vindicia®
An Amdocs Company

Vindicia® Retain™ API Reference Guide

May, 2020

Release: 27.00

Contents

Vindicia® Retain™ API Guide Preface

Vindicia Retain 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 Retain solution is accessed through a public API to the Vindicia Subscribe application, which is hosted and maintained on the Vindicia network.

The Vindicia Retain API leverages a Service Oriented Architecture (SOA), meaning that users are not required to install application software on their network. Instead, they 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 (http://soap.vindicia.com/1.0/Transaction.wsdl).

The SOAP documentation can be found here: http://developer.vindicia.com/docs/soap/index.html?ver=1.1

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

Vindicia Retain API Overview

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

Vindicia Retain 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 Retain 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 Retain clients do not use Vindicia Subscribe itself, they do have access to Vindicia's ChargeGuard services for any of their Transactions.*

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

- Transaction history across all Vindicia Subscribe 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 Retain 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 Retain 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 Retain 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 Retain processing cycle. (The returned results from this call will include any new payment method information available as a result of the Account Updater process. Vindicia Subscribe will encrypt the card using your public key before returning it to you. If the payment method did not change, Vindicia Retain will not pass any value in this field.) Vindicia Retain also allows you to retrieve Chargeback information using the `fetchChargebacks` call.

> *Note:*
>
> *To use the Vindicia Retain Account Updater, you must submit a PGP public key and encrypt any returned credit card information. Vindicia Retain will use this key to encrypt credit card numbers for return in `thefetchBillingResults` call. You must also have completed an Attestation of Compliance (AOC) as a declaration of the results of the service provider's assessment with the Payment Card Industry Data Security Standards (PCI DSS).*
>
> *Work with your Vindicia Client Services representative to enable Account Updater with Vindicia Retain.*

Client Settings

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

- Partial Authorization Threshold (percentage): for a partial authorization received above this threshold, Vindicia Subscribe will not perform additional logic to attempt to capture the full amount of the Transaction. Vindicia Subscribe will only capture the partial amount.
- Full Deposit Threshold (percentage): for a partial authorization received below this threshold, Vindicia Subscribe 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 Retain 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 Retain will apply ART.

Vindicia Retain Objects

Each Vindicia Retain 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 Retain'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 Retain 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 Retain 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 (Vindicia Subscribe 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 Retain. 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 Retain, 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 Vindicia Subscribe SOAP API version you will use.

For example, if you wish to call in to Vindicia Subscribe 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 Retain WSDL files:

Steps

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

   For Vindicia Subscribe 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 Vindicia Subscribe 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 Vindicia Subscribe 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 Vindicia Subscribe 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.

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 https://www.xe.com/iso4217.php) 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

Data Members	Data Type	Description
		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 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.
`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 https://www.xe.com/iso4217.php) of this transaction

Data Members	Data Type	Description
		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 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 Vindicia Subscribe.

3 The NameValuePair Object

The `NameValuePair` object is used to hold attributes not otherwise supported in a Vindicia Retain 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.

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 Retain API return a `Return` object, which contains the return codes for the call.

4.1 Return Data Members

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 Vindicia Subscribe failure to find any objects that match your input. See Retain 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 Retain API to generate a log of `returnString`, to help you debug your application in the development and production phases.

  (The `returnString` is not saved in the Vindicia Retain 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.

  (The `soapID` is a 40-byte string.)

4.2 Retain Return Codes

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 3.

Retain Return Codes

Return Code	Description
`200`	The call succeeded.
`400`	Your call failed, which could be due to an authentication failure or a Vindicia Subscribe 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 provides information about the failed transactions that are conveyed to Vindicia Retain for recovery. This information is passed to Retain via the `billTransactions` method.

When reporting a transaction to Vindicia for ChargeGuard, be sure to include the latest or final status information from the transaction object, such as whether the payment processor has approved the transaction and the reason code returned. The reason code values vary, depending on the payment method.

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

5.1 Transaction Data Members

The table below describes the `Transaction` object data members.

> *Note:*
>
> *These data members are both input for the `billTransactions` call, and are returned with the `fetchBillingResults` call.*
>
> *When invoking `billTransactions` to report failed transactions to Retain, populate the `Transaction` object with the most current information available.*
>
> *When Retain returns a processed `Transaction` object, it includes updated data member values. For example: `billTransactions` is called with a `Transaction` object with `status = Failed`. After Retain completes processing, it returns a `Transaction` object with the following status values: `Status = Captured`, `Failed` or `Refunded`. `fetchBillingResults` returns the final status of Retain processing. Note that the fields `authCode`, `avsCode`, and `timeStamp` contain responses from the payment processor.*

Table 4.

`Transaction` Object Data Members

Data Member	Data Type	Description
`accountHolderName`	string	255 or fewer characters, this is the name of the account holder.
`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 Retain.
`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 Retain.
`amount`	decimal	**Required**. The amount of the transaction; must be a positive number.
`authCode`	string	**Required** when calling `billTransactions`: response code returned by the payment processor for the failed transaction. When provided by `fetchBillingResults`, the `authCode` of the payment processor.
`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 (i.e. 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.
`billingAddressCity`	string	The city of the customer address.
`billingAddressCountry`	string	The customer address country, listed as the ISO-3166-1 two-letter code (for example, US, GB,

Data Member	Data Type	Description
		or FR).
`billingAddressCounty`	string	The county of the customer address.
`billingAddressDistrict`	string	The state, province, or district of the customer address.
`billingAddressLine1`	string	Customer address, first line.
`billingAddressLine2`	string	Customer address, second line.
`billingAddressLine3`	string	Customer address, third line.
`billingAddressPostalCode`	string	Customer address postal code (This field accepts 9-digit input.)
`billingFrequency`	enum	The billing frequency: hourly, daily, weekly, monthly, quarterly, or annually.
`billingStatementIdentifier`	string	String to display on customer billing statement when the transaction is captured. The format and value of this string depends on your payment processor. (This string can be used to override the default set by your payment processor. Work with Vindicia Client Services if you want to use this field.)
`creditCardAccount`	string	The credit card account number to use to charge this transaction. When calling the `reportTransactions` method, you need only provide a masked account that provides the 6 digit Bank Identifying Number (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. If you use tokenization and your processor is Worldpay (formerly, Vantive or Litle) or Authorize.net, this field must contain the BIN,

Data Member	Data Type	Description
		which is the first 6 digits of the card number. If you use tokenization and your processor is neither of the above, your Capture Rate may improve if you provide this value. This is recommended.
`creditCardAccountHash`	boolean	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 can be left blank and the hash will be calculated. For reference the HA1 hash of 4111111111111111 is 68bfb396f35af3876fc509665b3dc23a0930aab1 If you use tokenization with your processor this field is optional.
`creditCardAccountUpdated`	boolean	Returned by Vindicia Retain, this data member indicates whether the Credit Card was updated. If Vindicia was able to retrieve an updated credit card through Account Updater, this field will be set to true, and the new account information will be returned in the `creditCardAccount` and `creditCardExpirationDate` fields. To enable this feature, work with Vindicia Client Services, and provide a valid PGP public key. You must also have completed an Attestation of Compliance (AOC) as a declaration of the results of the service provider's assessment with the Payment Card Industry Data Security Standards (PCI DSS).
`creditCardExpirationDate`	string	**Required** when using a credit card, that is, when not using tokenization. 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. No validation is performed to check if the

Data Member	Data Type	Description
		expiration date is in the future. **Optional** when using tokenization.
`currency`	string	**Required**. The ISO 4217 currency code (see https://www.xe.com/iso4217.php) for the transaction.
`customerId`	string	**Required**. A 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 (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 division or group number your payment processor associates with this transaction. Chase Paymentech refers to this number as the Division ID; Litle calls it the Report Group; MeS calls it the Profile ID. **WorldPay Report Group**—A WorldPay transaction can also have a report group associated with it. This value must be conveyed in a name-value pair; see nameValues above. A WorldPay report group is not a division number.
`merchantTransactionId`	string	**Required**. A unique identifier for this Transaction.
`nameValues`	NameValuePair[]	An array of name-value pairs, used to hold attribute values that are not otherwise contained in this object. Can be used to store custom data for your own internal tracking purposes. See The NameValuePair Object.

Data Member	Data Type	Description
`paymentMethodId`	string	**Required**. Your ID associated with the Payment Method for the transaction. > ***Note:*** *When this value is a token, the length (number of characters) must be within a range acceptable to your payment processor. Worldpay, for example, limits this value to 25 characters.*
`paymentMethodIsTokenized`	boolean	Flag to indicate that payment method is tokenized. The `paymentMethodId` contains the token value. When `paymentMethodIsTokenized` is true, the `creditCardAccount` and `creditCardExpirationDate` fields are optional; otherwise they are required.
`previousBillingCount`	int	The number of times the customer has been successfully billed.
`previousBillingDate`	dateTime	The date of the last successful billing associated with the subscription.
`selectTransactionId`	string	Returned by Vindicia Retain; this is the unique ID, assigned by Retain, 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, seeTransactionStatusType Values. Typically this value is *Failed*.
`subscriptionId`	string	**Required**. Unique identifier of the subscription associated with this transaction.
`subscriptionStartDate`	dateTime	A timestamp indicating the date of the first successful billing associated with the subscription. If unspecified, the value defaults to today and the current time.

Data Member	Data Type	Description
`timestamp`	dateTime	**Required**. A timestamp specifying the date and time this transaction occurred. Be sure to include this attribute in reported transactions. If you do not, timestamp defaults to the current time. **Example**: '2019-08-21T09:12:34-04:00' is 21 August 2019, 9:12:34 AM in the US/Eastern timezone. **Note** that when omitted, the time zone defaults to US/Pacific, that is, '2019-08-21T09:12:34' becomes US/Pacific, or '2019-08-21T09:12:34-07:00'. The -07:00 is the offset from UTC, for US/Pacific on this date. Note also that since the offset from UTC varies by locale-specific seasonal offsets, (US daylight savings or EU summer time, among other others) this value should not be hard coded but set programmatically when the time stamp is defined.
`VID`	string	The Vindicia 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. The `merchantTransactionId` is generally used instead of the VID to identify a transaction. Do not specify this value when creating a new `Transaction` object. Vindicia Retain populates this field and returns it in `fetchBillingResults` calls.

5.1.1 TransactionStatusType Values

Defines the transaction status upon reporting to Vindicia.

Data Members	Data Type	Description
`BillingNotAttempted`	string	The transaction was not processed by Vindicia Retain.
`Cancelled`	string	Status returned when transaction submitted to Vindicia Retain is refunded before Retain processing has started.
`Captured`	string	Status indicating the payment processor has charged the customer payment method. Captured status means the payment processor has agreed to pay and money will be transferred.
`Failed`	string	The transaction did not authorize or capture successfully.
`Refunded`	string	Transaction has been refunded. This status is returned both when `refundTransaction` is called, and when Vindicia Subscribe proactively refunds a transaction to avoid a chargeback.

5.1.2 TransactionValidationResponse Values

Defines the status returned when transactions are submitted to Vindicia Retain.

> *Note:*
>
> *The `TransactionValidationResponse` array returns only transactions with errors; transactions processed without errors have no status values returned. (An empty array means there were no errors.) Use this array to take actions on the failures. Transactions with errors are not processed by Retain. Correct all errors and resubmit the transactions to Retain for processing.*

`TransactionValidationResponse` values are summarized in the table below.

Data Members	Data Type	Description
`code`	int	Numeric code indicating the type of issue that was encountered. For more information, see Retain Return Codes.
`description`	string	Description of the issue encountered. The `description` is not saved by Vindicia Retain, and has no size limit. The meaningful data is usually located within the first 512 bytes.
`merchantTransactionId`	string	Your unique ID for the submitted transaction.

6 Vindicia Retain Methods

The core Retain Methods include the following:

- `billTransactions`—(required) submit failed transactions to Retain for processing
- `fetchBillingResults`— (required) retrieve final results of Retain processing
- `refundTransactions`— (optional) cancel/refund a transaction submitted to Retain

Implementations of core Retain Methods are evaluated during merchant Retain Certification. Refer to the respective Use Cases for what Certification expects of implementations that use these methods.

See the The Transaction Object for information about the values passed to and returned by these Retain Methods.

Example Programs Calling Vindicia Retain Methods

Example programs in C#, Java, PHP and Python that demonstrate using the Retain Methods are available on github . An example in Ruby is also available on github.

Pseudo Code for Calling Vindicia Retain Methods

Although pseudo code for calling Vindicia Retain Methods is provided with the Methods described below, this API Guide is primarily a reference, and as such is more concerned with describing the inputs and outputs of the Methods than with how they should be used in an application. For more information about what Retain Certification expects in your implementation, follow the links to the associated Use Case in each of the required methods.

The following table summarizes the methods for Vindicia Retain.

Method	Description
`billTransactions` (Required to implement Use Case SEL-001)	Submit transactions to Vindicia Retain for processing.
`fetchBillingResults` (Required to implement Use Case SEL-002)	Returns the Vindicia Retain results over a specified time period. All transactions that have reached a terminal state during the specified time period are returned.
`fetchByMerchantTransactionId`	Returns the transaction specified by the merchant transaction ID.
`fetchChargebacks`	Returns an array of Chargeback objects received from your payment processor.
`refundTransactions` (Required to implement Use Cases SEL-003 and SEL-005)	Sends an array of transactions to Vindicia Retain to be canceled or refunded. Captured transactions are refunded, transactions not captured are canceled.
`reportTransactions`	Reports data to Vindicia Retain for use in fighting chargebacks resulting from transactions not processed through Vindicia Retain.

6.1 Select.billTransactions

Submits transactions to Vindicia Retain for processing. This call is required to implement Retain Use Case SEL-001.

Use `fetchBillingResults` to retrieve final results of Vindicia Retain processing on transactions submitted via `billTransactions`; see `fetchBillingResults` for details.

Input

transactions: an array of `Transaction` objects. For a given transaction that has failed at your payment processor, submit only the final failed attempt to Vindicia Retain.

> ***Note:*** *Do not populate the VID field in any new transaction submitted with this method. Vindicia Retain populates this data member in the* `fetchBillingResults` *call.*

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 Retain Return Codes.

Example

```
$tx = new Transaction();
$tx->setTimestamp('2012-09-11T22:34:32.265-4:00');
$tx->setAmount('9.90');
$tx->setCurrency('USD');
$tx->setStatus('Failed');
$tx->setDivisionNumber('54321');
// Merchant Transaction ID must be unique for each new
// transaction you want 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->setBillingInterval('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.
}
```

6.2 Select.fetchBillingResults

Returns an array of completed transactions. Only those transactions that have reached a terminal status (Captured, Canceled, Refunded, or Failed) during the time period specified are returned. This call is required to implement Retain Use Case SEL-002.

> *Note:*
>
> *Vindicia Retain requires that input parameters be entered in the order shown, and place-marked 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 Retain Account Updater during Deployment, contact your Vindicia Solutions Architect; after go-live, contact Vindicia Customer Support. You must also have completed an Attestation of Compliance (AOC) as a declaration of the results of the service provider's assessment with the Payment Card Industry Data Security Standards (PCI DSS).*
>
> *You must provide a PGP public key to Vindicia to encrypt returned credit card information.*

Input

timestamp: the starting timestamp for the range of (terminal state) `Transactions` you want to retrieve.

endTimestamp: the ending timestamp for the range of (terminal state) `Transactions` you want 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:*** *Each `Transaction` object returned will identify the original transaction submitted via the `merchantTransactionId` and the last transaction processed as the `selectTransactionId`. Thus, terminal results for every transaction submitted on a given day should have been retrieved by or as of the fetch following the final retry.*

Returns

This method returns the codes listed in Retain Return Codes.

6.3 Select.fetchByMerchantTransactionId

Returns the `Transaction` specified by the input `merchantTransactionId`. Use this method to fetch specific transactions. Use `fetchBillingResults` to retrieve all results in a single call.

Use this method also to confirm that Select received a transaction when the `merchantTransactionId` was provided. It returns the transaction that was submitted, but it does not return the final processing status of the transaction. Use `fetchBillingResults` to obtain the final processing status.

When a `transactionId`, that is, the transaction processing Id of a Select processing step (retry) after submission, is provided, that transaction is returned. Note that the only way to find out the final state of a `transactionId` is by using the `fetchBillingResults` call. This means that you should already have received all final state results. Making this call with a `transactionId` that you got from fetchBillingResults would only provide the same results.

This method is not required to implement any Retain Use Case. Per SEL-002, you must use `fetchBillingResults` for regular fetches in Production.

Input

merchantTransactionId: the `merchantTransactionId` for the transaction to fetch.

Output

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

transaction: the transaction identified by the `TransactionId`.

Returns

This method returns the codes listed in Retain 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.
}
```

6.4 Select.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 Retain 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 want to retrieve.

endTimestamp: the ending timestamp (upper limit) for the range of `Chargebacks` you want to retrieve. A null end time stamp defaults to *now* (the current time).

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

- Specifying 0 for *page* returns results from page 1 through 10.
- Specifying 2 for *page* returns results from page 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 success or failure of the call.

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

Returns

This method returns the codes listed in Retain 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);
```

6.5 Select.refundTransactions

Directs Vindicia Retain to discontinue further attempts to capture a transaction, or to refund a captured transaction. A call to `refund.Transactions` might be needed when a customer has canceled a subscription, or has paid a bill. This call results in Select either ceasing attempts (retries) to capture a transaction, or refunding one that was captured.

If a submitted transaction fails to validate it is returned in the response array, along with information indicating the validation issue.

After Select processes the `refundTransactions` call, the next call to `fetchBillingResults`, or to `fetchByMerchantTransactionId`, will return the status of the `refundTransactions` call.

Input

refunds: an array of `merchantTransactionIds` to refund/cancel.

Output

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

response: an object of type `TransactionValidationResponse`, which includes the `merchantTransactionID` of each transaction that returned 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 Retain 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.
}
```

6.6 Select.reportTransactions

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

Input

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

Output

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

response: an object of type `TransactionValidationResponse`, which includes the ID of an emh 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 Retain 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
}
```