vindicia
Build Subscription Revenue

Vindicia® Select™ API Guide

Vindicia Select 1.0
April, 2014

April 16, 2014

1 Table of Contents

Vindicia® Select™ API Guide Preface

Vindicia Select is an on-demand billing solution, available through an object-oriented application programming interface (API), based on the Simple Object Application Protocol (SOAP). The Vindicia Select solution is accessed through a public API to the CashBox application, which is hosted and maintained on the Vindicia network.

The Vindicia Select API leverages a Service Oriented Architecture (SOA), meaning that users are not required to install application software on their network. Instead, use SOAP to communicate with the application, either through a thin client provided by Vindicia, or through a WSDL published by the Vindicia SOAP servers (e.g. http://soap.vindicia.com/1.0/Transaction.wsdl). (These SOAP servers comprise the first tier of Vindicia's network, and it is the only tier that is publicly accessible.)

This manual, the ***Vindicia Select API Guide***, lists and describes the objects available in the Vindicia Select solution, and provides pseudo-code examples.

Vindicia Select API Overview

Vindicia Select was created for large clients, selling digital goods on a recurring basis, who manage their own billing system.

Vindicia Select is designed to perform the billing only for those credit card transactions that the client has been unable to capture, and for which the client will make no further attempts to collect from the customer.

Even with robust, successful billing systems in place, there is a percentage of customers lost every month because the merchant could not collect on the bill. On a monthly subscription service, you can often double your customer lifetime value by maintaining a connection with those customers whose payment method might otherwise have failed. This is critical because these customers want to use your service, and have not actively opted out.

Vindicia Select analyzes reported data to determine whether a transaction is likely to be successfully captured by ART. Because we understand which transactions have the highest likelihood of success, there is minimal impact to your chargeback volumes or rate, and we will fight those that are submitted on your behalf.

Note:	While Vindicia Select clients do not use CashBox itself, they do have access to Vindicia's ChargeGuard services for any of their Transactions.

CashBox uses Vindicia's patent pending Advanced Retention Technology™ or "ART" to capture these transactions. Vindicia identifies which transactions for any given Vindicia Select client are eligible for ART using a variety of factors including:

- Transaction history across all CashBox clients,
- Transaction history across clients that are similar to the specific client
- The client's successful and failed Transaction activity
- Reason codes for the failed transactions
- BIN data and individual bank behaviors
- Chargeback data cross all merchants
- Transaction price point

Using Vindicia Select not only allows you to capture failed transactions, it also grants you extended lifetime from your subscribers. Instead of a subscriber dropping from your service due to a single failed transaction, Vindicia Select may capture that transaction, allowing you to continue to bill for subsequent periods, as if you had successfully captured the transaction yourself.

Process Overview

Use the `billTransactions` method to report an array of Transactions to Vindicia Select for processing.

Vindicia will run Advanced Retention Technology on the Transactions (described below), which will include Account Updater, retry logic and partial authorization.

Use `fetchBillingResults` to retrieve Transactions which have completed their Vindicia Select processing cycle. (The returned results from this call will include any new payment method information available as a result of the Account Updater process. CashBox will encrypt the card using your public key before returning it to you. If the payment method did not change, Vindicia Select will not pass any value in this field.) Vindicia Select also allows you to retrieve Chargeback information using the `fetchChargebacks` call.

Note:	To use the Vindicia Select Account Updater, you must submit a PGP public key, to encrypt any returned credit card information. Vindicia Select will use this key to encrypt credit card numbers for return in the `fetchBillingResults` call. Work with your Vindicia Client Services representative to enable Account Updater with Vindicia Select.

Client Settings

Vindicia Select offers several settings by which merchants may customize billing attempt parameters:

- **Partial Authorization Threshold (percentage):** for a partial authorization received above this threshold, CashBox will not perform additional logic to attempt to capture the full amount of the Transaction. CashBox will ***only*** capture the partial amount.
- **Full Deposit Threshold (percentage):** for a partial authorization received ***below*** this threshold, CashBox will ***not*** perform additional logic to attempt to capture the full amount of the transaction, and the transaction attempt will be treated as a failure. For a partial authorization received ***above*** this threshold and ***below*** the Partial Authorization Threshold, an attempt to capture the full amount will be made.
- **Forced Deposit:** this flag indicates whether or not Vindicia Select will attempt to capture the full amount of a transaction, even when the authorization was declined (as opposed to only partially approved).
- **ART Attempt Threshold:** defines the first attempt upon which Vindicia Select will apply ART.

Vindicia Select Objects

Each Vindicia Select object consists of data members, which fall into one of the following categories:

- Standard, built-in data types, such as integers or strings, that are common to programming languages.
- Enumerations, which are scalar types coded as standard data types, but which are restricted to a specific set of legal values.
- Data structures, which consist of multiple data members, each of which can be of different data types.
- Arrays, containing zero or more data elements, all of which must be the same data type.

Vindicia Select's methods are functions which require one or more input arguments. Methods always return a code that indicates the success or failure of the function call. In the event of failure, the code value and description will indicate why the call failed.

The Vindicia Select API is a structured language, and requires input parameters to be entered in the order shown. Parameters must be place-marked if not specified.

This guide presents Objects, their data members, and methods alphabetically, for ease of reference. Variable parameters for the methods are presented in syntactical order.

1 Working with WSDL Files

Integrate with Vindicia Select by making SOAP calls directly to Vindicia's Web services.

Because of the prevalence of Web services with SOAP as a protocol of choice for integration of disparate systems, most programming languages have built-in support for developing SOAP client-server code. A third-party plug-in or library might also be available for your language of choice. For example, Python programmers can build SOAP client-server code with the SOAPy library. Programming a SOAP client in the language of your choice usually requires access to a Web Services Description Language (WSDL) file that describes the Web service for which you wish to create a client.

Vindicia Web services consist of a set of objects and the SOAP calls that they support (CashBox SOAP API), with the calls described in a set of WSDL files. These WSDL files support document-literal SOAP calls, as defined in the World Wide Web Consortium (W3C) standards. Each WSDL file corresponds to a logical object commonly used in billing solutions. Objects (complex types) shared across all WSDL files are defined in the `Select.xsd` file that every WSDL file includes in its definition.

Each WSDL file describes a set of calls supported by the logical object. For example, the `Transaction.wsdl` file describes the calls with which you can perform activities on a transaction (a `Transaction` object) in Vindicia Select. Make an `update` call to create or update an `Account` object; or a `fetchByMerchantTransactionId()` call to retrieve a `Transaction` object by the unique ID assigned by you when you created the object.

Each WSDL file defines only one SOAP port bound to the object-specific interface (a set of methods). For example, `Select.wsdl` defines a port called `SelectPort`, which supports only the SOAP calls that relate to the `Select` object.

The ports defined in each of the WSDL files are associated with the same SOAP address (endpoint). That address is a script on Vindicia servers that receives all SOAP calls and routes them to object-specific code for further processing, depending on which objects the calls are for. For example:

```
<service name="Select">
   <port binding="tns:SelectBinding" name="SelectPort">
      <soap:address location="https://soap.vindicia.com/soap.pl" />
   </port>
</service>
```

Each WSDL file imports the `Select.xsd` schema file, which defines the data structure of all top-level and helper objects. Your client code must be able to access this schema file.

1.1 Specifying the SOAP Address

By default, the SOAP address points to Vindicia's production servers. Before going live with Vindicia Select, test your integration code in a Vindicia sandbox server. If your language-specific implementation of a SOAP client does not override the SOAP address specified by the WSDL file, you can download the WSDL files from Vindicia, save them locally, and update the SOAP address to reflect the sandbox and CashBox SOAP API version you will use.

For example, if you wish to call in to CashBox on Vindicia's `Prodtest` sandbox, update the `service` attribute in your local WSDL file as follows:

```
<service name="Select">
   <port binding="tns:SelectBinding" name="SelectPort">
      <soap:address location="https://soap.prodtest.sj.vindicia.com/
               soap.pl" />
   </port>
</service>
```

1.2 Setting up your SOAP Environment

Before developing client code with the Vindicia Select WSDL files:

1. Download the WSDL files and the schema file from the Vindicia servers.

   For CashBox API production releases, download from the following sites:

   - **WSDL file:** `https://soap.vindicia.com/`***version***`/`***object***`.wsdl`, for example, `https://soap.vindicia.com/1.0/Select.wsdl`
   - **Schema file:** `https://soap.vindicia.com/`***version***`/Select.xsd`

   For CashBox API nonproduction releases that are in Vindicia's staging servers for testing prerelease client regression, download from the following sites:

   - **WSDL file:** `https://soap.staging.sj.vindicia.com/`***version***`/`***object***`.wsdl`
   - **Schema file:** `https://soap.staging.sj.vindicia.com/`***version***`/Select.xsd`

2. **Optional.** Update the SOAP endpoint address to reflect the server to which to send your SOAP calls, assuming that you cannot programmatically do so in your client code.

3. Generate local stub or proxy objects with language-specific tools. For example:

   - If you program in Java and are using the Apache Axis library to work with SOAP, generate local stub objects with the utility `WSDL2Java`.
   - If you program in .Net with C#, import the WSDL files into your project to automatically generate local stub objects.
   - If you program in another language, such as Python, for which no appropriate tools are available, consult the language- or package-specific SOAP documentation on how client code can make the SOAP calls as described in the WSDL files.

4. Ensure that your SOAP library supports making SOAP calls to external servers through HTTPS.

   For security, Vindicia does not support HTTP-based SOAP calls. You might need to install additional SSL-specific libraries on your system.

1.3 Tips for Developing SOAP Clients

Remember:

- In most SOAP libraries, you can set a timeout value for the SOAP calls that you make from the client to the server. Ensure that the value is globally configurable. You may wish to change the value to fine-tune it, depending on the amount of data you will be sending or receiving from the Vindicia servers.
- Every SOAP call you make to CashBox requires that you pass an `Authentication` object, which contains the SOAP login and password assigned to you by Vindicia. Make those credentials globally configurable. When you switch to production, you must log in with credentials that differ from those used in testing against Vindicia's sandboxes.
- You might also want to make the Vindicia server, to which your client code points globally, configurable. This can simplify the process of switching from testing to production.
- Log all calls made with the CashBox client library. At a minimum, log the following:
  - Timestamps
  - Classes and methods
  - The Vindicia `Return` data structure (all three fields)
  - SOAP envelopes that are sent to and received from Vindicia servers may be used to debug data-related errors. Most SOAP libraries allow you to add an option to log these envelopes.

2 The `Chargeback` Object

A chargeback is initiated by a customer to reverse a specific Transaction charge on their billing statement. Work with the `ChargeBack` object when you subscribe to Vindicia's ChargeGuard service to dispute chargebacks on your behalf.

The `ChargeBack` object holds information about a chargeback against a specific Transaction. Its status will change as Vindicia takes steps to dispute a chargeback on your behalf.

2.1 Chargeback Data Members

The ChargeBack object encapsulates information on a chargeback, including the amount, date, reference number, and status.

The following table lists and describes the data members of the ChargeBack object.

Table 2-1 Chargeback Object Data Members

Data Members	Data Type	Description
amount	decimal	**Required.** This chargeback's settlement amount, which usually matches the amount of the original Transaction. In some cases, customers charge back only part of a Transaction. **Note:** Given exchange-rate fluctuations, transactions across currencies might be charged back at amounts that differ from the original amounts.
caseNumber	string	Your bank's case number for this Chargeback object, if any.
currency	string	The ISO 4217 currency code (see www.xe.com/iso4217.htm) of this Chargeback object. This currency applies to the settlement amount. (See the amount attribute)
merchantNumber	string	Your bank's merchant number, which identifies you as the merchant.
merchantTransactionId	string	Your unique identifier for the transaction associated with this Chargeback object.This ID must match the order number you used when processing the transaction with your payment processor.
merchantTransactionTimestamp	dateTime	A timestamp that specifies the date and time when the original transaction occurred.
merchantUserId	string	Your unique identifier for the account of the customer who conducted the original transaction.
nameValues	NameValuePair[]	**Optional.** An array of name–value pairs for the Chargeback object. See Section 3: The NameValuePair Object.
note	string	Notes on the Chargeback object. (Vindicia personnel might make entries here during the dispute process.)
postedTimestamp	dateTime	**Required.** A timestamp that specifies the date and time the chargeback was posted in the Vindicia database. The difference in time between the chargeback, and this posted timestamp, will depend on the frequency at which Vindicia downloads chargebacks from your bank or payment processor.

Table 2-1 Chargeback Object Data Members

Data Members	Data Type	Description
`presentmentAmount`	decimal	The amount charged back (in the presentment currency), which usually matches the amount of the original transaction. Specify this attribute if the original transaction was processed with Chase Paymentech in a currency other than USD.
`presentmentCurrency`	string	The ISO 4217 currency code (see www.xe.com/iso4217.htm) of this transaction at presentment.
`processorReceivedTimestamp`	`dateTime`	**Required.** A timestamp that specifies the date and time when your bank received the chargeback from the customer.
`reasonCode`	string	**Required.** The reason code reported by your bank for this `Chargeback` object. Reason codes vary from bank to bank.
`referenceNumber`	string	Your bank's reference number for this `Chargeback` object, if any.
`status`	`ChargebackStatus`	**Required.** The current chargeback status in ChargeGuard. A chargeback goes through a life cycle as Vindicia disputes the chargeback on your behalf. See Table 2-2: ChargebackStatus Values.
`statusChangedTimestamp`	`dateTime`	A timestamp that specifies the date and time for the last status change.
`VID`	string	Vindicia's Globally Unique Identifier (GUID) for this object. When creating a new `Chargeback` object, leave this field blank; it will be automatically populated by CashBox.

`ChargebackStatus` Values

Table 2-2 `ChargebackStatus` Values

Value	Description
`Challenged`	Vindicia has submitted rebuttal documents to your payment processor to dispute this chargeback.
`Duplicate`	A duplicate chargeback has either been manually entered or received by Vindicia from the payment processor. Another chargeback in the queue exists with exactly the same information but is not marked duplicate.
`Expired`	The related documents or transaction details you reported were received too late by Vindicia to dispute this chargeback.
`Incomplete`	Vindicia has received chargeback information from the payment processor but does not have the original transaction details from you.
`Legitimate`	A valid chargeback because the original transaction was truly fraudulent. Vindicia does not represent or dispute legitimate transactions.
`Lost`	Vindicia challenged this chargeback but lost the case.
`New`	The first chargeback received by Vindicia, which is in the process of deciding how to pursue on your behalf.
`NewSecondChargeback`	A second chargeback has been received against a transaction that was initially charged back, disputed, and won.
`Pass`	Even though all the documentation is available, Vindicia will not dispute this chargeback because of one or more of the following reasons: • The chargeback is less than US$5. • Not enough evidence exists for a dispute. • Regulations do not allow Vindicia to respond. • Vindicia does not recommend taking the dispute to arbitration.
`Retrieval`	An incoming retrieval or ticket request.
`Responded`	Vindicia has responded to the retrieval or ticket request.
`Represented`	*As a result of Vindicia's intervention, the chargeback was reversed in your favor. However, the customer or issuing bank is continuing the dispute by issuing a second chargeback.* *(This status is not in use.)*
`Won`	Vindicia challenged this chargeback, which has been reversed in your favor.

3 The `NameValuePair` Object

The `NameValuePair` object is used to hold attributes not otherwise supported in a Vindicia Select object, and may be used to store data for your own, internal tracking purposes.

3.1 `NameValuePair` Data Members

The following table lists and describes the data members of the `NameValuePair` object.

Table 3-1 `NameValue` Object Data Members

Data Members	Data Type	Description
`name`	string	**Required.** The `name` for the name/value pair.
`value`	string	**Required.** The `value` for the name/value pair.

4 The `Return` Object

All methods in the Vindicia Select API return a `Return` object, which contains the return codes for the call.

The `Return` object contains three data members:

- `returnCode`: This data member contains a value that corresponds to a standard HTTP return code. For values of 400 or higher, assume that your call failed. The failure could be due to several reasons, such as an authentication failure or a CashBox failure to find any objects that match your input. See Table 4-1: Vindicia Select Return Codes for a list of the most common return codes.
- `returnString`: If `returnCode` indicates an error condition (a non-`200` return code), your application can check `returnString` for further information. Use the Vindicia Select API to generate a log of `returnString`, to help you debug your application in the development and production phases.

  (**Note:** The `returnString` is not saved in the Vindicia Select database, and has no size limit; however, the meaningful data is usually located within the first 512 bytes.)
- `soapId`: This ID is returned for certain calls to Vindicia, especially those made to submit a batch of data (for example, a batch of transactions or chargebacks) for ChargeGuard processing. This ID helps Vindicia track your batched data in Vindicia's system and, if the ID is available, you should log it in your application. If an incident arises that requires troubleshooting by Vindicia, a Vindicia representative might ask you for this ID to determine the status of your data.

  (**Note:** The `soapID` is a 40 byte string.)

Some return strings contain information specific to the call for which the return was generated. In some cases, these will take the format:

`Unable to load Transaction by merchantId` ***input-ID:*** `No match.`

where ***input-ID*** specifies the object or call to which the return error applies.

In some cases, these will take the format:

`Unable to load Transaction by merchantId` ***input-ID: error-description.***

where ***error-description*** more specifically explains the cause of the error. In both cases, variable text is displayed in bold-italic.

The following table lists and describes the most common return codes. If a method returns different return codes, they are listed with the method.

Table 4-1 Vindicia Select Return Codes

Return Code	Description
`200`	The call succeeded.
`400`	Your call failed, which could be due to an authentication failure or a CashBox failure to find any objects that match your input. 400 may also be one of the following: • Billing has already been attempted for Transaction ID ***merchantTransactionId.*** • Failed to deserialize Transaction. • Invalid Arguments - No transaction object.
`403`	The Vindicia server cannot authenticate your request.
`404`	One of the following: • Unable to load transaction: no match for `merchantTransactionId` ***merchantTransactionId***. • Unable to load transaction: no match for `VID` ***vid***.
`405`	Unable to save transaction.
`500`	The Vindicia server encountered an internal error. That error could occur for various reasons, the most common being an incorrectly populated input object, especially when you are making the call from a client library whose language does not support strict data-type checking. For resolution, especially during the development phase, contact Vindicia Technical Support.
`503`	A Vindicia back-end service, such as a database, is unavailable. Retry your call later.

5 The `Transaction` Object

The `Transaction` object encapsulates information about a financial transaction processed through your payment provider.

The `Transaction` object also includes the current status of the Transaction in the `TransactionStatusType` enum. This value may be used to track your Transaction through the processing sequence.

When reporting a Transaction to Vindicia for ChargeGuard, be sure to include the latest or final status information inside the `Transaction` object, such as information on whether your payment processor has approved the transaction and the reason code returned by the processor. The status cycle of a `Transaction` object and the reason codes vary, depending on the Transaction's payment method.

When Vindicia downloads chargebacks from your payment processor for ChargeGuard, it matches them to your transactions in its database. If you have not yet reported the Transaction with which the chargeback is associated, CashBox creates a stub `Transaction` object in its database using the `Transaction` information in the chargeback that was downloaded. Vindicia Select then fills the stub when the corresponding Transaction is reported.

5.1 `Transaction` Data Members

The following table lists and describes the data members of the `Transaction` object.

Note: These data members are both input for the `billTransactions` call, and returned with the `fetchBillingResults` call. When using the `billTransactions` method to report failed transactions to Vindicia Select, populate these data members with the latest information you have. When Vindicia Select returns a processed Transaction object, it will populate these data members with their updated values.

For example: You may call `billTransactions`, with a `Transaction` object with `status` = `Failed`. When Select has completed processing, it will return the `Transaction` object with `Status` = `Captured`, `Failed` or `Refunded`.

For fields such as `authCode`, `avsCode`, and `timeStamp`, input the latest result from your Payment Processor. The `fetchBillingResults` call will return the final result from the Vindicia Select process.

Table 5-1 `Transaction` Object Data Members

Data Member	Data Type	Description
`accountHolderName`	string	255 or fewer characters, this is the name of the account holder for the credit card.
`affiliateId`	string	Your unique identifier for the partner or affiliate who directed this Transaction to you. To implement affiliate tracking, enter this attribute when reporting transactions to Vindicia Select.
`affiliateSubId`	string	Your ID for the partner or sub-affiliate who directed this Transaction to you. To implement sub-affiliate tracking, enter this attribute when reporting transactions to Vindicia Select.
`amount`	decimal	**Required**. The total cost of the Transaction. This must be a positive number.
`authCode`	string	**Required.** The response code returned by your payment processor for this Transaction.
`avsCode`	string	The AVS code returned by the payment processor for this Transaction. To receive this code, enable AVS with the payment processor. The AVS response code returned by your payment processor. When reporting a Transaction, if you receive the AVS code as a string along with its authorization code (ie Y:2341234234 or Y-2341234234), simply copy it into this field. If you receive the AVS code separate from its authorization code, concatenate them with a colon as a separator, as shown in the previous example.

Table 5-1 `Transaction` Object Data Members *(Continued)*

Data Member	Data Type	Description
`billingAddressCity`	string	The city of the customer's address.
`billingAddressCountry`	string	The customer address country, listed as the ISO-3166-1 two-letter code (for example, US, GB, or FR).
`billingAddressCounty`	string	The county of the customer's address.
`billingAddressDistrict`	string	The state, province, or district of the customer's address.
`billingAddressLine1`	string	The first address line.
`billingAddressLine2`	string	The second, additional address line.
`billingAddressLine3`	string	The third, additional address line.
`billingAddressPostalCode`	string	The postal code of the customer's address. **Note:** This field will accept 9-digit input.
`billingInterval`	enum	Indicates the interval at which billing occurs: hourly, daily, weekly, monthly, quarterly, or annually.
`billingStatement-Identifier`	string	Optional string that should be displayed on a customer's billing statement when they are charged. Support for this string and its allowable format depends on your payment processor. (This string may be used to override the default set by your payment processor. Work with Vindicia Client Services if you wish to use this field.) **Note:** This value and its format are constrained by your payment processor.
`creditCardAccount`	string	The credit card account number used when billing the Transaction. When calling the `reportTransactions` method, you need only provide a masked account that provides the 6 digit BIN, and the last 4 digits (4111111111111111 would be sent as 411111xxxxxx1111). When providing a masked account enter the SHA1 hash in the `creditCardAccountHash` field. When calling `billTransactions`, the full account number is required, and Vindicia will calculate the SHA1 hash. If you use tokenization with your processor this field is optional.
`creditCardAccountHash`	string	A SHA1 hash of the full account number. Any non-numeric characters should be removed prior to hashing. If the account number is provided, this may be left blank and the hash will be calculated by Vindicia. For reference the HA1 hash of 4111111111111111 is 68bfb396f35af3876fc509665b3dc23a0930aab1. If you use tokenization with your processor, this field is optional.

Table 5-1 `Transaction` Object Data Members *(Continued)*

Data Member	Data Type	Description
`creditCardAccountUpdated`	boolean	Returned by Vindicia Select, this data member indicates whether or not the Credit Card has been updated. If Vindicia was able to retrieve an updated credit card account through Account Updater, this field will be set to `true`, and the new account information will be returned in the `creditCard-Account` and `creditCardExpirationDate` fields. To enable this feature, you must work with Vindicia Client Services, and provide a valid PGP public key.
`creditCardExpirationDate`	string	**Required.** The `CreditCard` expiration date in YYYYMM format, where YYYY is the four-digit year and MM is the two-digit month. For example, the string for July 2007 is 200707. If you use tokenization with your processor this field is optional. Note: No validation is performed to check that this date is in the future.
`currency`	string	**Required.** The ISO 4217 currency code (see www.xe.com/iso4217.htm) for the transaction.
`customerId`	string	**Required.** Your unique identifier for the customer associated with this transaction.
`cvnCode`	string	The CVN response code returned by your payment processor. When reporting a Transaction, if you receive the CVN code as a string along with its authorization code (ie M:2341234234 or M-2341234234), simply copy it into this field. If you receive the CVN code separate from its authorization code, concatenate them with a colon as a separator, as shown in the previous example.
`divisionNumber`	string	**Required.** The number of your division or group with your payment processor for this transaction. Chase Paymentech refers to this number as the Division ID; Litle calls it the Report Group; MeS calls it the Profile ID. CashBox ChargeGuard will use this value to match the Transaction to the appropriate chargeback from the payment processor.
`merchantTransactionId`	string	**Required.** Your unique identifier for this Transaction. If you are reporting this transaction to Vindicia for ChargeGuard only, ensure that this ID matches the order number you sent to the payment processor. That way, ChargeGuard can match this transaction with a chargeback received for this transaction from the processor.
`nameValues`	`NameValuePair[]`	An array of name–value pairs, which are used to hold attribute values not otherwise supported by this object. These may be used to store custom data for your own, internal tracking purposes. See Section 3: The NameValuePair Object.

Table 5-1 `Transaction` Object Data Members *(Continued)*

Data Member	Data Type	Description
`paymentMethodId`	string	**Required.** Your ID associated with the Payment Method for the Transaction.
`paymentMethodIsTokenized`	boolean	Use this flag to indicate when the `paymentMethodId` should be used rather than the `creditCardAccount`. If `paymentMethodIsTokenized` is `true`, the creditCard-Account and `creditCardExpirationDate` fields are optional; otherwise, they are required.
`previousBillingCount`	int	The number of times your customer has been successfully billed.
`previousBillingDate`	dateTime	The date of the last successful billing associated with the subscription.
`selectTransactionId`	string	Returned by Vindicia Select, this is the unique ID, assigned by Vindicia Select, used when billing the Transaction after receiving a `billTransactions` call.
`status`	`TransactionStatusType`	**Required.** Defines the Transaction status upon reporting to Vindicia. This is a 255 byte string. For a list of possible values, see Table 5-2: TransactionStatusType Values.
`subscriptionId`	string	**Required.** Your unique identifier for the subscription associated with this Transaction.
`subscriptionStartDate`	dateTime	A timestamp that indicates the date of the first successful billing associated with the subscription. If unspecified, the value defaults to today and the current time.
`timestamp`	`dateTime`	**Required.** A timestamp that specifies the date and time of when this Transaction occurred. Be certain to include this attribute in reported Transactions, or it will default to the current time.
`VID`	string	Vindicia's Globally Unique Identifier (GUID) for this object. This field is created by Vindicia, and is provided to serve as a unique ID for every `Transaction`. In general, you may use your `merchantTransactionId` instead of the VID. Do not specify this value when creating a new Transaction object. Vindicia will populate this field, and return it in any subsequent `fetch` calls. This is a 40-character freeform string.

`TransactionStatusType` Values

Defines the Transaction status upon reporting to Vindicia.

Table 5-2 `TransactionStatusType` Values

Data Members	Data Type	Description
`BillingNot-Attempted`	string	The Transaction was not processed by Vindicia ART.
`Cancelled`	string	This status will be returned if you call refund on a Select Transaction before CashBox has processed it.
`Captured`	string	A captured status, which indicates that the payment processor has charged the customer. A captured transaction means that the payment processor has accepted it and that money transfer will take place. For most successful transactions, this is the terminal status.
`Failed`	string	The Transaction did not authorize or capture successfully.
`Refunded`	string	The Transaction has been refunded. This status will be returned both if refundTransaction is called, and if CashBox proactively refunds a transaction to avoid a chargeback.

`TransactionValidationResponse` Values

Defines the Transaction status upon reporting to Vindicia.

> **Note:** The `TransactionValidationResponse` array will return only `Transactions` with errors; it will not include Transactions which are processed error-free. (An empty array indicates an error-free call.) Use this array to define a, actionable list of Transactions that require further investigation.

These errors will not be significant enough to invalidate the request (properly-formed, etc.), but will make the Transaction unable to be processed for the reason indicated by the validation message.

Table 5-3 `TransactionValidationResponse` Values

Data Members	Data Type	Description
`code`	int	A numeric code indicating the type of issue that was encountered. For more information, see Table 4-1: Vindicia Select Return Codes.
`description`	string	A description of the issue encountered. **Note:** The `description` is not saved in the Vindicia Select database, and has no size limit; however, the meaningful data is usually located within the first 512 bytes.
`merchantTransactionId`	string	Your unique ID for the submitted Transaction.

6 Vindicia Select Methods

The following table summarizes the methods for Vindicia Select.

Table 6-1 Vindicia Select Methods

Method	Description
billTransactions	Reports an array of Transactions to Vindicia Select to begin the Vindicia Select billing cycle.
fetchBillingResults	Returns the Vindicia Select Billing cycle results.
fetchByMerchantTransactionId	Returns the Transaction specified by the Merchant Transaction ID.
fetchChargebacks	Returns an array of Chargeback objects received from your payment processor.
refundTransactions	Sends an array of Transactions to Vindicia Select, for which a refund should be issued. Use this method to reverse any Transactions completed inhouse, after having been reported to Vindicia Select.
reportTransactions	Reports data to Vindicia Select for use in fighting Chargebacks resulting from Transactions not processed through Vindicia Select.

billTransactions

Submits Transactions to Vindicia Select for processing. The results of subsequent billing attempts will be returned through the `fetchBillingResults` method. If a submitted item fails to validate, it will be returned in the ***response*** array, with information indicating the validation issue.

If for any reason you wish to withdraw a `Transaction` from processing after calling this method, use the `refundTransactions` method.

Note:	For best results, submit 100 Transactions with your first call to `billTransactions`, then adjust the number submitted for greatest efficiency.

Input

transactions: an array of `Transaction` objects to submit for billing. (If you have attempted to process a single billing multiple times, submit only your final attempt to Vindicia Select. Do not submit multiple `Transaction` objects for a single billing attempt.)

Note:	Do not populate the VID field in any new Transaction submitted with this method. Vindicia will populate this data member, and return it to you in any relevant `fetch` calls.

Output

return: an object of type `Return` that indicates the success or failure of the call.

response: an object of type `TransactionValidationResponse`, which includes the ID of the Transaction returning an error.

Note: An empty `TransactionValidationResponse` object indicates that the call succeeded without error. For more information, see TransactionValidationResponse Values.

Returns

This method returns the codes listed in Table 4-1: Vindicia Select Return Codes.

Example

```
$tx = new Transaction();
$tx->setTimestamp('2012-09-11T22:34:32.265Z');
$tx->setAmount('9.90');
$tx->setCurrency('USD');
$tx->setStatus('Failed');
$tx->setDivisionNumber('54321');

// Merchant Transaction ID must be unique for each new
// transaction you wish Vindicia to process.
$tx->setMerchantTransactionId('TxPrfx123');

// Merchant Subscription ID should be unique for each new
// transaction-subscription you wish Vindicia to process.
$tx->setSubscriptionId('AbPrfx456');
$tx->setBillingFrequency('Yearly');
$tx->setPreviousBillingDate('2011-09-11');
$tx->setPreviousBillingCount('24');

// Merchant Customer ID should be unique for each unique Customer.
$tx->setCustomerId('McPrfx789');
$tx->setPaymentMethodId('PmPrfx012');
$tx->setPaymentMethodIsTokenized(false);
$tx->setCreditCardAccount('4111111111111111');
$tx->setCreditCardExpirationDate('20121212');
$tx->setAccountHolderName('Calvino Hobbes');
$tx->setBillingAddressLine1('Suite 200');
$tx->setBillingAddressLine2('23 George Street');
$tx->setBillingAddressCity('Larkspur');
$tx->setBillingAddressDistrict('CA');
$tx->setBillingAddressPostalCode('94964');
$tx->setBillingAddressCountry('US');

$nv1 = new NameValuePair();
$nv1->setName('CriminalOffense');
$nv1->setValue('Wire Fraud');
$nv2 = new NameValuePair();
$nv2->setName('Sentence');
$nv2->setValue('25 years');

$tx->setNameValues(array{$nv1, $nv2});
$tx->setAuthCode('110');
$select = new select();
$ret = $select->billTransactions(array{$tx});

if($ret['returnCode'] == 200)
{
    $failedTransactions = $ret['response'];
    if($failedTransactions && (sizeof($failedTransactions) > 0))
    {
        foreach ($failedTransactions as $transaction)
        {
            //Evaluate Transaction failure response here.
        }
    }
}
else
{
    //Handle error condition.
}
```

fetchBillingResults

Returns an array of completed Transactions. Only those Transactions which have reached a terminal status (Captured, Cancelled, Refunded, or Failed), or which have been refunded since the input ***timestamps*** will be returned for this call.

> **Note:** Vindicia Select requires that input parameters be entered in the order shown, and placemarked with `null` if not specified.

Paging is supported using ***page*** and ***pageSize***. Continue calling the function until the length of the resulting array is 0.

> **Note:** To use the Vindicia Select Account Updater, you must submit a PGP public key, to encrypt any returned credit card information. Vindicia Select will use this key to encrypt credit card numbers for return in the `fetchBillingResults` call.
>
> Work with your Vindicia Client Services representative to enable Account Updater with Vindicia Select.

Input

timestamp: the starting timestamp (lower limit) for the range of (completed) `Transactions` you wish to retrieve.

endTimestamp: the ending timestamp (upper limit) for the range of (completed) `Transactions` you wish to retrieve.

page: the page number, starting at 0, for which to return the results. For example, if the total number of results is 85 and ***pageSize*** is 10:

- Specifying 0 for ***page*** gets the results from 1 through 10.
- Specifying 2 for ***page*** gets the results from 21 through 30.

pageSize: the number of records to display per page per call. This value must be greater than 0.

Output

return: an object of type `Return` that indicates the success or failure of the call.

transactions: an array of `Transaction` objects that have reached a terminal status during the time period specified with ***timestamp*** and ***endTimestamp***.

> **Note:** These `Transaction` objects will list the results of the Vindicia Select billing attempt,, with the `merchantTransactionId` to identify the submitted `Transaction`, and the `selectTransactionId` to identify the Vindicia Select `Transaction` returned.

Returns

This method returns the codes listed in Table 4-1: Vindicia Select Return Codes.

Example

```
$select = new select();
$page = 0;
$pageSize = 50;

// Assume we have a function available to us that gives us
// the timestamp for the last time we ran this call:
$since = getLastCallTime();

do {
    $ret = $select->fetchBillingResults($since, null, $page, $pageSize);
    $count = 0;
    if($ret['returnCode'] == 200)
    {
        $fetchedTransactions = $ret['transactions'];
        if($fetchedTransactions != null)
        {
            $count = sizeof($fetchedTransactions)
            foreach ($fetchedTransactions as $transaction)
            {
                 //Process a fetched transaction here...
            }
            $page++;
        }
    }
    else
    {
    //Handle error condition.
    }
} while ($count == $pageSize);
```

fetchByMerchantTransactionId

Returns the `Transaction` specified by the input `merchantTransactionId`.

Use this method to fetch specific Transactions, to address issues within your working process. Use `fetchBillingResults` to retrieve all results in a single call.

Input

merchantTransactionId: the `merchantTransactionId` for the `Transaction` you wish to fetch.

Output

return: an object of type `Return` that indicates the success or failure of the call.

transaction: the `Transaction` with the input `merchantTransactionId`.

Returns

This method returns the codes listed in Table 4-1: Vindicia Select Return Codes.

Example

```
$select = new select();

//Assume we have a merchantTransactionId 'TxPrfx123'
//that we want to retrieve information on.
$merchantTxId = 'TxPrfx123';

$ret = $select->fetchByMerchantTransactionId($merchantTxId);
if($ret['returnCode'] == 200)
{
   $fetchedTransaction = $ret['transaction'];
   //Process fetched transaction here...
}
else
{
   //Handle error condition.
}
```

`fetchChargebacks`

Returns a list of `Chargeback` objects that have changed status since the input ***timestamps***. Paging is supported using ***page*** and ***pageSize***. Continue calling the function until the length of the resulting array is 0.

Note: Vindicia Select requires that input parameters be entered in the order shown, and place-marked with `null` if not specified.

Input

timestamp: the starting timestamp (lower limit) for the range of `Chargebacks` you wish to retrieve.

endTimestamp: the ending timestamp (upper limit) for the range of `Chargebacks` you wish to retrieve.

page: the page number, starting at 0, for which to return the results. For example, if the total number of results is 85 and ***pageSize*** is 10:

- Specifying 0 for ***page*** gets the results from 1 through 10.
- Specifying 2 for ***page*** gets the results from 21 through 30.

pageSize: the number of records to display per page per call. This value must be greater than 0.

Output

return: an object of type `Return` that indicates the success or failure of the call.

chargebacks: an array of `Chargeback` objects that have been updated during the time period specified with ***timestamp*** and ***endTimestamp***.

Returns

This method returns the codes listed in Table 4-1: Vindicia Select Return Codes.

Example

```
$select = new select();
$page = 0;
$pageSize = 50;

// Assume we have a function available to us that gives us
// the timestamp for the last time we ran this call:
$since = getLastCallTime();

do {
    $ret = $select->fetchChargebacks($since, null, $page, $pageSize);
    $count = 0;
    if($ret['returnCode'] == 200)
    {
        $fetchedChargebacks = $ret['chargebacks'];
        if($fetchedChargebacks != null)
        {
            $count = sizeof($fetchedChargebacks)
            foreach ($fetchedChargebacks as $chargeback)
            {
                 //Process a fetched chargeback here...
            }
            $page++;
        }
    }
    else
    {
        //Handle error condition.
    }
} while ($count == $pageSize);
```

refundTransactions

Instructs Vindicia Select to discontinue further attempts to bill a customer (which may occur if a customer has cancelled their subscription, or paid using a different Payment method than is included in the original `Transaction`). If Vindicia Select has not yet successfully billed this customer, this method cancels the attempt. If CashBox receives this request after having successfully billed, then CashBox will issue a refund against the Transaction to eliminate the unwanted billing. Confirmation of a refund will be returned in the `fetchBillingResults` method.

If a submitted item fails to validate, it will be returned in the ***response*** array along with information indicating the validation issue.

Input

transactions: an array of `Transaction` objects to submit for billing.

Output

return: an object of type `Return` that indicates the success or failure of the call.

response: an object of type `TransactionValidationResponse`, which includes the ID of the Transaction returning an error.

> **Note:** An empty `TransactionValidationResponse` object indicates that the call succeeded without error. For more information, see TransactionValidationResponse Values.

Returns

This method returns the codes listed in Table 4-1: Vindicia Select Return Codes.

Example

```
//Given existing (i.e. already billed)
//merchantTransactionIds TxPrfx123, TxPrfx234, and TxPrfx345:

$select = new select();
$ret = $select->refundTransactions(array{'TxPrfx123', 'TxPrfx234',
       'TxPrfx345'});

if($ret['returnCode'] == 200)
{
    $failedTransactions = $ret['response'];
    if($failedTransactions && (sizeof($failedTransactions) > 0))
    {
        foreach ($failedTransactions as $transaction)
        {
            //Evaluate Transaction failure response here
        }
    }
}
else
{
    //Handle error condition.
}
```

reportTransactions

Use this method to report data to Vindicia Select for use in fighting Chargebacks resulting from Transactions not processed through Vindicia Select.

Input

transactions: an array of `Transaction` objects to submit for billing.

Output

return: an object of type `Return` that indicates the success or failure of the call.

response: an object of type `TransactionValidationResponse`, which includes the ID of the Transaction returning an error.

Note: An empty `TransactionValidationResponse` object indicates that the call succeeded without error. For more information, see TransactionValidationResponse Values.

Returns

This method returns the codes listed in Table 4-1: Vindicia Select Return Codes.

Example

```
$tx = new Transaction();
// Populate the Transaction object as illustrated above
// for credit card based billTransactions call.

$select = new select();
$ret = $select->reportTransactions(array{$tx});

if($ret['returnCode'] == 200)
{
    $failedTransactions = $ret['response'];
    if($failedTransactions && (sizeof($failedTransactions) > 0))
    {
        foreach ($failedTransactions as $transaction)
        {
            //Evaluate Transaction failure response here.
        }
    }
}
else
{
    //handle error condition
}
```