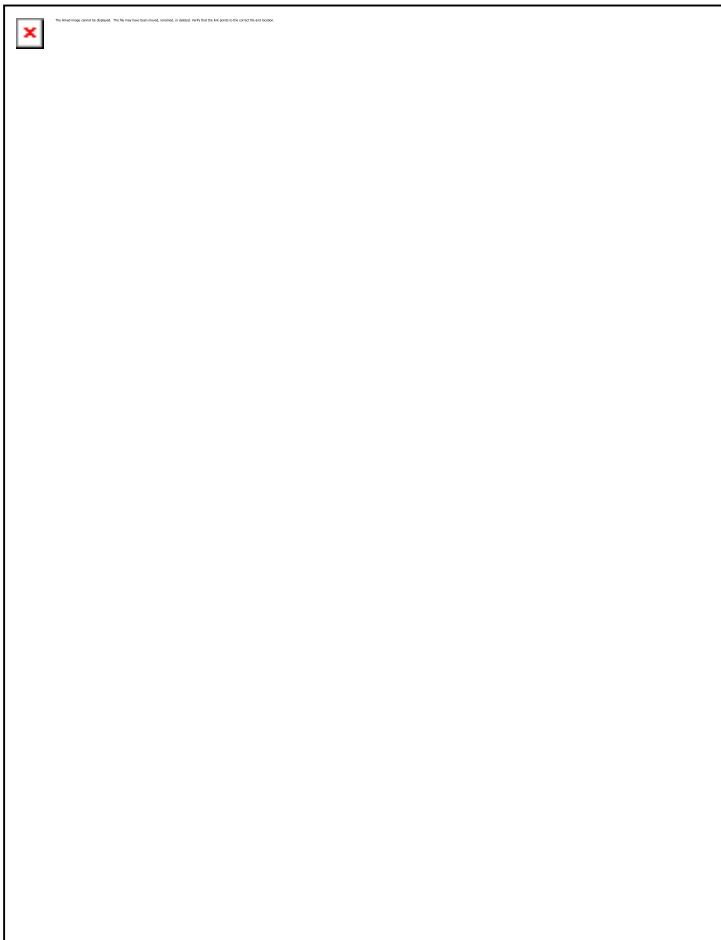
Les transactions de conclusion par carte Amex et les transaction forcée par carte Amex peuvent être annulés le jour même* où elles se produisent. Une annulation doit annuler le montant total de la transaction et supprimera toute trace de celle-ci sur le relevé du titulaire de la carte. * Une correction 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 Visa – (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.

6.4.2 Flux de transaction de niveaux 2 et 3 par carte Amex**6.4.3 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

6.4.3.1 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.

6.4.3.2 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	<pre>AxTable2 table2 = new AxTable2(); level23.setTable2(table2);</pre>

		description et une définition plus détaillées de l'objet 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	Numéro du bon de commande	22 caractères alphanumériques	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>REMARQUE : Cet élément est obligatoire si le client du commerçant fournit</p>

				un numéro de bon de commande.
N	Numéro de libération	30 caractères alphanumériques	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	Numéro de facture	8 caractères alphanumériques	table1.setBig10(big10) ;	Inclut le numéro de facture ou de référence Amex
N	N1Loop	Objet	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 Table1 – 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 l'expéditeur</td></tr> <tr> <td>ST</td><td>Nom du destinataire</td></tr> <tr> <td>40</td><td>Nom du récepteur</td></tr> </tbody> </table>	Code n101	Signification n102	R6	Nom du demandeur	BG	Nom du groupe acheteur	SF	Nom de l'expéditeur	ST	Nom du destinataire	40	Nom du récepteur
Code n101	Signification n102															
R6	Nom du demandeur															
BG	Nom du groupe acheteur															
SF	Nom de l'expéditeur															
ST	Nom du destinataire															
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</p>												

				axRef1 doit être définie; un maximum de deux valeurs axRef1 peuvent être définies.
--	--	--	--	--

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

Objet Table1 – 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

Requis*	Valeur	Limites	Variable	Description										
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> <p>Valeur n101 Dénotation ref01</p> <table border="1"> <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>40</td><td>S. O.</td></tr> </table>	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.	40	S. O.
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.													
40	S. O.													
Y	Identification de la référence	15 caractères alphanumériques	ref02	Ce champ doit être rempli pour chaque valeur ref01										

				fournie.
				Valeur ref01 Dénomination ref02
				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 n101 = R6) : Cet élément doit contenir le numéro de 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	Quantité d'article acheté Jusqu'à 2 décimales sont supportées Valeur minimale de 0,0 et maximale de 9999999999
Y	Code de l'unité ou de la base de mesure	2 caractères alphanumériques	it103	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
				EXEMPLE : EA = chaque, E5 = pouces
				Consultez le site ANSI X-12 EDI Allowable Units of Measure and Codes pour connaître la liste des codes.
Y	Prix unitaire	15 caractères décimaux	it104	Coût unitaire de chaque article Doit comporter 2 décimales Valeur minimale de 0,00 et maximale de 999 999,99
N	Code tarifaire de la base ou de l'unité	2 caractères alphanumériques	it105	Code identifiant le type de prix unitaire d'un article EXEMPLE : DR = vendeur (dealer), AP = prix conseillé (advise price) Consultez le site ASC X12 004010 Element 639 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	<p>Veuillez vous référer à la section ci-dessous pour obtenir une description et une définition des détails de l'objet.</p> <p>Un maximum de 12 valeurs AxTxi (ensembles de données sur les renseignements sur les taxes) peuvent être définies.</p> <p>REMARQUE : 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.</p>
Y	Montant final de l'article	8 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</p>

				99999,99
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	<pre>it106s[0].setIt10618(it10618[0]); it106s[1].setIt10618(it10618[1]);</pre>	<p>Valeurs acceptées :</p> <p>MG = Numéro de pièce du fabricant</p> <p>VC = Numéro de catalogue du fournisseur</p> <p>SK = Numéro de référence du fournisseur</p> <p>UP = Code universel du produit</p> <p>VP = Numéro de pièce du</p>

				fournisseur PO = Numéro du bon de commande AN = Code du bien défini par le client						
N	Numéro de produit ou de service	it1061 Taille ou 8 type it10719 <table border="1" style="margin-left: 20px;"> <tr> <td>VC</td><td>20 caractères alphanumériques</td></tr> <tr> <td>PO</td><td>22 caractères alphanumériques</td></tr> <tr> <td>Autre</td><td>30 caractères alphanumériques</td></tr> </table>	VC	20 caractères alphanumériques	PO	22 caractères alphanumériques	Autre	30 caractères alphanumériques	<pre>it106s[0].setIt10719(it10719[0]); it106s[1].setIt10719(it10719[1]);</pre>	Numéro du produit ou du service cqui orrespond au qualificateur précédent défini par la variable it10618 La longueur maximale dépend du qualificateur défini dans la variable it10618.
VC	20 caractères alphanumériques									
PO	22 caractères alphanumériques									
Autre	30 caractères alphanumériques									

* 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 – Champ de l'objet AxTxi

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> <p>CT = Taxe de comté</p>

				(facultatif) CA = Taxe municipale (facultatif) EV = Taxe environnementale (facultatif) GS = Taxe sur les biens et services (TPS) (facultatif) LS = Taxe de vente d'État et locale (facultatif) LT = Taxe de vente locale (facultatif) PG = Taxe de vente provinciale (TVP) (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	6 caractères décimaux	Cet élément peut contenir le montant de la taxe en numéraire qui correspond au code de type de taxe du champ txi01. REMARQUE : Si le champ txi02 est

				<p>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>
				<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 : 9 999.99</p> <p>Un crédit est entré comme suit : -9 999.99</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	Cet élément peut contenir le code d'exonération fiscale qui

				<p>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 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> <p>G = Exonéré (taxe de vente provinciale)</p> <p>L = Service local exonéré</p> <p>R = Exonération périodique</p> <p>U = Utilisation exonérée</p>
--	--	--	--	---

* 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);	<p>Veuillez vous référer à la section ci-dessous pour obtenir une description et une définition des détails de l'objet.</p> <p>REMARQUE : 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.</p>

* 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 dans Table 3, même si les données sur les taxes au niveau des détails des lignes d'articles sont indiquées dans Table 2.

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	<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"> CT = Taxe de comté (facultatif) CA = Taxe municipale (facultatif) EV = Taxe environnementale (facultatif) GS = Taxe sur les biens et services (TPS) (facultatif) LS = Taxe de vente d'État et locale (facultatif) LT = Taxe de vente locale (facultatif) PG = Taxe de vente provinciale (TVP)

				(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	6 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>REMARQUE : 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</p>

				<p>dans ce champ est « 9 999.99 », soit 9 999,99 \$ CA.</p> <p>Un débit est entré comme suit : 9 999.99</p> <p>Un crédit est entré comme suit : -9 999.99</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

6.4.4 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
ID de commande	Chaîne	axCompletion.setOrderId(order_id);

Variable	Type et limites	Méthode Set
	50 caractères alphanumériques	
Montant de la conclusion	<p><i>Chaîne</i> 10 caractères décimaux Jusqu'à 7 chiffres (dollars) + signe décimal (.) + 2 chiffres (sous) après le point décimal</p> <p>EXEMPLE : 1234567.89</p>	axCompletion.setCompAmount(comp_amount);
Numéro de transaction	<p><i>Chaîne</i> 255 caractères alphanumériques</p>	axCompletion.setTxnNumber(txn_number);
Indicateur de commerce électronique	<p><i>Chaîne</i> 1 caractère alphanumérique</p>	axCompletion.setCryptType(crypt);
Données de niveaux 2 et 3	<p><i>Objet</i> S. O.</p>	axCompletion.setAxLevel23(level23);

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

```

Exemple de conclusion par carte Amex

```

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(), 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]);

```

Exemple de conclusion par carte Amex

```

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);
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 or comment out this line 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);
}
}
}
}

```

6.4.5 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
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques	axForcePost.setOrderId(order_id);
Montant	<i>Chaîne</i> 10 caractères décimaux Jusqu'à 7 chiffres (dollars) + signe décimal (.) + 2 chiffres (sous) après le point décimal EXAMPLE : 1234567.89	axForcePost.setAmount(amount);
Numéro de carte de crédit	<i>Chaîne</i> 20 caractères alphanumériques	axForcePost.setPan(pan);
Date d'expiration	<i>Chaîne</i> 4 caractères alphanumériques (format AAMM)	axForcePost.setExpDate(expiry_date);
Code d'autorisation	<i>Chaîne</i>	axForcePost.setAuthCode(auth_cod

Valeur	Limites	Méthode Set
	8 caractères alphanumériques	e) ;
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	axForcePost.setCryptType(crypt) ;
Données de niveaux 2 et 3	<i>Objet</i> S. O.	axForcePost.setAxLevel23(level23) ;

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 = "P07758545"; //Purchase Order Number
    }
}

```

Exemple de transaction forcée par carte Amex

```

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 = {"DJF4R", "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(), 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], pam05[0],
pid05[0]);
it1Loop.setIt1Loop(it102[1], it103[1], it104[1], it105[1], it106s[1], txi[1], pam05[1],

```

Exemple de transaction forcée par carte Amex

6.4.6 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
ID 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
<pre>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";</pre>

Correction d'achat par carte Amex

```

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 or comment out this line 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);
}
}
}
}

```

6.4.7 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
ID 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) + signe décimal (.) + 2 chiffres (sous) après le point décimal EXEMPLE : 1234567.89	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";
```

Exemple de remboursement par carte Amex

```

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(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]);

```

Exemple de remboursement par carte Amex

```

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 or comment out this line 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)
}

```

Exemple de remboursement par carte Amex

```
{
    System.out.println(e);
}
}
}
```

6.4.8 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
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques	axIndependentRefund.setOrderId(order_id);
Montant	<i>Chaîne</i> 10 caractères décimaux Jusqu'à 7 chiffres (dollars) + signe décimal (.) + 2 chiffres (sous) après le point décimal EXEMPLE : 1234567.89	axIndependentRefund.setAmount(amount);
Numéro de carte de crédit	<i>Chaîne</i> 20 caractères alphanumériques	axIndependentRefund.setPan(pan);

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

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	<code>axIndependentRefund.setCustId(cust_id);</code>

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 = "P07758545"; //Purchase Order Number
String big05 = "RN0049858"; //Release Number
String big10 = "INV99870E"; //Invoice Number
AxRef axRef1 = new AxRef();

```

Exemple d'une transaction de remboursement indépendant par carte Amex

```

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", "FEP33", "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(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],

```

Exemple d'une transaction de remboursement indépendant par carte Amex

7 3-D Secure 2.0

- 7.1 À propos de la solution 3-D Secure 2.0
- 7.2 Créer votre intégration 3-D Secure 2.0
- 7.3 Mise en œuvre de la demande de recherche de carte (CardLookup)
- 7.4 Mise en œuvre de la demande d'authentification 3DS pour les modules d'extension des commerçants
- 7.5 Traitement du flux de contestation
- 7.6 Effectuer l'autorisation
- 7.7 Tester votre intégration 3-D Secure 2.0
- 7.8 Passage à la phase de production avec 3-D Secure 2.0
- 7.9 Codes TransStatus de la solution 3-D Secure 2.0
- 1 Codes TransStatusReason de la solution 3-D Secure 2.0
- 7.10 Codes de résultat du code de vérification d'authentification du titulaire de carte (CAVV)

7.1 À propos de la solution 3-D Secure 2.0

3-D Secure 2.0 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
- 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)

7.1.1 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.

7.1.2 Hors de portée ou non pris en charge

- Solutions intégrées aux applications
- 3RI

7.1.3 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.

7.1.4 Passage de l'outil 3-D Secure 1.0 à 2.0

L'API 3DS 2.0 est différent de l'API 3DS 1.0. Les développeurs doivent donc suivre les étapes de la section 7.2 Créer votre intégration 3-D Secure 2.0.

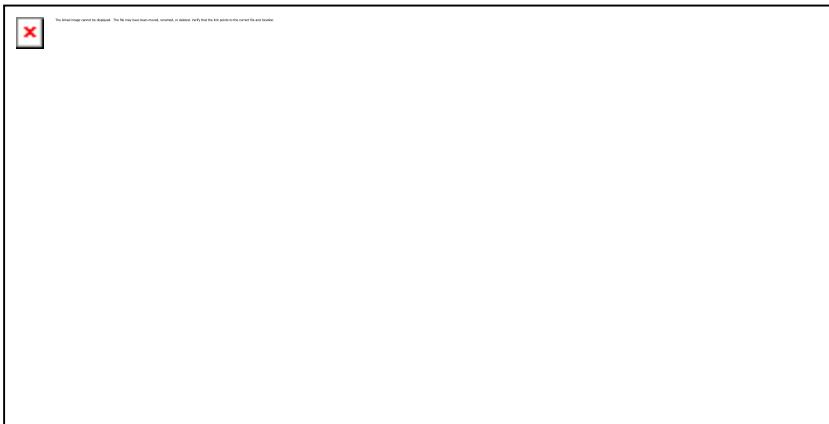
7.2 Créer votre intégration 3-D Secure 2.0

- 7.2.1 Activation de la fonction 3-D Secure
- 7.2.2 Flux des transactions avec la solution 3-D Secure 2.0

7.2.1 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.

7.2.2 Flux des transactions avec la solution 3-D Secure 2.0



L'API 3DS 2.0 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 2.0 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.

7.3 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 2.0 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 à la recherche de carte (CardLookup). 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.

Les données threeDSMethodData doivent être envoyées grâce à la méthode HTTP POST à l'URL threeDSMethodURL dans un iFrame 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

7.3.1 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	cardLookup.setOrderId(order_id);

Variable	Type et limites	
	alphanumériques a-z A-Z 0-9 _ - : . @ espaces	
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://YOURURL.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 = "4740611374762707";

        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("ThreeDSMethodURL = " + receipt.getMPIThreeDSMethodURL());
System.out.println("ThreeDSMethodData = " + receipt.getMPIThreeDSMethodData());
System.out.println("ThreeDSServerTransId = " + receipt.getMPIThreeDSServerTransId());
}
catch (Exception e)
{
e.printStackTrace();
}
}
} // end TestResMPITxn

```

7.4 Mise en œuvre de la demande d'authentification 3DS pour les modules d'extension des commerçants

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 2.0 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 :

esqa.monteris.com

Production :

www3.monteris.com

7.4.1 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

Production : 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);
```

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 de demande d'authentification 3DS des modules d'extension pour les commerçants (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	mpiThreeDSAuthentication.setOrderId(order_id);
Nom du titulaire de carte	<i>Chaîne</i> 45 caractères alphanumériques	mpiThreeDSAuthentication.setCardholderName ("CARDHOLDER_NAME_VALUE");

Variable	Type et limites	
	REMARQUE : Les caractères accentués ne sont pas autorisés.	
Numéro de carte de crédit	<i>Chaîne</i> Maximum 20 caractères alphanumériques	<code>mpiCardLookup.setPan(pan);</code>
Date d'expiration	<i>Chaîne</i> 4 caractères alphanumériques AAMM	<code>mpiThreeDSAuthentication.setExpDate(expiry_date);</code>
Clé de données	<i>Chaîne</i> 25 caractères alphanumériques	<code>mpiThreeDSAuthentication.setD ata(data_key);</code>
Montant	<i>Chaîne</i> 10 caractères décimaux Jusqu'à 7 chiffres (dollars) + signe décimal (.) + 2 chiffres (sous) après le point décimal EXEMPLE : 1234567.89	<code>mpiThreeDSAuthentication.setAmount(amount);</code>
Indicateur de conclusion 3DS	<i>Chaîne</i> 1 caractère alphanumérique Y = Conclu avec succès N = N'a pas été conclu avec succès U = Non disponible	<code>mpiThreeDSAuthentication.setThreeDSCompletionInd(THREEDSCO MPLETION_VALUE);</code>
Type de demande	<i>Chaîne</i>	<code>mpiThreeDSAuthentication.setRequestType("REQUEST_TYPE_VALU</code>

Variable	Type et limites	
	2 caractères alphanumériques 01 = Paiement entamé par le titulaire de la carte 02 = Périodique	E") ;
URL de notification	<i>Chaîne</i> 256 caractères alphanumériques	mpiThreeDSAuthentication.setNotificationURL("HTTPS://YOURURL.COM");
Taille de la fenêtre	<i>Chaîne</i> 2 caractères alphanumériques 01 = 250 x 400 02 = 390 x 400 03 = 500 x 600 04 = 600 x 400 05 = Écran complet	mpiThreeDSAuthentication.setChallengeWindowSize("CWS_VALUE");
Agent utilisateur du navigateur	<i>Chaîne</i> 2048 caractères alphanumériques	mpiThreeDSAuthentication.setBrowserUserAgent("BROWSER_USER_AGENT_VALUE");
Java activé dans le navigateur	<i>Chaîne</i> 1 caractère alphabétique T ou F	mpiThreeDSAuthentication.setBrowserJavaEnabled("BROWSER_JAVA_VALUE");
Hauteur de la fenêtre du navigateur	<i>Chaîne</i> 6 caractères numériques	mpiThreeDSAuthentication.setBrowserScreenHeight("BROWSER_SCREEN_HEIGHT_VALUE");
Largeur de la fenêtre du	<i>Chaîne</i>	mpiThreeDSAuthentication.setBrowserScreenWidth("BROWSER_SC

Variable	Type et limites	
navigateur	6 caractères numériques	REEN_WIDTH_VALUE");
Langue du navigateur	<i>Chaîne</i> 8 caractères alphanumériques	mpiThreeDSAuthentication.setBrowserLanguage("BROWSER_LANGUAGE_VALUE");
Adresse de facturation	<i>Chaîne</i> 50 caractères alphanumériques	mpiThreeDSAuthentication.setBillAddress1("BILL_STREET_ADDRESS_VALUE");
Province de facturation	<i>Chaîne</i> 3 caractères alphanumériques Défini dans la sous-division du pays ISO 3166-2	mpiThreeDSAuthentication.setBillProvince("BILL_PROV_VALUE");
Ville de facturation	<i>Chaîne</i> 50 caractères alphanumériques	mpiThreeDSAuthentication.setBillCity("BILL_CITY_VALUE");
Code postal de facturation	<i>Chaîne</i> 16 caractères alphanumériques	mpiThreeDSAuthentication.setBillPostalCode("BILL_POSTAL_CODE_VALUE");
Pays de facturation	<i>Chaîne</i> 3 caractères alphanumériques Correspond à un code de pays de 3 chiffres ISO 3166-1	mpiThreeDSAuthentication.setBillCountry("BILL_COUNTRY_VALUE");

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	<i>Chaîne</i> 3 caractères numériques	mpiThreeDSAuthentication.setCurrency("CURRENCY_VALUE");
REMARQUE : Ce champ ne devrait pas être envoyé, à moins que la tarification multidevise soit activée dans votre compte de commerçant.		
Adresse d'expédition	<i>Chaîne</i> 50 caractères alphanumériques	mpiThreeDSAuthentication.setShipAddress1("SHIP_STREET_ADDRESS_VALUE");
Province d'expédition	<i>Chaîne</i> 3 caractères alphanumériques Défini dans la sous-division du pays ISO 3166-2	mpiThreeDSAuthentication.setShipProvince("SHIP_PROV_VALUE");
Ville d'expédition	<i>Chaîne</i> 50 caractères alphanumériques	mpiThreeDSAuthentication.setShipCity("SHIP_CITY_VALUE");
Code postal d'expédition	<i>Chaîne</i> 16 caractères alphanumériques	mpiThreeDSAuthentication.setShipPostalCode("SHIP_POSTAL_CODE_VALUE");
Pays d'expédition	<i>Chaîne</i> 3 caractères alphanumériques Correspond à un code de pays de 3 chiffres ISO 3166-1	mpiThreeDSAuthentication.setShipCountry("SHIP_COUNTRY_VALUE");

Variable	Type et limites	Méthode Set
Courriel	<p><i>Chaîne</i></p> <p>254 caractères alphanumériques</p>	mpiThreeDSAuthentication.setEmail("EMAIL_VALUE");
Demande de contestation	<p><i>Chaîne</i></p> <p>1 caractère alphabétique</p> <p>Y = Oui</p> <p>N = Non</p>	mpiThreeDSAuthentication.setRequestChallenge("CHALLENGE_VALUE");

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 = "monca02932";
        String api_token = "CG8kYzGgzVU5z23irgMx";

        String processing_country_code = "CA";
        MpiThreeDSAuthentication mpiThreeDSAuthentication = new MpiThreeDSAuthentication();
        mpiThreeDSAuthentication.setOrderId("Test15978735193"); //must be the same one that was used
        in MpiCardLookup call
        mpiThreeDSAuthentication.setCardholderName("Moneris Test");
        mpiThreeDSAuthentication.setPan("4740611374762707");
        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.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.setEmail("test@email.com");
mpiThreeDSAuthentication.setRequestChallenge("Y"); // (Y|N Requesting challenge regardless of
outcome)
//*****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);
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("ChallengeURL = " + receipt.getMpiChallengeURL());
System.out.println("ChallengeData = " + receipt.getMpiChallengeData());
System.out.println("ThreeDSServerTransId = " + receipt.getMpiThreeDSServerTransId());

//In Frictionless flow, you may receive TransStatus as "Y",
//in which case you can then proceed directly to Cavy Purchase/Preauth with values below
if(receipt.getMpiTransStatus().equals("Y"))
{
System.out.println("Cavy = " + receipt.getMpiCavy());
System.out.println("ECI = " + receipt.getMpiEci());
}
}
catch (Exception e)
{
e.printStackTrace();
}
}
} // end TestResMpiTxn

```

7.5 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>
```

7.5.1 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>	mpiCavvLookup.setCRes(cres);

Variable	Type et limites
	200 caractères alphanumériques

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 =
"eyJhY3NUcmFuc01EIjoiNzQ0ZDI2NjUtNjU2Yy00ZGNiLTg3MWUtYTBkYmMwODA0OTYzIihibWVzc2FnZVR5cGUIoIJD
UmVzIiwiY2hhbGxlmdlQ29tcGxldGlvbkluZCI6IlkiLCJtZXNzYWdlVmVyc2lvbiI6IjIuMS4wIiwidHJhbnNTdGF0d
XMiOjZIiwidGhyZWVEU1NlcnZlc1RyYW5zSUQiOjJMTFkNDk4NS04ZDI1LTQwZWQtOTlkNiIjMzgwM2Z1NWU2OGYifQ
==";

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("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

```

7.6 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.

7.6.1 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^{MC}. 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^{MC} 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	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p> <p>a-Z A-Z 0-9 _ - : @ espaces</p>	cavv_purchase.setOrderId(order_id);
Montant	<p><i>Chaîne</i></p> <p>10 caractères décimaux</p> <p>Jusqu'à 7 chiffres (dollars) + signe décimal (.) + 2 chiffres (sous) après le point décimal</p>	cavv_purchase.setAmount(amount);
	EXEMPLE : 1234567.89	
Numéro de carte de crédit	<p><i>Chaîne</i></p> <p>Maximum 20 caractères alphanumériques</p>	cavvPurchase.setPan(pan);
Date d'expiration	<p><i>Chaîne</i></p> <p>4 caractères alphanumériques</p> <p>AAMM</p>	cavvPurchase.setExpDate(expiry_date);
Code de vérification d'authentification du titulaire de carte	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p>	cavvPurchase.setCavv(cavv);
REMARQUE : Pour les transactions d'achat et de préautorisation Apple Pay et Google Pay ^{MC} 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		

Variable	Type et limites	
de demande.		
Indicateur de commerce électronique REMARQUE : Pour les transactions d'achat et de préautorisation Apple Pay et Google Pay ^{MC} 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 cavvPurchase.setCryptType(crypt);	

3-D Secure 2.0 – Champs particuliers (obligatoires)

Variable	Type et limites	
Version de 3DS	<i>Chaîne</i> 1 caractère numérique cavv_purchase.setThreeDSVersion("ThreeDSVersion");	
REMARQUE : Obligatoire pour les transactions qui utilisent la version 2.0 de 3-D Secure		
ID de transaction du serveur 3DS	<i>Chaîne</i> 36 caractères numériques cavv_purchase.setThreeDSServerTransId("ThreeDSServerTransId");	
REMARQUE : Obligatoire pour les transactions utilisant la version 2.0+ de 3-D Secure, 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		

Variable	Type et limites
modules d'extension pour les commerçants.	

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 la solution 3-D Secure (facultatifs)

Variable	Type et limites
ID de client	<i>Chaîne</i> 50 caractères alphanumériques <div style="background-color: #e0f2e0; padding: 5px;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <code><> \$ % = ? ^ { } [] \</code> </div>
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;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <code><> \$ % = ? ^ { } [] \</code> </div>

Variable	Type et limites	
ID de correspondance de carte REMARQUE : Applicables à Offlinx ^{MC} seulement, chaque transaction doit avoir une valeur unique	<i>Chaîne</i> 50 caractères alphanumériques	cavv_purchase.setCmId(transaction_id);
Renseignements du client	<i>Objet</i> S. O.	cavv_purchase.setCustInfo(customer);
Renseignements du SVA	<i>Objet</i> S. O.	cavv_purchase.setAvsInfo(avsCheck);
Renseignements du NVC 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.	<i>Objet</i> S. O.	cavv_purchase.setCvdInfo(cvdCheck);
Information sur les frais de commodité REMARQUE : Ne s'applique pas lors du traitement de transactions Apple Pay ou Google Pay.	<i>Objet</i> S. O.	ConvFeeInfo convFeeInfo = new ConvFeeInfo(); cavv_purchase.setConvenienceFee(convFeeInfo);
Facturation périodique	<i>Objet</i>	cavv_purchase.setRecurInfo(recurIn

Variable	Type et limites	
recur	S. O.	fo);
REMARQUE : Pour un exemple de code concernant un achat avec la solution 3-D Secure incluant l'objet Recurring Billing Info, consultez la section 7.6.5 Achat avec la solution 3-D Secure et facturation périodique..		
Indicateur de portefeuille électronique	<p><i>Chaîne</i> 3 caractères alphanumériques</p> <p>REMARQUE : 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 Pay^{MC}. Pour plus d'information, consultez l'annexe A Définition des champs de demande.</p>	cavv_purchase.setWalletIndicator(wallet_indicator);
Renseignements d'identification au dossier cof	<p><i>Objet</i> S. O.</p> <p>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.</p>	cavv_purchase.setCofInfo(cof);

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> <p>REMARQUE : 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>REMARQUE : 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> <p>REMARQUE : 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>REMARQUE : 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 sur les paiements	<p><i>Chaîne</i></p> <p>1 caractère numérique</p>	<pre>cof.setPaymentInformation("PAYMENT_INFO_VALUE");</pre> <p>REMARQUE : 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 2.0 – Champs particuliers (facultatifs)

Variable	Type et limites

Variable	Type et limites	
ID de transaction DS	<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>De 1 à 999</p>	Il s'agit du nombre d'occurrences de la transaction.
Période	<p><i>Chaîne</i></p> <p>Valeur numérique</p> <p>De 1 à 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 AAAAMMJJ</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	<p><i>Chaîne</i></p> <p>true/false</p>	<p>Réglez cette variable à true si un montant est porté immédiatement à la carte. Autrement, réglez la variable à false.</p> <p>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.</p> <p>REMARQUE : Le montant à facturer immédiatement peut différer des montants</p>

Variable	Type et limites	Description
		subséquents.
Montant récurrent	<p><i>Chaîne</i></p> <p>10 caractères décimaux, minimum de 3 chiffres</p> <p>Jusqu'à 7 chiffres (dollars) + signe décimal (.) + 2 chiffres (sous) après le point décimal</p>	<p>Il s'agit du montant en dollars de la transaction périodique.</p> <p>Il s'agit du montant facturé à la date de départ (<code>start_date</code>) et qui sera ensuite facturé à répétition en fonction de l'intervalle défini par les valeurs <code>period</code> et <code>recur unit</code>.</p>
	EXEMPLE : 1234567.89	
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 <code>period</code> pour déterminer la fréquence de facturation.</p>

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 = "BwABApFSYyd4l2eQQFJjAAAAAA=";
        String dynamic_descriptor = "123456";
        String processing_country_code = "CA";
        String crypt_type = "5";
        boolean status_check = false;
        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.setData("3DSecure"); //set only for Interac e-commerce
        //cavvPurchase.setCmId("8nAK8712sGaAkls56"); //set only for usage with Offlinx - Unique max
    }
}

```

7.6.2 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^{MC}. 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^{MC} 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.

```
CavvPreAuth cavv_prauth = new CavvPreAuth();

HttpsPostRequest mpgReq = new HttpsPostRequest();

mpgReq.setTransaction(cavv_prauth);
```

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	<i>Chaîne</i> 50 caractères alphanumériques a-Z A-Z 0-9 _ - : . @ espaces	cavv_prauth.setorderId(order_id);
Montant	<i>Chaîne</i>	cavv_prauth.setAmount(amount);

Variable	Type et limites	
	10 caractères décimaux Jusqu'à 7 chiffres (dollars) + signe décimal (.) + 2 chiffres (sous) après le point décimal EXAMPLE : 1234567.89	
Numéro de carte de crédit	<i>Chaîne</i> Maximum 20 caractères alphanumériques	<code>cavv_preatuh.setPan(pan);</code>
Code de vérification d'authentification du titulaire de carte	<i>Chaîne</i> 50 caractères alphanumériques	<code>cavv_preatuh.setCavv(cavv);</code>
REMARQUE : Pour les transactions d'achat et de préautorisation Apple Pay et Google Pay ^{MC} 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	<code>cavv_preatuh.setExpDate(expiry_date);</code>
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	<code>cavv_preatuh.setCryptType(crypt);</code>
REMARQUE : Pour les transactions d'achat et de préautorisation Apple Pay et Google Pay ^{MC} utilisant le code de vérification d'authentification du titulaire de carte, l'indicateur de commerce électronique est un champ		

Variable	Type et limites
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 2.0 – Champs particuliers (obligatoires)

Variable	Type et limites
Version de 3DS	<i>Chaîne</i> 1 caractère numérique
REMARQUE : Obligatoire pour les transactions qui utilisent la version 2.0 de 3-D Secure	
ID de transaction du serveur 3DS	<i>Chaîne</i> 36 caractères numériques
REMARQUE : Obligatoire pour les transactions utilisant la version 2.0+ de 3-D Secure, 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.	

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

Variable	Type et limites	
ID de client	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p> <div style="background-color: #e0f2e0; padding: 5px;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <> \$ % = ? ^ { } [] \ </div>	cavv_preatuth.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;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <> \$ % = ? ^ { } [] \ </div>	cavv_preatuth.setDynamicDescriptor(dynamic_descriptor) ;
ID de correspondance de carte	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p>	cavv_preatuth.setCmId(transaction_id) ;
Renseignements du SVA	<p><i>Objet</i></p> <p>S. O.</p>	cavv_preatuth.setAvsInfo(avscheck) ;
Renseignements du NVC	<p><i>Objet</i></p> <p>S. O.</p>	cavv_preatuth.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		

Variable	Type et limites	
stockés, le NVC peut être envoyé avec les transactions entamées par le titulaire de carte seulement. Les commerçants ne doivent pas enregistrer le NVC.		
Indicateur de portefeuille électronique	<p><i>Chaîne</i></p> <p>3 caractères alphanumériques</p> <p>REMARQUE : 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 Pay^{MC}. Pour plus d'information, consultez l'annexe A Définition des champs de demande.</p>	cavv_preatuh.setWalletIndicator(wallet_indicator);
Autorisation finale	<p><i>Chaîne</i></p> <p>true/false</p> <p>REMARQUE : Applicable uniquement aux transactions par carte Mastercard</p>	cavv_preatuh.setFinalAuth("true");
Renseignements d'identification au dossier cof	<p><i>Objet</i></p> <p>S. O.</p> <p>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.</p>	cavv_preatuh.setCofInfo(cof);

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>REMARQUE : 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>REMARQUE : 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 sur les paiements	<p><i>Chaîne</i></p> <p>1 caractère numérique</p>	<pre>cof.setPaymentInformation("PAYMENT_INFO_VALUE");</pre> <p>REMARQUE : 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 2.0 – Champs particuliers (facultatifs)

Variable	Type et limites	
ID de transaction DS	<p><i>Chaîne</i></p> <p>36 caractères</p>	<pre>cavv_preatuh.setDsTransId("DsTransId");</pre>

Variable	Type et limites	
	alphanumériques	

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 = "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 = "AAABBjg0VhIOVniQEjRWAAAAAAA=";
        String dynamic_descriptor = "123456";
        String processing_country_code = "CA";
        String crypt_type = "5";
        boolean status_check = false;
        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
        alphanumerics 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("12345");//Optional - to be used only if you are using 3rd party
        3ds 2.0 service

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

        cavvPreauth.setCofInfo(cof);

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

```

7.6.3 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	<p><i>Chaîne</i></p> <p>25 caractères alphanumériques</p>	resCavvPurchaseCC.setData (data_key);
ID de commande	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p> <p>a-Z A-Z 0-9 _ - : . @ espaces</p>	resCavvPurchaseCC.setOrderId(order_id);
Montant	<p><i>Chaîne</i></p> <p>10 caractères décimaux</p> <p>Jusqu'à 7 chiffres (dollars) + signe décimal (.) + 2 chiffres (sous) après le point décimal</p> <p>EXEMPLE : 1234567.89</p>	resCavvPurchaseCC.setAmount(amount);
Code de vérification d'authentification du titulaire de carte	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p>	resCavvPurchaseCC.setCavv(cavv);
Indicateur de commerce électronique	<p><i>Chaîne</i></p> <p>1 caractère alphanumérique</p>	resCavvPurchaseCC.setCryptType(crypt);

3-D Secure 2.0 – Champs particuliers (obligatoires)

Variable	Type et limites	
Version de 3DS	<p><i>Chaîne</i></p> <p>1 caractère numérique</p>	resCavvPurchaseCC.setThreeDSVersion ("ThreeDSVersion");
REMARQUE : Obligatoire pour les transactions qui utilisent la		

Variable	Type et limites	
version 2.0 de 3-D Secure		
ID de transaction du serveur 3DS REMARQUE : Obligatoire pour les transactions utilisant la version 2.0+ de 3-D Secure, 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 caractères numériques	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	<i>Chaîne</i> 50 caractères alphanumériques REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <> \$ % = ? ^ { } [] \	resCavvPurchaseCC.setCustomerId(cust_id);
Date d'expiration	<i>Chaîne</i> 4 caractères alphanumériques AAMM	resCavvPurchaseCC.setExpDate(expiry_date);
Autorisation finale REMARQUE : Applicable uniquement aux transactions par carte Mastercard	<i>Chaîne</i> true/false	resCavvPurchaseCC.setFinalAuth("true");

3-D Secure 2.0 – Champs particuliers (facultatifs)

Variable	Type et limites	
ID de transaction DS	<p><i>Chaîne</i></p> <p>36 caractères alphanumériques</p>	resCavvPurchaseCC.setDsTransId("DsTransId");

Exemple d'achat avec la chambre forte et la solution 3-D Secure

```

package Canada;
import JavaAPI.*;
public class TestCanadaResCavvPurchaseCC
{
    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 = "AAABBJg0VhI0VniQEjRWAAAAAAA";
        String processing_country_code = "CA";
        String exp_date = "1901";
        boolean status_check = false;
        ResCavvPurchaseCC resCavvPurchaseCC = new ResCavvPurchaseCC();
        resCavvPurchaseCC.setOrderId(order_id);
        resCavvPurchaseCC.setDataKey(data_key);
        resCavvPurchaseCC.setCustomerId(cust_id);
        resCavvPurchaseCC.setAmount(amount);
        resCavvPurchaseCC.setCavv(cavv);
        resCavvPurchaseCC.setExpDate(exp_date);
        resCavvPurchaseCC.setThreeDSVersion("2"); //Mandatory for 3DS Version 2.0+
        resCavvPurchaseCC.setThreeDSServerTransId("e11d4985-8d25-40ed-99d6-c3803fe5e68f");
        //Mandatory for 3DS Version 2.0+ - obtained from MpiCavvLookup or MpiThreeDSAAuthentication
        //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 or comment out this line 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.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();
}
}
}

```

7.6.4 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> 10 caractères décimaux Jusqu'à 7 chiffres (dollars) + signe décimal (.) + 2 chiffres (sous) après le point décimal EXEMPLE : 1234567.89	resCavvPreauthCC.setAmount(amount);
Code de vérification d'authentification du titulaire de carte	<i>Chaîne</i> 50 caractères alphanumériques	resCavvPreauthCC.setCavv(cavv);

Variable	Type et limites	Méthode Set
Indicateur de commerce électronique	<p><i>Chaîne</i></p> <p>1 caractère alphanumérique</p>	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	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p> <div style="background-color: #e0f2fd; padding: 5px;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\</div>	resCavvPreauthCC.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: #e0f2fd; padding: 5px;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\</div>	resCavvPreauthCC.setDynamicDescriptor(dynamic_descriptor) ;
Date d'expiration	<p><i>Chaîne</i></p> <p>4 caractères alphanumériques</p> <p>AAMM</p>	resCavvPreauthCC.setExpDate(expiry_date) ;
ID de transaction DS	<p><i>Chaîne</i></p> <p>36 caractères alphanumériques</p>	resCavvPreauthCC.setDsTransId("DsTransId") ;

Variable	Type et limites	Méthode Set
Autorisation finale	<i>Chaîne</i>	resCavvPreauthCC.setFinalAuth("true");
REMARQUE : Applicable uniquement aux transactions par carte Mastercard	true/false	

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 = "yesguyl";
String data_key = "4INQR1A8ocxD0oafSz501ADXY";
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 expdate = "1901";
boolean status_check = false;
ResCavvPreauthCC resCavvPreauthCC = new ResCavvPreauthCC();
resCavvPreauthCC.setOrderId(order_id);
resCavvPreauthCC.setDataKey(data_key);
resCavvPreauthCC.setCustId(cust_id);
resCavvPreauthCC.setAmount(amount);
resCavvPreauthCC.setCavv(cavv);
resCavvPreauthCC.setExpDate(expdate);

resCavvPreauthCC.setThreeDSVersion("2"); //Mandatory for 3DS Version 2.0+
resCavvPreauthCC.setThreeDSServerTransId("e11d4985-8d25-40ed-99d6-c3803fe5e68f"); //Mandatory
for 3DS Version 2.0+ - 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 or comment out this line 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());
}
catch (Exception e)
{
e.printStackTrace();
}
}
}
}

```

7.6.5 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_recur = "12";
String period = "1";
String recur_amount = "5.00";
/**************** Recur Object Option1 *****/
Recur recurring_cycle = new Recur(recur_unit, start_now, start_date,
num_recur, period, recur_amount);
CavvPurchase cavvPurchase = new CavvPurchase();
cavvPurchase.setOrderId(order_id);
cavvPurchase.setCustId(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"); //Mandatory for 3DS Version 2.0+
cavvPurchase.setThreeDSServerTransId("e11d4985-8d25-40ed-99d6-c3803fe5e68f"); //Mandatory for
3DS Version 2.0+ - obtained from MpiCavvLookup or MpiThreeDSAAuthentication
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 or comment out this line 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();
    }
}
```

7.7 Tester votre intégration 3-D Secure 2.0

Dans la phase de test du développement :

1. Utilisez l'URL de test comme serveur pour vos demandes :

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.

7.8 Passage à la phase de production avec 3-D Secure 2.0

Après avoir terminé la phase de test de votre intégration 3D Secure 2.0, 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.

7.9 Codes TransStatus de la solution 3-D Secure 2.0

Valeur	Description	Commentaires
Y	Authentifié	Le titulaire de carte a été entièrement authentifié.

Valeur	Description	Commentaires
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.

7.10 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.

7.10.1 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.

7.10.2 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.

7.10.3 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

8 Tarification multidevise (TMD)

- 8.1 À propos de la tarification multidevise (TMD)
- 8.2 Méthode de traitement des transactions avec la TMD
- 8.3 Taux d'obtention de la TMD
- 8.4 Achat utilisant la TMD
- 8.5 Achat utilisant la TMD avec 3-D Secure
- 8.6 Achat utilisant la TMD avec 3-D Secure et la chambre forte
- 8.7 Préautorisation utilisant la TMD
- 8.8 Préautorisation utilisant la TMD avec 3-D Secure
- 8.9 Préautorisation utilisant la TMD avec 3-D Secure et la chambre forte
- 8.10 Conclusion de préautorisation utilisant la TMD
- 8.11 Correction d'achat utilisant la TMD
- 8.12 Remboursement utilisant la TMD
- 8.13 Remboursement indépendant utilisant la TMD
- 8.14 Achat utilisant la TMD avec la chambre forte
- 8.15 Préautorisation utilisant la TMD avec la chambre forte
- 8.16 Remboursement indépendant utilisant la TMD avec la chambre forte
- 8.17 Codes de devise de la TMD
- 8.18 Codes d'erreur de la TMD

8.1 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.

8.2 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.

8.3 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 getRate = 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
----------	-----------------	-------------

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 sous en 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 or comment out this line 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();
}
}
}

```

8.4 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

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>mcpPurchase.setMCPVersion("MCP_VERSION_NUM");</code>
Montant du titulaire de carte	<p><i>Chaîne</i></p> <p>12 caractères numériques</p> <p>La plus petite unité discrète de devise étrangère</p>	<code>mcpPurchase.setCardholderAmount("CARDHOLDER_AMOUNT");</code>
Code de devise du titulaire de carte	<p><i>Chaîne</i></p> <p>3 caractères numériques</p>	<code>mcpPurchase.setCardholderCurrencyCode("CARDHOLDER_CURRENCY_CODE");</code>

Champs de demande liés aux transactions d'achat 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: #e0f2fd; padding: 5px;"> <p>REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\</p> </div>	<code>mcpPurchase.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;"> <p>REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\</p> </div>	<code>mcpPurchase.setDynamicDescriptor(dynamic_descriptor);</code>

Variable	Type et limites	Méthode Set
Indicateur de portefeuille électronique	<i>Chaîne</i> 3 caractères alphanumériques	<code>mcpPurchase.setWalletIndicator(wallet_indicator);</code>
Renseignements d'identification au dossier cof	<i>Objet</i> S. O.	<code>mcpPurchase.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.	<code>mcpPurchase.setAvsInfo(avsCheck);</code>
Renseignements du NVC	<i>Objet</i> S. O.	<code>mcpPurchase.setCvdInfo(cvdCheck);</code>

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.	<code>mcpPurchase.setMCPRateToken("MCP_RATE_TOKEN");</code>

Exemple d'achat utilisant la TMD

```
package Canada;
import JavaAPI.*;
public class TestCanadaMCPPurchase
{
    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 or comment out this line 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();
}
}
}

```

8.5 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);

Variable	Type et limites	Méthode Set
Numéro de carte de crédit	<i>Chaîne</i> Maximum 20 caractères alphanumériques	<code>mcpCavvPurchase.setPan(pan);</code>
Date d'expiration	<i>Chaîne</i> 4 caractères alphanumériques AAMM	<code>mcpCavvPurchase.setExpDate(expiry_date);</code>
Code de vérification d'authentification du titulaire de carte	<i>Chaîne</i> 50 caractères alphanumériques	<code>mcpCavvPurchase.setCavv(cavv);</code>
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	<code>mcpCavvPurchase.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>mcpCavvPurchase.setMCPVersion("MCP_VERSION_NUM");</code>
Montant du titulaire de carte	<i>Chaîne</i> 12 caractères numériques La plus petite unité discrète de devise étrangère	<code>mcpCavvPurchase.setCardholderAmount("CARDHOLDER_AMOUNT");</code>
Code de devise du titulaire de carte	<i>Chaîne</i> 3 caractères numériques	<code>mcpCavvPurchase.setCardholderCurrencyCode("CARDHOLDER_CURRENCY_CODE");</code>

3-D Secure 2.0 – Champs particuliers (obligatoires)

Variable	Type et limites	Méthode Set
Version de 3DS REMARQUE : Obligatoire pour les transactions qui utilisent la version 2.0 de 3-D Secure	<i>Chaîne</i> 1 caractère numérique	<code>mcpCavvPurchase.setThreeDSVersion("ThreeDSVersion");</code>
ID de transaction du serveur 3DS REMARQUE : Obligatoire pour les transactions utilisant la version 2.0+ de 3-D Secure, 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 caractères numériques	<code>mcpCavvPurchase.setThreeDSServerTransId("ThreeDSServerTransId");</code>

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	<i>Chaîne</i> 50 caractères alphanumériques REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\`	<code>mcpCavvPurchase.setCustId(cust_id);</code>
Descripteur dynamique	<i>Chaîne</i> 20 caractères alphanumériques Total de 22 caractères incluant votre nom de commerçant et un	<code>mcpCavvPurchase.setDynamicDescriptor(dynamic_descriptor);</code>

Variable	Type et limites	Méthode Set
	séparateur REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\	
Indicateur de portefeuille électronique	<i>Chaîne</i> 3 caractères alphanumériques	<code>mcpCavvPurchase.setWalletIndicator(wallet_indicator);</code>
Renseignements d'identification au dossier cof	<i>Objet</i> S. O.	<code>mcpCavvPurchase.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.	<code>mcpCavvPurchase.setAvsInfo(avsCheck);</code>
Renseignements du NVC	<i>Objet</i> S. O.	<code>mcpCavvPurchase.setCvdInfo(cvdCheck);</code>

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>	<code>mcpCavvPurchase.setMCPRateToken("MCP_RATE_TOKEN");</code>

Variable	Type et limites	Méthode Set
	S. O.	

3-D Secure 2.0 – Champs particuliers (facultatifs)

Variable	Type et limites	Méthode Set
ID de transaction DS	<i>Chaîne</i> 36 caractères alphanumériques	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 = "BwABApFSYyd412eQQFJjAAAAAAA=";
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.setCustomerId(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.setDataType("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"); //Mandatory for 3DS Version 2.0+
mcpCavvPurchase.setThreeDSServerTransId("e11d4985-8d25-40ed-99d6-c3803fe5e68f"); //Mandatory for 3DS Version 2.0+ - 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 or comment out this line 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();
}
}
}
}

```

8.6 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	<i>Chaîne</i>	mcpResCavvPurchaseCC.setCavv(cavv

Variable	Type et limites	Méthode Set
titulaire de carte	50 caractères alphanumériques) ;
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 caractères numériques La plus petite unité discrète de devise étrangère	mcpResCavvPurchaseCC.setCardholderAmount("CARDHOLDER_AMOUNT") ;
Code de devise du titulaire de carte	<i>Chaîne</i> 3 caractères numériques	mcpResCavvPurchaseCC.setCardholderCurrencyCode("CARDHOLDER_CURRENCY_CODE") ;

3-D Secure 2.0 – Champs particuliers (obligatoires)

Variable	Type et limites	Méthode Set
Version de 3DS	<i>Chaîne</i>	mcpResCavvPurchaseCC.setThreeDSVersion("ThreeDSVersion") ;
REMARQUE : Obligatoire pour les transactions qui utilisent la version 2.0 de 3-D Secure	1 caractère numérique	
ID de transaction du serveur 3DS	<i>Chaîne</i>	mcpResCavvPurchaseCC.setThreeDSserverTransId("ThreeDSServerTr

Variable	Type et limites	Méthode Set
REMARQUE : Obligatoire pour les transactions utilisant la version 2.0+ de 3-D Secure, 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 caractères numériques	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;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\`</div>	mcpResCavvPurchaseCC.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: #e0f2fd; padding: 5px;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\`</div>	mcpResCavvPurchaseCC.setDynamicDescriptor (dynamic_descriptor) ;
Indicateur de portefeuille électronique	<p><i>Chaîne</i></p> <p>3 caractères alphanumériques</p>	mcpResCavvPurchaseCC.setWalletIndicator (wallet_indicator) ;

Variable	Type et limites	Méthode Set
Renseignements d'identification au dossier cof	<i>Objet</i> S. O.	mcpResCavvPurchaseCC.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.	mcpResCavvPurchaseCC.setAvsInfo (avsCheck) ;
Renseignements du NVC	<i>Objet</i> S. O.	mcpResCavvPurchaseCC.setCvdInfo (cvdCheck) ;

3-D Secure 2.0 – Champs particuliers (facultatifs)

Variable	Type et limites	Méthode Set
ID de transaction DS	<i>Chaîne</i> 36 caractères alphanumériques	mcpResCavvPurchaseCC.setDsTransId ("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.setCustId(cust_id);
mcpResCavvPurchaseCC.setAmount(amount);
mcpResCavvPurchaseCC.setCavv(cavv);
mcpResCavvPurchaseCC.setCryptType(crypt_type);
mcpResCavvPurchaseCC.setExpDate(exp_date);
mcpResCavvPurchaseCC.setThreeDSVersion("2"); //Mandatory for 3DS Version 2.0+
mcpResCavvPurchaseCC.setThreeDSServerTransId("e11d4985-8d25-40ed-99d6-c3803fe5e68f");
//Mandatory for 3DS Version 2.0+ - 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 or comment out this line 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.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.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();
}
}
}

```

8.7 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	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p> <p>a-Z A-Z 0-9 _ - : @ espaces</p>	<code>mcpPreauth.setOrderId(order_id);</code>
Numéro de carte de crédit	<p><i>Chaîne</i></p> <p>Maximum 20 caractères alphanumériques</p>	<code>mcpPreauth.setPan(pan);</code>
Date d'expiration	<p><i>Chaîne</i></p> <p>4 caractères alphanumériques</p> <p>AAMM</p>	<code>mcpPreauth.setExpDate(expiry_date);</code>
Indicateur de commerce électronique	<p><i>Chaîne</i></p> <p>1 caractère alphanumérique</p>	<code>mcpPreauth.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	<p><i>Chaîne</i></p> <p>Valeur numérique</p> <p>La version actuelle est 1.0</p>	<code>mcpPreauth.setMCPVersion("MCP_VERSION_NUM");</code>
Montant du titulaire de carte	<p><i>Chaîne</i></p> <p>12 caractères numériques</p> <p>La plus petite unité discrète de devise étrangère</p>	<code>mcpPreauth.setCardholderAmount("CARDHOLDER_AMOUNT");</code>

Variable	Type et limites	Méthode Set
Code de devise du titulaire de carte	<p><i>Chaîne</i></p> <p>3 caractères numériques</p>	<code>mcpPreauth.setCardholderCurrencyCode ("CARDHOLDER_CURRENCY_CODE");</code>

Champs de demande pour les transactions de préautorisation 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="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <code><>\$%=?^{}[]\</code> </div>	<code>mcpPreauth.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> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <code><>\$%=?^{}[]\</code> </div>	<code>mcpPreauth.setDynamicDescriptor(dynamic_descriptor);</code>
Indicateur de portefeuille électronique	<p><i>Chaîne</i></p> <p>3 caractères alphanumériques</p>	<code>mcpPreauth.setWalletIndicator(wallet_indicator);</code>
Autorisation finale	<p><i>Chaîne</i></p> <p>true/false</p>	<code>mcpPreauth.setFinalAuth("true");</code>
REMARQUE : Applicable uniquement aux transactions par carte Mastercard		

Variable	Type et limites	Méthode Set
Renseignements d'identification au dossier cof	<i>Objet</i> S. O.	mcpPreauth.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.	mcpPreauth.setAvsInfo(avsCheck);
Renseignements du NVC	<i>Objet</i> S. O.	mcpPreauth.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.	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 mcpPrauth = new MCPPreAuth();
mcpPrauth.setOrderId(order_id);
mcpPrauth.setAmount(amount);
mcpPrauth.setPan(pan);
mcpPrauth.setExpdate(expdate);
mcpPrauth.setCryptType(crypt);
//mcpPrauth.setWalletIndicator(""); //Refer documentation for possible values
//mcpPrauth.setFinalAuth("true");
//optional - Credential on File details
CofInfo cof = new CofInfo();
cof.setPaymentIndicator("U");
cof.setPaymentInformation("2");
cof.setIssuerId("139X3130ASCXAS9");

//mcpPrauth.setCofInfo(cof);

//MCP Fields
mcpPrauth.setMCPVersion("1.0");
mcpPrauth.setCardholderAmount("500");
mcpPrauth.setCardholderCurrencyCode("840");
mcpPrauth.setMCPRateToken("P1538681661706745");

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(mcpPrauth);
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();
}
}
}

```

8.8 Préautorisation utilisant la TMD avec 3-D Secure

MCP Définition de l'objet de transaction MCP Pre-Authorization with 3-D Secure

```
MCPCavvPreAuth mcpCavvPreauth = new MCPCavvPreAuth();
```

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> 4 caractères alphanumériques	mcpCavvPreauth.setExpDate(expiry_date);

Variable	Type et limites	Méthode Set
	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("MC_P_VERSION_NUM");</code>
Montant du titulaire de carte	<i>Chaîne</i> 12 caractères numériques 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 caractères numériques	<code>mcpCavvPreauth.setCardholderCurrencyCode("CARDHOLDER_CURRENCY_CODE");</code>

3-D Secure 2.0 – Champs particuliers (obligatoires)

Variable	Type et limites	Méthode Set
Version de 3DS	<i>Chaîne</i>	<code>mcpCavvPreauth.setThreeDSVersion("ThreeDSVersion");</code>
REMARQUE : Obligatoire pour les transactions qui utilisent la	1 caractère numérique	

Variable	Type et limites	Méthode Set
version 2.0 de 3-D Secure		
ID de transaction du serveur 3DS REMARQUE : Obligatoire pour les transactions utilisant la version 2.0+ de 3-D Secure, 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 caractères numériques	mcpCavvPreauth.setThreeDSServerTransId("ThreeDSServerTransId");

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 REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <> \$ % = ? ^ { } [] \	mcpCavvPreauth.setCustomerId(cust_id);
Descripteur dynamique	<i>Chaîne</i> 20 caractères alphanumériques Total de 22 caractères incluant votre nom de commerçant et un séparateur	mcpCavvPreauth.setDynamicDescriptor(dynamic_descriptor);

Variable	Type et limites	Méthode Set
	<p>REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <> \$ % = ? ^ { } [] \</p>	
Indicateur de portefeuille électronique	<i>Chaîne</i> 3 caractères alphanumériques	<code>mcpCavvPreauth.setWalletIndicator(wallet_indicator);</code>
Renseignements d'identification au dossier cof	<i>Objet</i> S. O.	<code>mcpCavvPreauth.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.	<code>mcpCavvPreauth.setAvsInfo(avsCheck);</code>
Renseignements du NVC	<i>Objet</i> S. O.	<code>mcpCavvPreauth.setCvdInfo(cvdCheck);</code>

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.	<code>mcpCavvPreauth.setMCPRateToken("MCP_RATE_TOKEN");</code>

3-D Secure 2.0 – Champs particuliers (facultatifs)

Variable	Type et limites	Méthode Set
ID de transaction DS	<i>Chaîne</i> 36 caractères alphanumériques	<code>mcpCavvPreauth.setDsTransId("DsTransId");</code>

Exemple de préautorisation utilisant la TMD avec 3-D Secure

```

package Canada;
import JavaAPI.*;
public class TestCanadaMCPGavvPurchase
{
  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 = "BwABApFSYyd4l2eQQFJjAAAAAAA=";
    String dynamic_descriptor = "123456";
    String processing_country_code = "CA";
    String crypt_type = "5";
    boolean status_check = false;
    MCPGavvPurchase mcpGavvPurchase = new MCPGavvPurchase();
    mcpGavvPurchase.setOrderId(order_id);
    mcpGavvPurchase.setCustomerId(cust_id);
    mcpGavvPurchase.setAmount(amount);
    mcpGavvPurchase.setPan(pan);
    mcpGavvPurchase.setExpdate(expdate);
    mcpGavvPurchase.setCavv(cavv);
    mcpGavvPurchase.setCryptType(crypt_type); //Mandatory for AMEX only
    mcpGavvPurchase.setDynamicDescriptor(dynamic_descriptor);
    //mcpGavvPurchase.setWalletIndicator("APP"); //set only for wallet transactions. e.g APPLE PAY
    //mcpGavvPurchase.setNetwork("Interac"); //set only for Interac e-commerce
    //mcpGavvPurchase.setDataType("3DSecure"); //set only for Interac e-commerce
    //mcpGavvPurchase.setCmid("8nAK8712sGaAkls56"); //set only for usage with Offlinx - Unique max 50 alphanumeric characters transaction id generated by merchant

    mcpGavvPurchase.setThreeDSVersion("2"); //Mandatory for 3DS Version 2.0+
    mcpGavvPurchase.setThreeDSServerTransId("e11d4985-8d25-40ed-99d6-c3803fe5e68f"); //Mandatory for 3DS Version 2.0+ - obtained from MpiGavvLookup or MpiThreeDSAuthentication
    //mcpGavvPurchase.setDsTransId("12345"); //Optional - to be used only if you are using 3rd party 3ds 2.0 service

    //MCP Fields
    mcpGavvPurchase.setMCPVersion("1.0");
    mcpGavvPurchase.setCardholderAmount("500");
    mcpGavvPurchase.setCardholderCurrencyCode("840");
    mcpGavvPurchase.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 or comment out this line 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();
}
}
}

```

8.9 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	mcpResCavvPreauthCC.setCavv(cavv);

Variable	Type et limites	Méthode Set
	alphanumériques	
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	<code>mcpResCavvPreauthCC.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>mcpResCavvPreauthCC.setMCPVersion("MCP_VERSION_NUM");</code>
Montant du titulaire de carte	<i>Chaîne</i> 12 caractères numériques La plus petite unité discrète de devise étrangère	<code>mcpResCavvPreauthCC.setCardholderAmount("CARDHOLDER_AMOUNT");</code>
Code de devise du titulaire de carte	<i>Chaîne</i> 3 caractères numériques	<code>mcpResCavvPreauthCC.setCardholderCurrencyCode("CARDHOLDER_CURRENCY_CODE");</code>

3-D Secure 2.0 – Champs particuliers (obligatoires)

Variable	Type et limites	Méthode Set
Version de 3DS	<i>Chaîne</i> 1 caractère numérique	<code>mcpResCavvPreauthCC.setThreeDSVersion("ThreeDSVersion");</code>
REMARQUE : Obligatoire pour les transactions qui utilisent la version 2.0 de 3-D Secure		
ID de transaction du serveur 3DS	<i>Chaîne</i> 36 caractères numériques	<code>mcpResCavvPreauthCC.setThreeDSServerTransId("ThreeDSServerTransId");</code>

Variable	Type et limites	Méthode Set
REMARQUE : Obligatoire pour les transactions utilisant la version 2.0+ de 3-D Secure, 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.		

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: #e0f2fd; padding: 5px;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\ </div>	mcpResCavvPreauthCC.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: #e0f2fd; padding: 5px;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\ </div>	mcpResCavvPreauthCC.updateDynamicDescriptor(dynamic_descriptor);
Indicateur de portefeuille électronique	<p><i>Chaîne</i></p> <p>3 caractères alphanumériques</p>	mcpResCavvPreauthCC.setWalletIndicator(wallet_indicator);

Variable	Type et limites	Méthode Set
Renseignements d'identification au dossier cof	<i>Objet</i> S. O.	mcpResCavvPreauthCC.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.	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 2.0 – 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 = "AAABBJg0VhI0VniQEjRWAAAAAAA";
        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.setCustomerId(cust_id);
        mcpResCavvPreauthCC.setAmount(amount);
        mcpResCavvPreauthCC.setCavv(cavv);
        mcpResCavvPreauthCC.setExpDate(expdate);
        mcpResCavvPreauthCC.setCryptType(crypt_type);

        mcpResCavvPreauthCC.setThreeDSVersion("2"); //Mandatory for 3DS Version 2.0+
        mcpResCavvPreauthCC.setThreeDSServerTransId("e11d4985-8d25-40ed-99d6-c3803fe5e68f");
        //Mandatory for 3DS Version 2.0+ - 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 or comment out this line 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();
}
}
}

```

8.10 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	Chaîne	mpgReq.setstoreId(store_id);

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

Champs de demande pour les transactions de conclusion de préautorisation utilisant la TMD (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>	<pre>mcpCompletion.setOrderId(order_id);</pre>
Numéro de transaction	<p><i>Chaîne</i></p> <p>255 caractères (alphanumériques, trait d'union ou trait de soulignement)</p> <p>Longueur variable</p>	<pre>mcpCompletion.setTxnNumber(txn_number);</pre>
Indicateur de commerce électronique	<p><i>Chaîne</i></p> <p>1 caractère alphanumérique</p>	<pre>mcpCompletion.setCryptType(crypt); ;</pre>

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>	<pre>mcpCompletion.setMCPVersion("MC_P_VERSION_NUM"); ;</pre>

Variable	Type et limites	Méthode Set
Montant du titulaire de carte	<p><i>Chaîne</i></p> <p>12 caractères numériques</p> <p>La plus petite unité discrète de devise étrangère</p>	<code>mcpCompletion.setCardholderAmount("CARDHOLDER_AMOUNT");</code>
Code de devise du titulaire de carte	<p><i>Chaîne</i></p> <p>3 caractères numériques</p>	<code>mcpCompletion.setCardholderCurrencyCode ("CARDHOLDER_CURRENCY_CODE");</code>

Champs de demande pour les transactions de conclusion de préautorisation 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: #e0f2e0; padding: 5px;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : < > \$ % = ? ^ { } [] \ </div>	<code>mcpCompletion.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> <div style="background-color: #e0f2e0; padding: 5px;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : < > \$ % = ? ^ { } [] \ </div>	<code>mcpCompletion.setDynamicDescriptor(dynamic_descriptor);</code>

Champs de demande liés aux transactions d'achat utilisant la TMD (facultatifs)

Variable	Type et limites	Méthode Set

Variable	Type et limites	Méthode Set
Jeton du taux de la TMD	<i>Chaîne</i> S. O.	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 or comment out this line 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();
}
}
}

```

8.11 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	<code>mcpPurchasecorrection.setOrderId(order_id);</code>
Numéro de transaction	<i>Chaîne</i> 255 caractères (alphanumériques, trait d'union ou trait de soulignement) Longueur variable	<code>mcpPurchasecorrection.setTxnNumber(txn_number);</code>
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	<code>mcpPurchasecorrection.setCryptType(crypt);</code>

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	<code>mcpPurchasecorrection.setDynamicDescriptor(dynamic_descriptor);</code> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : < > \$ % = ? ^ { } [] \ </div>
ID de client	<i>Chaîne</i> 50 caractères	<code>mcpPurchasecorrection.setCustId(cust_id);</code>

Variable	Type et limites	Description
	alphanumériques <div style="border: 1px solid black; padding: 5px; background-color: #e0f2fd;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\ </div>	

Exemple de transaction de correction d'achat utilisant la TMD

```

package Canada;
import JavaAPI.*;
public class TestCanadaMCPurchaseCorrection
{
  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 or comment out this line 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();
    }
  }
}

```

}

8.12 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	mcpRefund.setOrderId(order_id);

Variable	Type et limites	Méthode Set
	_ - : . @ espaces	
Numéro de transaction	<p><i>Chaîne</i></p> <p>255 caractères (alphanumériques, trait d'union ou trait de soulignement)</p> <p>Longueur variable</p>	mcpRefund.setTxnNumber(txn_number);
Indicateur de commerce électronique	<p><i>Chaîne</i></p> <p>1 caractère alphanumérique</p>	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	<p><i>Chaîne</i></p> <p>Valeur numérique</p> <p>La version actuelle est 1.0</p>	mcpRefund.setMCPVersion("MCP_VERSION_NUM");
Montant du titulaire de carte	<p><i>Chaîne</i></p> <p>12 caractères numériques</p> <p>La plus petite unité discrète de devise étrangère</p>	mcpRefund.setCardholderAmount("CARDHOLDER_AMOUNT");
Code de devise du titulaire de carte	<p><i>Chaîne</i></p> <p>3 caractères numériques</p>	mcpRefund.setCardholderCurrencyCode("CARDHOLDER_CURRENCY_CODE");

Champs de demande liés aux transactions de remboursement indépendant utilisant la TMD (facultatifs)

Variable	Type et limites	Description

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; margin-top: 10px;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\ </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; margin-top: 10px;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\ </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.setCustId(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 or comment out this line 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();
}
}
}
}

```

8.13 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);

Variable	Type et limites	Méthode Set
Numéro de carte de crédit	<p><i>Chaîne</i></p> <p>Maximum 20 caractères alphanumériques</p>	<code>mcplndrefund.setPan(pan);</code>
Date d'expiration	<p><i>Chaîne</i></p> <p>4 caractères alphanumériques</p> <p>AAMM</p>	<code>mcplndrefund.setExpDate(expiry_date);</code>
Indicateur de commerce électronique	<p><i>Chaîne</i></p> <p>1 caractère alphanumérique</p>	<code>mcplndrefund.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	<p><i>Chaîne</i></p> <p>Valeur numérique</p> <p>La version actuelle est 1.0</p>	<code>mcplndrefund.setMCPVersion("MCP_VERSION_NUM");</code>
Montant du titulaire de carte	<p><i>Chaîne</i></p> <p>12 caractères numériques</p> <p>La plus petite unité discrète de devise étrangère</p>	<code>mcplndrefund.setCardholderAmount("CARDHOLDER_AMOUNT");</code>
Code de devise du titulaire de carte	<p><i>Chaîne</i></p> <p>3 caractères numériques</p>	<code>mcplndrefund.setCardholderCurrencyCode("CARDHOLDER_CURRENCY_CODE");</code>

Champs de demande liés aux transactions de remboursement indépendant utilisant la TMD (facultatifs)

Variable	Type et limites	Méthode Set
----------	-----------------	-------------

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;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\ </div>	mcplndrefund.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;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\ </div>	mcplndrefund.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>	mcplndrefund.setMCPRateToken("MC_P_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;
MCPIIndependentRefund mcpIndrefund = new MCPIIndependentRefund();
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 or comment out this line 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();
}
}
}

```

8.14 Achat utilisant la TMD avec la chambre forte

Cette transaction utilise la clé de données 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	mcpResPurchaseCC.setOrderId(order_id);

Variable	Type et limites	Méthode Set
	alphanumériques a-z A-Z 0-9 _ - : . @ espaces	
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	<i>Chaîne</i> Valeur numérique La version actuelle est 1.0	mcpResPurchaseCC.setMCPVersion("MCP_VERSION_NUM") ;
Montant du titulaire de carte	<i>Chaîne</i> 12 caractères numériques La plus petite unité discrète de devise étrangère	mcpResPurchaseCC.setCardholderAmount("CARDHOLDER_AMOUNT") ;
Code de devise du titulaire de carte	<i>Chaîne</i> 3 caractères numériques	mcpResPurchaseCC.setCardholderCurrencyCode("CARDHOLDER_CURRENCY_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	<i>Chaîne</i> 50 caractères alphanumériques <div style="background-color: #e0f2e0; padding: 5px;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\ </div>	mcpResPurchaseCC.setCustId(cust_id) ;

Variable	Type et limites	Méthode Set
Renseignements d'identification au dossier cof	<i>Objet</i> S. O.	mcpResPurchaseCC.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.	mcpResPurchaseCC.setAvsInfo(avheck);
Renseignements du NVC	<i>Objet</i> S. O.	mcpResPurchaseCC.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.	mcpResPurchaseCC.setMCPRateToken("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.setCustId(cust_id);
mcpResPurchaseCC.setCryptType(crypt_type);
//resPurchaseCC.setDynamicDescriptor(descriptor);
//resPurchaseCC.setExpDate(expdate); //Temp Tokens only

//MCP Fields
mcpResPurchaseCC.setMCPVersion("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 or comment out this line 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.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();
    }
}
```

8.15 Préautorisation utilisant la TMD avec la chambre forte

Cette transaction utilise la clé de données 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>	mcpResPreauthCC.setData(data_key

Variable	Type et limites	Méthode Set
	25 caractères alphanumériques) ;
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	<i>Chaîne</i> Valeur numérique La version actuelle est 1.0	mcpResPreauthCC.setMCPVersion("MCP_VERSION_NUM") ;
Montant du titulaire de carte	<i>Chaîne</i> 12 caractères numériques La plus petite unité discrète de devise étrangère	mcpResPreauthCC.setCardholderAmount("CARDHOLDER_AMOUNT") ;
Code de devise du titulaire de carte	<i>Chaîne</i> 3 caractères numériques	mcpResPreauthCC.setCardholderCurrencyCode("CARDHOLDER_CURRENCY_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	mcpResPreauthCC.setCustomerId(cust_id) ;

Variable	Type et limites	Méthode Set
	alphanumériques <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <> \$ % = ? ^ { } [] \ </div>	
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; margin-top: 10px;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <> \$ % = ? ^ { } [] \ </div>	<code>mcpResPreauthCC.setDynamicDescriptor(dynamic_descriptor);</code>
Pour les transactions de préautorisation REMARQUE : 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.		
Autorisation finale	<i>Chaîne</i> true/false	<code>mcpResPreauthCC.setFinalAuth("true");</code>
Renseignements d'identification au dossier cof	<i>Objet</i> S. O.	<code>mcpResPreauthCC.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.		

Variable	Type et limites	Méthode Set
Renseignements du SVA	<i>Objet</i> S. O.	mcpResPreauthCC.setAvsInfo(avscHECK);
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;
        MCPResPreauthCC mcpResPreauthCC = new MCPResPreauthCC();
        mcpResPreauthCC.setDataKey(data_key);
        mcpResPreauthCC.setOrderId(order_id);
        mcpResPreauthCC.setCustId(cust_id);
        mcpResPreauthCC.setAmount(amount);
        mcpResPreauthCC.setCryptType(crypt_type);
        mcpResPreauthCC.setDynamicDescriptor(dynamic_descriptor);
        //mcpResPreauthCC.setExpDate(expdate); //Temp Tokens only
        //mcpResPreauthCC.setFinalAuth("true");

        //MCP Fields
        mcpResPreauthCC.setMCPVersion("1.0");
        mcpResPreauthCC.setCardholderAmount("500");
        mcpResPreauthCC.setCardholderCurrencyCode("840");
        mcpResPreauthCC.setMCPRateToken("P1538681661706745");

        //Mandatory - Credential on File details
        CofInfo cof = new CofInfo();
        cof.setPaymentIndicator("U");
    }
}

```

```

cof.setPaymentInformation("2");
cof.setIssuerId("139X3130ASCXAS9");

mcpResPreauthCC.setCofInfo(cof);
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(mcpResPreauthCC);
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();
}
}
}

```

8.16 Remboursement indépendant utilisant la TMD avec la chambre forte

Cette transaction utilise la clé de données 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(key);

Variable	Type et limites	Méthode Set
ID de commande	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques a-z A-Z 0-9 _ - : . @ espaces</p>	<code>mcpResIndRefundCC.setOrderId(order_id);</code>
Indicateur de commerce électronique	<p><i>Chaîne</i></p> <p>1 caractère alphanumérique</p>	<code>mcpIndrefund.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	<p><i>Chaîne</i></p> <p>Valeur numérique</p> <p>La version actuelle est 1.0</p>	<code>mcpResIndRefundCC.setMCPVersion("MCP_VERSION_NUM");</code>
Montant du titulaire de carte	<p><i>Chaîne</i></p> <p>12 caractères numériques</p> <p>La plus petite unité discrète de devise étrangère</p>	<code>mcpResIndRefundCC.setCardholderAmount("CARDHOLDER_AMOUNT");</code>
Code de devise du titulaire de carte	<p><i>Chaîne</i></p> <p>3 caractères numériques</p>	<code>mcpResIndRefundCC.setCardholderCurrencyCode("CARDHOLDER_CURRENCY_CODE");</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	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p> <p>REMARQUE : Certains caractères</p>	<code>mcpResIndRefundCC.setCustId(cust_id);</code>

Variable	Type et limites	Méthode Set
	spéciaux ne sont pas autorisés : < > \$ % = ? ^ { } [] \	

Champs de demande liés aux transactions d'achat utilisant la TMD (facultatifs)

Variable	Type et limites	Méthode Set
Jeton du taux de la TMD	Chaîne 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.setMCVersion("1.0");
        mcpResIndRefundCC.setCardholderAmount("500");
        mcpResIndRefundCC.setCardholderCurrencyCode("840");
        mcpResIndRefundCC.setMCPRateToken("R1538679861330690");
        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(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.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.getResExppdate());
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();
}
}
}

```

8.17 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)

Code de devise (ISO)	Devise et acronyme
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)
156	Yuan (CNY)
170	Peso colombien (COP)

Code de devise (ISO)	Devise et acronyme
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)
340	Lempira hondurien (HNL)
344	Dollar de Hong Kong (HKD)

Code de devise (ISO)	Devise et acronyme
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)
454	Kwacha malawien (MWK)
458	Ringgit de Malaisie (MYR)

Code de devise (ISO)	Devise et acronyme
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)
598	Kina (PGK)
600	Guarani (PYG)

Code de devise (ISO)	Devise et acronyme
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)
764	Baht (THB)
780	Dollar de la Trinité et de Tobago (TTD)

Code de devise (ISO)	Devise et acronyme
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 belorusse (BYN)
934	Manat turkmène (TMT)
941	Dinar serbe (RSD)
943	Metical (MZM)
944	Manat azerbaïdjanais (AZN)

Code de devise (ISO)	Devise et acronyme
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)
986	Real brésilien (BRL)

8.18 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

9 Versements Visa

- 9.1 À propos de la solution Versements Visa
- 9.2 Types de transaction Versements Visa
- 9.3 Envoi de transactions avec la solution Versements Visa
- 9.4 Recherche de plan de versements
- 9.5 Recherche de plan de versements dans la chambre forte
- 9.6 Objet Installment Info

9.1 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.

9.2 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

9.3 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.

9.4 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) + signe décimal (.) + 2 chiffres (sous) après le point décimal</p> <p>EXAMPLE : 1234567.89</p>	installmentLookup.setAmount(amount);
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 or comment out this line 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();
}
}
}
}

```

9.5 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>	mpgReq.setApiToken(api_token);

Variable	Type et limites
	S. O.

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	<code>resInstallmentLookup.setOrderId(order_id);</code>
Montant	<i>Chaîne</i> 10 caractères décimaux Jusqu'à 7 chiffres (dollars) + signe décimal (.) + 2 chiffres (sous) après le point décimal EXEMPLE : 1234567.89	<code>resInstallmentLookup.setAmount(amount);</code>
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>
REMARQUE : Envoyez seulement ce champ si un jeton temporaire est utilisé. Sinon, n'envoyez pas ce champ.		

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)
  {
    ...
  }
}

```



```

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();
}
}
}

```

9.6 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	installmentLookup.setPlanId("VALU E");

Variable	Type et limites	Méthode Set
Référence du plan de versements	<p><i>Chaîne</i></p> <p>10 caractères alphanumériques</p> <p>Longueur fixe</p>	<code>installmentLookup.setPlanIdRef("VALUE");</code>
Version des modalités	<p><i>Chaîne</i></p> <p>10 caractères alphanumériques</p> <p>Longueur variable (de 1 à 10 caractères)</p>	<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

10. Outils de protection contre la fraude en ligne

- 10.1 Service de vérification d'adresse
- 10.2 Numéro de vérification de carte (NVC)
- 10.3 Outil de gestion des risques transactionnels

10.1 Service de vérification d'adresse

- 10.1.1 À propos du service de vérification d'adresse (SVA)
- 10.1.2 Objet AVS Info
- 1. Objet AVS Information
- 10.1.3 Codes de réponse du SVA
- 10.1.4 Exemple d'une transaction utilisant le SVA

10.1.1 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.

10.1.2 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>REMARQUE : 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>REMARQUE : 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("Plyton 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

10.1.3 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

Code	Visa	Mastercard/Discover	American Express/JCB
A	L'adresse municipale correspond, mais pas le code postal ou ZIP. Les droits de l'acquéreur ne sont pas implicites.	L'adresse municipale correspond, mais pas le code postal.	L'adresse de facturation correspond, mais pas le code postal.
B	L'adresse municipale correspond, mais le code postal ou ZIP n'a pas été vérifié en raison de formats incompatibles. (L'acquéreur de transactions a envoyé l'adresse municipale et le code postal ou ZIP.)	S. O.	S. O.
C	Les adresses municipales n'ont pas été vérifiées en raison de formats incompatibles. (L'acquéreur de transactions a envoyé l'adresse municipale et le code postal ou ZIP.)	S. O.	S. O.
D	L'adresse municipale et le code postal ou ZIP correspondent.	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	<i>S'applique seulement au Royaume-Uni</i> : L'adresse municipale et le code postal ou ZIP correspondent.	S. O.	Le nom du client est incorrect. L'adresse de facturation correspond.
G	Les renseignements liés à	S. O.	S. O.

Code	Visa	Mastercard/Discover	American Express/JCB
	<p>l'adresse n'ont pas été vérifiés dans le cadre d'une transaction internationale.</p> <p>Tous les éléments suivants peuvent être vrais :</p> <ul style="list-style-type: none"> l'émetteur ne participe pas au programme de SVA; les données du SVA étaient présentes dans la demande, mais l'émetteur n'a pas renvoyé de résultat; Visa a effectué le SVA au nom de l'émetteur et aucune adresse n'était enregistrée au dossier du client. 		
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	L'adresse municipale et le code postal ou ZIP correspondent.	S. O.	Le nom du client, l'adresse de facturation et le code postal ou ZIP correspondent.
N	Aucune correspondance. L'acquéreur a envoyé : <ul style="list-style-type: none"> uniquement le code postal ou ZIP; uniquement l'adresse municipale; le code postal ou ZIP et l'adresse municipale. Ce code est également	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.

Code	Visa	Mastercard/Discover	American Express/JCB
	utilisé lorsque l'acquéreur de transactions demande l'utilisation du SVA, mais n'envoie aucune donnée y étant liée.		
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	Nouvel essai : le système n'est pas disponible ou le temps du système est écoulé. L'émetteur utilise habituellement le SVA, mais l'émetteur n'était pas disponible. REMARQUE : Visa utilise le code R lorsque les émetteurs ne sont pas disponibles. Les émetteurs doivent s'abstenir d'utiliser ce code.	Nouvel essai : le système n'a pas été en mesure de traiter la demande.	Le système n'est pas disponible. Essayez de nouveau.
S	S. O.	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.

Code	Visa	Mastercard/Discover	American Express/JCB
U	L'adresse n'a pas été vérifiée dans le cadre d'une transaction nationale pour l'une ou plusieurs des raisons suivantes : l'émetteur ne participe pas au programme de SVA; les données du SVA étaient présentes dans la demande, mais l'émetteur n'a pas renvoyé de résultat; Visa a effectué le SVA au nom de l'émetteur et aucune adresse n'était enregistrée au dossier du client.	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 pas. Si présent, remplacez ce code par le code Z de Visa. <i>Ce code est disponible pour les émetteurs des États-Unis seulement.</i>	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	S. O.	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	L'adresse municipale et le code postal ou ZIP correspondent.	L'adresse de facturation et le code postal ou ZIP correspondent.	L'adresse de facturation et le code postal ou ZIP correspondent.
Z	Le code postal ou ZIP correspond. L'adresse municipale ne correspond	Pour les adresses des États-Unis, le code ZIP de cinq chiffres correspond,	Le code postal ou ZIP correspond, mais pas

Code	Visa	Mastercard/Discover	American Express/JCB
	pas ou l'adresse municipale n'est pas incluse dans la demande.	mais pas l'adresse	l'adresse de facturation.

10.1.4 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("1212");
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);
```

10.2 Numéro de vérification de carte (NVC)

- 10.2.1 À propos du numéro de vérification de carte (NVC)
- 10.2.3 Objet CVD Information
- 10.2.4 Codes de résultat liés au NVC
- 10.2.5 Exemple d'une transaction d'achat avec l'objet CVD Info

10.2.1 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 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 du NVC.

10.2.2 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.

10.2.3 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 CVD Information

```
<!-- start CVD -->

<!ELEMENT cvd_info (cvd_indicator, cvd_value)>
<!ELEMENT cvd_indicator (#PCDATA)>
<!ELEMENT cvd_value (#PCDATA)>
```

Définition de l'objet CvdInfo

```
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 <code><cvd_indicator></code>	<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

Variable	Type et limites	Description
		pas de CVC sur sa carte.
Valeur du NVC <cvd_value>	<i>Chaîne</i> 4 caractères numériques	NVC imprimé sur la carte 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.

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 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.

10.2.4 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

10.2.5 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);
```

10.3 Outil de gestion des risques transactionnels

- 10.3.1 À propos de l'outil de gestion des risques transactionnels
- 10.3.2 Introduction aux demandes d'information (queries)
- 10.3.3 Demande d'information sur la session (Session Query)
- 10.3.4 Demande d'information sur les attributs (Attribute Query)
- 10.3.6 Ajout de balises de profilage à votre site Web
- 10.3.6 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**.

10.3.1 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).

10.3.2 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 379)
- Les demandes d'information sur les attributs (page 386)

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.

10.3.3 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 caractères numériques 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	sq.setAccountAddressStreet1 ("3300 Bloor St W");
	Première partie de la rue de l'adresse de facturation		
Rue 2 de l'adresse du compte	Chaîne	32 caractères alphanumériques	sq.setAccountAddressStreet2 ("4th Flr West Tower");
	Deuxième partie de la rue de l'adresse de facturation		
Ville de l'adresse du compte	Chaîne	50 caractères alphanumériques	sq.setAccountAddressCity ("Toronto");
	Ville de l'adresse de facturation		
Province ou état de l'adresse du compte	Chaîne	64 caractères alphanumériques	sq.setAccountAddressState ("Ontario");
	Province ou état de l'adresse de facturation		
Pays de l'adresse du compte	Chaîne	2 caractères alphanumériques	sq.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	sq.setAccountAddressZip ("M8X2X2");
	Code postal ou ZIP de l'adresse de facturation		
Rue 1 de l'adresse d'expédition du compte	Chaîne	32 caractères alphanumériques	sq.setShippingAddressStreet1 ("3300 Bloor St W");
	Première partie de la rue de l'adresse d'expédition		
Rue 2 de l'adresse d'expédition du compte	Chaîne	32 caractères alphanumériques	sq.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	sq.setShippingAddressCity ("Toronto");
	Ville de l'adresse d'expédition		
Province ou état de	Chaîne	64 caractères alphanumériques	sq.setShippingAddressState ("Ontario");

Valeur	Type	Limites	Méthode Set
	Description		
l'adresse d'expédition	Province ou état de l'adresse d'expédition		
Pays de l'adresse d'expédition	Chaîne	2 caractères alphanumériques	sq.setShippingAddressCountry ("CA");
	Code ISO2 représentant le pays de l'adresse d'expédition		
Code postal ou ZIP de l'adresse d'expédition	Chaîne	8 caractères alphanumériques	sq.setAccountAddressZip ("M8X2X2");
	Code postal ou ZIP de l'adresse d'expédition		
Attribut 1-5 local	Chaîne	255 caractères alphanumériques	sq.setLocalAttrib1 ("a");
	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	sq.setTransactionAmount ("1.00");
	Montant numérique de la transaction		
Devise de la transaction	Chaîne	10 caractères numériques	sq.setTransactionCurrency ("CAN");
	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 compte	Chaîne	255 caractères alphanumériques	sq.setAccountLogin ("13195417-8CA0-46cd-960D-14C158E4DBB2");
	Nom de l'identifiant du compte		

Valeur	Type	Limites		Méthode Set
		Description		
Algorithme de hachage du mot de passe	Chaîne	40 caractères alphanumériques	sq.setPasswordHash ("489c830f10f7c601d30599a0deaf66e64d2aa50a");	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		
Nom du compte	Chaîne	255 caractères alphanumériques	sq.setAccountNumber ("3E17A905-AC8A-4c8d-A417-3DADA2A55220");	Numéro du compte
		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		
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 de l'adresse 1		
Rue de l'adresse 2	Chaîne	32 caractères alphanumériques		Première partie de la rue de l'adresse du compte Deuxième partie de la rue de l'adresse du compte
		Ville de l'adresse		
Province ou état de l'adresse	Chaîne	50 caractères alphanumériques		Ville de l'adresse du compte Province ou état de l'adresse du compte

Valeur	Type	Limites	Méthode Set
	Description		
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	Chaîne	32 caractères alphanumériques	
		Première partie de la rue de l'adresse d'expédition	
Rue 2 de l'adresse d'expédition	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 ZIP de l'adresse d'expédition	Chaîne	8 caractères alphanumériques	
		Code postal ou ZIP de l'adresse d'expédition	
Algorithme de	Chaîne	255 caractères alphanumériques	

Valeur	Type	Limites	Méthode Set
	Description		
hachage du numéro de carte de crédit	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");
    }
}

```

Exemple d'une transaction de demande d'information sur la session – CA

```

//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 or comment out this line 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();
}
}
}

```

10.3.3.1 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.

Les balises HTML doivent être placées sur des pages que le titulaire de carte consultera pendant quelques secondes au moins pour obtenir le plus vaste éventail de données possible.

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.

10.3.4 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.	aq.setServiceType(service_type); Indique les champs de sortie inclus dans la réponse session -- renvoie les attributs concernant l'adresse IP et l'appareil
ID de l'appareil	Chaîne	36 caractères alphanumériques	aq.setDeviceId("");

Valeur	Type	Limites	Méthode Set
	Description		
(device ID)	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 caractères numériques Sans espace ou tiret	aq.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.
Adresse IP	Chaîne	64 caractères alphanumériques	aq.setIPAddress ("192.168.0.1") ; Il s'agit de l'adresse IP réelle. Les résultats seront notamment true_ip_geo et true_ip_score.
Adresse IP transférée	Chaîne	64 caractères alphanumériques	aq.setIPForwarded ("192.168.1.0") ; Il s'agit de l'adresse IP du serveur mandataire (proxy). Si l'adresse IP est fournie, les résultats seront proxy_ip_geo et proxy_ip_score. Si aucune adresse IP n'est fournie, cette adresse IP sera considérée comme l'adresse IP réelle et les résultats seront notamment true_ip_geo et true_ip_score.
Rue 1 de l'adresse du compte	Chaîne	32 caractères alphanumériques	aq.setAccountAddressStreet1 ("3300 Bloor St W") ; Première partie de la rue de l'adresse de facturation
Rue 2 de l'adresse du compte	Chaîne	32 caractères alphanumériques	aq.setAccountAddressStreet2 ("4th Flr West Tower") ; Deuxième partie de la rue de l'adresse de facturation
Ville de l'adresse du compte	Chaîne	50 caractères alphanumériques	aq.setAccountAddressCity ("Toronto") ; 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

Valeur	Type	Limites	Méthode Set
	Description		
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";

```

Exemple d'une transaction de demande d'information sur les attributs

```

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-AF9DE98B84D");
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]));
    }
}
}

```

10.3.4.1 Flux d'une transaction de demande d'information sur les attributs

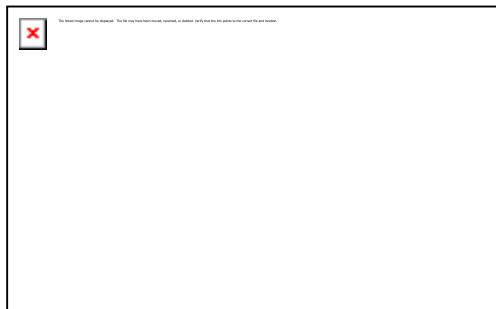


Image 4 : Flux d'une transaction de demande d'information sur les attributs

1. Le titulaire de carte se connecte sur le site Web du commerçant et effectue une transaction.
2. L'application Web du commerçant effectue une transaction de demande d'information sur les attributs incluant l'ID de session à Passerelle Moneris.
3. Passerelle Moneris transfère les données de la demande d'information sur les attributs à ThreatMetrix.
4. 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.
5. 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.

10.3.5 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

10.3.5.1 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	Chaîne	3 caractères alphanumériques	<code>receipt.getResponseCode();</code>

Valeur	Type	Limites	Méthode Get
	Définition		
réponse		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 98 = Impossible de trouver la transaction précédente	
Message	Chaîne	S. O.	receipt.getMessage(); 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 la demande	Chaîne	S. O.	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	Chaîne	S. O.	receipt.getRiskResult();

Valeur	Type	Limites	Méthode Get
	Définition		
de la demande	Voir la section 10.3.5.1 (page).		
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
Sommairie 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
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

Valeur	Définition
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 n'était pas valide.
fail_invalid_characters	Des caractères non valides ont été soumis.
fail_invalid_charset	La valeur character set n'était pas valide.
fail_invalid_currency_code	Le format de la valeur currency_code n'était pas valide.
fail_invalid_currency_format	Le format de la valeur currency_format n'était pas valide.
fail_invalid_telephone_number	Le format du numéro de téléphone fourni n'était pas 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.
fail_invalid_device_id	Le format de la valeur device_id fournie n'était pas valide.
fail_invalid_email_address	Le format de l'adresse courriel fournie n'était pas valide.
fail_invalid_fuzzy_device_id	Le format de la valeur fuzzy_device_id n'était pas

Valeur	Définition
	valide.
fail_invalid_ip_address_parameter	Le format du paramètre ip_address fourni n'était pas valide.
fail_invalid_parameter	Le format du paramètre n'était pas valide, ou la valeur dépasse les limites.
fail_invalid_sha_hash	Le format d'un paramètre précisé comme un hachage sha n'était pas valide. Le hachage sha comprenait le hachage sha1/2/3.
fail_invalid_submitter_id	Le format de l'ID du demandeur n'était pas 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 en acceptent uniquement une.
fail_verification	La limite de requête de l'API a été atteinte
success	ThreatMetrix a réussi à traiter la demande.

10.3.5.2 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. Remarque : 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.

10.3.5.3 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 L'appareil est sur la liste blanche. Cela signifie que l'appareil est signalé comme étant toujours « ok ». Remarque : Cette règle n'est pas offerte actuellement.
IPWhitelisted	WL002	Adresse IP sur la liste blanche

Nom de règle	Numéro de règle	Message de règle
	Explication de la règle	
	L'adresse IP est sur la liste blanche. Cela signifie que l'appareil est signalé comme étant toujours « ok ». Remarque : Cette règle n'est pas offerte actuellement.	
EmailWhitelisted	WL003	Adresse courriel sur la liste blanche
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.	
2IPPaymentVelocity	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.	
3DevicePerEmailDay	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.	

Nom de règle	Numéro de règle	Message de règle
	Explication de la règle	
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 courriel 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. Remarque : 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
		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

Nom de règle	Numéro de règle	Message de règle
	Explication de la règle	
	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.	
3DevicePerCreditCardWeek	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
	La carte de crédit a été émise dans une région différente de l'adresse de facturation fournie.	
CreditCardDeviceGeoMismatch	CP008	L'emplacement de la carte de crédit et de l'appareil ne correspondent pas

Nom de règle	Numéro de règle	Message de règle
	Explication de la règle	
	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
	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.	

Nom de règle	Numéro de règle	Message de règle
	Explication de la règle	
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.
	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.	

Nom de règle	Numéro de règle	Message de règle
	Explication de la règle	
Géolocalisation		
DeviceCountriesNotAllowed	GE001	<p>La géolocalisation véritable fait partie de la liste noire des pays non autorisés.</p> <p>Cet appareil se connecte depuis un emplacement géographique à haut risque.</p>
DeviceCountriesNotAllowed	GE002	<p>La géolocalisation véritable fait partie de la liste blanche négative des pays non autorisés.</p> <p>L'appareil provient d'une région qui ne figure pas sur la liste blanche des régions acceptées.</p>
DeviceProxyGeoMismatch	GE003	<p>La géolocalisation véritable est différente de la géolocalisation du serveur mandataire.</p> <p>L'emplacement géographique véritable de cet appareil est différente de l'emplacement géographique du serveur mandataire.</p>
DeviceAccountGeoMismatch	GE004	<p>L'adresse du compte est différente de la géolocalisation véritable.</p> <p>Cet appareil a présenté une adresse de facturation de compte qui ne correspond pas à la géolocalisation de l'appareil.</p>
DeviceShipGeoMismatch	GE005	<p>Incohérence entre la géolocalisation de l'appareil et l'adresse d'expédition</p> <p>L'emplacement de l'appareil et l'adresse d'expédition ne correspondent pas.</p>
DeviceShipGeoMismatch	GE006	<p>Incohérence entre la géolocalisation de l'appareil et l'adresse d'expédition</p> <p>L'emplacement de l'appareil et l'adresse d'expédition ne correspondent pas.</p>
Appareil		
SatelliteISP	DV001	<p>Fournisseur de service internet par satellite</p> <p>Cette transaction provient d'une source qui utilise un fournisseur de service Internet par satellite.</p>
MidsessionChange	DV002	Session modifiée au milieu de la session

Nom de règle	Numéro de règle	Message de règle
	Explication de la règle	
	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	Témoin (cookies) des images Flash désactivés
	Les fonctions ou identificateurs clés du navigateur ont été désactivés sur cet appareil.	
FlashCookiesDisabled	DV011	Témoin (cookies) Flash désactivés
	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.	

Nom de règle	Numéro de règle	Message de règle
	Explication de la règle	
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.

10.3.5.4 Exemples de réponses de risque

Demande de session

Exemple de réponse de risque – Demande de session
<pre><?xml version="1.0"?> <response> <receipt> <ResponseCode>001</ResponseCode> <Message>Success</Message> <Result> <session_id>abc123</session_id> <unknown_session>yes</unknown_session> <event_type>payment</event_type> <service_type>session</service_type> <policy_score>-25</policy_score> <transaction_id>riskcheck42</transaction_id> <org_id>11kue096</org_id> <request_id>91C1879B-33D4-4D72-8FCB-B60A172B3CAC</request_id> <risk_rating>medium</risk_rating> <request_result>success</request_result> <summary_risk_score>-25</summary_risk_score> <Policy>default</policy> <review_status>review</review_status> </Result> <Rule></pre>

Exemple de réponse de risque – Demande de session

```

<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>
```

10.3.6 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. 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>
```

10.4 Intégration de tous les outils de protection contre la fraude offerts

- 10.4.1 Options de mise en œuvre de l'outil de gestion des risques transactionnels (TRMT)
- 10.4.2 Liste de contrôle de mise en œuvre
- 10.4.3 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.

10.4.1 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.

10.4.2 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 390) 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 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.

10.4.3 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 \$.

11 Intégration d'Apple Pay et de Google Pay^{MC}

- 11.1 À propos de l'intégration d'Apple Pay et de Google Pay^{MC}
- 11.2 Sommaire du processus de transaction Apple Pay
- 11.3 Sommaire du processus de transaction Google Pay^{MC}
- 11.4 À propos de l'intégration de l'API d'Apple Pay et de Google Pay^{MC}
- 11.5 Achat utilisant le code de vérification d'authentification du titulaire de carte – Apple Pay et Google Pay^{MC}
- 11.6 Préautorisation utilisant le code de vérification d'authentification du titulaire de carte – Apple Pay et Google Pay^{MC}

11.1 À propos de l'intégration d'Apple Pay et de Google Pay^{MC}

Passerelle Moneris permet aux commerçants de traiter des transactions dans des applications fonctionnant sur des appareils mobiles iOS (Apple Pay) ou Android (Google Pay^{MC}), 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^{MC}).

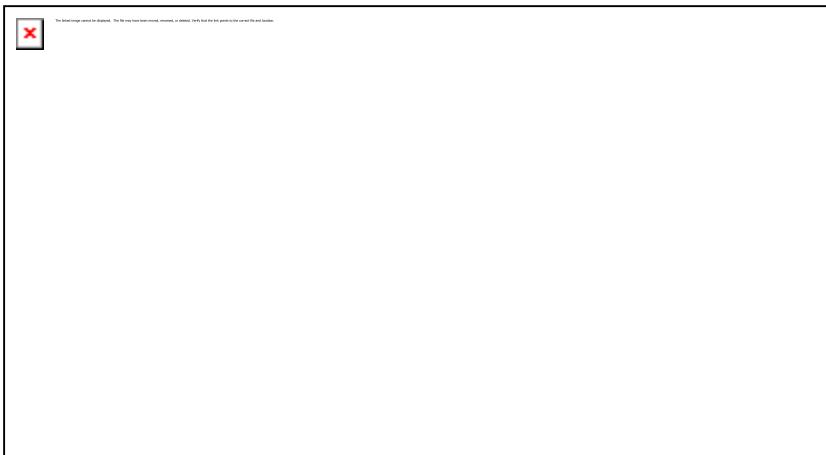
Solutions Moneris offre deux méthodes d'intégration pour le traitement des transactions Apple Pay et Google Pay^{MC}. Les commerçants peuvent choisir d'utiliser l'une des deux méthodes suivantes :

- 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>.

11.2 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™ ou Cvv Pre-Authorization – Apple Pay & Google Pay™ à la page 417transaction
 - 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

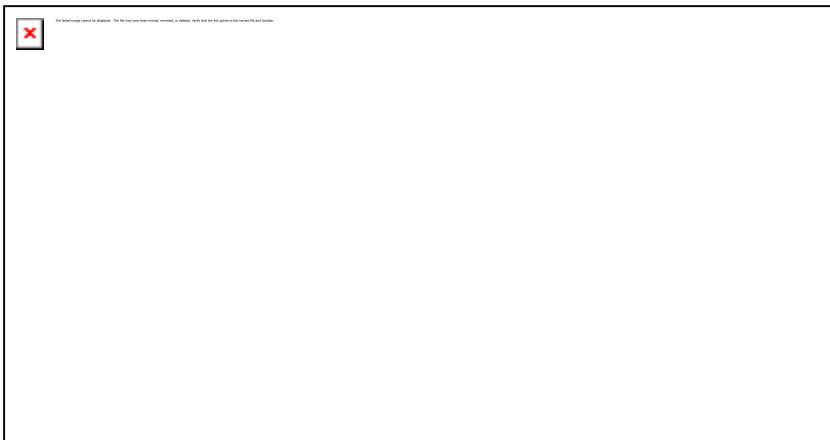
1. L'application mobile ou la page Web du commerçant demande et reçoit les données chiffrées.
2. Les données chiffrées sont envoyées du serveur du commerçant au serveur de Passerelle Moneris où elles sont déchiffrées et traitées.

Ce guide ne traite que de la méthode API; pour plus de renseignements sur la méthode d'intégration utilisant la troussse SDK, consultez le portail des développeurs de Moneris à l'adresse <https://developer.moneris.com>.

11.3 Sommaire du processus de transaction Google Pay^{MC}

Pour les méthodes d'intégration API et SDK, l'application ou le site Web du commerçant utilise le cadre Google Pay^{MC} pour demander et recevoir les détails de paiement chiffrées de Google. 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.

REMARQUE : Dans la méthode API où le serveur du commerçant est responsable de déchiffrer les données, le commerçant doit signer une entente avec Google directement. Google peut alors vous fournir les clés pour déchiffrer les données.



Étapes du processus de paiement Google Pay^{MC}

API

1. L'application 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™ ou Cvv Pre-Authorization – Apple Pay & Google Pay™ à la page 417transaction
 - a. Veuillez vous assurer que l'indicateur de portefeuille est correctement rempli avec la bonne valeur (GPP pour Google Pay^{MC} In-App ou GPW pour Google Pay^{MC} 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.

11.4 À propos de l'intégration de l'API d'Apple Pay et de Google Pay^{MC}

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^{MC} 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^{MC}.

11.4.1 Types de transaction utilisées pour Apple Pay et Google Pay^{MC}

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^{MC} :

- 11.5 Achat utilisant le code de vérification d'authentification du titulaire de carte – Apple Pay et Google Pay^{MC}
- 11.6 Préautorisation utilisant le code de vérification d'authentification du titulaire de carte – Apple Pay et Google Pay^{MC}

Commerce électronique par carte INTERAC^{MD} **REMARQUE :** Cette fonction peut uniquement être utilisée pour les transactions d'achat utilisant le code de vérification d'authentification.

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

11.5 Achat utilisant le code de vérification d'authentification du titulaire de carte – Apple Pay et Google Pay^{MC}

Une transaction d'achat utilisant le code de vérification d'authentification du titulaire de carte pour Apple Pay et Google Pay^{MC} 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, 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^{MC}. 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^{MC} 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^{MC}

```
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 Pay^{MC} (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) + signe décimal (.) + 2 chiffres (sous) après le point décimal	cavvPurchase.setAmount(amount);
	EXAMPLE : 1234567.89	
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);
REMARQUE : Pour les transactions d'achat et de préautorisation Apple Pay et Google Pay ^{MC} 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 REMARQUE : Pour les transactions d'achat et de préautorisation Apple Pay et Google Pay ^{MC} 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 Pay^{MC} (facultatifs)

Variable	Type et limites	Méthode Set
----------	-----------------	-------------

Variable	Type et limites	Méthode Set
Vérification d'état	Valeur booléenne true/false	mpgReq.setStatusCheck(status_check);
ID de client	Chaîne 50 caractères alphanumériques REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\	cavvPurchase.setCustId(cust_id);
Descripteur dynamique	Chaîne 20 caractères alphanumériques Total de 22 caractères incluant votre nom de commerçant et un séparateur REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\	cavvPurchase.setDynamicDescriptor(dynamic_descriptor);
ID de correspondance de carte REMARQUE : Applicables à Offlinx ^{MC} seulement, chaque transaction doit avoir une valeur unique	Chaîne 50 caractères alphanumériques	cavvPurchase.setCmId(transaction_id);
Renseignements du client	Objet S. O.	cavvPurchase.setCustInfo(customer);

Exemple d'achat utilisant le code de vérification d'authentification du titulaire de carte pour Apple Pay et Google Pay^{MC}

```
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 = "BwABApFSYyd4l2eQQFJjAAAAAAA=";
String dynamic_descriptor = "123456";
String processing_country_code = "CA";
String crypt_type = "5";
boolean status_check = false;
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.setData("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"); //Mandatory for 3DS Version 2.0+
cavvPurchase.setThreeDSServerTransId("e11d4985-8d25-40ed-99d6-c3803fe5e68f"); //Mandatory for
3DS Version 2.0+ - obtained from MpiCavvLookup or MpiThreeDSAuthentication
//cavvPurchase.setDsTransId("12345");//Optional - to be used only if you are using 3rd party
3ds 2.0 service

//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 or comment out this line 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());
}
}

```

```

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

```

11.6 Préautorisation utilisant le code de vérification d'authentification du titulaire de carte – Apple Pay et Google Pay^{MC}

Une transaction de préautorisation utilisant le code de vérification d'authentification du titulaire de carte pour Apple Pay et Google Pay^{MC} 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^{MC}. 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^{MC} 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^{MD}^{MD} **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 PayTM

```
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^{MC}

```
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 Pay^{MC} (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>	cavvPreauth.setOrderId(order_id);
Montant	<p><i>Chaîne</i></p> <p>10 caractères décimaux</p> <p>Jusqu'à 7 chiffres (dollars) + signe décimal (.) + 2 chiffres (sous) après le point décimal</p> <p>EXEMPLE : 1234567.89</p>	cavvPreauth.setAmount(amount);
Numéro de carte de crédit	<p><i>Chaîne</i></p> <p>Maximum 20 caractères alphanumériques</p>	cavvPreauth.setPan(pan);
Code de vérification d'authentification du titulaire de carte	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p>	cavvPreauth.setCavv(cavv);
REMARQUE : Pour les transactions d'achat et de préautorisation Apple Pay et Google Pay ^{MC} 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>	cavvPreauth.setExpDate(expiry_date);

Variable	Type et limites	Méthode Set
Indicateur de commerce électronique REMARQUE : Pour les transactions d'achat et de préautorisation Apple Pay et Google Pay ^{MC} 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^{MC} (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 REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <> \$ % = ? ^ { } [] \	cavvPreauth.setCustId(cust_id);
Descripteur dynamique	<i>Chaîne</i>	cavvPreauth.setDynamicDescriptor(dynamic_descriptor);
Pour les transactions de préautorisation REMARQUE : La valeur du champ Dynamic descriptor est uniquement transférée à une conclusion de	20 caractères alphanumériques Total de 22 caractères incluant votre nom de	

Variable	Type et limites	Méthode Set
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 REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\	
ID de correspondance de carte	<i>Chaîne</i> 50 caractères alphanumériques	cavvPreauth.setCmId(transaction_id);
REMARQUE : Applicables à Offlinx ^{MC} seulement, chaque transaction doit avoir une valeur unique		
Réseau	<i>Chaîne</i> Caractère alphabétique	cavvPurchase.setNetwork(network);
REMARQUE : Cette variable de demande est obligatoire pour les transactions de commerce électronique INTERAC ^{MD} effectuées via Apple Pay ou Google Pay ^{MC} uniquement, et ne doit pas être utilisée pour les transactions par carte de crédit.		
Type de données	<i>Chaîne</i> 3 caractères alphanumériques	cavvPurchase.setDataTipe(data_type);
REMARQUE : Cette variable de demande est obligatoire pour les transactions de commerce électronique INTERAC ^{MD} effectuées via Apple Pay ou Google Pay ^{MC} uniquement, et ne doit pas être utilisée pour les transactions par carte de crédit.		

Exemple de préautorisation utilisant le code de vérification d'authentification du titulaire de carte pour Apple Pay et Google Pay^{MC}

```
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 = "AAABBBJg0VhI0VniQEjRWAAAAAAA=";
String dynamic_descriptor = "123456";
String processing_country_code = "CA";
String crypt_type = "5";
boolean status_check = false;
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"); //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("12345");//Optional - to be used only if you are using 3rd party
3ds 2.0 service

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

cavvPreauth.setCofInfo(cof);

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());
}
catch (Exception e)
}

```

```
{  
e.printStackTrace();  
}  
}  
}
```

12 Offlinx^{MC}

- Qu'est-ce qu'un pixel invisible?
- Offlinx^{MC} et les transactions API

12.1 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^{MC} 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

12.2 Offlinx^{MC} et les transactions API

La fonction Offlinx^{MC} 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^{MC}. La variable Card Match ID doit être une valeur unique pour chaque transaction.

Pour plus de renseignementssur la solution Offlinx^{MC}, consultez le guide de configuration des pixels invisibles de Offlinx^{MC}, 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_preatuh)
- Achat utilisant le code de vérification d'authentification du titulaire de carte – Apple Pay et Google Pay^{MC}
- Préautorisation utilisant le code de vérification d'authentification du titulaire de carte – Apple Pay et Google Pay^{MC}

13 Frais de commodité

- 13.1 À propos des frais de commodité
- 13.3 Achat avec frais de commodité
- 13.4 Achat avec renseignements sur le client et frais commodité
- 13.5 Achat avec frais de commodité utilisant 3-D Secure

13.1 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.

13.2 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

13.3 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) + signe décimal (.) + 2 chiffres (sous) après le point décimal	purchase.setAmount(amount);

Variable	Type et limites	Méthode Set
	EXEMPLE : 1234567.89	
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>
REMARQUE : 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	<code>purchase.setCustId(cust_id);</code>

Variable	Type et limites	Méthode Set
	<p>spéciaux ne sont pas autorisés :</p> <p>< > \$ % = ? ^ { } [] \</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>REMARQUE : Certains caractères spéciaux ne sont pas autorisés :</p> <p>< > \$ % = ? ^ { } [] \</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);
    }
}

```

Exemple d'achat avec frais de commodité

```

purchase.setPan(pan);
purchase.setExpdate(expdate);
purchase.setCryptType(crypt);
purchase.setConvFeeInfo(convFeeInfo);
HttpsPostRequest mpgReq = new HttpsPostRequest();
mpgReq.setProcCountryCode(procCountryCode);
mpgReq.setTestMode(true); //false or comment out this line 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();
}
}
}
}

```

13.4 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.	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) + signe décimal (.) + 2 chiffres (sous) après le point décimal EXEMPLE : 1234567.89	purchase.setAmount(amount);
Numéro de carte de crédit	<i>Chaîne</i> Maximum 20 caractères alphanumériques	purchase.setPan(pan);
Date d'expiration	<i>Chaîne</i> 4 caractères alphanumériques AAMM	purchase.setExpDate(expiry_date);
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère	purchase.setCryptType(crypt);

Variable	Type et limites	Méthode Set
	alphanumérique	
Montant des frais de commodité	<i>Chaîne</i> 9 caractères décimaux	<code>purchase.setConvFee(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	<i>Chaîne</i> 50 caractères alphanumériques <div style="border: 1px solid black; padding: 5px;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\ </div>	<code>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="border: 1px solid black; padding: 5px;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\ </div>	<code>purchase.setDynamicDescriptor(dynamic_descriptor);</code>
Renseignements du client	<i>Objet</i> S. O.	<code>purchase.setCustInfo(customer);</code>
Renseignements du SVA	<i>Objet</i> S. O.	<code>purchase.setAvsInfo(avscCheck);</code>
Renseignements du NVC	<i>Objet</i>	<code>purchase.setCvdInfo(cvdCheck);</code>

Variable	Type et limites	Méthode Set
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.	

Exemple d'achat avec renseignements sur le client et frais commodité

```

package Canada;

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 */
        /*
    }
}

```

Exemple d'achat avec renseignements sur le client et frais commodité

```

***** **** 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");

```

Exemple d'achat avec renseignements sur le client et frais commodité

```

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.setExpdate(expdate);
purchase.setCryptType(crypt);
purchase.setCustInfo(customer);
purchase.setConvFeeInfo(convFeeInfo);
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(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();
}
}
}
}

```

13.5 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) + signe décimal (.) + 2 chiffres (sous) après le point décimal EXEMPLE : 1234567.89	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	cavvPurchase.setExpDate(expiry_date);

Variable	Type et limites	Méthode Set
	AAMM	
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	cavvPurchase.setCryptType(crypt);
Code de vérification d'authentification du titulaire de carte	<i>Chaîne</i> 50 caractères alphanumériques	cavvPurchase.setCavv(cavv);
Montant des frais de commodité	<i>Chaîne</i> 9 caractères décimaux	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	<i>Valeur booléenne</i> true/false	mpgReq.setStatusCheck(status_check);
ID de client	<i>Chaîne</i> 50 caractères alphanumériques <div style="background-color: #e0f2e0; padding: 5px;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\' </div>	cavvPurchase.setCustId(cust_id);
Descripteur dynamique	<i>Chaîne</i> 20 caractères alphanumériques Total de 22 caractères incluant votre nom de commerçant et un séparateur	cavvPurchase.setDynamicDescriptor(dynamic_descriptor);

Variable	Type et limites	Méthode Set
	<p>REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\`</p>	
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);
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.		

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";

```

Exemple d'achat avec frais de commodité utilisant 3-D Secure

```

String expdate = "1901"; //YYMM
String cavv = "AAABBJg0Vhi0VniQEjRWAAAAAA=";
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 or comment out this line 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();
}
}
}
}

```

14 Facturation périodique

- 14.1 À propos de la facturation périodique
- 14.2 Achat avec la facturation périodique
- 1 Achat avec la facturation périodique
- 14.3 Mise à jour de la facturation périodique
- 1 Mise à jour de la facturation périodique
- 14.4 Codes et champs de réponse liés à la facturation périodique
- 14.5 Renseignements d'identification au dossier et facturation périodique

14.1 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.

14.2 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	<p><i>Chaîne</i></p> <p>Valeur numérique</p> <p>De 1 à 999</p>	Il s'agit du nombre d'occurrences de la transaction.
Période	<p><i>Chaîne</i></p> <p>Valeur numérique</p> <p>De 1 à 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 AAAAMMJJ</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	<p><i>Chaîne</i></p> <p>true/false</p>	<p>Réglez cette variable à true si un montant est porté immédiatement à la carte. Autrement, réglez la variable à false.</p> <p>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.</p> <p>REMARQUE : Le montant à facturer immédiatement peut différer des montants subséquents.</p>
Montant récurrent	<p><i>Chaîne</i></p> <p>10 caractères décimaux, minimum de 3 chiffres</p> <p>Jusqu'à 7 chiffres (dollars) + signe décimal (.) + 2 chiffres (sous) après le point décimal</p>	<p>Il s'agit du montant en dollars de la transaction périodique.</p> <p>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.</p>

Variable	Type et limites	Description
	EXEMPLE : 1234567.89	
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 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");
  }
}

```

Exemple de transaction d'achat avec la facturation périodique

```

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 or comment out this line 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();
}
}
}

```

14.3 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
ID 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 pan	<i>Chaîne</i> 20 caractères alphanumériques	recurUpdate.setPan(pan);
Date d'expiration expiry_date	<i>Chaîne</i> AAMM	recurUpdate.setExppdate(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
Montant récurrent recur_amount	<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.
Ajout du nombre d'occurrences add_num	<i>Chaîne</i> Valeur numérique, entre 1 et 999	recurUpdate.setAddNumRecurs(add_num);	Cette variable ajoute 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	recurUpdate.setTotalNumRecurs(total_num);	Cette variable remplace le nombre actuel de transactions périodiques restantes.
Suspendre la facturation périodique hold	<i>Chaîne</i> true/false	recurUpdate.setHold(hold);	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	recurUpdate.setTerminate(terminate);	Cette variable met fin à une transaction périodique. REMARQUE : Lorsque vous mettez fin à une transaction périodique, celle-ci ne peut pas être réactivée. Une nouvelle transaction d'achat

Variable	Type et limites	Méthode Set	Description
			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.setCustomerId(cust_id);
recurUpdate.setRecurAmount(recur_amount);
recurUpdate.setPan(pan);
recurUpdate.setExpiryDate(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 or comment out this line 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();
}
}

```

Exemple de transaction de mise à jour d'une facturation périodique
}

14.4 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		
Objet de transaction avec les champs de réponse liés à la facturation périodique				
Code de réponse	Chaîne	3 caractères numériques	receipt.getResponseCode () ;	
	Consultez le tableau 20 pour obtenir une description des codes de réponse possibles.			
Réussite de la répétition	Chaîne	À confirmer	receipt.getRecurSuccess () ;	
	Indique si la transaction a été correctement enregistrée			
Champs de réponse liés à l'objet Recur update				
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			
Prochaine date de l'occurrence	Chaîne	Format aaaa-mm-jj	receipt.getNextRecurDate () ;	
	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			

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
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

14.5 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).

15. Renseignements du client

- 15.1 Utiliser l'objet Customer Information
- 15.2 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 ´.
- Les données incluses dans les champs Billing et Shipping Address ne seront pas utilisées à des fins de vérification d'adresse.

15.1 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);
```

15.1.1 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);

15.1.2 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	50	"company_name"

Valeur	Limite	Clé de la table de hachage
Adresse	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"

15.1.2.1 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 15.1.2.1 Méthodes Set pour les renseignements d'expédition et de facturation.

Pour obtenir un exemple de code, consultez la section 15.2 Exemple d'une transaction incluant l'objet Customer Information.

15.1.2.2 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 15.1.2 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 15.2 Exemple d'une transaction incluant l'objet Customer Information.

15.1.3 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 caractères numériques	"quantity"
Code de produit de l'article	20 caractères alphanumériques	"product_code"
Montant final de l'article	9 caractères décimaux composés d'au moins 3 chiffres et de 2 cents De 0,01 à 999999,99	"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.

15.1.3.1 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 15.1.3 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 15.2 Exemple d'une transaction incluant l'objet Customer Information.

15.1.3.2 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 :

- Instanciez un objet CustInfo.
- 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.)
- 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 15.1.3 Objet Customer Info – Information sur les articles.
- Appelez la méthode setItem de l'objet CustInfo afin de transférer les renseignements de la table de hachage à l'objet CustInfo.
- 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 15.2 Exemple d'une transaction incluant l'objet Customer Information.

15.2 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();
```

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);
```

Exemple d'une transaction d'achat incluant les renseignements du client – Table de hachage

```

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();

```

16 Vérification d'état

- 16.1 À propos de la vérification d'état
- 16.2 Utiliser les champs de réponse liés à la vérification d'état
- 16.3 Exemple d'achat avec la vérification d'état

16.1 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.

16.2 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.

16.3 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 or comment out this line 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();
        }
    }
}
```

17 Visa Checkout

- 17.1 À propos de Visa Checkout
- 17.2 Intégration de Visa Checkout Lightbox
- 17.3 Flux de transactions pour Visa Checkout
- 17.4 Achat avec Visa Checkout
- 17.5 Préautorisation avec Visa Checkout – VdotMePreAuth
- 17.6 Conclusion avec Visa Checkout
- 17.7 Correction d'achat avec Visa Checkout
- 17.8 Remboursement avec Visa Checkout
- 17.9 Renseignements avec Visa Checkout

17.1 À propos de Visa Checkout

Visa Checkout est un service de portefeuille électronique offert aux clients utilisant des cartes de crédit. La fonction Visa Checkout peut être intégrée à Passerelle Moneris par l'entremise de l'API.

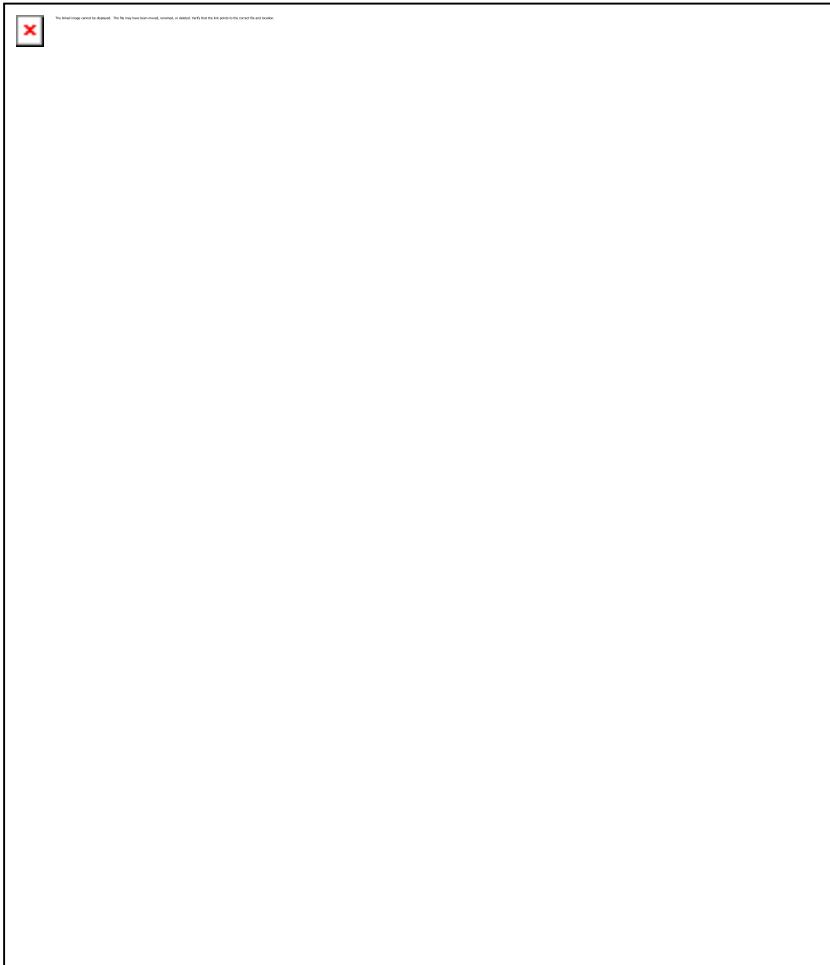
17.2 Intégration de Visa Checkout Lightbox

1. À l'aide de la clé API obtenue lors de la configuration de votre magasin Visa Checkout, créez l'intégration Visa Checkout Lightbox avec JavaScript en suivant la documentation Visa, qui est disponible sur le portail Visa Developer (en anglais seulement) :
 1. **Renseignements généraux sur Visa Checkout (téléchargement de la trousse SDK de JavaScript)**
 2. https://developer.visa.com/products/visa_checkout
 3. **Démarrer avec Visa Checkout**
 4. https://developer.visa.com/products/visa_checkout/guides#getting_started
 5. **Ajouter Visa Checkout à votre page Web**
 6. https://developer.visa.com/products/visa_checkout/guides#adding_to_page
 7. **Envoyer la demande de paiement du consommateur**
 8. https://developer.visa.com/products/visa_checkout/guides#submitting_csr
2. Si vous obtenez un événement de succès de paiement à partir du JavaScript Visa Lightbox, vous devrez analyser et obtenir le `callid` de la réponse JSON. Les renseignements supplémentaires sont obtenus à l'aide de l'objet `VdotMeInfo` `VdotMeInfo`.

Une fois que vous avez obtenu le `callid` de Visa Lightbox, vous pouvez effectuer l'appel de transaction Visa Checkout `VdotMe` approprié à Moneris pour traiter votre transaction et obtenir vos fonds.

REMARQUE : Lors des tests de Visa Checkout dans notre environnement de tests d'assurance qualité, veuillez utiliser la clé API que vous avez générée dans la configuration de Visa Checkout pour l'appel `V.Init` dans votre JavaScript.

17.3 Flux de transactions avec Visa Checkout



17.4 Achat avec Visa Checkout

Une transaction d'achat avec Visa Checkout appelle Moneris pour obtenir des fonds liés au `callId` de Visa Checkout et les préparer à être déposés dans le compte du commerçant. Cette transaction met également à jour l'historique des transactions Visa Checkout du client.

Définition de l'objet de transaction Visa Checkout Purchase

```
VdotMePurchase vmePurchase = new VdotMePurchase();
```

Objet HttpsPostRequest pour les transactions d'achat avec Visa Checkout

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

Champs de demande liés aux transactions d'achat avec Visa Checkout (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>	<code>vmepurchase.setOrderId(order_id);</code>
ID d'appel	<p><i>Chaîne</i></p> <p>20 caractères numériques</p>	<code>vmepurchase.setCallId(call_id);</code>
Montant	<p><i>Chaîne</i></p> <p>10 caractères décimaux</p> <p>Jusqu'à 7 chiffres (dollars) + signe décimal (.) + 2 chiffres (sous) après le point décimal</p>	<code>vmepurchase.setAmount(amount);</code>
	EXAMPLE : 1234567.89	
Indicateur de commerce électronique	<p><i>Chaîne</i></p> <p>1 caractère alphanumérique</p>	<code>vmepurchase.setCryptType(crypt);</code>

Champs de demande liés aux transactions d'achat avec Visa Checkout (facultatifs)

Variable	Type et limites	Méthode Set
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; margin-top: 10px;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\`</div>	<code>vmepurchase.setDynamicDescriptor(dynamic_descriptor);</code>

Variable	Type et limites	Méthode Set
Vérification d'état	Valeur booléenne true/false	mpgReq.setStatusCheck(status_check);

Exemple d'achat avec Visa Checkout

```

package Canada;
import JavaAPI.*;
public class TestCanadaVdotMePurchase
{
    public static void main(String[] args)
    {
        String store_id = "store2";
        String api_token = "yesguy";
        String cust_id = "Joe Doe";
        java.util.Date createDate = new java.util.Date();
        String order_id = "Test"+createDate.getTime();
        String amount = "8.00";
        String crypt_type = "7";
        String call_id = "9104624497663077101";
        String dynamic_descriptor = "inv123";
        String processing_country_code = "CA";
        boolean status_check = false;
        VdotMePurchase vmepurchase = new VdotMePurchase();
        vmepurchase.setOrderId(order_id);
        vmepurchase.setCustId(cust_id);
        vmepurchase.setAmount(amount);
        vmepurchase.setCallId(call_id);
        vmepurchase.setCryptType(crypt_type);
        vmepurchase.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(vmepurchase);
        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("StatusCode = " + receipt.getStatusCode());
            System.out.println("StatusMessage = " + receipt.getStatusMessage());
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}

```

Exemple d'achat avec Visa Checkout
}

17.5 Préautorisation avec Visa Checkout – VdotMePreAuth

Cette transaction appelle Moneris pour vérifier les fonds liées au `callId` de Visa Checkout et réserver ces fonds pour votre compte de commerçant. Ces fonds sont bloqués pour une durée prédéterminée, en fonction de l'émetteur de carte. Pour récupérer les fonds provenant de cet appel afin qu'ils puissent être déposés dans le compte du commerçant, une transaction `VdotMeCompletion` doit être effectuée. Cette transaction met également à jour l'historique des transactions Visa Checkout du client.

La transaction `VdotMePreAuth` est presque identique à la transaction `VdotMePurchase`, à l'exception du nom du type de transaction.

Si la commande n'a pas pu être complétée pour une raison quelconque, par exemple elle a été annulée, elle a été traitée par erreur ou elle n'est pas réalisable, la transaction `VdotMePreAuth` doit être annulée dans les 72 heures.

Pour annuler une autorisation, effectuez une transaction `VdotMeCompletion` de 0,00 \$ (zéro dollars).

Définition de l'objet de transaction Visa Checkout Pre-Authorization

```
VdotMePreauth vMePreauthRequest = new VdotMePreauth();
```

Objet `HttpsPostRequest` pour les transactions de préautorisation avec Visa Checkout

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

Champs de demande pour les transactions de préautorisation avec Visa Checkout (obligatoires)

Variable	Type et limites	Méthode Set
Montant	<p><i>Chaîne</i></p> <p>10 caractères décimaux</p> <p>Jusqu'à 7 chiffres (dollars) + signe décimal (.) + 2 chiffres (sous) après le point décimal</p> <p>EXEMPLE : 1234567.89</p>	<code>vDotMeReauthRequest.setAmount(amount);</code>
ID d'appel	<p><i>Chaîne</i></p> <p>20 caractères numériques</p>	<code>vDotMeReauthRequest.setCallId(call_id);</code>

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>	vDotMeReauthRequest.setOrderId(order_id);
Indicateur de commerce électronique	<p><i>Chaîne</i></p> <p>1 caractère alphanumérique</p>	vDotMeReauthRequest.setCryptType(crypt);

Champs de demande pour les transactions de préautorisation avec Visa Checkout (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: 10px;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\</div>	vMePreauthRequest.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: #e0f2fd; padding: 10px;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\</div>	vMePreauthRequest.setDynamicDescriptor(dynamic_descriptor);

Exemple de préautorisation avec Visa Checkout

```
package Canada;
import JavaAPI.*;
public class TestCanadaVdotMePreauth
{
    public static void main(String[] args)
    {
```

Exemple de préautorisation avec Visa Checkout

```
String store_id = "store2";
String api_token = "yesguy";
String amount = "5.00";
String crypt_type = "7";
java.util.Date createDate = new java.util.Date();
String order_id = "Test"+createDate.getTime();
String call_id = "9104624497663077101";
String cust_id = "my customer id";
String processing_country_code = "CA";
boolean status_check = false;
VdotMePreauth vMePreauthRequest = new VdotMePreauth();
vMePreauthRequest.setOrderId(order_id);
vMePreauthRequest.setAmount(amount);
vMePreauthRequest.setCallId(call_id);
vMePreauthRequest.setCustId(cust_id);
vMePreauthRequest.setCryptType(crypt_type);
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(vMePreauthRequest);
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("StatusCode = " + receipt.getStatusCode());
System.out.println("StatusMessage = " + receipt.getStatusMessage());
}
catch (Exception e)
{
e.printStackTrace();
}
}
}
```

17.6 Conclusion avec Visa Checkout

Cette transaction appelle Moneris pour obtenir les fonds réservés par l'appel `VdotMePreAuth`. Cet appel de transaction récupère les fonds bloqués et les prépare à être déposés dans le compte du commerçant. Cet appel doit généralement être effectué dans les 72 heures suivant l'exécution de la transaction `VdotMePreAuth`. Cette transaction met également à jour l'historique des transactions Visa Checkout du client.

La transaction `VdotMeCompletion` est utilisée pour sécuriser les fonds verrouillés par une transaction `VdotMePreAuth`.

Vous pouvez également effectuer une transaction de ce type d'une valeur de 0,00 \$ (zéro dollar) pour annuler une transaction VdotMePreAuthh que vous n'êtes pas en mesure d'exécuter.

Définition de l'objet de transaction Visa Checkout Completion

```
VdotMeCompletion vmecompletion = new VdotMeCompletion();
```

Objet HttpsPostRequest pour les transactions de conclusion avec Visa Checkout

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

Champs de demande pour les transactions de conclusion avec Visa Checkout (obligatoires)

Variable	Type et limites	Méthode Set
ID de commande	<i>Chaîne</i> 50 caractères alphanumériques a-Z A-Z 0-9 _ - : @ espaces	vmecompletion.setOrderId(order_id);
Numéro de transaction	<i>Chaîne</i> 255 caractères (alphanumériques, trait d'union ou trait de soulignement) Longueur variable	vmecompletion.setTxnNumber(txn_number);
Montant de conclusion	<i>Chaîne</i> 10 caractères décimaux Jusqu'à 7 chiffres (dollars) + signe décimal (.) + 2 chiffres (sous) après le point décimal	vmecompletion.setCompAmount(comp_amount); EXAMPLE : 1234567.89
Indicateur de commerce électronique	<i>Chaîne</i> 1 caractère alphanumérique	vmecompletion.setCryptType(crypt);

Champs de demande pour les transactions de conclusion avec Visa Checkout (facultatifs)

Variable	Type et limites	Méthode Set
----------	-----------------	-------------

Variable	Type et limites	Méthode Set
ID de client	<p><i>Chaîne</i></p> <p>50 caractères alphanumériques</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\ </div>	vmecompletion.setCustId(cust_id);
<p>Descripteur dynamique</p> <p>Pour les transactions de préautorisation REMARQUE : 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><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; margin-top: 10px;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\ </div>	vmecompletion.setDynamicDescriptor(dynamic_descriptor);

Exemple de conclusion avec Visa Checkout

```

package Canada;
import JavaAPI.*;
public class TestCanadaVdotMeCompletion
{
  public static void main(String[] args)
  {
    String store_id = "store2";
    String api_token = "yesguy";
    String order_id = "Test1432134710264";
    String txn_number = "724379-0_10";
    String comp_amount = "1.00";
    String ship_indicator = "P";
    String crypt_type = "7";
    String cust_id = "mycustomerid";
    String dynamic_descriptor = "inv 123";
    String processing_country_code = "CA";
    boolean status_check = false;
    VdotMeCompletion vmecompletion = new VdotMeCompletion();
    vmecompletion.setOrderId(order_id);
    vmecompletion.setTxnNumber(txn_number);
    vmecompletion.setAmount(comp_amount);
    vmecompletion.setCryptType(crypt_type);
    vmecompletion.setDynamicDescriptor(dynamic_descriptor);
    vmecompletion.setCustId(cust_id);
    vmecompletion.setShipIndicator(ship_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);
  }
}

```

Exemple de conclusion avec Visa Checkout

```

mpgReq.setApiToken(api_token);
mpgReq.setTransaction(vmecompletion);
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("StatusCode = " + receipt.getStatusCode());
    System.out.println("StatusMessage = " + receipt.getStatusMessage());
}
catch (Exception e)
{
    e.printStackTrace();
}
}
}

```

17.7 Correction d'achat avec Visa Checkout

Cette transaction appelle Moneris pour annuler les transactions VdotMePurchase et VdotMeCompletion le jour même où elles ont eu lieu. Cette transaction met également à jour l'historique des transactions Visa Checkout du client.

La transaction `VdotMePurchaseCorrection` est utilisé pour annuler une transaction `VdotMeCompletion` ou `VdotMePurchase` qui a été effectuée dans le lot actuel. Aucun autre type de transaction ne peut être corrigé à l'aide de cette méthode.

Aucun montant n'est requis, car il s'agit toujours de 100 % de la transaction originale.

Définition de l'objet de transaction Visa Checkout Purchase Correction

```
VdotMePurchaseCorrection vDotMePurchaseCorrection = new VdotMePurchaseCorrection();
```

Objet `HttpsPostRequest` pour les transactions de correction d'achat avec Visa Checkout

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

Champs de demande liés aux transactions de correction d'achat avec Visa Checkout (obligatoires)

Variable	Type et limites	Méthode Set
----------	-----------------	-------------

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>	vDotMePurchaseCorrection.setOrderID(order_id);
Numéro de transaction	<p><i>Chaîne</i></p> <p>255 caractères (alphanumériques, trait d'union ou trait de soulignement)</p> <p>Longueur variable</p>	vDotMePurchaseCorrection.setTxnNumber(txn_number);

Champs de demande liés aux transactions de correction d'achat avec Visa Checkout (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;"> REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\`</div>	vDotMePurchaseCorrection.setCustId(cust_id);
Vérification d'état	<p><i>Valeur booléenne</i></p> <p><i>Chaîne</i></p>	mpgReq.setStatusCheck(status_check);

Exemple de correction d'achat avec Visa Checkout

```

package Canada;
import JavaAPI.*;
public class TestCanadaVdotMePurchaseCorrection
{
public static void main(String[] args)
{
String store_id = "store2";
String api_token = "yesguy";
String order_id = "Test1432134533159";
String txn_number = "724377-0_10";

```

Exemple de correction d'achat avec Visa Checkout

```

String crypt_type = "7";
String cust_id = "my customer id";
String processing_country_code = "CA";
boolean status_check = false;
VdotMePurchaseCorrection vDotMePurchaseCorrection = new VdotMePurchaseCorrection();
vDotMePurchaseCorrection.setOrderId(order_id);
vDotMePurchaseCorrection.setCustId(cust_id);
vDotMePurchaseCorrection.setTxnNumber(txn_number);
vDotMePurchaseCorrection.setCryptType(crypt_type);
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(vDotMePurchaseCorrection);
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("StatusCode = " + receipt.getStatusCode());
System.out.println("StatusMessage = " + receipt.getStatusMessage());
}
catch (Exception e)
{
e.printStackTrace();
}
}
}

```

17.8 Remboursement avec Visa Checkout

Cette transaction appelle Moneris pour annuler une transaction `VdotMePurchase` ou `VdotMeCompletion` et rembourser une partie ou la totalité de la transaction. Cette transaction met également à jour l'historique des transactions Visa Checkout du client.

La transaction `VdotMeRefund` créditera un montant déterminé sur la carte de crédit du titulaire de la carte et mettra à jour l'historique de ses transactions Visa Checkout. Un remboursement allant jusqu'à la valeur totale de la transaction `VdotMeCompletion` or `VdotMePurchase` originale peut être effectué.

Définition de l'objet de transaction Visa Checkout Refund

```
VdotMeRefund vDotMeRefundRequest = new VdotMeRefund();
```

Objet HttpsPostRequest pour les transactions de remboursement avec Visa Checkout

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

Champs de demande liés aux transactions de remboursement avec Visa Checkout (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>	vDotMeRefundRequest.setOrderId(order_id);
Montant	<p><i>Chaîne</i></p> <p>10 caractères décimaux</p> <p>Jusqu'à 7 chiffres (dollars) + signe decimal (.) + 2 chiffres (sous) après le point decimal</p> <p>EXAMPLE : 1234567.89</p>	vDotMeRefundRequest.setAmount(amount);
Numéro de transaction	<p><i>Chaîne</i></p> <p>255 caractères (alphanumériques, trait d'union ou trait de soulignement)</p> <p>Longueur variable</p>	vDotMeRefundRequest.setTxnNumber(txn_number);
Indicateur de commerce électronique	<p><i>Chaîne</i></p> <p>1 caractère alphanumérique</p>	vDotMeRefundRequest.setCryptType(crypt);

Champs de demande liés aux transactions de remboursement avec Visa Checkout (facultatifs)

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

Variable	Type et limites	Méthode Set
	<p>REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\</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>REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\</p>	vDotMeRefundRequest.setDynamicDescriptor(dynamic_descriptor);
Vérification d'état	<p><i>Valeur booléenne</i></p> <p>true/false</p>	mpgReq.setStatusCheck(status_check);

Exemple de remboursement avec Visa Checkout

```

package Canada;
import JavaAPI.*;
public class TestCanadaVdotMeRefund
{
public static void main(String[] args)
{
String store_id = "store2";
String api_token = "yesguy";
String order_id = "Test1432134710264";
String txn_number = "724380-1_10";
String amount = "1.00";
String crypt_type = "7";
String dynamic_descriptor = "inv 123";
String cust_id = "my customer id";
String processing_country_code = "CA";
boolean status_check = false;
VdotMeRefund vDotMeRefundRequest = new VdotMeRefund();
vDotMeRefundRequest.setOrderId(order_id);
vDotMeRefundRequest.setAmount(amount);
vDotMeRefundRequest.setCustomerId(cust_id);
vDotMeRefundRequest.setTxnNumber(txn_number);
vDotMeRefundRequest.setCryptType(crypt_type);
vDotMeRefundRequest.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(vDotMeRefundRequest);
}

```

Exemple de remboursement avec Visa Checkout

```

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("StatusCode = " + receipt.getStatusCode());
    System.out.println("StatusMessage = " + receipt.getStatusMessage());
}
catch (Exception e)
{
    e.printStackTrace();
}
}
}
}

```

17.9 Renseignements avec Visa Checkout

Cette transaction appelle Moneris pour obtenir les renseignements du titulaire de la carte, tels que le nom affiché sur la carte, le numéro partiel de la carte, la date d'expiration ainsi que les renseignements sur l'expédition et la facturation.

La transaction `VdotMeInfo` obtiendra des renseignements sur le client à partir de son portefeuille Visa Checkout. Les détails renvoyés dépendent de ce qui est stocké par le client dans Visa Checkout.

Définition de l'objet de transaction Visa Checkout Information

```
VdotMeInfo vmeinfo = new VdotMeInfo();
```

Objet `HttpsPostRequest` pour les transactions de renseignement avec Visa Checkout

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

Champs de demande liés aux transactions de renseignement avec Visa Checkout (obligatoires)

Variable	Type et limites	Méthode Set
ID d'appel	<i>Chaîne</i> 20 caractères numériques	<code>vmeinfo.setCallId(call_id);</code>

Exemple de renseignements avec Visa Checkout

```

package Canada;
import java.util.Hashtable;
import java.util.Set;
import JavaAPI.*;
public class TestCanadaVdotMeInfo
{
public static void main(String[] args)
{
String store_id = "store2";
String api_token = "yesguy";
String call_id = "8620484083629792701";
String processing_country_code = "CA";
boolean status_check = false;
VdotMeInfo vmeinfo = new VdotMeInfo();
vmeinfo.setCallId(call_id);
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(vmeinfo);
mpgReq.setStatusCheck(status_check);
mpgReq.send();

try
{
Receipt receipt = mpgReq.getReceipt();

System.out.println("dump of vmeDataHash variables:");
Hashtable<String, String>vmeDataHash = new Hashtable<String, String>();
vmeDataHash = receipt.getVmeDataHash();

Set<String> keys = vmeDataHash.keySet();
for(String key: keys){
System.out.println("Value of "+key+" is: "+vmeDataHash.get(key));
}
System.out.println("Response Code: " + receipt.getResponseCode());
System.out.println("Response Message: " + receipt.getMessage());
System.out.println("Currency Code: " + receipt.getCurrencyCode());
System.out.println("Payment Totals: " + receipt.getPaymentTotal());
System.out.println("User First Name: " + receipt.getUserFirstName());
System.out.println("User Last Name: " + receipt.getUserLastName());
System.out.println("Username: " + receipt.getUserName());
System.out.println("User Email: " + receipt.getUserEmail());
System.out.println("Encrypted User ID: " + receipt.getEncUserId());
System.out.println("Creation Time Stamp: " + receipt.getCreationTimeStamp());
System.out.println("Name on Card: " + receipt.getNameOnCard());
System.out.println("Expiration Month: " + receipt.getExpirationDateMonth());
System.out.println("Expiration Year: " + receipt.getExpirationDateYear());
System.out.println("Last 4 Digits: " + receipt.getLastFourDigits());
System.out.println("Bin Number (6 Digits): " + receipt.getBinSixDigits());
System.out.println("Card Brand: " + receipt.getCardBrand());
System.out.println("Card Type: " + receipt.getVdotMeCardType());
System.out.println("Billing Person Name: " + receipt.getPersonName());
System.out.println("Billing Address Line 1: " + receipt.getBillingAddressLine1());
System.out.println("Billing City: " + receipt.getBillingCity());
System.out.println("Billing State/Province Code: " + receipt.getBillingStateProvinceCode());
System.out.println("Billing Postal Code: " + receipt.getBillingPostalCode());
System.out.println("Billing Country Code: " + receipt.getBillingCountryCode());
System.out.println("Billing Phone: " + receipt.getBillingPhone());
System.out.println("Billing ID: " + receipt.getBillingId());
System.out.println("Billing Verification Status: " + receipt.getBillingVerificationStatus());
System.out.println("Partial Shipping Country Code: " +
receipt.getPartialShippingCountryCode());
System.out.println("Partial Shipping Postal Code: " +
receipt.getPartialShippingPostalCode());
System.out.println("Shipping Person Name: " + receipt.getShippingPersonName());
System.out.println("Shipping Address Line 1: " + receipt.getShipAddressLine1());
}

```

Exemple de renseignements avec Visa Checkout

```
System.out.println("Shipping City: " + receipt.getShippingCity());
System.out.println("Shipping State/Province Code: " +
receipt.getShippingStateProvinceCode());
System.out.println("Shipping Postal Code: " + receipt.getShippingPostalCode());
System.out.println("Shipping Country Code: " + receipt.getShippingCountryCode());
System.out.println("Shipping Phone: " + receipt.getShippingPhone());
System.out.println("Shipping Default: " + receipt.getShippingDefault());
System.out.println("Shipping ID: " + receipt.getShippingId());
System.out.println("Shipping Verification Status: " +
receipt.getShippingVerificationStatus());
System.out.println("isExpired: " + receipt.getIsExpired());
System.out.println("Base Image File Name: " + receipt.getBaseImageFileName());
System.out.println("Height: " + receipt.getHeight());
System.out.println("Width: " + receipt.getWidth());
System.out.println("Issuer Bid: " + receipt.getIssuerBid());
System.out.println("Risk Advice: " + receipt.getRiskAdvice());
System.out.println("Risk Score: " + receipt.getRiskScore());
System.out.println("AVS Response Code: " + receipt.getAvsResponseCode());
System.out.println("CVV Response Code: " + receipt.getCvvResponseCode());
System.out.println("\r\nPress the enter key to exit");
}
catch (Exception e)
{
e.printStackTrace();
}
}
```

18 Mise à l'essai d'une solution

- 18.1 À propos du centre de ressources pour commerçants
- 18.2 Se connecter à l'environnement d'assurance de la qualité du centre de ressources pour commerçants
- 18.3 Renseignements d'identification test du centre de ressources pour commerçants
- 18.4 Obtenir un ID de commerce et un jeton API uniques pour l'environnement de test
- 18.5 Traiter une transaction
- 18.6 Mise à l'essai d'une solution de paiement INTERAC^{MD} en ligne
- 18.7 Mise à l'essai d'un module d'extension pour les commerçants
- 18.8 Mise à l'essai de Visa Checkout
- 1 Cartes de test
- 18.10 Simulation du serveur de traitement

18.1 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.monteris.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.

18.2 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.monteris.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 18.3 Renseignements d'identification test du centre de ressources pour commerçants.

18.3 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.
monca01428	mcmpguy	demouser	password	Utilisez ce jeton pour tester l'outil

store_id	api_token	Username	Password	Autre information
				Masterpass de Mastercard.

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 476).

18.4 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).

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 475).

18.5 Traiter une transaction

- 1.1 Sommaire
- 1.2 Objet HttpsPostRequest
- 1.3 Objet Receipt

18.5.1 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 18.5.

La section 18.5 (page) 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 Java.

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.setExdate(exdate); 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(); HttpsPostRequest(); mpgReq.setProcCountryCode(processing_country_code); mpgReq.setTestMode(true); mpgReq.setStoreId(store_id); mpgReq.setApiToken(api_token); mpgReq.setTransaction(purchase); mpgReq.setStatusCheck(status_check); mpgReq.send();</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>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(); }</pre>	<p>Invoquer la méthode <code>send()</code> de l'objet de connexion</p> <p>Instancier l'objet <code>Receipt</code> et utiliser ses méthodes <code>get</code> pour récupérer les données de réponse souhaitées</p>

18.5.2 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 caractère alphabétique CA pour Canada, US pour États-Unis	mpgReq.setProcCountryCode(processing_country_code);
Mode de mise à l'essai	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 Identifiant unique fourni par Moneris lors de la création du compte de commerçant Voir la section 18.1 À propos du centre de ressources pour commerçants pour obtenir plus de détails sur l'environnement de mise à l'essai	mpgReq.setstoreId(store_id);
Jeton API	Chaîne	20 caractères alphanumériques 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 18.3 Renseignements d'identification test du centre de ressources pour commerçants pour obtenir des détails sur l'environnement de mise en essai.	mpgReq.setApiToken(api_token);
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	Valeu r boolé enne	true/fa lse	mpgReq.setStatusCheck(status_check);
Consultez l'annexe A Définition des champs de demande.			
REMARQUE : 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.			

18.5.3 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.

18.6 Mise à l'essai d'une solution de paiement INTERAC^{MD} en ligne

Acxsys dispose de deux sites Web où les commerçants peuvent effectuer des transactions pour tester le transfert de la garantie de fonds des transactions de paiement INTERAC^{MD} en ligne. La valeur IDEBIT_MERCHNUM de test est fournie par Moneris après l'inscription à l'environnement de test.

Après l'inscription, les deux liens suivants seront accessibles :

- Outil de test pour commerçants
- Outil de test de certification

Outil de test pour commerçants

https://merchant-test.interacidebit.ca/gateway/merchant_test_processor.do

Cette URL est utilisée pour simuler le processus de réponse de la transaction, pour valider les variables de réponse et pour intégrer correctement votre processus de paiement.

Lorsque vous testez les transactions de paiement en ligne INTERAC^{MD}, vous êtes redirigé vers l'outil de test pour commerçants de paiement en ligne INTERAC^{MD}. Un écran apparaît là où certains champs doivent être remplis.

Pour que votre réponse soit approuvée, ne modifiez aucun des champs, à l'exception de ceux qui sont énumérés ici.

IDEBIT_TRACK2

Pour former une piste track2 lors d'un test avec Passerelle Moneris, utilisez l'un de ces trois numéros :

3728024906540591206=01121122334455000

5268051119993326=011211223344550000000

453781122255=0112112233445500000000000

IDEBIT_ISSNAME

RBC

IDEBIT_ISSCONF

123456

Pour une réponse refusée, fournissez toute autre valeur pour le champ IDEBIT_TRACK2. Cliquez sur **Post to Merchant**.

Que la transaction soit approuvée ou refusée, **ne cliquez pas sur Validate Data**. Cela créera des erreurs de validation.

Outil de test de certification

https://merchant-test.interacidebit.ca/gateway/merchant_certification_processor.do

Cette adresse URL est utilisée pour effectuer les cas de test requis pour la certification frontale du commerçant en matière de paiement INTERAC^{MD} en ligne, qui sont décrits à l'annexe E (page 585) et à l'annexe F (page 589).

Pour confirmer les fonds garanti ci-dessus, il faut envoyer une transaction d'achat INTERAC^{MD} en ligne au service d'assurance-qualité de Passerelle Moneris en utilisant les renseignements du magasin test suivants :

Serveur de traitement : esqa.monteris.com

ID de commerce : store3

Jeton API : yesguy

Vous pouvez toujours vous connecter au centre de ressources pour les commerçants pour vérifier les résultats en utilisant les renseignements suivants :

URL : <https://esqa.monteris.com/mpg>

ID de commerce : store3

Notez que toutes les variables de réponse renvoyées par la passerelle IOP à l'étape 5.4 de la section 5.4 doivent être validées pour la longueur du champ, les caractères autorisés et les caractères non valides.

18.7 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	Mesure
4012001037141112	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
4242424242424242			
4012001038488884	U	NA	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	NA	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	Mesure
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.
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.
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\$.

18.8 Mise à l'essai de Visa Checkout

Afin de mettre à l'essai Visa Checkout, vous devez :

1. Créer un profil de configuration Visa Checkout dans l'environnement d'assurance qualité du centre de ressources pour commerçants à la page <https://esqa.monteris.com/mpg>. Pour en apprendre davantage à ce sujet, consultez la section Crédit d'une configuration de page de paiement Visa pour la mise en essai à la page 484.

2. Obtenir une clé API Lightbox à utiliser pour l'intégration Lightbox. Pour en apprendre davantage à ce sujet, consultez la section Intégration de Visa Checkout Lightbox à la page 457.
3. Pour obtenir les numéros de cartes de test à utiliser uniquement lors du test de Visa Checkout, consultez la section Cartes de test pour Visa Checkout à la page 485

18.8.1 Crédit d'une configuration Visa Checkout pour la mise en essai

Une fois que vous avez créé un magasin de test, vous devez activer Visa Checkout dans l'environnement d'assurance qualité.

Pour activer Visa Checkout dans l'environnement d'assurance qualité, procédez de la manière suivante :

1. Ouvrez une session dans l'environnement d'assurance qualité sur la page <https://esqa.moneris.com/mpg>.
2. Dans le menu Admin, sélectionnez Visa Checkout.
3. Remplissez les champs applicables.
4. Cliquez sur Enregistrer.

18.9 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

18.9.1 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	5454545442424242
Visa	4242424254545454
Amex	373269005095005

18.9.2 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

18.10 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.moneris.com>.

REMARQUE : Ces réponses peuvent changer sans préavis. Consultez le portail pour développeurs de Moneris (<https://developer.moneris.com>, en anglais seulement) pour accéder aux plus récents documents et fichiers à télécharger.

19 Passage à l'environnement de production

- 19.1 Activation d'un compte de magasin dans l'environnement de production
- 19.2 Configuration d'un commerce pour l'environnement de production
- 19.3 Exigences de reçu
- 1 Obténir de l'aide

19.1 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 487)

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>).

19.2 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 487).
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 : mpgReq.setTestMode(true);	Méthode Set pour l'environnement de production mpgReq.setTestMode(false);
2	Chaîne : String store_id = "store5"; Méthode Set associée : mpgReq.setstoreId(store_id);	Chaîne pour l'environnement de production : String store_id = " monXXXXXXXXX ";
3	Chaîne : String api_token = "yesguy"; Méthode Set associée : mpgReq.setApiToken(api_token);	Chaîne pour l'environnement de production : String api_token = " xxxx ";

19.2.1 Configurer un commerce de paiement en ligne d'INTERAC^{MD} pour l'environnement de production

Avant de traiter des transactions d'INTERAC^{MD} en ligne par l'entremise de votre site Web, vous devez compléter le processus d'enregistrement de la certification auprès de Moneris, comme indiqué ci-dessous. Moneris fournit la valeur de l'environnement de production IDEBIT_MERCHNUM après que vous avez complété le processus de certification avec succès.

L'URL de l'environnement de production Acxsys de la passerelle de paiement d'INTERAC^{MD} en ligne est : https://gateway.interaconline.com/merchant_processor.do.

Pour accéder à l'URL de l'environnement de production de Passerelle Moneris, utilisez ce qui suit :

ID de commerce : Fourni par Moneris

Jeton API : Généré durant le processus d'activation du commerce

Code du pays de traitement : CAN :

L'URL de l'environnement de **production** du centre de ressources pour commerçants est :

<https://www3.moneris.com/mpg/>.

19.2.1.1 Compléter le processus d'enregistrement de la certification des commerçants

Pour compléter le processus d'enregistrement de la certification, envoyez les renseignements ci-dessous par télécopie ou par courriel à notre centre d'aide à l'intégration :

- Logo du commerçant à afficher sur la passerelle de paiement INTERAC^{MD} en ligne
- En français et en anglais
- 120 x 30 pixels
- Format PNG (seul format pris en charge)

- Nom de l'entreprise du commerçant
- En anglais et en français
- Maximum de 30 caractères

- Liste de toutes les URL de renvoi Soit les URL qui peuvent rediriger le client sur la passerelle de paiement INTERAC^{MD} en ligne
- Liste de toutes les URL pouvant apparaître dans le champ IDEBIT_FUNDEDURL du formulaire https POST envoyé à la passerelle de paiement INTERAC^{MD} en ligne
- Liste de toutes les URL pouvant apparaître dans le champ IDEBIT_NOTFUNDEDURL du formulaire https POST envoyé à la passerelle de paiement INTERAC^{MD} en ligne

19.2.1.2 Fournisseurs de service tiers et de panier d'achat

Dans la documentation de votre produit, indiquez à vos clients de fournir les renseignements ci-dessous au centre d'assistance à l'intégration de Passerelle Moneris pour l'enregistrement de la certification :

- Logo du commerçant à afficher sur la passerelle de paiement INTERAC^{MD} en ligne
- En français et en anglais
- 120 x 30 pixels
- Format PNG (seul format pris en charge)

- Nom de l'entreprise du commerçant
- En anglais et en français
- Maximum de 30 caractères

- Liste de toutes les URL de renvoi Soit les URL qui peuvent rediriger le client sur la passerelle de paiement INTERAC^{MD} en ligne
- Liste de toutes les URL pouvant apparaître dans le champ IDEBIT_FUNDEDURL du formulaire https POST envoyé à la passerelle de paiement INTERAC^{MD} en ligne
- Liste de toutes les URL pouvant apparaître dans le champ IDEBIT_NOTFUNDEDURL du formulaire https POST envoyé à la passerelle de paiement INTERAC^{MD} en ligne

Consultez la section 5.3.3, à la page 122 pour connaître les exigences supplémentaires du client.

19.3 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.

19.3.1 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 1 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^{MC}
- 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 : https://esqa.moneris.com/mpg/?chlan

Variable	Type et limites	Description
		<p><u>g=fr</u></p> <p>Production : https://www3.moneris.com/mpg/?chl=ang=fr</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) + signe 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>
	EXAMPLE : 1234567.89	
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) + signe 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>
	EXAMPLE : 1234567.89	
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: #e0f2f1; padding: 10px;"> <p>REMARQUE : Certains caractères spéciaux ne sont pas autorisés : <>\$%=?^{}[]\`</p> </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> <hr/> <p>REMARQUE : La limite maximale de 22 caractères doit inclure le « / » comme l'un des caractères.</p>
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>Lors d'une transaction utilisant les renseignements d'identification au dossier pour laquelle le champ de demande e-commerce indicator est</p>

Variable	Type et limites	Description
		<p>également inclus, les valeurs permises de ce champ dépendent de la valeur envoyée pour la variable payment indicator. 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>Pour les transactions Apple Pay ou Google Pay^{MC} où vous effectuez un déchiffrement : envoyez la valeur du champ ecilIndicator ou 3dsEcilIndicator 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^{MC} sont :</p> <ul style="list-style-type: none"> 5 : Transaction de commerce électronique authentifiée 7 : Commerçant supportant le SSL
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	REMARQUE : Il s'agit de l'inverse du format de date MMAA qui est affichée sur la carte.
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 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. Valeurs possibles : P = Partiel

Variable	Type et limites	Description
		F = Final
Numéro de transaction txn_number	<i>Chaîne</i> 255 caractères (alphanumériques, trait d'union ou trait de soulignement) Longueur variable	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) Cette valeur est renvoyée dans la réponse de la transaction originale. Conclusion de préautorisation : fait référence à une préautorisation Remboursement ou correction d'achat : fait référence à un achat ou à la conclusion d'une préautorisation
Indicateur de portefeuille électronique wallet_indicator	<i>Chaîne</i> 3 caractères alphanumériques	Indique qu'un numéro de carte a été obtenu par l'entremise d'un portefeuille électronique, comme Apple Pay, Google Pay ^{MC} , Visa Checkout et Mastercard MasterPass, ou par l'entremise d'un jeton provenant de la marque de carte Requis pour les transactions Apple Pay et Google Pay ^{MC} pour lesquelles vous utilisez votre propre API pour déchiffrer les données Valeurs possibles : APP = Apple Pay intégré à une application APW = Apple Pay intégrée à un site Web GPP = Google Pay intégré à une application GPW = Google Pay intégré à un site Web VCO = Visa Checkout MMP = Mastercard Masterpass MVN = Transformation en jetons de Moneris du réseau Visa

Variable	Type et limites	Description
		<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>REMARQUE : Veuillez noter que si ce champ est inclus pour indiquer Apple Pay ou Google Pay^{MC}, alors les frais de commodité ne sont pas pris en charge.</p> <p>REMARQUE : 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>Longueur variable</p> <p>REMARQUE : 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>	<p>Identifiant unique pour les renseignements d'identification stockés du titulaire de la carte</p> <p>Renvoyé dans la réponse de la marque 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</p>

Variable	Type et limites	Description
		<p>I'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>
Indicateur de paiement	<i>Chaîne</i> 1 caractère alphabétique	<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 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</p>

Variable	Type et limites	Description
		<p>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 sur les paiements	<i>Chaîne</i> 1 caractère numérique	<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 – Chambre forte

Variable	Type et limites	Description
Clé de données	<i>Chaîne</i> 25 caractères alphanumériques	<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</p>

Variable	Type et limites	Description
		enregistrement du profil.
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>0U = Clé de données alphanumériques unique à 25 caractères</p>
Durée	<p><i>Chaîne</i></p> <p>3 caractères numériques</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>
Note	<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	<p><i>Chaîne</i></p> <p>30 caractères alphanumériques</p>	<p>Numéro de téléphone du client</p> <p>Peut être envoyée lors de la création ou de la mise à jour d'un profil de chambre forte</p>

A.5 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>REMARQUE : Il ne doit pas y avoir que des espaces ou que des zéros.</p>
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</p>

Requis*	Nom du champ	Limites	Méthode Set	Description
				<p>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 (TVP ou TVQ) s'applique</p>
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

Requis*	Nom du champ	Limites	Méthode Set	Description
				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 alphanumériques	Nom de l'acheteur ou 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 appliquée en fonction du

Requis	Variable	Nom du champ	Taille/Type	Description
				<p>montant de taxe locale</p> <p>EXEMPLE : Une TVP de 8 % devrait être 8,0.</p>
				<p>Valeur maximale de 99,99</p> <p>* Doit être fourni si la taxe locale (TVP ou TVQ) s'applique.</p>
N	Droits de douane	duty_amount	9 caractères décimaux	<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>
N	Traitemet des rabais sur la facture	discount_treatment	1 caractère numérique	<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</p>

Requis	Variable	Nom du champ	Taille/Type	Description
				l'application des rabais
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>Code postal ou code ZIP auquel la marchandise sera expédiée</p> <p>* Requis s'il y a une expédition.</p> <p>Code postal alphanumérique complet – Format ANA<space>NAN requis si expédié à une adresse canadienne</p>
C	Code postal d'origine	ship_from_pos_code	10 caractères alphanumériques	<p>Code postal ou code ZIP d'origine de la marchandise</p> <p>Pour les adresses canadiennes, un code postal alphanumérique complet au format ANA<espace>NAN valide est requis pour</p>

Requis	Variable	Nom du champ	Taille/Type	Description
				le commerçant.
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>REMARQUE : 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 d'une facture incluant la 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	Traitement fiscal	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 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>

Requis	Variable	Nom du champ	Taille/Type	Description
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 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</p>

Requis	Variable	Nom du champ	Taille/Type	Description
				13,00
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 appliqué 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 de 999 999,99 (sans symbole)</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 l'être)
Y	Code du produit	product_code	12 caractères alphanumériques	Code de produit pour cette ligne d'article – code de produit du commerçant, code de produit du fabricant ou code de produit de

Requis	Variable	Nom du champ	Taille/Type	Description
				<p>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 9999999999</p>
Y	Unité de mesure de l'article	item_uom	2 caractères alphanumériques	<p>Unité de mesure</p> <p>Utilisez les unités de mesure et codes permis par la norme ANSI X-12 EDI.</p>
Y	Coût unitaire de l'article	unit_cost	12 caractères décimaux	Coût unitaire de chaque article

Requis	Variable	Nom du champ	Taille/Type	Description
				<p>De 2 à 4 décimales sont acceptées</p> <p>Valeur minimale de 0,0001</p> <p>Valeur maximale de 999999,9999</p>
N	Montant de la TVA	vat_tax_amt	12 caractères décimaux	<p>Tout montant de taxe sur la valeur ajoutée ou autre taxe de vente</p> <p>Doit comporter 2 décimales</p> <p>Valeur minimale de 0,01</p> <p>Valeur maximale de 999 999,99</p>
N	Taux de la TVA	vat_tax_rate	4 caractères décimaux	<p>Taux de la taxe de vente</p> <p>EXEMPLE : Une TVP de 8 % devrait être 8,0.</p> <p>Valeur maximale de 99,99</p>
Y	TraITEMENT des rabais	discount_treatmentL	1 caractère numérique	<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	discount_amtL	12 caractères décimaux	Montant du rabais,

Requis	Variable	Nom du champ	Taille/Type	Description
	rabais			<p>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.6 Définition des champs de demande – Transactions de niveaux 2 et 3 de Mastercard

Tableau 1 Objets - Niveau 2 et 3 Mastercard

Objets MCCorpais	Description
MCCorpac	Données communes de carte d'entreprise
MCCoral	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 NAICS	15 caractères alphanumériques	Code du système de classification des industries de l'Amérique du Nord

Requis	Variable	Nom du champ	Taille/Type	Description
				(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 caractères numériques	Date d'achat de l'article Si présent, doit contenir une date valide au format AAMMJJ
N	CorporationVatNumber	Numéro de TVA d'entreprise	20 caractères alphanumériques	Contient le numéro de taxe sur la valeur ajoutée (TVA) d'une entreprise
N	CustomerVatNumber	Numéro de TVA du client	20 caractères alphanumériques	Indique 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	FreightAmount	Montant des	12 caractères	Frais d'expédition sur

Requis	Variable	Nom du champ	Taille/Type	Description
		frais de transport	décimaux	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 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

Requis	Variable	Nom du champ	Taille/Type	Description
				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	<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"> Société par actions Inconnu Individuel ou propriétaire unique Partenariat Association, état ou fiducie Organisations exonérées d'impôts (501C) Organisation internationale Société à responsabilité limitée (SARL) Agence gouvernementale <p>Le deuxième caractère représente le type de</p>

Requis	Variable	Nom du champ	Taille/Type	Description
				<p>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 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</p>

Requis	Variable	Nom du champ	Taille/Type	Description
				<p>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>0 = 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 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>

Requis	Variable	Nom du champ	Taille/Type	Description
				<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 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</p>

Requis	Variable	Nom du champ	Taille/Type	Description
				<p>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	Documents fiscaux	Jusqu'à 6 tableaux	<p>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 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</p>

Requis	Variable	Nom du champ	Taille/Type	Description
				taxes, comme indiqué dans les champs de demande du tableau des taxes ci-dessous.

Tableau 3 Mastercard – Détails de ligne d'article (MCCorpal) – Champ 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 caractères numériques	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 caractères numériques	Date à laquelle la marchandise a été expédiée à la destination Format AAMMJJ
N	OrderDate	Date de la commande	6 caractères numériques	Date d'achat de l'article Format AAMMJJ
Y	ProductCode	Code du produit	12 caractères	Code du produit de la

Requis	Variable	Nom du champ	Taille/Type	Description
			alphanumériques	<p>ligne d'article (si ce champ n'est pas envoyé, le champ itemComCode doit l'être).</p> <p>Si une ligne d'article « Transport/Expédition » apparaît dans la commande, la valeur productCode doit être « Freight/Shipping » (Transport/Expédition)</p> <p>.</p> <p>Si une ligne d'article « Rabais » apparaît dans la commande, la valeur productCode doit être « Discount » (Rabais).</p>
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	<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 999999,99999</p>
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

Requis	Variable	Nom du champ	Taille/Type	Description
				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 attribué par le commerçant qui catégorise le mieux les articles achetés
C	Tax	Documents fiscaux	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 chaque champ. * 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.

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>EXEMPLE : Une TVP de 5 % devrait être « 5,0 », alors qu'une TVP de ou 9,975 % devrait être « 9,975 »</p>
				Peut contenir jusqu'à 3 décimales, avec une valeur minimale de 0,001 ainsi qu'une valeur maximale de 9 999,9
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	Il s'agit de l'indicateur utilisé

Requis	Variable	Nom du champ	Taille/Type	Description
				<p>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.7 Définition des champs de demande – Transactions de niveaux 2 et 3 de Amex

Tableau 1 Champs de demande de niveau 2 et 3 de Amex - Tableau 1 - 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 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	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

Requis	Variable	Nom du champ	Taille/Type	Description										
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	Valeurs acceptées : R6 = Demandeur (obligatoire) BG = Groupe d'achat (facultatif) SF = Expéditeur (facultatif) ST = Destinataire (facultatif) 40 = Récepteur (facultatif)										
Y	n102	Nom	40 caractères alphanumériques	<p>Code n101 Signification n102</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>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	40	Nom du récepteur
R6	Nom du demandeur													
BG	Nom du groupe acheteur													
SF	Nom de l'expéditeur													
ST	Nom du destinataire													
40	Nom du récepteur													
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	Cet élément peut contenir les valeurs suivantes pour les occurrences correspondantes de l'objet										

Requis	Variable	Nom du champ	Taille/Type	Description												
				<p>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)</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>40</td> <td>S. O.</td> </tr> </tbody> </table>	Valeur n101	Dénotation ref01	R6	Valeurs acceptées : 4C = Code de destination de la marchandise (obligatoire)	BG	S. O.	SF	S. O.	ST	S. O.	40	S. O.
Valeur n101	Dénotation ref01															
R6	Valeurs acceptées : 4C = Code de destination de la marchandise (obligatoire)															
BG	S. O.															
SF	S. O.															
ST	S. O.															
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 destinataire (obligatoire)</td> </tr> <tr> <td>CR</td> <td>Numéro de référence du titulaire de la carte (facultatif)</td> </tr> </tbody> </table>	Code ref01	Dénotation ref02	4C	Code postal canadien ou code ZIP de l'adresse du destinataire (obligatoire)	CR	Numéro de référence du titulaire de la carte (facultatif)						
Code ref01	Dénotation ref02															
4C	Code postal canadien ou code ZIP de l'adresse du destinataire (obligatoire)															
CR	Numéro de référence du titulaire de la carte (facultatif)															

Tableau 2 Champs de demande de niveau 2 et 3 de Amex - Tableau 2 - 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

Requis	Variable	Nom du champ	Taille/Type	Description
				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
				EXEMPLE : 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 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
				EXEMPLE : DR = vendeur (dealer), AP = prix conseillé (advise price)

Requis	Variable	Nom du champ	Taille/Type	Description						
				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	it10618 Taille ou type it10719 <table border="1" style="margin-left: 20px;"> <tr> <td>VC</td> <td>20 caractères alphanumériques</td> </tr> <tr> <td>PO</td> <td>22 caractères alphanumériques</td> </tr> <tr> <td>Autre</td> <td>30 caractères alphanumériques</td> </tr> </table>	VC	20 caractères alphanumériques	PO	22 caractères alphanumériques	Autre	30 caractères alphanumériques	Le numéro du produit ou du service correspond au qualificateur précédent défini dans la variable it10618. La longueur maximale dépend du qualificateur défini dans la variable it10618.
VC	20 caractères alphanumériques									
PO	22 caractères alphanumériques									
Autre	30 caractères alphanumériques									
C	txi01	Code du type de taxe	2 caractères alphanumériques	Valeurs acceptées : CA = Taxe municipale (facultatif) CT = Taxe de comté						

Requis	Variable	Nom du champ	Taille/Type	Description
				<p>(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>
C	txi02	Montant en numéraire	6 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 #00AEEF; padding: 5px; background-color: #E0F2F1;"> REMARQUE : Si le champ txi02 est utilisé dans l'occurrence obligatoire de txi01=TX, txi02 doit contenir le montant </div>

Requis	Variable	Nom du champ	Taille/Type	Description
				<p>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> <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 : 9 999.99</p> <p>Un crédit est entré comme suit : - 9 999.99</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	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

Requis	Variable	Nom du champ	Taille/Type	Description
				<p>taxe indiqué dans le champ txi01.</p> <p>Valeurs acceptées :</p> <ul style="list-style-type: none"> 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	pam05	Montant final de l'article	8 caractères décimaux	<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 99999,99</p>

Requis	Variable	Nom du champ	Taille/Type	Description
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 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>

Requis	Variable	Nom du champ	Taille/Type	Description
				VA = Taxe sur la valeur ajoutée aussi appelée taxe de vente harmonisée du Canada (TVH) (facultatif)
C	txi02	Montant en numéraire	6 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;"> REMARQUE : 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.</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 : 9 999.99</p> <p>Un crédit est entré comme suit : - 9 999.99</p>
C	txi03	Pourcentage	10 caractères décimaux	Indique le pourcentage de

Requis	Variable	Nom du champ	Taille/Type	Description
				taxe (sous forme décimale) qui correspond au code de type de taxe défini dans le champ txi01 Jusqu'à 2 décimales sont supportées
C	txi06	Code d'exonération fiscale	1 caractère alphanumérique	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

A.8 Définition des champs de demande – 3-D Secure 2.0

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 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 ISO 3166-2
Java activé dans le navigateur browserJavaEnabled	<i>Chaîne</i> 1 caractère alphabétique	Indique si Java est activé dans le navigateur 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	<i>Chaîne</i>	Hauteur en pixels de l'écran du titulaire de carte

Variable	Type et limites	Description
browserScreenHeight	6 caractères numériques	
Largeur de la fenêtre du navigateur browserScreenWidth	<i>Chaîne</i> 6 caractères numériques	Hauteur 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	Nom du titulaire de carte
	REMARQUE : Les caractères accentués ne sont pas autorisés.	
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 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 caractères numériques	Code de devise à 3 chiffres ISO 4217 (CAD = 124, USD = 840)

Variable	Type et limites	Description
REMARQUE : 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
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
Type de demande requestType	<i>Chaîne</i> 2 caractères alphanumériques	Valeurs acceptées : 01 = Paiement entamé par le titulaire de la carte 02 = Périodique
Adresse d'expédition shipAddress1	<i>Chaîne</i> 50 caractères alphanumériques	Adresse d'expédition
Ville d'expédition	<i>Chaîne</i> 50 caractères	Ville d'expédition

Variable	Type et limites	Description
shipCity	alphanumériques	
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 ISO 3166-2
Indicateur de conclusion 3DS threedsCompletionInd	<i>Chaîne</i> 1 caractère alphanumérique	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

A.9 Définition des champs de demande – TMD

Variable	Type et limites	Description
Numéro de la version de la TMD	<i>Chaîne</i> Valeur numérique La version actuelle est 1.0	Numéro de version de la TMD
Montant du titulaire de carte	<i>Chaîne</i> 12 caractères numériques	Montant, en unités de devise étrangère, qui sera facturé au titulaire de la carte pour la transaction

Variable	Type et limites	Description
	La plus petite unité discrète de devise étrangère	
Code de devise du titulaire de carte	<i>Chaîne</i> 3 caractères numériques	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	<i>Chaîne</i> S. O.	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	<i>Chaîne</i> Valeur numérique La version actuelle est 1.0	Numéro de version de la TMD
Type de transaction de taux	<i>Chaîne</i> 1 caractère alphabétique	Valeur représentant le type de demande de transaction ultérieure pour laquelle le jeton de taux sera utilisé Valeurs acceptées : P = Achat R = Remboursement
Renseignements sur le taux de la TMD	<i>Objet</i>	Objet imbriqué dans la transaction du taux d'obtention de la TMD contenant les champs add cardholder amount et

Variable	Type et limites	Description
S. O.		add merchant settlement

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 sous en 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.10 Définition des champs de demande – Offlinx^{MC}

S'applique uniquement à l'intégration d'Offlinx^{MC}

Variable	Type et limites	Description
ID de correspondance de carte	<i>Chaîne</i> 50 caractères alphanumériques	Correspond à l'ID de transaction utilisé pour la fonction Offlinx ^{MC} Match pour le pixel invisible; il s'agit d'un identifiant unique créé par le commerçant. Doit être unique pour chaque

Variable	Type et limites	Description
		transaction

A.11 Définition des champs de demande – Frais de commodité

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
Montant des frais de commodité	<i>Chaîne</i> 9 caractères décimaux	Montant en dollars facturé au client en tant que frais de commodité

A.12 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 De 1 à 999	Il s'agit du nombre d'occurrences de la transaction.
Période	<i>Chaîne</i> Valeur numérique De 1 à 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. REMARQUE : 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>signe décimal (.) + 2 chiffres (sous) après le point décimal</p> <p>EXEMPLE : 1234567.89</p>	<p>fonction de l'intervalle défini par les valeurs period et recur unit.</p>
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>

A.13 Définition des champs de demande – Renseignements d'identification au dossier

Variable	Type et limites	Description
Information sur le versement	<p><i>Objet</i></p> <p>S. O.</p>	Contient les champs de demande liés aux versements
ID du plan de versements	<p><i>Chaîne</i></p> <p>36 caractères alphanumériques</p> <p>Longueur fixe</p>	Identifiant généré par la marque de carte pour un plan de versements
Référence du plan de versements	<p><i>Chaîne</i></p> <p>10 caractères alphanumériques</p> <p>Longueur fixe</p>	Nom unique et humain du plan de versements
Version des modalités	<p><i>Chaîne</i></p> <p>10 caractères alphanumériques</p> <p>Longueur variable (de 1 à</p>	<p>Version des modalités du plan de versements accepté par le titulaire de carte</p> <p>Cette version augmente automatiquement chaque fois que le</p>

Variable	Type et limites	Description
	10 caractères)	plan est mis à jour par l'émetteur.

Annexe B Définition des champs de réponse

Tableau 28 : Valeurs de réponse de l'objet Receipt

Valeur	Type	Limites	Méthode Get
		Description	
Champs de réponse généraux			
Type de carte	Chaîne	2 caractères alphabétiques (min. 1)	receipt.getCardType();
<p>Représente le type de carte utilisée dans la transaction, par exemple, Visa ou Mastercard</p> <p>Valeurs possibles :</p> <ul style="list-style-type: none"> 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 Jusqu'à 7 chiffres (dollars) + signe décimal (.) + 2 chiffres (sous) après le point décimal	receipt.getTransactionAmount();
<p>EXEMPLE : 1234567.89</p> <p>Montant de la transaction qui a été traitée</p>			
Numéro de transaction	Chaîne	255 caractères alphanumériques	receipt.getTransactionNumber();
<p>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</p>			
ID de reçu	Chaîne	50 caractères alphanumériques	receipt.getReceiptId();
<p>Numéro de commande qui a été précisé dans la demande de transaction</p>			

Valeur	Type	Limites	Méthode Get
		Description	
Type de transaction	Chaîne	2 caractères alphanumériques	receipt.getTransType();
	0 = Achat 1 = Préautorisation 2 = Conclusion 4 = Remboursement 11 = Nul		
Numéro de référence	Chaîne	18 caractères numériques	receipt.getReferenceNum();
	Terminal utilisé pour traiter la transaction ainsi que numéro du quart, du lot et de la séquence Ces données sont généralement utilisées pour faire 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 : 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 caractères numériques	receipt.getResponseCode();
	érieur à 50 : Transaction approuvée érieur ou égal à 50 : Transaction refusée 1 : Transaction incomplète Pour plus de détails sur les codes de réponse, consultez le document sur les codes de réponse qui se trouve dans le portail pour développeurs (https://developer.moneris.com).		
Organisations de vente indépendantes :	Chaîne	2 caractères numériques	receipt.getISO();
	Code de réponse ISO		
Totaux de banques	Objet		receipt.getBankTotals();
	Données de réponse reçues dans une demande de fermeture de lot et de totaux des lots ouverts Consultez l'Annexe A Définitions des champs de réponse à la page .		
Message	Chaîne	100 caractères alphanumériques	receipt.getMessage();

Valeur	Type	Limites	Méthode Get
	Description		
	Description de la réponse renvoyée par l'émetteur Le message renvoyé par l'émetteur est destiné à informer le commerçant uniquement, il ne doit pas être affiché sur les reçus des clients.		
Code d'autorisation	Chaîne	8 caractères alphanumériques	receipt.getAuthCode();
	Code d'autorisation reçu de l'institution émettrice		
Compléter	Chaîne	true/false	receipt.getComplete();
	La transaction a été envoyée au serveur d'autorisation, et une réponse a été reçue.		
Date de la 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.getTransactionTime();
	Horodateur du serveur de traitement		
Billet	Chaîne	S. O.	receipt.getTicket();
	Champ réservé		
Délai écoulé	Chaîne	true/false	receipt.getTimeOut();
	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.		
Champs de réponse de fermeture ou d'ouverture de lot			
Types de cartes traitées	Tableau chaînes	S. O.	receipt.getCreditCards(ecr_no);
	Renvoie tous les types de cartes traités dans le lot actuel pour l'ID du terminal et le numéro CEE de la demande		

Valeur	Type	Limites	Méthode Get
		Description	
ID du terminal	Chaîne	8 caractères alphanumériques	receipt.getTerminalIDs(); Code à venir
Retourne l'ID du terminal et le numéro CEE de la demande			
Nombre d'achats	Chaîne	4 caractères numériques	receipt.purchaseCount(ecr, cardType);
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	receipt.purchaseAmount(ecr, cardType);
Indique le montant en dollars traité pour les transactions d'achat, de conclusion de préautorisation ou les transactions forcées 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			
Nombre de remboursements	Chaîne	4 caractères numériques	receipt.refundCount(ecr, cardType);
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	receipt.refundAmount(ecr, cardType);
Indique le montant en dollars du traitement des transactions de remboursement, de remboursement indépendant ou de crédit de chambre de compensation automatisée 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			

Valeur	Type	Limites	Méthode Get
		Description	
Nombre de corrections d'achat	Chaîne	4 caractères numériques	receipt.getCorrectionCount(ecr, cardType);
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	receipt.getCorrectionAmount(ecr, cardType);
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)			
Succès de la facturation périodique	Chaîne	true/false	receipt.getRecurSuccess();
	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	receipt.getRecurUpdateSuccess();
	Indique la réussite de la mise à jour de la facturation périodique		
Prochaine date de l'occurrence	Chaîne	aaaa-mm-jj	receipt.getNextRecurDate();
	Indique la prochaine date de facturation périodique		
Date de fin de la répétition	Chaîne	aaaa-mm-jj	receipt.getRecurEndDate();
	Indique la dernière date de facturation périodique		
Champs de réponse pour la vérification de l'état (consulter)			
Code d'état	Chaîne	3 caractères alphanumériques	receipt.getStatusCode();

Valeur	Type	Limites	Méthode Get
	Description		
		Inférieur à 50 : Transaction trouvée et réussie Supérieur ou égal à 50 : Transaction non trouvée et échouée	
		REMARQUE : Le code d'état n'est généré que si la valeur true est attribuée à la propriété Status Check de l'objet de connexion.	
Message d'état	Chaîne	found (trouvé) ou not found (non trouvé)	receipt.getStat usMessage();
	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.		
	REMARQUE : Le message d'état n'est généré que si la valeur true est attribuée à la propriété Status Check de l'objet de connexion.		
Champs de réponse du SVA (consulter la section 10.1, à la page 367)			
Code de résultat du SVA	Chaîne	1 caractère alphanumérique	receipt.getAvsR esultCode();
	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.getCvdR esultCode();
	Indique le résultat de la validation du NVC Le premier octet est l'indicateur numérique du NVC envoyé avec la demande, et le second est le code de réponse. Les codes de réponse possibles sont indiqués à l'annexe B		
Champs de réponse des modules d'extension pour les commerçants (consulter la section « Module d'extension pour les commerçants » 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.getMpis uccess();
	True si la tentative a été réussie, false si la tentative a échoué		
Message	Chaîne	100 caractère alphabétique	receipt.getMpim essage();

Valeur	Type	Limites	Méthode Get
		Description	
		<p>Les transactions MPI TXN peuvent produire les valeurs suivantes :</p> <ul style="list-style-type: none"> • Y : Crée une fenêtre contextuelle du formulaire de vérification Vérifié par Visa • N : Envoie l'achat ou la préautorisation avec la valeur crypt type = 6 • U : Envoie l'achat ou la préautorisation avec la valeur crypt type = 7 <p>Les transactions MPI ACS peuvent produire les valeurs suivantes :</p> <ul style="list-style-type: none"> • Y ou A : (Également receipt.getMpiSuccess () =true) Poursuivre la transaction d'achat ou de préautorisation utilisant le code de vérification d'authentification du titulaire de carte (CAVV). • N : Échec de l'authentification ou transaction à haut risque Il est recommandé de ne pas poursuivre la transaction. <p>Selon la tolérance au risque du commerçant et les résultats d'autres méthodes de détection de la fraude, la transaction peut se poursuivre avec la valeur crypt type = 7.</p> <ul style="list-style-type: none"> • U ou time out : Envoie l'achat ou la préautorisation avec la valeur crypt type = 7. 	
URL du terme	Chaîne	255 caractères alphanumériques	
	URL à laquelle la valeur PARes est renvoyée		
Marchand Direct	Chaîne	1024 caractères alphanumériques	
	Données définies par le commerçant qui ont été renvoyées		
URL du serveur de contrôle d'accès	Chaîne	255 caractères alphanumériques	
	URL utilisée pour la fenêtre contextuelle générée		
Code de vérification d'authentification du titulaire de carte des modules d'extension pour les commerçants	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();
Code de résultat du code de vérification d'authentification du titulaire de carte	Chaîne	1 caractère alphanumérique	receipt.getCavvResultCode(); CODE HERE CODE HERE

Valeur	Type	Limites	Méthode Get
		Description	
	Indique le résultat du code de vérification d'authentification du titulaire de carte (CAVV) Visa Pour plus de renseignements, consultez la section 1 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 section4.1, à la page 61)			
Clé de données	Chaîne	28 caractères alphanumériques	receipt.getDataKey();
	Le champ de réponse data key (clé de données) est rempli lorsque vous envoyez les transactions suivantes : Ajout de carte de crédit à la chambre forte – ResAddCC (page 64), Ajout d'une carte de crédit chiffrée à la chambre forte – EncResAddCC (page 68), Transformation en jeton d'une carte de crédit dans la chambre forte – ResTokenizeCC (page 93), Ajout d'un jeton temporaire à la chambre forte – ResTempAdd (page 71) ou Ajout d'un jeton à la chambre forte – ResAddToken (page 89). 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.		
Type de paiement associé à la chambre forte	Chaîne	cc	receipt.getPaymentType();
	Indique le type de paiement associé à un profil de chambre forte		
Type de paiement pour la carte arrivant à expiration	Chaîne	cc	receipt.getExpPaymentType();
	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		

Valeur	Type	Limites		Méthode Get
		Description		
Numéro de carte de crédit masqué par la chambre forte	Chaîne	20 caractères numériques		receipt.getResMaskedPan();
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 caractères numériques		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				
Réussite de la chambre forte	Chaîne	true/false		receipt.getResSuccess();
Indique si la transaction de la chambre forte a été réussie				
ID de client dans la chambre forte	Chaîne	30 caractères alphanumériques		receipt.getResCustomerId();
Renvoie l'ID de client enregistré dans le profil				
ID de client lié à la carte arrivant à expiration	Chaîne	30 caractères alphanumériques		receipt.getExpCustomerId();
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		receipt.getResPhone();
Renvoie le numéro de téléphone enregistré dans le profil				
Numéro de téléphone de la carte arrivant à expiration	Chaîne	30 caractères alphanumériques		receipt.getExpPhone();
Renvoie le numéro de téléphone enregistré dans le profil Applicable au type de transaction d'obtention des cartes expirées dans la chambre forte				

Valeur	Type	Limites	Méthode Get
		Description	
Adresse courriel dans la chambre forte	Chaîne	30 caractères alphanumériques	receipt.getResEmail();
Renvoie l'adresse courriel enregistrée dans le profil			
Adresse courriel de la carte arrivant à expiration	Chaîne	30 caractères alphanumériques	receipt.getExpEmail();
	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	receipt.getResNote();
	Renvoie la note enregistrée dans le profil		
Note de la carte arrivant à expiration	Chaîne	30 caractères alphanumériques	receipt.getExpNote();
	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	receipt.getResExpdte();
	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	receipt.getExpExpdte();
	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		

Valeur	Type	Limites	Méthode Get
		Description	
Numéro d'immeuble pour le SVA dans la chambre forte	Chaîne	19 caractères alphanumériques	<code>receipt.getResAvsStreetNumber();</code>
	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	<code>receipt.getExpAvsStreetNumber();</code>
	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	<code>receipt.getResAvsStreetName();</code>
	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	<code>receipt.getExpAvsStreetName();</code>
	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	<code>receipt.getResAvsZipcode();</code>
	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	<code>receipt.getExpAvsZipcode();</code>
	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		

Valeur	Type	Limites		Méthode Get				
		Description						
Numéro de carte de crédit dans la chambre forte	Chaîne	20 caractères numériques		receipt.getResPan();				
	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		receipt.getCorporateCard();				
	Indique si la carte associée au profil de la chambre forte est une carte d'entreprise							
Champs de réponse pour la bande magnétique chiffrée (consulter la Section 1, à la page 1)								
Numéro masqué de carte de crédit	Chaîne	20 caractères alphanumériques		receipt.getMaskedPan();				
Champs de réponse pour les frais de commodité (consulter l'annexe A, à la page 1)								
Réussite des frais de commodité	Chaîne	true/false		receipt.get CfSuccess();				
	Indique si la transaction de frais de commodité a été traitée avec succès							
État des frais de commodité	Chaîne	3 caractères alphanumériques		receipt.get CfStatus();				
	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		receipt.getFeeAmount();				

Valeur	Type	Limites	Méthode Get
		Description	
	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	receipt.getFeeRate();
	Taux des frais de commodité défini dans le profil du commerçant Par exemple : 1,00 = Un montant fixe 10,0 = Un montant en pourcentage		
Type de frais de commodité	Chaîne	AMT ou PCT	receipt.getFeeType();
	Type de frais de commodité défini dans le profil du commerçant Voici les options possibles : AMT = Montant fixe PCT = Pourcentage		

Tableau 29 : Codes de réponse de transaction financière

Code	Description
< 50	Transaction approuvée
≥ 50	Transaction refusée
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
-------------	--------------------

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 exiprant 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>	<code>receipt.getMPIChallengeURL()</code> Si la valeur de transStatus est « C », ce champ

Variable	Type et limites	Description
	Caractères alphanumériques	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> 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> Caractères alphanumériques	<code>receipt.getMPIThreeDSServerTransId()</code> Identifiant de transaction unique de 3-D Secure
Version de 3DS	<i>Chaîne</i> 1 caractère numérique	<code>receipt.getThreeDSVersion();</code> Requis lors de l'envoi d'une transaction 3-D Secure 2.0 Valeur par défaut de 1 lorsqu'aucune valeur n'est fournie Valeurs acceptées : 1 ou 2
État de la transaction	<i>Chaîne</i> 1 caractère alphanumérique Y ou N	<code>receipt.getMPITransStatus()</code> Indique le résultat du serveur de contrôle d'accès (ACS)

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 MCP rate token 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 caractères numériques</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) + signe décimal (.) + 2 chiffres (sous) après le point décimal</p>	<pre>receipt.getMerchantSettlementAmount();</pre> <p>Montant qui sera versé au commerçant, en dollars canadiens</p>
	<p>EXEMPLE : 1234567.89</p>	
Code de devise du titulaire de carte	<p><i>Chaîne</i></p> <p>3 caractères numériques</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 caractères numériques</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
	Longueur variable	
Code d'état d'erreur de la TMD (MCP error status code)	<i>Chaîne</i> 4 caractères numériques Longueur variable	<code>receipt.getMCPErrorStatusCode();</code> 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	<code>receipt.getMCPErrorMessage();</code> Message correspondant à un code d'erreur de la TMD
ID du serveur	<i>Chaîne</i> 15 caractères alphanumériques	<code>receipt.getHostId();</code> 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)	<code>receipt.getRateTxnType();</code> Correspond au type de transaction envoyé dans la demande
Jeton du taux de la TMD	<i>Chaîne</i> 17 caractères alphanumériques	<code>receipt.GetMCPRateToken();</code> 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	<i>Chaîne</i>	<code>receipt.getRateInqStartTime();</code>

Variable	Type et limites	Méthode Get et description
demande de taux	24 caractères alphanumériques	Heure locale (ISO 8601) à laquelle le taux est demandé
Heure de fin de la demande de taux	<i>Chaîne</i> 24 caractères alphanumériques	<code>receipt.getRateInqEndTime();</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 caractères numériques	<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 caractères numériques	<code>receipt.getRateValidityEndTime();</code> Heure (unix UTC) jusqu'à laquelle le taux est valide
Période de validité du taux	<i>Chaîne</i> 3 caractères numériques 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

Variable	Type et limites	Description
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 représenté par un objet Plan Details précis.
Taux annuel en pourcentage	<i>Chaîne</i> Valeur numérique	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 Valeurs acceptées : De 0 à 10000 Le taux en pourcentage est affiché avec deux décimales implicites. EXEMPLE : 320 = 3,2 %
Fréquence des versements	<i>Chaîne</i> Maximum 10 caractères alphabétiques	Fréquence des versements du plan Valeurs possibles : WEEKLY BIWEEKLY MONTHLY BIMONTHLY
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 Utilisé comme champ de demande dans l'objet Installment Info
Nom du plan de versements	<i>Chaîne</i> Maximum 255 caractères alphanumériques	Nom du plan de versements, peut ne pas être unique

Variable	Type et limites	Description
Référence du plan de versements	<p><i>Chaîne</i></p> <p>10 caractères alphanumériques</p> <p>Longueur fixe</p>	<p>Nom unique et humain du plan de versements</p> <p>Utilisé comme champ de demande dans l'objet Installment Info</p>
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"> ISSUER_PROMOTION BI_LATERAL ISSUER_DEFAULT MARKET
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>EXEMPLE : 123112 = 1 231,12 \$</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>

Variable	Type et limites	Description
		EXEMPLE : 123112 = 1 231,12 \$
Frais initiaux	<i>Chaîne</i> Valeur numérique	Frais initiaux facturés au titulaire de carte pour le plan de versements; facturés uniquement lors du premier versement
Dernier versement	<i>Objet</i> S. O.	Contient les détails liés 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. EXEMPLE : 123112 = 1 231,12 \$
Frais du dernier versement	<i>Chaîne</i> Maximum 9 caractères numériques	Frais facturés lors du dernier versement Les deux derniers chiffres représentent les sous. EXEMPLE : 123112 = 1 231,12 \$
Information liée à la promotion	<i>Objet</i> S. O.	Contient les renseignements liés 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 fourni par l'émetteur qui identifie un programme ou une promotion

Variable	Type et limites	Description
Modalités	<i>Objet tableau</i> S. O.	Contient les champs liés aux modalités présentées aux titulaires de carte
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>	Frais totaux liés au plan

Variable	Type et limites	Description
	Maximum 9 caractères numériques	Les deux derniers chiffres représentent les sous. EXEMPLE : 123112 = 1 231,12 \$
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 EXEMPLE : 123112 en dollars canadiens = 1 231,12 \$ CA

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	Version des modalités du plan de versement accepté par le titulaire de carte

Variable	Type et limites	Description
	alphanumériques Longueur variable (de 1 à 10 caractères)	Cette version augmente automatiquement chaque fois que le plan est mis à jour par l'émetteur.
ID d'acceptation du plan	<i>Chaîne</i> 36 caractères alphanumériques Longueur fixe	Nom humain court et unique créé par Visa en caractère alphanumérique du plan de versements
État du plan de versements	<i>Chaîne</i> 1 caractère alphabétique Longueur fixe	Valeurs possibles : N = Nouveau plan, pas encore accepté A = Plan accepté C = Plan annulé
Réponse du plan	<i>Chaîne</i> Maximum 50 caractères numériques	Code de réponse du plan de versements Valeurs possibles : 00 = Plan traité et accepté Si la réponse n'est pas 00, cela signifie qu'un problème est survenu avec le plan de versements. Un message d'erreur verbal indiquant que la demande a été reçue par Visa est retourné.

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
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
	Demande refusée : Échec du CAC Réservé Échec du processus de sécurité Pas d'arriérés (le reçu de transaction n'est pas imprimé) Type de fichier non valide Pas de fichier de ce type Fichier verrouillé Échoué Longueur de fichier incorrecte Erreur de décompression de fichier Erreur de nom de fichier Le fichier ne peut pas être reçu Refuser – Ne pas honorer
051	Carte expirée
052	Nombre de tentatives de saisie du code NIP dépassé Limite de tentatives de code NIP dépassée Nombre autorisé de tentatives de code NIP dépassé
053	Pas de partage
054	Aucun module de sécurité
055	Transaction non valide
056	Pas de support ou transaction non autorisée à l'acquéreur Transaction non supportée par l'institution financière ou non supportée par l'acquéreur

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
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
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.

Transaction non valideInvalid 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.

Annexe E Listes de contrôle pour les commerçants concernant la certification des paiements en INTERAC^{MD} ligne

Renseignements du commerçant

Nom et URL	Nom du commerçant (anglais)	
	URL de la page d'accueil (anglais)	
	Nom du commerçant (français)	
	URL de la page d'accueil (français)	
Numéro	Numéro de commerçant	
Catégorie de frais de transaction (Encerclez-en une)	Gouvernement Éducation Général	

Liste de contrôle pour les tests frontaux

No du cas	Date à laquelle le formulaire a été rempli	Commentaires
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		

No du cas	Date à laquelle le formulaire a été rempli	Commentaires
13		
14		
15		
16		
17		
18		
19		
20		
21		
22		
23		
24		
25		
26		
27		
28		
29		
30		
31		
32		
33		
34		
35		
36		
37		
38		

No du cas	Date à laquelle le formulaire a été rempli	Commentaires
39		

Exigences pour le commerçant

Tableau 31 : Liste de contrôle des exigences en matière d'affichage Web

Fait	Exigence obligatoire
Page de paiement	
	Afficher le logo INTERAC en ligne, le mot-symbole (texte « INTERAC en ligne ») ou les deux
Exigences relatives au dessin et au mot-symbole (toute page)	
	<p>Logos des autres options de paiement :</p> <ul style="list-style-type: none"> • Afficher le dessin (logo) INTERAC en ligne si le commerçant affiche les marques ou les logos d'autres options de paiement. • Le dessin doit être de taille égale et ne doit pas être moins visible que ceux des autres marques d'options de paiement
	<p>Mot-symbole INTERAC :</p> <ul style="list-style-type: none"> • INTERAC toujours écrit en majuscules ou en italique (comme dans « le service INTERAC en ligne »). • Dans la première utilisation du mot-symbole INTERAC en ligne, INTERAC est suivi de la notation MD en exposant. Par exemple, « <i>Interac[®]</i> » (anglais) ou « <i>Interac^{MD}</i> » (français). • Sur la même page que la première occurrence du mot-symbole, la note de bas de page suivante, qui correspond à la langue, apparaît : <ul style="list-style-type: none"> •[®] Trademark of Interac Inc. Used under licence" •^{MD} Marque de commerce d'Interac Inc. Utilisée sous licence
Version du dessin	
	<p>Il faut utiliser le dessin bicolore sur le Web :</p> <ul style="list-style-type: none"> • Version horizontale : la hauteur ne doit pas être inférieure à 25 pixels (rapport largeur/hauteur de 2:37:1). • Version verticale : la largeur ne doit pas être inférieure à 30 pixels (rapport largeur/hauteur de 1:01:37).
Renseignements sur la section « En apprendre davantage (Learn more) »	
	Il faut fournir aux consommateurs un lien vers la page www.interaconline.com/learn (de préférence sur la page de paiement).
Page de confirmation	
	Doit indiquer que la transaction est réussie
	Doit afficher le nom et le numéro de confirmation de l'institution financière

Fait	Exigence obligatoire
	Doit offrir la possibilité d'imprimer la page
Page d'erreur	
	Doit indiquer que le paiement a échoué
	Doit indiquer que la commande est annulée ou afficher d'autres options de paiement
Message de délai de paiement	
	Doit être affiché si le consommateur dispose de moins de 30 minutes pour effectuer le paiement
Paiement	
	Doit afficher le total en dollars canadiens

Tableau 32 : Liste de contrôle des exigences en matière de sécurité et de confidentialité

Fait	Exigence obligatoire
Commerçant	
	Utilise un chiffrement SSL d'au moins 128 bits lors de la collecte de renseignements personnels
	Protège les renseignements des consommateurs conformément aux lois fédérales et provinciales sur la protection des renseignements personnels
	Adhère au Code canadien de pratiques pour la protection des consommateurs dans le commerce électronique
Captures d'écran fournies	
	Page de paiement (où le client sélectionne l'option INTERAC en ligne)
	Page de confirmation (un des cas de test suivants : 1, 2 ou 3)
	Page d'erreur (cas de test 4)

Annexe F Listes de contrôle des fournisseurs de services tiers pour les tests de certification des paiements INTERAC^{MD} en ligne

Renseignements sur les fournisseur de service tiers

Nom	Anglais	
	Français	
Application Web du commerçant	Nom de la solution	
	Version	
Acquéreur		

Renseignements sur l'affichage des sites Web Interaconline.com/Interacenligne.com

Consultez la page http://www.interaconline.com/merchants_thirdparty.php pour obtenir des exemples.

Coordonnées en anglais	Un maximum de 5 lignes est accepté. Un maximum de 35 caractères par ligne est accepté. Par exemple, le nom et le titre du contact, le service, le téléphone, le site Web, le courriel.
Logo en anglais	Type de fichier : PNG Taille maximale : 120x120 pixels

Coordonnées en français	Un maximum de 5 lignes est accepté. Un maximum de 35 caractères par ligne est accepté. Par exemple, le nom et le titre du contact, le service, le téléphone, le site Web, le courriel.
Logo en français	Type de fichier : PNG Taille maximale : 120x120 pixels

Tableau 33 : Liste de contrôle pour les tests frontaux

No du cas	Date à laquelle le formulaire a été rempli	Commentaires
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		

No du cas	Date à laquelle le formulaire a été rempli	Commentaires
17		
18		
19		
20		
21		
22		
23		
24		
25		
26		
27		
28		
29		
30		
31		
32		
33		
34		
35		
36		
37		
38		
39		

Exigences pour le commerçant

Tableau 34 : Liste de contrôle des exigences en matière d'affichage Web

Fait	Exigence obligatoire
Page de paiement	
	Afficher le logo INTERAC en ligne, le mot-symbole (texte « INTERAC en ligne ») ou les deux
Exigences relatives au dessin et au mot-symbole (toute page)	
	<p>Logos des autres options de paiement :</p> <ul style="list-style-type: none"> › Afficher le dessin (logo) INTERAC en ligne si le commerçant affiche les marques ou les logos d'autres options de paiement. › Le dessin doit être de taille égale et ne doit pas être moins visible que ceux des autres marques d'options de paiement
	<p>Mot-symbole INTERAC :</p> <ul style="list-style-type: none"> › INTERAC toujours être écrit en majuscules ou en italique (comme dans « le service INTERAC en ligne »). › Dans la première utilisation du mot-symbole INTERAC en ligne, INTERAC est suivi de la notation MD en exposant. Par exemple, « <i>Interac[®]</i> » (anglais) ou « <i>Interac^{MD}</i> » (français). › Sur la même page que la première occurrence du mot-symbole, la note de bas de page suivante, qui correspond à la langue, apparaît : <ul style="list-style-type: none"> •[®] Trademark of Interac Inc. Used under licence" •^{MD} Marque de commerce d'Interac Inc. Utilisée sous licence
Version du dessin	
	<p>Il faut utiliser le dessin bicolore sur le Web :</p> <ul style="list-style-type: none"> › Version horizontale : la hauteur ne doit pas être inférieure à 25 pixels (rapport largeur/hauteur de 2:37:1). › Version verticale : la largeur ne doit pas être inférieure à 30 pixels (rapport largeur/hauteur de 1:01:37).
Renseignements sur la section « En apprendre davantage (Learn more) »	
	Il faut fournir aux consommateurs un lien vers la page www.interaconline.com/learn (de préférence sur la page de paiement).
Page de confirmation	
	Doit indiquer que la transaction est réussie
	Doit afficher le nom et le numéro de confirmation de l'institution financière
	Doit offrir la possibilité d'imprimer la page
Page d'erreur	
	Doit indiquer que le paiement a échoué
	Doit indiquer que la commande est annulée ou afficher d'autres options de paiement

Fait	Exigence obligatoire
Message de délai de paiement	
	Doit être affiché si le consommateur dispose de moins de 30 minutes pour effectuer le paiement
Paiement	
	Doit afficher le total en dollars canadiens

Tableau 35 : Liste de contrôle des exigences en matière de sécurité et de confidentialité

Fait	Exigence obligatoire
Commerçant	
	Utilise un chiffrement SSL d'au moins 128 bits lors de la collecte de renseignements personnels
	Protège les renseignements des consommateurs conformément aux lois fédérales et provinciales sur la protection des renseignements personnels
	Adhère au Code canadien de pratiques pour la protection des consommateurs dans le commerce électronique

Tableau 36 : Liste de contrôle des captures d'écran requises

Fait	Exigence obligatoire
Captures d'écran fournies	
	Page de paiement (où le client sélectionne l'option INTERAC en ligne)
	Page de confirmation (un des cas de test suivants : 1, 2 ou 3)
	Page d'erreur (cas de test 4)

Annexe G Listes de contrôle pour les commerçants concernant la certification des paiements INTERAC^{MD} en ligne

Renseignements du commerçant

Nom et URL	Nom du commerçant (anglais)	
	URL de la page d'accueil (anglais)	
	Nom du commerçant (français)	
	URL de la page d'accueil (français)	
Numéro	Numéro de commerçant	
Catégorie de frais de transaction (Encerclez-en une)	Gouvernement Éducation Général	
Fournisseurs de service tiers	Nom de l'entreprise	
Application Web du commerçant du fournisseur de services	Nom de la solution Version	

Exigences pour le commerçant

Tableau 37 : Liste de contrôle des exigences en matière d'affichage Web

Fait	Exigence obligatoire
Page de paiement	Afficher le logo INTERAC en ligne, le mot-symbole (texte « INTERAC en ligne ») ou les deux
Exigences relatives au dessin et au mot-symbole (toute page)	Logos des autres options de paiement : •Afficher le dessin (logo) INTERAC en ligne si le commerçant affiche les marques ou les logos d'autres options de paiement. •Le dessin doit être de taille égale et ne doit pas être moins visible que ceux des autres marques d'options de paiement

Fait	Exigence obligatoire
	<p>Mot-symbole INTERAC :</p> <ul style="list-style-type: none"> • INTERAC toujours être écrit en majuscules ou en italique (comme dans « le service INTERAC en ligne »). • Dans la première utilisation du mot-symbole INTERAC en ligne, INTERAC est suivi de la notation MD en exposant. Par exemple, « <i>Interac®</i> » (anglais) ou « <i>Interac™</i> » (français). • Sur la même page que la première occurrence du mot-symbole, la note de bas de page suivante, qui correspond à la langue, apparaît : <ul style="list-style-type: none"> •® Trademark of Interac Inc. Used under licence" •MD Marque de commerce d'Interac Inc. Utilisée sous licence
Version du dessin	
	<p>Il faut utiliser le dessin bicolore sur le Web :</p> <ul style="list-style-type: none"> • Version horizontale : la hauteur ne doit pas être inférieure à 25 pixels (rapport largeur/hauteur de 2:37:1). • Version verticale : la largeur ne doit pas être inférieure à 30 pixels (rapport largeur/hauteur de 1:01:37).
Renseignements sur la section « En apprendre davantage (Learn more) »	
	<p>Il faut fournir aux consommateurs un lien vers la page www.interaconline.com/learn (de préférence sur la page de paiement).</p>
Page de confirmation	
	Doit indiquer que la transaction est réussie
	Doit afficher le nom et le numéro de confirmation de l'institution financière
	Doit offrir la possibilité d'imprimer la page
Page d'erreur	
	Doit indiquer que le paiement a échoué
	Doit indiquer que la commande est annulée ou afficher d'autres options de paiement
Message de délai de paiement	
	Doit être affiché si le consommateur dispose de moins de 30 minutes pour effectuer le paiement
Paiement	
	Doit afficher le total en dollars canadiens

Tableau 38 : Liste de contrôle des exigences en matière de sécurité et de confidentialité

Fait	Exigence obligatoire
Commerçant	
	Utilise un chiffrement SSL d'au moins 128 bits lors de la collecte de renseignements personnels
	Protège les renseignements des consommateurs conformément aux lois fédérales et provinciales sur la protection des renseignements personnels
	Adhère au Code canadien de pratiques pour la protection des consommateurs dans le commerce électronique
Captures d'écran fournies	
	Page de paiement (où le client sélectionne l'option INTERAC en ligne)
	Page de confirmation (un des cas de test suivants : 1, 2 ou 3)
	Page d'erreur (cas de test 4)

Annexe H Détail du cas de test pour la certification des paiements INTERAC^{MD} en ligne

- H.1 Validations communes
- H.2 Cas de test
- H.3 Valeurs des cas de tests frontaux de commerçants

H.1 Validations communes

Le commerçant envoie une demande à l'outil de test en ligne pour commerçants d'INTERAC, qui valide les champs de la manière suivante :

- Tous les champs obligatoires sont présents.
- Tous les champs sont valides selon leur définition dans les *spécifications fonctionnelles d'INTERAC en ligne (INTERAC Online Functional Specifications)* (y compris la longueur des champs, les caractères valides, etc.).
- Le numéro du commerçant est celui d'un commerçant enregistré valide.
- L'URL pour les achats avec fonds suffisants correspond à l'une des URL pour les achats avec fonds suffisants enregistrées du commerçant qui ont été fournies lors de son inscription.
- L'URL pour les achats avec fonds non suffisants correspond à l'une des URL pour les achats avec fonds non suffisants enregistrées du commerçant qui ont été fournies lors de son inscription.
- Aucun champ supplémentaire n'est présent.

H.2 Cas de test

Tableau 39 : Cas 1 à 3

Déroulement	Tester que le commerçant peut effectuer tout ce qui suit : Envoyer une demande valide à la page de Passerelle Recevoir une confirmation de financement valide provenant de l'application bancaire en ligne de l'émetteur Envoyer une demande de conclusion d'achat à l'acquéreur Recevoir une réponse approuvée de l'acquéreur
Prérequis	Aucun
Configuration	Le commerçant envoie des messages de formulaire à l'outil de test pour commerçants, qui répond à son tour à l'URL pour les achats sans fonds suffisants ou avec fonds suffisants. Le commerçant est connecté à un émulateur d'acquéreur, qui peut être configuré pour confirmer toute demande de confirmation de paiement. (C'est-à-dire que le processus dorsal d'envoi d'un message 0200 à l'émetteur est émulé pour toujours accepter la demande d'achat).

Outils spéciaux requis	Aucun
Exigences en matière de données d'entrée	<p>L'acquéreur doit avoir enregistré le commerçant à l'aide du système d'administration et avoir fourni les éléments suivants :</p> <ul style="list-style-type: none"> • IDEBIT_FUNDEDURL(S) • IDEBIT_NOTFUNDEDURL(S) HTTP REFERERURL(S) <p>Des données fournies par l'outil de test pour commerçants</p>
Stratégie d'exécution	Entamez un paiement chez le commerçant. Les deux chiffres les moins significatifs du montant en dollars doivent être égaux au numéro du cas de test. Par exemple, si vous exécutez le cas de test 3, le format du montant doit être ### ### #03,##.
Résultat attendu	<p>Le commerçant indique au client que l'achat a été effectué et présente un écran de confirmation qui comprend (selon le cas de test) le montant correct, le nom de l'émetteur et le numéro de confirmation de l'émetteur.</p> <p>Cas de test 1</p> <ul style="list-style-type: none"> • Nom de l'émetteur : 123Bank Numéro de confirmation de l'émetteur : CONF#123 <p>Cas de test 2</p> <ul style="list-style-type: none"> • Nom de l'émetteur : Bank Éàéëï#\$.,-/=?@' Numéro de confirmation de l'émetteur : #\$.,-=/?@'UPdn9 <p>Cas de test 3</p> <ul style="list-style-type: none"> • Nom de l'émetteur : B Numéro de confirmation de l'émetteur : C
Registres applicables	<ul style="list-style-type: none"> • Registres de l'outil de test pour commerçants • Capture d'écran de la page de confirmation du commerçant

Tableau 40 : Cas 4

Déroulement	Vérifier que le commerçant traite un rejet en réponse à l'acquéreur
Prérequis	Aucun
Configuration	Identique aux cas de test 1 à 3 sauf que l'émulateur de l'acquéreur doit être configuré pour refuser la demande de confirmation de paiement. (C'est-à-dire pour émuler le scénario dans lequel un émetteur envoie un refus dans la réponse 0210 au message 0200 de l'acquéreur).
Outils spéciaux requis	Aucun

Exigences en matière de données d'entrée	L'acquéreur doit avoir enregistré le commerçant à l'aide du système d'administration et avoir fourni les éléments suivants : • IDEBIT_FUNDEDURL(S) • IDEBIT_NOTFUNDEDURL(S) HTTP REFERERURL(S) Des données fournies par l'outil de test pour commerçants
Stratégie d'exécution	Lancer un paiement chez le commerçant pour tout montant dont les deux chiffres les moins significatifs du dollar sont 04. (C'est-à-dire, au format ### ### #04.##)
Résultat attendu	Le commerçant informe le client que l'achat a été refusé. Le nom de l'émetteur et le numéro de confirmation de l'émetteur ne sont pas affichés.
Registres applicables	Registres de l'outil de test pour commerçants

Tableau 41 : Cas 5 à 22

Déroulement	Vérifier qu'un commerçant traite de manière sûre les redirections vers l'URL pour les achats avec fonds suffisants avec des données non valides, et traite la transaction conclue en raison de fonds suffisants
Prérequis	Aucun
Configuration	Aucun L'émulateur de l'acquéreur n'est pas nécessaire, car le commerçant ne soumet aucune demande de confirmation de paiement.
Outils spéciaux requis	Aucun
Exigences en matière de données d'entrée	L'acquéreur doit avoir enregistré le commerçant à l'aide du système d'administration et avoir fourni les éléments suivants : • IDEBIT_FUNDEDURL(S) • IDEBIT_NOTFUNDEDURL(S) HTTP REFERERURL(S) Des données fournies par l'outil de test pour commerçants
Stratégie d'exécution	Entamez un paiement chez le commerçant. Les deux chiffres les moins significatifs du montant en dollars doivent être égaux au numéro du cas de test. Par exemple, si vous exéutez le cas de test 13, le format du montant doit être ### ### #13,##.

Résultat attendu	Le commerçant informe le client que l'achat a été refusé. Le nom de l'émetteur et le numéro de confirmation de l'émetteur ne sont pas affichés.
Registres applicables	Registres de l'outil de test pour commerçants

Tableau 42 : Cas 23

Déroulement	Vérifier qu'un commerçant peut recevoir une redirection valide de l'émetteur qui indique que le paiement n'a pas été fait en raison de fonds non suffisants
Prérequis	Aucun
Configuration	Aucun L'émulateur de l'acquéreur n'est pas nécessaire, car le commerçant ne soumet aucune demande de confirmation de paiement.
Outils spéciaux requis	Aucun
Exigences en matière de données d'entrée	L'acquéreur doit avoir enregistré le commerçant à l'aide du système d'administration et avoir fourni les éléments suivants : • IDEBIT_FUNDEDURL(S) • IDEBIT_NOTFUNDEDURL(S) HTTP REFERERURL(S) Des données fournies par l'outil de test pour commerçants
Stratégie d'exécution	Lancer un paiement chez le commerçant pour tout montant dont les deux chiffres les moins significatifs du dollar sont 23. (C'est-à-dire, au format ### ### #23.##)
Résultat attendu	Le commerçant informe le client que l'achat a été refusé. Le nom de l'émetteur et le numéro de confirmation de l'émetteur ne sont pas affichés.
Registres applicables	Registres de l'outil de test pour commerçants

Tableau 43 : Cas 24 à 39

Déroulement	Vérifier qu'un commerçant traite de manière sûre les redirections vers l'URL pour les achats avec fonds non suffisants avec des données non valides, et traite la transaction comme non effectuée en raison de fonds non suffisants
Prérequis	Aucun
Configuration	Aucun L'émulateur de l'acquéreur n'est pas nécessaire, car le commerçant ne soumet aucune demande de confirmation de paiement.
Outils spéciaux requis	Aucun
Exigences en matière de données d'entrée	L'acquéreur doit avoir enregistré le commerçant à l'aide du système d'administration et avoir fourni les éléments suivants : • IDEBIT_FUNDEDURL(S) • IDEBIT_NOTFUNDEDURL(S) HTTP REFERERURL(S) Des données fournies par l'outil de test pour commerçants
Stratégie d'exécution	Entamez un paiement chez le commerçant. Les deux chiffres les moins significatifs du montant en dollars doivent être égaux au numéro du cas de test. Par exemple, si vous exécutez le cas de test 27, le format du montant doit être ### ### #27,##.
Résultat attendu	Le commerçant informe le client que l'achat a été refusé. Le nom de l'émetteur et le numéro de confirmation de l'émetteur ne sont pas affichés.
Registres applicables	Registres de l'outil de test pour commerçants

H.3 Valeurs des cas de tests frontaux de commerçants

Ces valeurs sont automatiquement envoyées par l'outil de test pour commerçants d'INTERAC en ligne. Elles sont fournies ici à titre de référence seulement.

Tableau 44 : Cas de test 1 et 4 – URL pour les achats avec fonds suffisants

URL de redirection	Fonds suffisants
ISSLANG	en
TRACK2	3728024906540591206=1201 0123456789XYZ
ISSCONF	CONF#123

ISSNAME	123Bank
INVOICE	(Identique à celle fournie par le commerçant)
MERCHDATA	(Identique à celle fournie par le commerçant)
VERSION	1

Tableau 45 : Cas de test 2 – URL pour les achats avec fonds suffisants

URL de redirection	Fonds suffisants
ISSLANG	en
TRACK2	5268051119993326=2912999 999999999000
ISSCONF	#\$.,-/=?@'UPdn9
ISSNAME	987Bank Éàêëï#\$.,- /=?@'Àôùûüýç
INVOICE	(Identique à celle fournie par le commerçant)
MERCHDATA	(Identique à celle fournie par le commerçant)
VERSION	1

Tableau 46 : Cas de test 3 – URL pour les achats avec fonds suffisants

URL de redirection	Fonds suffisants
ISSLANG	fr
TRACK2	453781122255=1001ABC1122 3344550000000
ISSCONF	C
ISSNAME	B
INVOICE	(Identique à celle fournie par le commerçant)
MERCHDATA	(Identique à celle fournie par le commerçant)
VERSION	123

Tableau 47 : Cas de test 5 à 22 – Champs non valides, URL pour les achats avec fonds suffisants

Cas de test	Finalité	Champ	Valeur
5	Champ manquant	IDEBIT_INVOICE	(manquant)
6	Champ manquant	IDEBIT_MERCHDATA	(manquant)
7	Champ manquant	IDEBIT_ISSLANG	(manquant)
8	Champ manquant	IDEBIT_TRACK2	(manquant)
9	Champ manquant	IDEBIT_ISSCONF	(manquant)
10	Champ manquant	IDEBIT_ISSNAME	(manquant)
11	Champ manquant	IDEBIT_VERSION	(manquant)
12	Champ manquant	IDEBIT_TRACK2, IDEBIT_ISSCONF, IDEBIT_ISSNAME	(manquant)
13	Valeur incorrecte	IDEBIT_INVOICE	XXX
14	Valeur incorrecte	IDEBIT_MERCHDATA	XXX
15	Valeur non valide	IDEBIT_ISSLANG	de
16	Valeur trop longue	IDEBIT_TRACK2	3728024906540591206=12010123456789XYZA
17	Chiffre de vérification non valide	IDEBIT_TRACK2	3728024906540591207=12010123456789XYZ
18	Champ trop long	IDEBIT_ISSCONF	Trop long confirmer
19	Caractère non valide	IDEBIT_ISSCONF	CONF<123

Cas de test	Finalité	Champ	Valeur
20	Champ trop long	IDEBIT_ISSNAME	Nom de l'émetteur très long
21	Caractère non valide	IDEBIT_ISSNAME	123<Bank
22	Valeur non valide	IDEBIT_VERSION	2

Tableau 48 : Cas de test 23 – Données valides, URL pour les achats avec fonds non suffisants

URL de redirection	Fonds insuffisants
ISSLANG	en
INVOICE	(Identique à celle fournie par le commerçant)
MERCHDATA	(Identique à celle fournie par le commerçant)
VERSION	1

Tableau 49 : Cas de test 5 à 22 – Champs non valides, URL pour les achats avec fonds suffisants

Cas de test	Finalité	Champ	Valeur
24	Champ manquant	IDEBIT_INVOICE	(manquant)
25	Champ manquant	IDEBIT_MERCHDATA	(manquant)
26	Champ manquant	IDEBIT_ISSLANG	(manquant)
27	La valeur IDEBIT_TRACK2 est présente et valide	IDEBIT_TRACK2	3728024906540591206=12010123456789XYZ

Cas de test	Finalité	Champ	Valeur
28	La valeur IDEBIT_ISSCONF est présente et valide	IDEBIT_ISSCONF	CONF#123
29	La valeur IDEBIT_ISSNAME est présente et valide	IDEBIT_ISSNAME	12Bank
30	Champ manquant	IDEBIT_VERSION	(manquant)
31	Valeur incorrecte	IDEBIT_INVOICE	XXX
32	Valeur non valide	IDEBIT_INVOICE	Données délicates </html> non valides (invalid </html> tricky data)
33	Valeur incorrecte	IDEBIT_MERCHD ATA	XXX
34	Valeur non valide	IDEBIT_MERCHD ATA	<2 000 caractères dans la catégorie hex 20-7E
35	Valeur non valide	IDEBIT_ISSLANG	de
36	Valeur IDEBIT_TRACK2 non valide présente	IDEBIT_TRACK2	INVALIDTRACK2, format incorrect et trop long
37	Valeur IDEBIT_ISSCONF non valide présente	IDEBIT_ISSCONF	Trop long confirmer
38	Valeur IDEBIT_ISSNAME non valide présente	IDEBIT_ISSNAME	Nom de l'émetteur très long

Cas de test	Finalité	Champ	Valeur
39	Valeur non valide	IDEBIT_VERSION	2

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