PHP Models

# Address

<?php

namespace Swagger\Client\Model;

use \ArrayAccess;

use \Swagger\Client\ObjectSerializer;

class Address implements ModelInterface, ArrayAccess

{

const DISCRIMINATOR = null;

protected static $swaggerModelName = 'Address';

protected static $swaggerTypes = [

'address\_line1' => 'string',

'address\_line2' => 'string',

'city' => 'string',

'state' => 'string',

'zip\_code' => 'string'

];

protected static $swaggerFormats = [

'address\_line1' => null,

'address\_line2' => null,

'city' => null,

'state' => null,

'zip\_code' => null

];

public static function swaggerTypes()

{

return self::$swaggerTypes;

}

public static function swaggerFormats()

{

return self::$swaggerFormats;

}

protected static $attributeMap = [

'address\_line1' => 'AddressLine1',

'address\_line2' => 'AddressLine2',

'city' => 'City',

'state' => 'State',

'zip\_code' => 'ZipCode'

];

protected static $setters = [

'address\_line1' => 'setAddressLine1',

'address\_line2' => 'setAddressLine2',

'city' => 'setCity',

'state' => 'setState',

'zip\_code' => 'setZipCode'

];

protected static $getters = [

'address\_line1' => 'getAddressLine1',

'address\_line2' => 'getAddressLine2',

'city' => 'getCity',

'state' => 'getState',

'zip\_code' => 'getZipCode'

];

public static function attributeMap()

{

return self::$attributeMap;

}

public static function setters()

{

return self::$setters;

}

public static function getters()

{

return self::$getters;

}

public function getModelName()

{

return self::$swaggerModelName;

}

protected $container = [];

public function \_\_construct(array $data = null)

{

$this->container['address\_line1'] = isset($data['address\_line1']) ? $data['address\_line1'] : null;

$this->container['address\_line2'] = isset($data['address\_line2']) ? $data['address\_line2'] : null;

$this->container['city'] = isset($data['city']) ? $data['city'] : null;

$this->container['state'] = isset($data['state']) ? $data['state'] : null;

$this->container['zip\_code'] = isset($data['zip\_code']) ? $data['zip\_code'] : null;

}

public function listInvalidProperties()

{

$invalidProperties = [];

return $invalidProperties;

}

public function valid()

{

return count($this->listInvalidProperties()) === 0;

}

public function getAddressLine1()

{

return $this->container['address\_line1'];

}

public function setAddressLine1($address\_line1)

{

$this->container['address\_line1'] = $address\_line1;

return $this;

}

public function getAddressLine2()

{

return $this->container['address\_line2'];

}

public function setAddressLine2($address\_line2)

{

$this->container['address\_line2'] = $address\_line2;

return $this;

}

public function getCity()

{

return $this->container['city'];

}

public function setCity($city)

{

$this->container['city'] = $city;

return $this;

}

public function getState()

{

return $this->container['state'];

}

public function setState($state)

{

$this->container['state'] = $state;

return $this;

}

public function getZipCode()

{

return $this->container['zip\_code'];

}

public function setZipCode($zip\_code)

{

$this->container['zip\_code'] = $zip\_code;

return $this;

}

public function offsetExists($offset)

{

return isset($this->container[$offset]);

}

public function offsetGet($offset)

{

return isset($this->container[$offset]) ? $this->container[$offset] : null;

}

public function offsetSet($offset, $value)

{

if (is\_null($offset)) {

$this->container[] = $value;

} else {

$this->container[$offset] = $value;

}

}

public function offsetUnset($offset)

{

unset($this->container[$offset]);

}

public function \_\_toString()

{

if (defined('JSON\_PRETTY\_PRINT')) { // use JSON pretty print

return json\_encode(

ObjectSerializer::sanitizeForSerialization($this),

JSON\_PRETTY\_PRINT

);

}

return json\_encode(ObjectSerializer::sanitizeForSerialization($this));

}

}

# Batch

<?php

namespace Swagger\Client\Model;

use \ArrayAccess;

use \Swagger\Client\ObjectSerializer;

class Batch implements ModelInterface, ArrayAccess

{

const DISCRIMINATOR = null;

protected static $swaggerModelName = 'Batch';

protected static $swaggerTypes = [

'batch\_number' => 'string',

'started' => '\DateTime',

'completed' => '\DateTime',

'sales\_count' => 'string',

'sales\_sum' => 'double',

'return\_count' => 'string',

'return\_sum' => 'double',

'batch\_error' => 'bool'

];

protected static $swaggerFormats = [

'batch\_number' => null,

'started' => 'date-time',

'completed' => 'date-time',

'sales\_count' => null,

'sales\_sum' => 'double',

'return\_count' => null,

'return\_sum' => 'double',

'batch\_error' => null

];

public static function swaggerTypes()

{

return self::$swaggerTypes;

}

public static function swaggerFormats()

{

return self::$swaggerFormats;

}

protected static $attributeMap = [

'batch\_number' => 'BatchNumber',

'started' => 'Started',

'completed' => 'Completed',

'sales\_count' => 'SalesCount',

'sales\_sum' => 'SalesSum',

'return\_count' => 'ReturnCount',

'return\_sum' => 'ReturnSum',

'batch\_error' => 'BatchError'

];

protected static $setters = [

'batch\_number' => 'setBatchNumber',

'started' => 'setStarted',

'completed' => 'setCompleted',

'sales\_count' => 'setSalesCount',

'sales\_sum' => 'setSalesSum',

'return\_count' => 'setReturnCount',

'return\_sum' => 'setReturnSum',

'batch\_error' => 'setBatchError'

];

protected static $getters = [

'batch\_number' => 'getBatchNumber',

'started' => 'getStarted',

'completed' => 'getCompleted',

'sales\_count' => 'getSalesCount',

'sales\_sum' => 'getSalesSum',

'return\_count' => 'getReturnCount',

'return\_sum' => 'getReturnSum',

'batch\_error' => 'getBatchError'

];

public static function attributeMap()

{

return self::$attributeMap;

}

public static function setters()

{

return self::$setters;

}

public static function getters()

{

return self::$getters;

}

public function getModelName()

{

return self::$swaggerModelName;

}

protected $container = [];

public function \_\_construct(array $data = null)

{

$this->container['batch\_number'] = isset($data['batch\_number']) ? $data['batch\_number'] : null;

$this->container['started'] = isset($data['started']) ? $data['started'] : null;

$this->container['completed'] = isset($data['completed']) ? $data['completed'] : null;

$this->container['sales\_count'] = isset($data['sales\_count']) ? $data['sales\_count'] : null;

$this->container['sales\_sum'] = isset($data['sales\_sum']) ? $data['sales\_sum'] : null;

$this->container['return\_count'] = isset($data['return\_count']) ? $data['return\_count'] : null;

$this->container['return\_sum'] = isset($data['return\_sum']) ? $data['return\_sum'] : null;

$this->container['batch\_error'] = isset($data['batch\_error']) ? $data['batch\_error'] : null;

}

public function listInvalidProperties()

{

$invalidProperties = [];

return $invalidProperties;

}

public function valid()

{

return count($this->listInvalidProperties()) === 0;

}

public function getBatchNumber()

{

return $this->container['batch\_number'];

}

public function setBatchNumber($batch\_number)

{

$this->container['batch\_number'] = $batch\_number;

return $this;

}

public function getStarted()

{

return $this->container['started'];

}

public function setStarted($started)

{

$this->container['started'] = $started;

return $this;

}

public function getCompleted()

{

return $this->container['completed'];

}

public function setCompleted($completed)

{

$this->container['completed'] = $completed;

return $this;

}

public function getSalesCount()

{

return $this->container['sales\_count'];

}

public function setSalesCount($sales\_count)

{

$this->container['sales\_count'] = $sales\_count;

return $this;

}

public function getSalesSum()

{

return $this->container['sales\_sum'];

}

public function setSalesSum($sales\_sum)

{

$this->container['sales\_sum'] = $sales\_sum;

return $this;

}

public function getReturnCount()

{

return $this->container['return\_count'];

}

public function setReturnCount($return\_count)

{

$this->container['return\_count'] = $return\_count;

return $this;

}

public function getReturnSum()

{

return $this->container['return\_sum'];

}

public function setReturnSum($return\_sum)

{

$this->container['return\_sum'] = $return\_sum;

return $this;

}

public function getBatchError()

{

return $this->container['batch\_error'];

}

public function setBatchError($batch\_error)

{

$this->container['batch\_error'] = $batch\_error;

return $this;

}

public function offsetExists($offset)

{

return isset($this->container[$offset]);

}

public function offsetGet($offset)

{

return isset($this->container[$offset]) ? $this->container[$offset] : null;

}

public function offsetSet($offset, $value)

{

if (is\_null($offset)) {

$this->container[] = $value;

} else {

$this->container[$offset] = $value;

}

}

public function offsetUnset($offset)

{

unset($this->container[$offset]);

}

public function \_\_toString()

{

if (defined('JSON\_PRETTY\_PRINT')) { // use JSON pretty print

return json\_encode(

ObjectSerializer::sanitizeForSerialization($this),

JSON\_PRETTY\_PRINT

);

}

return json\_encode(ObjectSerializer::sanitizeForSerialization($this));

}

}

# Business

<?php

namespace Swagger\Client\Model;

use \ArrayAccess;

use \Swagger\Client\ObjectSerializer;

class Business implements ModelInterface, ArrayAccess

{

const DISCRIMINATOR = null;

protected static $swaggerModelName = 'Business';

protected static $swaggerTypes = [

'address' => '\Swagger\Client\Model\Address',

'name' => 'string',

'phone' => 'string',

'fax' => 'string',

'web\_site' => 'string'

];

protected static $swaggerFormats = [

'address' => null,

'name' => null,

'phone' => null,

'fax' => null,

'web\_site' => null

];

public static function swaggerTypes()

{

return self::$swaggerTypes;

}

public static function swaggerFormats()

{

return self::$swaggerFormats;

}

protected static $attributeMap = [

'address' => 'Address',

'name' => 'Name',

'phone' => 'Phone',

'fax' => 'Fax',

'web\_site' => 'WebSite'

];

protected static $setters = [

'address' => 'setAddress',

'name' => 'setName',

'phone' => 'setPhone',

'fax' => 'setFax',

'web\_site' => 'setWebSite'

];

protected static $getters = [

'address' => 'getAddress',

'name' => 'getName',

'phone' => 'getPhone',

'fax' => 'getFax',

'web\_site' => 'getWebSite'

];

public static function attributeMap()

{

return self::$attributeMap;

}

public static function setters()

{

return self::$setters;

}

public static function getters()

{

return self::$getters;

}

public function getModelName()

{

return self::$swaggerModelName;

}

protected $container = [];

public function \_\_construct(array $data = null)

{

$this->container['address'] = isset($data['address']) ? $data['address'] : null;

$this->container['name'] = isset($data['name']) ? $data['name'] : null;

$this->container['phone'] = isset($data['phone']) ? $data['phone'] : null;

$this->container['fax'] = isset($data['fax']) ? $data['fax'] : null;

$this->container['web\_site'] = isset($data['web\_site']) ? $data['web\_site'] : null;

}

public function listInvalidProperties()

{

$invalidProperties = [];

return $invalidProperties;

}

public function valid()

{

return count($this->listInvalidProperties()) === 0;

}

public function getAddress()

{

return $this->container['address'];

}

public function setAddress($address)

{

$this->container['address'] = $address;

return $this;

}

public function getName()

{

return $this->container['name'];

}

public function setName($name)

{

$this->container['name'] = $name;

return $this;

}

public function getPhone()

{

return $this->container['phone'];

}

public function setPhone($phone)

{

$this->container['phone'] = $phone;

return $this;

}

public function getFax()

{

return $this->container['fax'];

}

public function setFax($fax)

{

$this->container['fax'] = $fax;

return $this;

}

public function getWebSite()

{

return $this->container['web\_site'];

}

public function setWebSite($web\_site)

{

$this->container['web\_site'] = $web\_site;

return $this;

}

public function offsetExists($offset)

{

return isset($this->container[$offset]);

}

public function offsetGet($offset)

{

return isset($this->container[$offset]) ? $this->container[$offset] : null;

}

public function offsetSet($offset, $value)

{

if (is\_null($offset)) {

$this->container[] = $value;

} else {

$this->container[$offset] = $value;

}

}

public function offsetUnset($offset)

{

unset($this->container[$offset]);

}

public function \_\_toString()

{

if (defined('JSON\_PRETTY\_PRINT')) { // use JSON pretty print

return json\_encode(

ObjectSerializer::sanitizeForSerialization($this),

JSON\_PRETTY\_PRINT

);

}

return json\_encode(ObjectSerializer::sanitizeForSerialization($this));

}

}

# CheckAccountType

<?php

namespace Swagger\Client\Model;

use \Swagger\Client\ObjectSerializer;

class CheckAccountType

{

const UNKNOWN = 'Unknown';

const CHECKING = 'Checking';

const SAVINGS = 'Savings';

public static function getAllowableEnumValues()

{

return [

self::UNKNOWN,

self::CHECKING,

self::SAVINGS,

];

}

}

# CheckTransaction

<?php

namespace Swagger\Client\Model;

use \ArrayAccess;

use \Swagger\Client\ObjectSerializer;

class CheckTransaction implements ModelInterface, ArrayAccess

{

const DISCRIMINATOR = null;

protected static $swaggerModelName = 'CheckTransaction';

protected static $swaggerTypes = [

'terminal\_id' => 'string',

'account\_number' => 'string',

'account\_type' => '\Swagger\Client\Model\CheckAccountType',

'check\_number' => 'string',

'routing\_number' => 'string',

'micr\_data' => 'string',

'first\_name' => 'string',

'last\_name' => 'string',

'company\_name' => 'string',

'address1' => 'string',

'address2' => 'string',

'city' => 'string',

'state' => 'string',

'zip' => 'string',

'phone\_number' => 'string',

'dl\_number' => 'string',

'dl\_state' => 'string',

'courtesy\_card\_id' => 'string',

'ssn4' => 'string',

'dob\_year' => 'string',

'check\_amount' => 'string',

'tran\_type' => '\Swagger\Client\Model\CheckTransactionType',

'proc\_id' => 'string',

'ip\_address' => 'string',

'customer\_number' => 'string',

'po\_number' => 'string',

'invoice\_number' => 'string',

'tran\_note' => 'string',

'email\_address' => 'string',

'sec\_code' => 'string',

'proc\_resp' => 'string',

'proc\_code' => 'string',

'proc\_mess' => 'string'

];

protected static $swaggerFormats = [

'terminal\_id' => null,

'account\_number' => null,

'account\_type' => null,

'check\_number' => null,

'routing\_number' => null,

'micr\_data' => null,

'first\_name' => null,

'last\_name' => null,

'company\_name' => null,

'address1' => null,

'address2' => null,

'city' => null,

'state' => null,

'zip' => null,

'phone\_number' => null,

'dl\_number' => null,

'dl\_state' => null,

'courtesy\_card\_id' => null,

'ssn4' => null,

'dob\_year' => null,

'check\_amount' => null,

'tran\_type' => null,

'proc\_id' => null,

'ip\_address' => null,

'customer\_number' => null,

'po\_number' => null,

'invoice\_number' => null,

'tran\_note' => null,

'email\_address' => null,

'sec\_code' => null,

'proc\_resp' => null,

'proc\_code' => null,

'proc\_mess' => null

];

public static function swaggerTypes()

{

return self::$swaggerTypes;

}

public static function swaggerFormats()

{

return self::$swaggerFormats;

}

protected static $attributeMap = [

'terminal\_id' => 'Terminal\_ID',

'account\_number' => 'Account\_Number',

'account\_type' => 'Account\_Type',

'check\_number' => 'Check\_Number',

'routing\_number' => 'Routing\_Number',

'micr\_data' => 'MICR\_Data',

'first\_name' => 'First\_Name',

'last\_name' => 'Last\_Name',

'company\_name' => 'Company\_Name',

'address1' => 'Address1',

'address2' => 'Address2',

'city' => 'City',

'state' => 'State',

'zip' => 'Zip',

'phone\_number' => 'Phone\_Number',

'dl\_number' => 'DL\_Number',

'dl\_state' => 'DL\_State',

'courtesy\_card\_id' => 'Courtesy\_Card\_ID',

'ssn4' => 'SSN4',

'dob\_year' => 'DOB\_Year',

'check\_amount' => 'Check\_Amount',

'tran\_type' => 'Tran\_Type',

'proc\_id' => 'Proc\_ID',

'ip\_address' => 'IP\_Address',

'customer\_number' => 'CustomerNumber',

'po\_number' => 'PONumber',

'invoice\_number' => 'InvoiceNumber',

'tran\_note' => 'TranNote',

'email\_address' => 'EmailAddress',

'sec\_code' => 'SEC\_Code',

'proc\_resp' => 'Proc\_Resp',

'proc\_code' => 'Proc\_Code',

'proc\_mess' => 'Proc\_Mess'

];

protected static $setters = [

'terminal\_id' => 'setTerminalId',

'account\_number' => 'setAccountNumber',

'account\_type' => 'setAccountType',

'check\_number' => 'setCheckNumber',

'routing\_number' => 'setRoutingNumber',

'micr\_data' => 'setMicrData',

'first\_name' => 'setFirstName',

'last\_name' => 'setLastName',

'company\_name' => 'setCompanyName',

'address1' => 'setAddress1',

'address2' => 'setAddress2',

'city' => 'setCity',

'state' => 'setState',

'zip' => 'setZip',

'phone\_number' => 'setPhoneNumber',

'dl\_number' => 'setDlNumber',

'dl\_state' => 'setDlState',

'courtesy\_card\_id' => 'setCourtesyCardId',

'ssn4' => 'setSsn4',

'dob\_year' => 'setDobYear',

'check\_amount' => 'setCheckAmount',

'tran\_type' => 'setTranType',

'proc\_id' => 'setProcId',

'ip\_address' => 'setIpAddress',

'customer\_number' => 'setCustomerNumber',

'po\_number' => 'setPoNumber',

'invoice\_number' => 'setInvoiceNumber',

'tran\_note' => 'setTranNote',

'email\_address' => 'setEmailAddress',

'sec\_code' => 'setSecCode',

'proc\_resp' => 'setProcResp',

'proc\_code' => 'setProcCode',

'proc\_mess' => 'setProcMess'

];

protected static $getters = [

'terminal\_id' => 'getTerminalId',

'account\_number' => 'getAccountNumber',

'account\_type' => 'getAccountType',

'check\_number' => 'getCheckNumber',

'routing\_number' => 'getRoutingNumber',

'micr\_data' => 'getMicrData',

'first\_name' => 'getFirstName',

'last\_name' => 'getLastName',

'company\_name' => 'getCompanyName',

'address1' => 'getAddress1',

'address2' => 'getAddress2',

'city' => 'getCity',

'state' => 'getState',

'zip' => 'getZip',

'phone\_number' => 'getPhoneNumber',

'dl\_number' => 'getDlNumber',

'dl\_state' => 'getDlState',

'courtesy\_card\_id' => 'getCourtesyCardId',

'ssn4' => 'getSsn4',

'dob\_year' => 'getDobYear',

'check\_amount' => 'getCheckAmount',

'tran\_type' => 'getTranType',

'proc\_id' => 'getProcId',

'ip\_address' => 'getIpAddress',

'customer\_number' => 'getCustomerNumber',

'po\_number' => 'getPoNumber',

'invoice\_number' => 'getInvoiceNumber',

'tran\_note' => 'getTranNote',

'email\_address' => 'getEmailAddress',

'sec\_code' => 'getSecCode',

'proc\_resp' => 'getProcResp',

'proc\_code' => 'getProcCode',

'proc\_mess' => 'getProcMess'

];

public static function attributeMap()

{

return self::$attributeMap;

}

public static function setters()

{

return self::$setters;

}

public static function getters()

{

return self::$getters;

}

public function getModelName()

{

return self::$swaggerModelName;

}

protected $container = [];

public function \_\_construct(array $data = null)

{

$this->container['terminal\_id'] = isset($data['terminal\_id']) ? $data['terminal\_id'] : null;

$this->container['account\_number'] = isset($data['account\_number']) ? $data['account\_number'] : null;

$this->container['account\_type'] = isset($data['account\_type']) ? $data['account\_type'] : null;

$this->container['check\_number'] = isset($data['check\_number']) ? $data['check\_number'] : null;

$this->container['routing\_number'] = isset($data['routing\_number']) ? $data['routing\_number'] : null;

$this->container['micr\_data'] = isset($data['micr\_data']) ? $data['micr\_data'] : null;

$this->container['first\_name'] = isset($data['first\_name']) ? $data['first\_name'] : null;

$this->container['last\_name'] = isset($data['last\_name']) ? $data['last\_name'] : null;

$this->container['company\_name'] = isset($data['company\_name']) ? $data['company\_name'] : null;

$this->container['address1'] = isset($data['address1']) ? $data['address1'] : null;

$this->container['address2'] = isset($data['address2']) ? $data['address2'] : null;

$this->container['city'] = isset($data['city']) ? $data['city'] : null;

$this->container['state'] = isset($data['state']) ? $data['state'] : null;

$this->container['zip'] = isset($data['zip']) ? $data['zip'] : null;

$this->container['phone\_number'] = isset($data['phone\_number']) ? $data['phone\_number'] : null;

$this->container['dl\_number'] = isset($data['dl\_number']) ? $data['dl\_number'] : null;

$this->container['dl\_state'] = isset($data['dl\_state']) ? $data['dl\_state'] : null;

$this->container['courtesy\_card\_id'] = isset($data['courtesy\_card\_id']) ? $data['courtesy\_card\_id'] : null;

$this->container['ssn4'] = isset($data['ssn4']) ? $data['ssn4'] : null;

$this->container['dob\_year'] = isset($data['dob\_year']) ? $data['dob\_year'] : null;

$this->container['check\_amount'] = isset($data['check\_amount']) ? $data['check\_amount'] : null;

$this->container['tran\_type'] = isset($data['tran\_type']) ? $data['tran\_type'] : null;

$this->container['proc\_id'] = isset($data['proc\_id']) ? $data['proc\_id'] : null;

$this->container['ip\_address'] = isset($data['ip\_address']) ? $data['ip\_address'] : null;

$this->container['customer\_number'] = isset($data['customer\_number']) ? $data['customer\_number'] : null;

$this->container['po\_number'] = isset($data['po\_number']) ? $data['po\_number'] : null;

$this->container['invoice\_number'] = isset($data['invoice\_number']) ? $data['invoice\_number'] : null;

$this->container['tran\_note'] = isset($data['tran\_note']) ? $data['tran\_note'] : null;

$this->container['email\_address'] = isset($data['email\_address']) ? $data['email\_address'] : null;

$this->container['sec\_code'] = isset($data['sec\_code']) ? $data['sec\_code'] : null;

$this->container['proc\_resp'] = isset($data['proc\_resp']) ? $data['proc\_resp'] : null;

$this->container['proc\_code'] = isset($data['proc\_code']) ? $data['proc\_code'] : null;

$this->container['proc\_mess'] = isset($data['proc\_mess']) ? $data['proc\_mess'] : null;

}

public function listInvalidProperties()

{

$invalidProperties = [];

return $invalidProperties;

}

public function valid()

{

return count($this->listInvalidProperties()) === 0;

}

public function getTerminalId()

{

return $this->container['terminal\_id'];

}

public function setTerminalId($terminal\_id)

{

$this->container['terminal\_id'] = $terminal\_id;

return $this;

}

public function getAccountNumber()

{

return $this->container['account\_number'];

}

public function setAccountNumber($account\_number)

{

$this->container['account\_number'] = $account\_number;

return $this;

}

public function getAccountType()

{

return $this->container['account\_type'];

}

public function setAccountType($account\_type)

{

$this->container['account\_type'] = $account\_type;

return $this;

}

public function getCheckNumber()

{

return $this->container['check\_number'];

}

public function setCheckNumber($check\_number)

{

$this->container['check\_number'] = $check\_number;

return $this;

}

public function getRoutingNumber()

{

return $this->container['routing\_number'];

}

public function setRoutingNumber($routing\_number)

{

$this->container['routing\_number'] = $routing\_number;

return $this;

}

public function getMicrData()

{

return $this->container['micr\_data'];

}

public function setMicrData($micr\_data)

{

$this->container['micr\_data'] = $micr\_data;

return $this;

}

public function getFirstName()

{

return $this->container['first\_name'];

}

public function setFirstName($first\_name)

{

$this->container['first\_name'] = $first\_name;

return $this;

}

public function getLastName()

{

return $this->container['last\_name'];

}

public function setLastName($last\_name)

{

$this->container['last\_name'] = $last\_name;

return $this;

}

public function getCompanyName()

{

return $this->container['company\_name'];

}

public function setCompanyName($company\_name)

{

$this->container['company\_name'] = $company\_name;

return $this;

}

public function getAddress1()

{

return $this->container['address1'];

}

public function setAddress1($address1)

{

$this->container['address1'] = $address1;

return $this;

}

public function getAddress2()

{

return $this->container['address2'];

}

public function setAddress2($address2)

{

$this->container['address2'] = $address2;

return $this;

}

public function getCity()

{

return $this->container['city'];

}

public function setCity($city)

{

$this->container['city'] = $city;

return $this;

}

public function getState()

{

return $this->container['state'];

}

public function setState($state)

{

$this->container['state'] = $state;

return $this;

}

public function getZip()

{

return $this->container['zip'];

}

public function setZip($zip)

{

$this->container['zip'] = $zip;

return $this;

}

public function getPhoneNumber()

{

return $this->container['phone\_number'];

}

public function setPhoneNumber($phone\_number)

{

$this->container['phone\_number'] = $phone\_number;

return $this;

}

public function getDlNumber()

{

return $this->container['dl\_number'];

}

public function setDlNumber($dl\_number)

{

$this->container['dl\_number'] = $dl\_number;

return $this;

}

public function getDlState()

{

return $this->container['dl\_state'];

}

public function setDlState($dl\_state)

{

$this->container['dl\_state'] = $dl\_state;

return $this;

}

public function getCourtesyCardId()

{

return $this->container['courtesy\_card\_id'];

}

public function setCourtesyCardId($courtesy\_card\_id)

{

$this->container['courtesy\_card\_id'] = $courtesy\_card\_id;

return $this;

}

public function getSsn4()

{

return $this->container['ssn4'];

}

public function setSsn4($ssn4)

{

$this->container['ssn4'] = $ssn4;

return $this;

}

public function getDobYear()

{

return $this->container['dob\_year'];

}

public function setDobYear($dob\_year)

{

$this->container['dob\_year'] = $dob\_year;

return $this;

}

public function getCheckAmount()

{

return $this->container['check\_amount'];

}

public function setCheckAmount($check\_amount)

{

$this->container['check\_amount'] = $check\_amount;

return $this;

}

public function getTranType()

{

return $this->container['tran\_type'];

}

public function setTranType($tran\_type)

{

$this->container['tran\_type'] = $tran\_type;

return $this;

}

public function getProcId()

{

return $this->container['proc\_id'];

}

public function setProcId($proc\_id)

{

$this->container['proc\_id'] = $proc\_id;

return $this;

}

public function getIpAddress()

{

return $this->container['ip\_address'];

}

public function setIpAddress($ip\_address)

{

$this->container['ip\_address'] = $ip\_address;

return $this;

}

public function getCustomerNumber()

{

return $this->container['customer\_number'];

}

public function setCustomerNumber($customer\_number)

{

$this->container['customer\_number'] = $customer\_number;

return $this;

}

public function getPoNumber()

{

return $this->container['po\_number'];

}

public function setPoNumber($po\_number)

{

$this->container['po\_number'] = $po\_number;

return $this;

}

public function getInvoiceNumber()

{

return $this->container['invoice\_number'];

}

public function setInvoiceNumber($invoice\_number)

{

$this->container['invoice\_number'] = $invoice\_number;

return $this;

}

public function getTranNote()

{

return $this->container['tran\_note'];

}

public function setTranNote($tran\_note)

{

$this->container['tran\_note'] = $tran\_note;

return $this;

}

public function getEmailAddress()

{

return $this->container['email\_address'];

}

public function setEmailAddress($email\_address)

{

$this->container['email\_address'] = $email\_address;

return $this;

}

public function getSecCode()

{

return $this->container['sec\_code'];

}

public function setSecCode($sec\_code)

{

$this->container['sec\_code'] = $sec\_code;

return $this;

}

public function getProcResp()

{

return $this->container['proc\_resp'];

}

public function setProcResp($proc\_resp)

{

$this->container['proc\_resp'] = $proc\_resp;

return $this;

}

public function getProcCode()

{

return $this->container['proc\_code'];

}

public function setProcCode($proc\_code)

{

$this->container['proc\_code'] = $proc\_code;

return $this;

}

public function getProcMess()

{

return $this->container['proc\_mess'];

}

public function setProcMess($proc\_mess)

{

$this->container['proc\_mess'] = $proc\_mess;

return $this;

}

public function offsetExists($offset)

{

return isset($this->container[$offset]);

}

public function offsetGet($offset)

{

return isset($this->container[$offset]) ? $this->container[$offset] : null;

}

public function offsetSet($offset, $value)

{

if (is\_null($offset)) {

$this->container[] = $value;

} else {

$this->container[$offset] = $value;

}

}

public function offsetUnset($offset)

{

unset($this->container[$offset]);

}

public function \_\_toString()

{

if (defined('JSON\_PRETTY\_PRINT')) { // use JSON pretty print

return json\_encode(

ObjectSerializer::sanitizeForSerialization($this),

JSON\_PRETTY\_PRINT

);

}

return json\_encode(ObjectSerializer::sanitizeForSerialization($this));

}

}

# CheckTransactionSettings

<?php

namespace Swagger\Client\Model;

use \ArrayAccess;

use \Swagger\Client\ObjectSerializer;

class CheckTransactionSettings implements ModelInterface, ArrayAccess

{

const DISCRIMINATOR = null;

protected static $swaggerModelName = 'CheckTransactionSettings';

protected static $swaggerTypes = [

'reader' => '\Swagger\Client\Model\FieldInfo',

'contact\_name' => '\Swagger\Client\Model\FieldInfo',

'contact\_title' => '\Swagger\Client\Model\FieldInfo',

'email\_address' => '\Swagger\Client\Model\FieldInfo',

'phone\_default' => 'string',

'fax' => '\Swagger\Client\Model\FieldInfo',

'billing\_address\_default' => 'string',

'billing\_address2' => '\Swagger\Client\Model\FieldInfo',

'billing\_city\_default' => 'string',

'billing\_state\_default' => 'string',

'billing\_zip\_default' => 'string',

'po\_number' => '\Swagger\Client\Model\FieldInfo',

'invoice\_number' => '\Swagger\Client\Model\FieldInfo',

'business\_name' => '\Swagger\Client\Model\FieldInfo',

'customer\_number' => '\Swagger\Client\Model\FieldInfo',

'reference\_note' => '\Swagger\Client\Model\FieldInfo',

'default\_sec\_code' => 'string'

];

protected static $swaggerFormats = [

'reader' => null,

'contact\_name' => null,

'contact\_title' => null,

'email\_address' => null,

'phone\_default' => null,

'fax' => null,

'billing\_address\_default' => null,

'billing\_address2' => null,

'billing\_city\_default' => null,

'billing\_state\_default' => null,

'billing\_zip\_default' => null,

'po\_number' => null,

'invoice\_number' => null,

'business\_name' => null,

'customer\_number' => null,

'reference\_note' => null,

'default\_sec\_code' => null

];

public static function swaggerTypes()

{

return self::$swaggerTypes;

}

public static function swaggerFormats()

{

return self::$swaggerFormats;

}

protected static $attributeMap = [

'reader' => 'Reader',

'contact\_name' => 'ContactName',

'contact\_title' => 'ContactTitle',

'email\_address' => 'EmailAddress',

'phone\_default' => 'PhoneDefault',

'fax' => 'Fax',

'billing\_address\_default' => 'BillingAddressDefault',

'billing\_address2' => 'BillingAddress2',

'billing\_city\_default' => 'BillingCityDefault',

'billing\_state\_default' => 'BillingStateDefault',

'billing\_zip\_default' => 'BillingZipDefault',

'po\_number' => 'PONumber',

'invoice\_number' => 'InvoiceNumber',

'business\_name' => 'BusinessName',

'customer\_number' => 'CustomerNumber',

'reference\_note' => 'ReferenceNote',

'default\_sec\_code' => 'DefaultSECCode'

];

protected static $setters = [

'reader' => 'setReader',

'contact\_name' => 'setContactName',

'contact\_title' => 'setContactTitle',

'email\_address' => 'setEmailAddress',

'phone\_default' => 'setPhoneDefault',

'fax' => 'setFax',

'billing\_address\_default' => 'setBillingAddressDefault',

'billing\_address2' => 'setBillingAddress2',

'billing\_city\_default' => 'setBillingCityDefault',

'billing\_state\_default' => 'setBillingStateDefault',

'billing\_zip\_default' => 'setBillingZipDefault',

'po\_number' => 'setPoNumber',

'invoice\_number' => 'setInvoiceNumber',

'business\_name' => 'setBusinessName',

'customer\_number' => 'setCustomerNumber',

'reference\_note' => 'setReferenceNote',

'default\_sec\_code' => 'setDefaultSecCode'

];

protected static $getters = [

'reader' => 'getReader',

'contact\_name' => 'getContactName',

'contact\_title' => 'getContactTitle',

'email\_address' => 'getEmailAddress',

'phone\_default' => 'getPhoneDefault',

'fax' => 'getFax',

'billing\_address\_default' => 'getBillingAddressDefault',

'billing\_address2' => 'getBillingAddress2',

'billing\_city\_default' => 'getBillingCityDefault',

'billing\_state\_default' => 'getBillingStateDefault',

'billing\_zip\_default' => 'getBillingZipDefault',

'po\_number' => 'getPoNumber',

'invoice\_number' => 'getInvoiceNumber',

'business\_name' => 'getBusinessName',

'customer\_number' => 'getCustomerNumber',

'reference\_note' => 'getReferenceNote',

'default\_sec\_code' => 'getDefaultSecCode'

];

public static function attributeMap()

{

return self::$attributeMap;

}

public static function setters()

{

return self::$setters;

}

public static function getters()

{

return self::$getters;

}

public function getModelName()

{

return self::$swaggerModelName;

}

protected $container = [];

public function \_\_construct(array $data = null)

{

$this->container['reader'] = isset($data['reader']) ? $data['reader'] : null;

$this->container['contact\_name'] = isset($data['contact\_name']) ? $data['contact\_name'] : null;

$this->container['contact\_title'] = isset($data['contact\_title']) ? $data['contact\_title'] : null;

$this->container['email\_address'] = isset($data['email\_address']) ? $data['email\_address'] : null;

$this->container['phone\_default'] = isset($data['phone\_default']) ? $data['phone\_default'] : null;

$this->container['fax'] = isset($data['fax']) ? $data['fax'] : null;

$this->container['billing\_address\_default'] = isset($data['billing\_address\_default']) ? $data['billing\_address\_default'] : null;

$this->container['billing\_address2'] = isset($data['billing\_address2']) ? $data['billing\_address2'] : null;

$this->container['billing\_city\_default'] = isset($data['billing\_city\_default']) ? $data['billing\_city\_default'] : null;

$this->container['billing\_state\_default'] = isset($data['billing\_state\_default']) ? $data['billing\_state\_default'] : null;

$this->container['billing\_zip\_default'] = isset($data['billing\_zip\_default']) ? $data['billing\_zip\_default'] : null;

$this->container['po\_number'] = isset($data['po\_number']) ? $data['po\_number'] : null;

$this->container['invoice\_number'] = isset($data['invoice\_number']) ? $data['invoice\_number'] : null;

$this->container['business\_name'] = isset($data['business\_name']) ? $data['business\_name'] : null;

$this->container['customer\_number'] = isset($data['customer\_number']) ? $data['customer\_number'] : null;

$this->container['reference\_note'] = isset($data['reference\_note']) ? $data['reference\_note'] : null;

$this->container['default\_sec\_code'] = isset($data['default\_sec\_code']) ? $data['default\_sec\_code'] : null;

}

public function listInvalidProperties()

{

$invalidProperties = [];

return $invalidProperties;

}

public function valid()

{

return count($this->listInvalidProperties()) === 0;

}

public function getReader()

{

return $this->container['reader'];

}

public function setReader($reader)

{

$this->container['reader'] = $reader;

return $this;

}

public function getContactName()

{

return $this->container['contact\_name'];

}

public function setContactName($contact\_name)

{

$this->container['contact\_name'] = $contact\_name;

return $this;

}

public function getContactTitle()

{

return $this->container['contact\_title'];

}

public function setContactTitle($contact\_title)

{

$this->container['contact\_title'] = $contact\_title;

return $this;

}

public function getEmailAddress()

{

return $this->container['email\_address'];

}

public function setEmailAddress($email\_address)

{

$this->container['email\_address'] = $email\_address;

return $this;

}

public function getPhoneDefault()

{

return $this->container['phone\_default'];

}

public function setPhoneDefault($phone\_default)

{

$this->container['phone\_default'] = $phone\_default;

return $this;

}

public function getFax()

{

return $this->container['fax'];

}

public function setFax($fax)

{

$this->container['fax'] = $fax;

return $this;

}

public function getBillingAddressDefault()

{

return $this->container['billing\_address\_default'];

}

public function setBillingAddressDefault($billing\_address\_default)

{

$this->container['billing\_address\_default'] = $billing\_address\_default;

return $this;

}

public function getBillingAddress2()

{

return $this->container['billing\_address2'];

}

public function setBillingAddress2($billing\_address2)

{

$this->container['billing\_address2'] = $billing\_address2;

return $this;

}

public function getBillingCityDefault()

{

return $this->container['billing\_city\_default'];

}

public function setBillingCityDefault($billing\_city\_default)

{

$this->container['billing\_city\_default'] = $billing\_city\_default;

return $this;

}

public function getBillingStateDefault()

{

return $this->container['billing\_state\_default'];

}

public function setBillingStateDefault($billing\_state\_default)

{

$this->container['billing\_state\_default'] = $billing\_state\_default;

return $this;

}

public function getBillingZipDefault()

{

return $this->container['billing\_zip\_default'];

}

public function setBillingZipDefault($billing\_zip\_default)

{

$this->container['billing\_zip\_default'] = $billing\_zip\_default;

return $this;

}

public function getPoNumber()

{

return $this->container['po\_number'];

}

public function setPoNumber($po\_number)

{

$this->container['po\_number'] = $po\_number;

return $this;

}

public function getInvoiceNumber()

{

return $this->container['invoice\_number'];

}

public function setInvoiceNumber($invoice\_number)

{

$this->container['invoice\_number'] = $invoice\_number;

return $this;

}

public function getBusinessName()

{

return $this->container['business\_name'];

}

public function setBusinessName($business\_name)

{

$this->container['business\_name'] = $business\_name;

return $this;

}

public function getCustomerNumber()

{

return $this->container['customer\_number'];

}

public function setCustomerNumber($customer\_number)

{

$this->container['customer\_number'] = $customer\_number;

return $this;

}

public function getReferenceNote()

{

return $this->container['reference\_note'];

}

public function setReferenceNote($reference\_note)

{

$this->container['reference\_note'] = $reference\_note;

return $this;

}

public function getDefaultSecCode()

{

return $this->container['default\_sec\_code'];

}

public function setDefaultSecCode($default\_sec\_code)

{

$this->container['default\_sec\_code'] = $default\_sec\_code;

return $this;

}

public function offsetExists($offset)

{

return isset($this->container[$offset]);

}

public function offsetGet($offset)

{

return isset($this->container[$offset]) ? $this->container[$offset] : null;

}

public function offsetSet($offset, $value)

{

if (is\_null($offset)) {

$this->container[] = $value;

} else {

$this->container[$offset] = $value;

}

}

public function offsetUnset($offset)

{

unset($this->container[$offset]);

}

public function \_\_toString()

{

if (defined('JSON\_PRETTY\_PRINT')) { // use JSON pretty print

return json\_encode(

ObjectSerializer::sanitizeForSerialization($this),

JSON\_PRETTY\_PRINT

);

}

return json\_encode(ObjectSerializer::sanitizeForSerialization($this));

}

}

# CheckTransactionType

<?php

namespace Swagger\Client\Model;

use \Swagger\Client\ObjectSerializer;

class CheckTransactionType

{

const UNKNOWN = 'Unknown';

const AUTHORIZE = 'Authorize';

const VOID = 'Void';

const OVERRIDE = 'Override';

const PAYROLL = 'Payroll';

const RECURRING = 'Recurring';

const REFUND = 'Refund';

public static function getAllowableEnumValues()

{

return [

self::UNKNOWN,

self::AUTHORIZE,

self::VOID,

self::OVERRIDE,

self::PAYROLL,

self::RECURRING,

self::REFUND,

];

}

}

# Contact

<?php

namespace Swagger\Client\Model;

use \ArrayAccess;

use \Swagger\Client\ObjectSerializer;

class Contact implements ModelInterface, ArrayAccess

{

const DISCRIMINATOR = null;

protected static $swaggerModelName = 'Contact';

protected static $swaggerTypes = [

'first\_name' => 'string',

'last\_name' => 'string',

'name' => 'string',

'title' => 'string',

'phone' => 'string',

'email' => 'string'

];

protected static $swaggerFormats = [

'first\_name' => null,

'last\_name' => null,

'name' => null,

'title' => null,

'phone' => null,

'email' => null

];

public static function swaggerTypes()

{

return self::$swaggerTypes;

}

public static function swaggerFormats()

{

return self::$swaggerFormats;

}

protected static $attributeMap = [

'first\_name' => 'FirstName',

'last\_name' => 'LastName',

'name' => 'Name',

'title' => 'Title',

'phone' => 'Phone',

'email' => 'Email'

];

protected static $setters = [

'first\_name' => 'setFirstName',

'last\_name' => 'setLastName',

'name' => 'setName',

'title' => 'setTitle',

'phone' => 'setPhone',

'email' => 'setEmail'

];

protected static $getters = [

'first\_name' => 'getFirstName',

'last\_name' => 'getLastName',

'name' => 'getName',

'title' => 'getTitle',

'phone' => 'getPhone',

'email' => 'getEmail'

];

public static function attributeMap()

{

return self::$attributeMap;

}

public static function setters()

{

return self::$setters;

}

public static function getters()

{

return self::$getters;

}

public function getModelName()

{

return self::$swaggerModelName;

}

protected $container = [];

public function \_\_construct(array $data = null)

{

$this->container['first\_name'] = isset($data['first\_name']) ? $data['first\_name'] : null;

$this->container['last\_name'] = isset($data['last\_name']) ? $data['last\_name'] : null;

$this->container['name'] = isset($data['name']) ? $data['name'] : null;

$this->container['title'] = isset($data['title']) ? $data['title'] : null;

$this->container['phone'] = isset($data['phone']) ? $data['phone'] : null;

$this->container['email'] = isset($data['email']) ? $data['email'] : null;

}

public function listInvalidProperties()

{

$invalidProperties = [];

return $invalidProperties;

}

public function valid()

{

return count($this->listInvalidProperties()) === 0;

}

public function getFirstName()

{

return $this->container['first\_name'];

}

public function setFirstName($first\_name)

{

$this->container['first\_name'] = $first\_name;

return $this;

}

public function getLastName()

{

return $this->container['last\_name'];

}

public function setLastName($last\_name)

{

$this->container['last\_name'] = $last\_name;

return $this;

}

public function getName()

{

return $this->container['name'];

}

public function setName($name)

{

$this->container['name'] = $name;

return $this;

}

public function getTitle()

{

return $this->container['title'];

}

public function setTitle($title)

{

$this->container['title'] = $title;

return $this;

}

public function getPhone()

{

return $this->container['phone'];

}

public function setPhone($phone)

{

$this->container['phone'] = $phone;

return $this;

}

public function getEmail()

{

return $this->container['email'];

}

public function setEmail($email)

{

$this->container['email'] = $email;

return $this;

}

public function offsetExists($offset)

{

return isset($this->container[$offset]);

}

public function offsetGet($offset)

{

return isset($this->container[$offset]) ? $this->container[$offset] : null;

}

public function offsetSet($offset, $value)

{

if (is\_null($offset)) {

$this->container[] = $value;

} else {

$this->container[$offset] = $value;

}

}

public function offsetUnset($offset)

{

unset($this->container[$offset]);

}

public function \_\_toString()

{

if (defined('JSON\_PRETTY\_PRINT')) { // use JSON pretty print

return json\_encode(

ObjectSerializer::sanitizeForSerialization($this),

JSON\_PRETTY\_PRINT

);

}

return json\_encode(ObjectSerializer::sanitizeForSerialization($this));

}

}

# CreditCardTransactionSettings

<?php

namespace Swagger\Client\Model;

use \ArrayAccess;

use \Swagger\Client\ObjectSerializer;

class CreditCardTransactionSettings implements ModelInterface, ArrayAccess

{

const DISCRIMINATOR = null;

protected static $swaggerModelName = 'CreditCardTransactionSettings';

protected static $swaggerTypes = [

'cvv' => '\Swagger\Client\Model\FieldInfo',

'contact\_name' => '\Swagger\Client\Model\FieldInfo',

'contact\_title' => '\Swagger\Client\Model\FieldInfo',

'email\_address' => '\Swagger\Client\Model\FieldInfo',

'phone' => '\Swagger\Client\Model\FieldInfo',

'fax' => '\Swagger\Client\Model\FieldInfo',

'billing\_address' => '\Swagger\Client\Model\FieldInfo',

'billing\_address2' => '\Swagger\Client\Model\FieldInfo',

'billing\_city' => '\Swagger\Client\Model\FieldInfo',

'billing\_state' => '\Swagger\Client\Model\FieldInfo',

'billing\_zip' => '\Swagger\Client\Model\FieldInfo',

'po\_number' => '\Swagger\Client\Model\FieldInfo',

'invoice\_number' => '\Swagger\Client\Model\FieldInfo',

'business\_name' => '\Swagger\Client\Model\FieldInfo',

'customer\_number' => '\Swagger\Client\Model\FieldInfo',

'reference\_note' => '\Swagger\Client\Model\FieldInfo',

'sales\_tax' => '\Swagger\Client\Model\FieldInfo',

'tax\_exempt' => '\Swagger\Client\Model\FieldInfo',

'card\_tracks' => '\Swagger\Client\Model\FieldInfo',

'card\_zip' => '\Swagger\Client\Model\FieldInfo',

'card\_address' => '\Swagger\Client\Model\FieldInfo',

'item\_description' => 'string',

'default\_transaction\_type' => '\Swagger\Client\Model\TransactionType'

];

protected static $swaggerFormats = [

'cvv' => null,

'contact\_name' => null,

'contact\_title' => null,

'email\_address' => null,

'phone' => null,

'fax' => null,

'billing\_address' => null,

'billing\_address2' => null,

'billing\_city' => null,

'billing\_state' => null,

'billing\_zip' => null,

'po\_number' => null,

'invoice\_number' => null,

'business\_name' => null,

'customer\_number' => null,

'reference\_note' => null,

'sales\_tax' => null,

'tax\_exempt' => null,

'card\_tracks' => null,

'card\_zip' => null,

'card\_address' => null,

'item\_description' => null,

'default\_transaction\_type' => null

];

public static function swaggerTypes()

{

return self::$swaggerTypes;

}

public static function swaggerFormats()

{

return self::$swaggerFormats;

}

protected static $attributeMap = [

'cvv' => 'CVV',

'contact\_name' => 'ContactName',

'contact\_title' => 'ContactTitle',

'email\_address' => 'EmailAddress',

'phone' => 'Phone',

'fax' => 'Fax',

'billing\_address' => 'BillingAddress',

'billing\_address2' => 'BillingAddress2',

'billing\_city' => 'BillingCity',

'billing\_state' => 'BillingState',

'billing\_zip' => 'BillingZip',

'po\_number' => 'PONumber',

'invoice\_number' => 'InvoiceNumber',

'business\_name' => 'BusinessName',

'customer\_number' => 'CustomerNumber',

'reference\_note' => 'ReferenceNote',

'sales\_tax' => 'SalesTax',

'tax\_exempt' => 'TaxExempt',

'card\_tracks' => 'CardTracks',

'card\_zip' => 'CardZip',

'card\_address' => 'CardAddress',

'item\_description' => 'ItemDescription',

'default\_transaction\_type' => 'DefaultTransactionType'

];

protected static $setters = [

'cvv' => 'setCvv',

'contact\_name' => 'setContactName',

'contact\_title' => 'setContactTitle',

'email\_address' => 'setEmailAddress',

'phone' => 'setPhone',

'fax' => 'setFax',

'billing\_address' => 'setBillingAddress',

'billing\_address2' => 'setBillingAddress2',

'billing\_city' => 'setBillingCity',

'billing\_state' => 'setBillingState',

'billing\_zip' => 'setBillingZip',

'po\_number' => 'setPoNumber',

'invoice\_number' => 'setInvoiceNumber',

'business\_name' => 'setBusinessName',

'customer\_number' => 'setCustomerNumber',

'reference\_note' => 'setReferenceNote',

'sales\_tax' => 'setSalesTax',

'tax\_exempt' => 'setTaxExempt',

'card\_tracks' => 'setCardTracks',

'card\_zip' => 'setCardZip',

'card\_address' => 'setCardAddress',

'item\_description' => 'setItemDescription',

'default\_transaction\_type' => 'setDefaultTransactionType'

];

protected static $getters = [

'cvv' => 'getCvv',

'contact\_name' => 'getContactName',

'contact\_title' => 'getContactTitle',

'email\_address' => 'getEmailAddress',

'phone' => 'getPhone',

'fax' => 'getFax',

'billing\_address' => 'getBillingAddress',

'billing\_address2' => 'getBillingAddress2',

'billing\_city' => 'getBillingCity',

'billing\_state' => 'getBillingState',

'billing\_zip' => 'getBillingZip',

'po\_number' => 'getPoNumber',

'invoice\_number' => 'getInvoiceNumber',

'business\_name' => 'getBusinessName',

'customer\_number' => 'getCustomerNumber',

'reference\_note' => 'getReferenceNote',

'sales\_tax' => 'getSalesTax',

'tax\_exempt' => 'getTaxExempt',

'card\_tracks' => 'getCardTracks',

'card\_zip' => 'getCardZip',

'card\_address' => 'getCardAddress',

'item\_description' => 'getItemDescription',

'default\_transaction\_type' => 'getDefaultTransactionType'

];

public static function attributeMap()

{

return self::$attributeMap;

}

public static function setters()

{

return self::$setters;

}

public static function getters()

{

return self::$getters;

}

public function getModelName()

{

return self::$swaggerModelName;

}

protected $container = [];

public function \_\_construct(array $data = null)

{

$this->container['cvv'] = isset($data['cvv']) ? $data['cvv'] : null;

$this->container['contact\_name'] = isset($data['contact\_name']) ? $data['contact\_name'] : null;

$this->container['contact\_title'] = isset($data['contact\_title']) ? $data['contact\_title'] : null;

$this->container['email\_address'] = isset($data['email\_address']) ? $data['email\_address'] : null;

$this->container['phone'] = isset($data['phone']) ? $data['phone'] : null;

$this->container['fax'] = isset($data['fax']) ? $data['fax'] : null;

$this->container['billing\_address'] = isset($data['billing\_address']) ? $data['billing\_address'] : null;

$this->container['billing\_address2'] = isset($data['billing\_address2']) ? $data['billing\_address2'] : null;

$this->container['billing\_city'] = isset($data['billing\_city']) ? $data['billing\_city'] : null;

$this->container['billing\_state'] = isset($data['billing\_state']) ? $data['billing\_state'] : null;

$this->container['billing\_zip'] = isset($data['billing\_zip']) ? $data['billing\_zip'] : null;

$this->container['po\_number'] = isset($data['po\_number']) ? $data['po\_number'] : null;

$this->container['invoice\_number'] = isset($data['invoice\_number']) ? $data['invoice\_number'] : null;

$this->container['business\_name'] = isset($data['business\_name']) ? $data['business\_name'] : null;

$this->container['customer\_number'] = isset($data['customer\_number']) ? $data['customer\_number'] : null;

$this->container['reference\_note'] = isset($data['reference\_note']) ? $data['reference\_note'] : null;

$this->container['sales\_tax'] = isset($data['sales\_tax']) ? $data['sales\_tax'] : null;

$this->container['tax\_exempt'] = isset($data['tax\_exempt']) ? $data['tax\_exempt'] : null;

$this->container['card\_tracks'] = isset($data['card\_tracks']) ? $data['card\_tracks'] : null;

$this->container['card\_zip'] = isset($data['card\_zip']) ? $data['card\_zip'] : null;

$this->container['card\_address'] = isset($data['card\_address']) ? $data['card\_address'] : null;

$this->container['item\_description'] = isset($data['item\_description']) ? $data['item\_description'] : null;

$this->container['default\_transaction\_type'] = isset($data['default\_transaction\_type']) ? $data['default\_transaction\_type'] : null;

}

public function listInvalidProperties()

{

$invalidProperties = [];

return $invalidProperties;

}

public function valid()

{

return count($this->listInvalidProperties()) === 0;

}

public function getCvv()

{

return $this->container['cvv'];

}

public function setCvv($cvv)

{

$this->container['cvv'] = $cvv;

return $this;

}

public function getContactName()

{

return $this->container['contact\_name'];

}

public function setContactName($contact\_name)

{

$this->container['contact\_name'] = $contact\_name;

return $this;

}

public function getContactTitle()

{

return $this->container['contact\_title'];

}

public function setContactTitle($contact\_title)

{

$this->container['contact\_title'] = $contact\_title;

return $this;

}

public function getEmailAddress()

{

return $this->container['email\_address'];

}

public function setEmailAddress($email\_address)

{

$this->container['email\_address'] = $email\_address;

return $this;

}

public function getPhone()

{

return $this->container['phone'];

}

public function setPhone($phone)

{

$this->container['phone'] = $phone;

return $this;

}

public function getFax()

{

return $this->container['fax'];

}

public function setFax($fax)

{

$this->container['fax'] = $fax;

return $this;

}

public function getBillingAddress()

{

return $this->container['billing\_address'];

}

public function setBillingAddress($billing\_address)

{

$this->container['billing\_address'] = $billing\_address;

return $this;

}

public function getBillingAddress2()

{

return $this->container['billing\_address2'];

}

public function setBillingAddress2($billing\_address2)

{

$this->container['billing\_address2'] = $billing\_address2;

return $this;

}

public function getBillingCity()

{

return $this->container['billing\_city'];

}

public function setBillingCity($billing\_city)

{

$this->container['billing\_city'] = $billing\_city;

return $this;

}

public function getBillingState()

{

return $this->container['billing\_state'];

}

public function setBillingState($billing\_state)

{

$this->container['billing\_state'] = $billing\_state;

return $this;

}

public function getBillingZip()

{

return $this->container['billing\_zip'];

}

public function setBillingZip($billing\_zip)

{

$this->container['billing\_zip'] = $billing\_zip;

return $this;

}

public function getPoNumber()

{

return $this->container['po\_number'];

}

public function setPoNumber($po\_number)

{

$this->container['po\_number'] = $po\_number;

return $this;

}

public function getInvoiceNumber()

{

return $this->container['invoice\_number'];

}

public function setInvoiceNumber($invoice\_number)

{

$this->container['invoice\_number'] = $invoice\_number;

return $this;

}

public function getBusinessName()

{

return $this->container['business\_name'];

}

public function setBusinessName($business\_name)

{

$this->container['business\_name'] = $business\_name;

return $this;

}

public function getCustomerNumber()

{

return $this->container['customer\_number'];

}

public function setCustomerNumber($customer\_number)

{

$this->container['customer\_number'] = $customer\_number;

return $this;

}

public function getReferenceNote()

{

return $this->container['reference\_note'];

}

public function setReferenceNote($reference\_note)

{

$this->container['reference\_note'] = $reference\_note;

return $this;

}

public function getSalesTax()

{

return $this->container['sales\_tax'];

}

public function setSalesTax($sales\_tax)

{

$this->container['sales\_tax'] = $sales\_tax;

return $this;

}

public function getTaxExempt()

{

return $this->container['tax\_exempt'];

}

public function setTaxExempt($tax\_exempt)

{

$this->container['tax\_exempt'] = $tax\_exempt;

return $this;

}

public function getCardTracks()

{

return $this->container['card\_tracks'];

}

public function setCardTracks($card\_tracks)

{

$this->container['card\_tracks'] = $card\_tracks;

return $this;

}

public function getCardZip()

{

return $this->container['card\_zip'];

}

public function setCardZip($card\_zip)

{

$this->container['card\_zip'] = $card\_zip;

return $this;

}

public function getCardAddress()

{

return $this->container['card\_address'];

}

public function setCardAddress($card\_address)

{

$this->container['card\_address'] = $card\_address;

return $this;

}

public function getItemDescription()

{

return $this->container['item\_description'];

}

public function setItemDescription($item\_description)

{

$this->container['item\_description'] = $item\_description;

return $this;

}

public function getDefaultTransactionType()

{

return $this->container['default\_transaction\_type'];

}

public function setDefaultTransactionType($default\_transaction\_type)

{

$this->container['default\_transaction\_type'] = $default\_transaction\_type;

return $this;

}

public function offsetExists($offset)

{

return isset($this->container[$offset]);

}

public function offsetGet($offset)

{

return isset($this->container[$offset]) ? $this->container[$offset] : null;

}

public function offsetSet($offset, $value)

{

if (is\_null($offset)) {

$this->container[] = $value;

} else {

$this->container[$offset] = $value;

}

}

public function offsetUnset($offset)

{

unset($this->container[$offset]);

}

public function \_\_toString()

{

if (defined('JSON\_PRETTY\_PRINT')) { // use JSON pretty print

return json\_encode(

ObjectSerializer::sanitizeForSerialization($this),

JSON\_PRETTY\_PRINT

);

}

return json\_encode(ObjectSerializer::sanitizeForSerialization($this));

}

}

# CVVPresenceType

<?php

namespace Swagger\Client\Model;

use \Swagger\Client\ObjectSerializer;

class CVVPresenceType

{

const NORMAL = 'Normal';

const BYPASS = 'Bypass';

const ILLEGIBLE = 'Illegible';

const NOT\_AVAILABLE = 'NotAvailable';

public static function getAllowableEnumValues()

{

return [

self::NORMAL,

self::BYPASS,

self::ILLEGIBLE,

self::NOT\_AVAILABLE,

];

}

}

# EmailReceipt

<?php

namespace Swagger\Client\Model;

use \ArrayAccess;

use \Swagger\Client\ObjectSerializer;

class EmailReceipt implements ModelInterface, ArrayAccess

{

const DISCRIMINATOR = null;

protected static $swaggerModelName = 'EmailReceipt';

protected static $swaggerTypes = [

'transaction\_id' => 'string',

'to\_addresses' => 'string',

'subject' => 'string',

'header' => 'string',

'footer' => 'string'

];

protected static $swaggerFormats = [

'transaction\_id' => null,

'to\_addresses' => null,

'subject' => null,

'header' => null,

'footer' => null

];

public static function swaggerTypes()

{

return self::$swaggerTypes;

}

public static function swaggerFormats()

{

return self::$swaggerFormats;

}

protected static $attributeMap = [

'transaction\_id' => 'TransactionID',

'to\_addresses' => 'ToAddresses',

'subject' => 'Subject',

'header' => 'Header',

'footer' => 'Footer'

];

protected static $setters = [

'transaction\_id' => 'setTransactionId',

'to\_addresses' => 'setToAddresses',

'subject' => 'setSubject',

'header' => 'setHeader',

'footer' => 'setFooter'

];

protected static $getters = [

'transaction\_id' => 'getTransactionId',

'to\_addresses' => 'getToAddresses',

'subject' => 'getSubject',

'header' => 'getHeader',

'footer' => 'getFooter'

];

public static function attributeMap()

{

return self::$attributeMap;

}

public static function setters()

{

return self::$setters;

}

public static function getters()

{

return self::$getters;

}

public function getModelName()

{

return self::$swaggerModelName;

}

protected $container = [];

public function \_\_construct(array $data = null)

{

$this->container['transaction\_id'] = isset($data['transaction\_id']) ? $data['transaction\_id'] : null;

$this->container['to\_addresses'] = isset($data['to\_addresses']) ? $data['to\_addresses'] : null;

$this->container['subject'] = isset($data['subject']) ? $data['subject'] : null;

$this->container['header'] = isset($data['header']) ? $data['header'] : null;

$this->container['footer'] = isset($data['footer']) ? $data['footer'] : null;

}

public function listInvalidProperties()

{

$invalidProperties = [];

return $invalidProperties;

}

public function valid()

{

return count($this->listInvalidProperties()) === 0;

}

public function getTransactionId()

{

return $this->container['transaction\_id'];

}

public function setTransactionId($transaction\_id)

{

$this->container['transaction\_id'] = $transaction\_id;

return $this;

}

public function getToAddresses()

{

return $this->container['to\_addresses'];

}

public function setToAddresses($to\_addresses)

{

$this->container['to\_addresses'] = $to\_addresses;

return $this;

}

public function getSubject()

{

return $this->container['subject'];

}

public function setSubject($subject)

{

$this->container['subject'] = $subject;

return $this;

}

public function getHeader()

{

return $this->container['header'];

}

public function setHeader($header)

{

$this->container['header'] = $header;

return $this;

}

public function getFooter()

{

return $this->container['footer'];

}

public function setFooter($footer)

{

$this->container['footer'] = $footer;

return $this;

}

public function offsetExists($offset)

{

return isset($this->container[$offset]);

}

public function offsetGet($offset)

{

return isset($this->container[$offset]) ? $this->container[$offset] : null;

}

public function offsetSet($offset, $value)

{

if (is\_null($offset)) {

$this->container[] = $value;

} else {

$this->container[$offset] = $value;

}

}

public function offsetUnset($offset)

{

unset($this->container[$offset]);

}

public function \_\_toString()

{

if (defined('JSON\_PRETTY\_PRINT')) { // use JSON pretty print

return json\_encode(

ObjectSerializer::sanitizeForSerialization($this),

JSON\_PRETTY\_PRINT

);

}

return json\_encode(ObjectSerializer::sanitizeForSerialization($this));

}

}

# FieldAccess

<?php

namespace Swagger\Client\Model;

use \Swagger\Client\ObjectSerializer;

class FieldAccess

{

const DISABLED = 'Disabled';

const ENABLED = 'Enabled';

const REQUIRED = 'Required';

public static function getAllowableEnumValues()

{

return [

self::DISABLED,

self::ENABLED,

self::REQUIRED,

];

}

}

# FieldInfo

<?php

namespace Swagger\Client\Model;

use \ArrayAccess;

use \Swagger\Client\ObjectSerializer;

class FieldInfo implements ModelInterface, ArrayAccess

{

const DISCRIMINATOR = null;

protected static $swaggerModelName = 'FieldInfo';

protected static $swaggerTypes = [

'access' => '\Swagger\Client\Model\FieldAccess',

'default' => 'string'

];

protected static $swaggerFormats = [

'access' => null,

'default' => null

];

public static function swaggerTypes()

{

return self::$swaggerTypes;

}

public static function swaggerFormats()

{

return self::$swaggerFormats;

}

protected static $attributeMap = [

'access' => 'Access',

'default' => 'Default'

];

protected static $setters = [

'access' => 'setAccess',

'default' => 'setDefault'

];

protected static $getters = [

'access' => 'getAccess',

'default' => 'getDefault'

];

public static function attributeMap()

{

return self::$attributeMap;

}

public static function setters()

{

return self::$setters;

}

public static function getters()

{

return self::$getters;

}

public function getModelName()

{

return self::$swaggerModelName;

}

protected $container = [];

public function \_\_construct(array $data = null)

{

$this->container['access'] = isset($data['access']) ? $data['access'] : null;

$this->container['default'] = isset($data['default']) ? $data['default'] : null;

}

public function listInvalidProperties()

{

$invalidProperties = [];

return $invalidProperties;

}

public function valid()

{

return count($this->listInvalidProperties()) === 0;

}

public function getAccess()

{

return $this->container['access'];

}

public function setAccess($access)

{

$this->container['access'] = $access;

return $this;

}

public function getDefault()

{

return $this->container['default'];

}

public function setDefault($default)

{

$this->container['default'] = $default;

return $this;

}

public function offsetExists($offset)

{

return isset($this->container[$offset]);

}

public function offsetGet($offset)

{

return isset($this->container[$offset]) ? $this->container[$offset] : null;

}

public function offsetSet($offset, $value)

{

if (is\_null($offset)) {

$this->container[] = $value;

} else {

$this->container[$offset] = $value;

}

}

public function offsetUnset($offset)

{

unset($this->container[$offset]);

}

public function \_\_toString()

{

if (defined('JSON\_PRETTY\_PRINT')) { // use JSON pretty print

return json\_encode(

ObjectSerializer::sanitizeForSerialization($this),

JSON\_PRETTY\_PRINT

);

}

return json\_encode(ObjectSerializer::sanitizeForSerialization($this));

}

}

# QueryTransaction

<?php

namespace Swagger\Client\Model;

use \ArrayAccess;

use \Swagger\Client\ObjectSerializer;

class QueryTransaction implements ModelInterface, ArrayAccess

{

const DISCRIMINATOR = null;

protected static $swaggerModelName = 'QueryTransaction';

protected static $swaggerTypes = [

'tran\_sequence' => 'string',

'tran\_batch' => 'string',

'tran\_modified' => 'string',

'tran\_processed' => 'string',

'tran\_settled' => 'string',

'submit\_time' => 'string',

'tran\_type' => 'string',

'tran\_amt' => 'double',

'tran\_tax' => 'double',

'tran\_inv' => 'string',

'tran\_po' => 'string',

'tran\_c\_num' => 'string',

'tran\_note' => 'string',

'card\_num' => 'string',

'card\_name' => 'string',

'card\_exp' => 'string',

'avs\_street' => 'string',

'avs\_zip' => 'string',

'cvv\_num' => 'string',

'proc\_id' => 'string',

'proc\_resp' => 'string',

'proc\_code' => 'string',

'proc\_mess' => 'string',

'avs\_code' => 'string',

'avs\_resp' => 'string',

'cvv\_code' => 'string',

'cvv\_resp' => 'string',

'items' => '\Swagger\Client\Model\TransactionItem[]',

'business\_name' => 'string',

'business\_address' => 'string',

'business\_address2' => 'string',

'business\_city' => 'string',

'business\_state' => 'string',

'business\_zip' => 'string',

'business\_country' => 'string',

'business\_website' => 'string',

'contact\_first\_name' => 'string',

'contact\_last\_name' => 'string',

'contact\_phone' => 'string',

'contact\_fax' => 'string',

'contact\_email' => 'string',

'contact\_title' => 'string',

'ip' => 'string',

'tax\_exempt' => 'bool'

];

protected static $swaggerFormats = [

'tran\_sequence' => null,

'tran\_batch' => null,

'tran\_modified' => null,

'tran\_processed' => null,

'tran\_settled' => null,

'submit\_time' => null,

'tran\_type' => null,

'tran\_amt' => 'double',

'tran\_tax' => 'double',

'tran\_inv' => null,

'tran\_po' => null,

'tran\_c\_num' => null,

'tran\_note' => null,

'card\_num' => null,

'card\_name' => null,

'card\_exp' => null,

'avs\_street' => null,

'avs\_zip' => null,

'cvv\_num' => null,

'proc\_id' => null,

'proc\_resp' => null,

'proc\_code' => null,

'proc\_mess' => null,

'avs\_code' => null,

'avs\_resp' => null,

'cvv\_code' => null,

'cvv\_resp' => null,

'items' => null,

'business\_name' => null,

'business\_address' => null,

'business\_address2' => null,

'business\_city' => null,

'business\_state' => null,

'business\_zip' => null,

'business\_country' => null,

'business\_website' => null,

'contact\_first\_name' => null,

'contact\_last\_name' => null,

'contact\_phone' => null,

'contact\_fax' => null,

'contact\_email' => null,

'contact\_title' => null,

'ip' => null,

'tax\_exempt' => null

];

public static function swaggerTypes()

{

return self::$swaggerTypes;

}

public static function swaggerFormats()

{

return self::$swaggerFormats;

}

protected static $attributeMap = [

'tran\_sequence' => 'Tran\_Sequence',

'tran\_batch' => 'Tran\_Batch',

'tran\_modified' => 'Tran\_Modified',

'tran\_processed' => 'Tran\_Processed',

'tran\_settled' => 'Tran\_Settled',

'submit\_time' => 'Submit\_Time',

'tran\_type' => 'Tran\_Type',

'tran\_amt' => 'Tran\_Amt',

'tran\_tax' => 'Tran\_Tax',

'tran\_inv' => 'Tran\_Inv',

'tran\_po' => 'Tran\_PO',

'tran\_c\_num' => 'Tran\_CNum',

'tran\_note' => 'Tran\_Note',

'card\_num' => 'Card\_Num',

'card\_name' => 'Card\_Name',

'card\_exp' => 'Card\_Exp',

'avs\_street' => 'AVS\_Street',

'avs\_zip' => 'AVS\_Zip',

'cvv\_num' => 'CVV\_Num',

'proc\_id' => 'Proc\_ID',

'proc\_resp' => 'Proc\_Resp',

'proc\_code' => 'Proc\_Code',

'proc\_mess' => 'Proc\_Mess',

'avs\_code' => 'AVS\_Code',

'avs\_resp' => 'AVS\_Resp',

'cvv\_code' => 'CVV\_Code',

'cvv\_resp' => 'CVV\_Resp',

'items' => 'Items',

'business\_name' => 'Business\_Name',

'business\_address' => 'Business\_Address',

'business\_address2' => 'Business\_Address2',

'business\_city' => 'Business\_City',

'business\_state' => 'Business\_State',

'business\_zip' => 'Business\_Zip',

'business\_country' => 'Business\_Country',

'business\_website' => 'Business\_Website',

'contact\_first\_name' => 'Contact\_FirstName',

'contact\_last\_name' => 'Contact\_LastName',

'contact\_phone' => 'Contact\_Phone',

'contact\_fax' => 'Contact\_Fax',

'contact\_email' => 'Contact\_Email',

'contact\_title' => 'Contact\_Title',

'ip' => 'IP',

'tax\_exempt' => 'TaxExempt'

];

protected static $setters = [

'tran\_sequence' => 'setTranSequence',

'tran\_batch' => 'setTranBatch',

'tran\_modified' => 'setTranModified',

'tran\_processed' => 'setTranProcessed',

'tran\_settled' => 'setTranSettled',

'submit\_time' => 'setSubmitTime',

'tran\_type' => 'setTranType',

'tran\_amt' => 'setTranAmt',

'tran\_tax' => 'setTranTax',

'tran\_inv' => 'setTranInv',

'tran\_po' => 'setTranPo',

'tran\_c\_num' => 'setTranCNum',

'tran\_note' => 'setTranNote',

'card\_num' => 'setCardNum',

'card\_name' => 'setCardName',

'card\_exp' => 'setCardExp',

'avs\_street' => 'setAvsStreet',

'avs\_zip' => 'setAvsZip',

'cvv\_num' => 'setCvvNum',

'proc\_id' => 'setProcId',

'proc\_resp' => 'setProcResp',

'proc\_code' => 'setProcCode',

'proc\_mess' => 'setProcMess',

'avs\_code' => 'setAvsCode',

'avs\_resp' => 'setAvsResp',

'cvv\_code' => 'setCvvCode',

'cvv\_resp' => 'setCvvResp',

'items' => 'setItems',

'business\_name' => 'setBusinessName',

'business\_address' => 'setBusinessAddress',

'business\_address2' => 'setBusinessAddress2',

'business\_city' => 'setBusinessCity',

'business\_state' => 'setBusinessState',

'business\_zip' => 'setBusinessZip',

'business\_country' => 'setBusinessCountry',

'business\_website' => 'setBusinessWebsite',

'contact\_first\_name' => 'setContactFirstName',

'contact\_last\_name' => 'setContactLastName',

'contact\_phone' => 'setContactPhone',

'contact\_fax' => 'setContactFax',

'contact\_email' => 'setContactEmail',

'contact\_title' => 'setContactTitle',

'ip' => 'setIp',

'tax\_exempt' => 'setTaxExempt'

];

protected static $getters = [

'tran\_sequence' => 'getTranSequence',

'tran\_batch' => 'getTranBatch',

'tran\_modified' => 'getTranModified',

'tran\_processed' => 'getTranProcessed',

'tran\_settled' => 'getTranSettled',

'submit\_time' => 'getSubmitTime',

'tran\_type' => 'getTranType',

'tran\_amt' => 'getTranAmt',

'tran\_tax' => 'getTranTax',

'tran\_inv' => 'getTranInv',

'tran\_po' => 'getTranPo',

'tran\_c\_num' => 'getTranCNum',

'tran\_note' => 'getTranNote',

'card\_num' => 'getCardNum',

'card\_name' => 'getCardName',

'card\_exp' => 'getCardExp',

'avs\_street' => 'getAvsStreet',

'avs\_zip' => 'getAvsZip',

'cvv\_num' => 'getCvvNum',

'proc\_id' => 'getProcId',

'proc\_resp' => 'getProcResp',

'proc\_code' => 'getProcCode',

'proc\_mess' => 'getProcMess',

'avs\_code' => 'getAvsCode',

'avs\_resp' => 'getAvsResp',

'cvv\_code' => 'getCvvCode',

'cvv\_resp' => 'getCvvResp',

'items' => 'getItems',

'business\_name' => 'getBusinessName',

'business\_address' => 'getBusinessAddress',

'business\_address2' => 'getBusinessAddress2',

'business\_city' => 'getBusinessCity',

'business\_state' => 'getBusinessState',

'business\_zip' => 'getBusinessZip',

'business\_country' => 'getBusinessCountry',

'business\_website' => 'getBusinessWebsite',

'contact\_first\_name' => 'getContactFirstName',

'contact\_last\_name' => 'getContactLastName',

'contact\_phone' => 'getContactPhone',

'contact\_fax' => 'getContactFax',

'contact\_email' => 'getContactEmail',

'contact\_title' => 'getContactTitle',

'ip' => 'getIp',

'tax\_exempt' => 'getTaxExempt'

];

public static function attributeMap()

{

return self::$attributeMap;

}

public static function setters()

{

return self::$setters;

}

public static function getters()

{

return self::$getters;

}

public function getModelName()

{

return self::$swaggerModelName;

}

protected $container = [];

public function \_\_construct(array $data = null)

{

$this->container['tran\_sequence'] = isset($data['tran\_sequence']) ? $data['tran\_sequence'] : null;

$this->container['tran\_batch'] = isset($data['tran\_batch']) ? $data['tran\_batch'] : null;

$this->container['tran\_modified'] = isset($data['tran\_modified']) ? $data['tran\_modified'] : null;

$this->container['tran\_processed'] = isset($data['tran\_processed']) ? $data['tran\_processed'] : null;

$this->container['tran\_settled'] = isset($data['tran\_settled']) ? $data['tran\_settled'] : null;

$this->container['submit\_time'] = isset($data['submit\_time']) ? $data['submit\_time'] : null;

$this->container['tran\_type'] = isset($data['tran\_type']) ? $data['tran\_type'] : null;

$this->container['tran\_amt'] = isset($data['tran\_amt']) ? $data['tran\_amt'] : null;

$this->container['tran\_tax'] = isset($data['tran\_tax']) ? $data['tran\_tax'] : null;

$this->container['tran\_inv'] = isset($data['tran\_inv']) ? $data['tran\_inv'] : null;

$this->container['tran\_po'] = isset($data['tran\_po']) ? $data['tran\_po'] : null;

$this->container['tran\_c\_num'] = isset($data['tran\_c\_num']) ? $data['tran\_c\_num'] : null;

$this->container['tran\_note'] = isset($data['tran\_note']) ? $data['tran\_note'] : null;

$this->container['card\_num'] = isset($data['card\_num']) ? $data['card\_num'] : null;

$this->container['card\_name'] = isset($data['card\_name']) ? $data['card\_name'] : null;

$this->container['card\_exp'] = isset($data['card\_exp']) ? $data['card\_exp'] : null;

$this->container['avs\_street'] = isset($data['avs\_street']) ? $data['avs\_street'] : null;

$this->container['avs\_zip'] = isset($data['avs\_zip']) ? $data['avs\_zip'] : null;

$this->container['cvv\_num'] = isset($data['cvv\_num']) ? $data['cvv\_num'] : null;

$this->container['proc\_id'] = isset($data['proc\_id']) ? $data['proc\_id'] : null;

$this->container['proc\_resp'] = isset($data['proc\_resp']) ? $data['proc\_resp'] : null;

$this->container['proc\_code'] = isset($data['proc\_code']) ? $data['proc\_code'] : null;

$this->container['proc\_mess'] = isset($data['proc\_mess']) ? $data['proc\_mess'] : null;

$this->container['avs\_code'] = isset($data['avs\_code']) ? $data['avs\_code'] : null;

$this->container['avs\_resp'] = isset($data['avs\_resp']) ? $data['avs\_resp'] : null;

$this->container['cvv\_code'] = isset($data['cvv\_code']) ? $data['cvv\_code'] : null;

$this->container['cvv\_resp'] = isset($data['cvv\_resp']) ? $data['cvv\_resp'] : null;

$this->container['items'] = isset($data['items']) ? $data['items'] : null;

$this->container['business\_name'] = isset($data['business\_name']) ? $data['business\_name'] : null;

$this->container['business\_address'] = isset($data['business\_address']) ? $data['business\_address'] : null;

$this->container['business\_address2'] = isset($data['business\_address2']) ? $data['business\_address2'] : null;

$this->container['business\_city'] = isset($data['business\_city']) ? $data['business\_city'] : null;

$this->container['business\_state'] = isset($data['business\_state']) ? $data['business\_state'] : null;

$this->container['business\_zip'] = isset($data['business\_zip']) ? $data['business\_zip'] : null;

$this->container['business\_country'] = isset($data['business\_country']) ? $data['business\_country'] : null;

$this->container['business\_website'] = isset($data['business\_website']) ? $data['business\_website'] : null;

$this->container['contact\_first\_name'] = isset($data['contact\_first\_name']) ? $data['contact\_first\_name'] : null;

$this->container['contact\_last\_name'] = isset($data['contact\_last\_name']) ? $data['contact\_last\_name'] : null;

$this->container['contact\_phone'] = isset($data['contact\_phone']) ? $data['contact\_phone'] : null;

$this->container['contact\_fax'] = isset($data['contact\_fax']) ? $data['contact\_fax'] : null;

$this->container['contact\_email'] = isset($data['contact\_email']) ? $data['contact\_email'] : null;

$this->container['contact\_title'] = isset($data['contact\_title']) ? $data['contact\_title'] : null;

$this->container['ip'] = isset($data['ip']) ? $data['ip'] : null;

$this->container['tax\_exempt'] = isset($data['tax\_exempt']) ? $data['tax\_exempt'] : null;

}

public function listInvalidProperties()

{

$invalidProperties = [];

return $invalidProperties;

}

public function valid()

{

return count($this->listInvalidProperties()) === 0;

}

public function getTranSequence()

{

return $this->container['tran\_sequence'];

}

public function setTranSequence($tran\_sequence)

{

$this->container['tran\_sequence'] = $tran\_sequence;

return $this;

}

public function getTranBatch()

{

return $this->container['tran\_batch'];

}

public function setTranBatch($tran\_batch)

{

$this->container['tran\_batch'] = $tran\_batch;

return $this;

}

public function getTranModified()

{

return $this->container['tran\_modified'];

}

public function setTranModified($tran\_modified)

{

$this->container['tran\_modified'] = $tran\_modified;

return $this;

}

public function getTranProcessed()

{

return $this->container['tran\_processed'];

}

public function setTranProcessed($tran\_processed)

{

$this->container['tran\_processed'] = $tran\_processed;

return $this;

}

public function getTranSettled()

{

return $this->container['tran\_settled'];

}

public function setTranSettled($tran\_settled)

{

$this->container['tran\_settled'] = $tran\_settled;

return $this;

}

public function getSubmitTime()

{

return $this->container['submit\_time'];

}

public function setSubmitTime($submit\_time)

{

$this->container['submit\_time'] = $submit\_time;

return $this;

}

public function getTranType()

{

return $this->container['tran\_type'];

}

public function setTranType($tran\_type)

{

$this->container['tran\_type'] = $tran\_type;

return $this;

}

public function getTranAmt()

{

return $this->container['tran\_amt'];

}

public function setTranAmt($tran\_amt)

{

$this->container['tran\_amt'] = $tran\_amt;

return $this;

}

public function getTranTax()

{

return $this->container['tran\_tax'];

}

public function setTranTax($tran\_tax)

{

$this->container['tran\_tax'] = $tran\_tax;

return $this;

}

public function getTranInv()

{

return $this->container['tran\_inv'];

}

public function setTranInv($tran\_inv)

{

$this->container['tran\_inv'] = $tran\_inv;

return $this;

}

public function getTranPo()

{

return $this->container['tran\_po'];

}

public function setTranPo($tran\_po)

{

$this->container['tran\_po'] = $tran\_po;

return $this;

}

public function getTranCNum()

{

return $this->container['tran\_c\_num'];

}

public function setTranCNum($tran\_c\_num)

{

$this->container['tran\_c\_num'] = $tran\_c\_num;

return $this;

}

public function getTranNote()

{

return $this->container['tran\_note'];

}

public function setTranNote($tran\_note)

{

$this->container['tran\_note'] = $tran\_note;

return $this;

}

public function getCardNum()

{

return $this->container['card\_num'];

}

public function setCardNum($card\_num)

{

$this->container['card\_num'] = $card\_num;

return $this;

}

public function getCardName()

{

return $this->container['card\_name'];

}

public function setCardName($card\_name)

{

$this->container['card\_name'] = $card\_name;

return $this;

}

public function getCardExp()

{

return $this->container['card\_exp'];

}

public function setCardExp($card\_exp)

{

$this->container['card\_exp'] = $card\_exp;

return $this;

}

public function getAvsStreet()

{

return $this->container['avs\_street'];

}

public function setAvsStreet($avs\_street)

{

$this->container['avs\_street'] = $avs\_street;

return $this;

}

public function getAvsZip()

{

return $this->container['avs\_zip'];

}

public function setAvsZip($avs\_zip)

{

$this->container['avs\_zip'] = $avs\_zip;

return $this;

}

public function getCvvNum()

{

return $this->container['cvv\_num'];

}

public function setCvvNum($cvv\_num)

{

$this->container['cvv\_num'] = $cvv\_num;

return $this;

}

public function getProcId()

{

return $this->container['proc\_id'];

}

public function setProcId($proc\_id)

{

$this->container['proc\_id'] = $proc\_id;

return $this;

}

public function getProcResp()

{

return $this->container['proc\_resp'];

}

public function setProcResp($proc\_resp)

{

$this->container['proc\_resp'] = $proc\_resp;

return $this;

}

public function getProcCode()

{

return $this->container['proc\_code'];

}

public function setProcCode($proc\_code)

{

$this->container['proc\_code'] = $proc\_code;

return $this;

}

public function getProcMess()

{

return $this->container['proc\_mess'];

}

public function setProcMess($proc\_mess)

{

$this->container['proc\_mess'] = $proc\_mess;

return $this;

}

public function getAvsCode()

{

return $this->container['avs\_code'];

}

public function setAvsCode($avs\_code)

{

$this->container['avs\_code'] = $avs\_code;

return $this;

}

public function getAvsResp()

{

return $this->container['avs\_resp'];

}

public function setAvsResp($avs\_resp)

{

$this->container['avs\_resp'] = $avs\_resp;

return $this;

}

public function getCvvCode()

{

return $this->container['cvv\_code'];

}

public function setCvvCode($cvv\_code)

{

$this->container['cvv\_code'] = $cvv\_code;

return $this;

}

public function getCvvResp()

{

return $this->container['cvv\_resp'];

}

public function setCvvResp($cvv\_resp)

{

$this->container['cvv\_resp'] = $cvv\_resp;

return $this;

}

public function getItems()

{

return $this->container['items'];

}

public function setItems($items)

{

$this->container['items'] = $items;

return $this;

}

public function getBusinessName()

{

return $this->container['business\_name'];

}

public function setBusinessName($business\_name)

{

$this->container['business\_name'] = $business\_name;

return $this;

}

public function getBusinessAddress()

{

return $this->container['business\_address'];

}

public function setBusinessAddress($business\_address)

{

$this->container['business\_address'] = $business\_address;

return $this;

}

public function getBusinessAddress2()

{

return $this->container['business\_address2'];

}

public function setBusinessAddress2($business\_address2)

{

$this->container['business\_address2'] = $business\_address2;

return $this;

}

public function getBusinessCity()

{

return $this->container['business\_city'];

}

public function setBusinessCity($business\_city)

{

$this->container['business\_city'] = $business\_city;

return $this;

}

public function getBusinessState()

{

return $this->container['business\_state'];

}

public function setBusinessState($business\_state)

{

$this->container['business\_state'] = $business\_state;

return $this;

}

public function getBusinessZip()

{

return $this->container['business\_zip'];

}

public function setBusinessZip($business\_zip)

{

$this->container['business\_zip'] = $business\_zip;

return $this;

}

public function getBusinessCountry()

{

return $this->container['business\_country'];

}

public function setBusinessCountry($business\_country)

{

$this->container['business\_country'] = $business\_country;

return $this;

}

public function getBusinessWebsite()

{

return $this->container['business\_website'];

}

public function setBusinessWebsite($business\_website)

{

$this->container['business\_website'] = $business\_website;

return $this;

}

public function getContactFirstName()

{

return $this->container['contact\_first\_name'];

}

public function setContactFirstName($contact\_first\_name)

{

$this->container['contact\_first\_name'] = $contact\_first\_name;

return $this;

}

public function getContactLastName()

{

return $this->container['contact\_last\_name'];

}

public function setContactLastName($contact\_last\_name)

{

$this->container['contact\_last\_name'] = $contact\_last\_name;

return $this;

}

public function getContactPhone()

{

return $this->container['contact\_phone'];

}

public function setContactPhone($contact\_phone)

{

$this->container['contact\_phone'] = $contact\_phone;

return $this;

}

public function getContactFax()

{

return $this->container['contact\_fax'];

}

public function setContactFax($contact\_fax)

{

$this->container['contact\_fax'] = $contact\_fax;

return $this;

}

public function getContactEmail()

{

return $this->container['contact\_email'];

}

public function setContactEmail($contact\_email)

{

$this->container['contact\_email'] = $contact\_email;

return $this;

}

public function getContactTitle()

{

return $this->container['contact\_title'];

}

public function setContactTitle($contact\_title)

{

$this->container['contact\_title'] = $contact\_title;

return $this;

}

public function getIp()

{

return $this->container['ip'];

}

public function setIp($ip)

{

$this->container['ip'] = $ip;

return $this;

}

public function getTaxExempt()

{

return $this->container['tax\_exempt'];

}

public function setTaxExempt($tax\_exempt)

{

$this->container['tax\_exempt'] = $tax\_exempt;

return $this;

}

public function offsetExists($offset)

{

return isset($this->container[$offset]);

}

public function offsetGet($offset)

{

return isset($this->container[$offset]) ? $this->container[$offset] : null;

}

public function offsetSet($offset, $value)

{

if (is\_null($offset)) {

$this->container[] = $value;

} else {

$this->container[$offset] = $value;

}

}

public function offsetUnset($offset)

{

unset($this->container[$offset]);

}

public function \_\_toString()

{

if (defined('JSON\_PRETTY\_PRINT')) { // use JSON pretty print

return json\_encode(

ObjectSerializer::sanitizeForSerialization($this),

JSON\_PRETTY\_PRINT

);

}

return json\_encode(ObjectSerializer::sanitizeForSerialization($this));

}

}

# Receipt

<?php

namespace Swagger\Client\Model;

use \ArrayAccess;

use \Swagger\Client\ObjectSerializer;

class Receipt implements ModelInterface, ArrayAccess

{

const DISCRIMINATOR = null;

protected static $swaggerModelName = 'Receipt';

protected static $swaggerTypes = [

'header' => '\Swagger\Client\Model\ReceiptCustomLines',

'footer' => '\Swagger\Client\Model\ReceiptCustomLines'

];

protected static $swaggerFormats = [

'header' => null,

'footer' => null

];

public static function swaggerTypes()

{

return self::$swaggerTypes;

}

public static function swaggerFormats()

{

return self::$swaggerFormats;

}

protected static $attributeMap = [

'header' => 'Header',

'footer' => 'Footer'

];

protected static $setters = [

'header' => 'setHeader',

'footer' => 'setFooter'

];

protected static $getters = [

'header' => 'getHeader',

'footer' => 'getFooter'

];

public static function attributeMap()

{

return self::$attributeMap;

}

public static function setters()

{

return self::$setters;

}

public static function getters()

{

return self::$getters;

}

public function getModelName()

{

return self::$swaggerModelName;

}

protected $container = [];

public function \_\_construct(array $data = null)

{

$this->container['header'] = isset($data['header']) ? $data['header'] : null;

$this->container['footer'] = isset($data['footer']) ? $data['footer'] : null;

}

public function listInvalidProperties()

{

$invalidProperties = [];

return $invalidProperties;

}

public function valid()

{

return count($this->listInvalidProperties()) === 0;

}

public function getHeader()

{

return $this->container['header'];

}

public function setHeader($header)

{

$this->container['header'] = $header;

return $this;

}

public function getFooter()

{

return $this->container['footer'];

}

public function setFooter($footer)

{

$this->container['footer'] = $footer;

return $this;

}

public function offsetExists($offset)

{

return isset($this->container[$offset]);

}

public function offsetGet($offset)

{

return isset($this->container[$offset]) ? $this->container[$offset] : null;

}

public function offsetSet($offset, $value)

{

if (is\_null($offset)) {

$this->container[] = $value;

} else {

$this->container[$offset] = $value;

}

}

public function offsetUnset($offset)

{

unset($this->container[$offset]);

}

public function \_\_toString()

{

if (defined('JSON\_PRETTY\_PRINT')) { // use JSON pretty print

return json\_encode(

ObjectSerializer::sanitizeForSerialization($this),

JSON\_PRETTY\_PRINT

);

}

return json\_encode(ObjectSerializer::sanitizeForSerialization($this));

}

}

# ReceiptCustomLines

<?php

namespace Swagger\Client\Model;

use \ArrayAccess;

use \Swagger\Client\ObjectSerializer;

class ReceiptCustomLines implements ModelInterface, ArrayAccess

{

const DISCRIMINATOR = null;

protected static $swaggerModelName = 'ReceiptCustomLines';

protected static $swaggerTypes = [

'custom\_line1' => 'string',

'custom\_line2' => 'string',

'custom\_line3' => 'string',

'custom\_line4' => 'string'

];

protected static $swaggerFormats = [

'custom\_line1' => null,

'custom\_line2' => null,

'custom\_line3' => null,

'custom\_line4' => null

];

public static function swaggerTypes()

{

return self::$swaggerTypes;

}

public static function swaggerFormats()

{

return self::$swaggerFormats;

}

protected static $attributeMap = [

'custom\_line1' => 'CustomLine1',

'custom\_line2' => 'CustomLine2',

'custom\_line3' => 'CustomLine3',

'custom\_line4' => 'CustomLine4'

];

protected static $setters = [

'custom\_line1' => 'setCustomLine1',

'custom\_line2' => 'setCustomLine2',

'custom\_line3' => 'setCustomLine3',

'custom\_line4' => 'setCustomLine4'

];

protected static $getters = [

'custom\_line1' => 'getCustomLine1',

'custom\_line2' => 'getCustomLine2',

'custom\_line3' => 'getCustomLine3',

'custom\_line4' => 'getCustomLine4'

];

public static function attributeMap()

{

return self::$attributeMap;

}

public static function setters()

{

return self::$setters;

}

public static function getters()

{

return self::$getters;

}

public function getModelName()

{

return self::$swaggerModelName;

}

protected $container = [];

public function \_\_construct(array $data = null)

{

$this->container['custom\_line1'] = isset($data['custom\_line1']) ? $data['custom\_line1'] : null;

$this->container['custom\_line2'] = isset($data['custom\_line2']) ? $data['custom\_line2'] : null;

$this->container['custom\_line3'] = isset($data['custom\_line3']) ? $data['custom\_line3'] : null;

$this->container['custom\_line4'] = isset($data['custom\_line4']) ? $data['custom\_line4'] : null;

}

public function listInvalidProperties()

{

$invalidProperties = [];

return $invalidProperties;

}

public function valid()

{

return count($this->listInvalidProperties()) === 0;

}

public function getCustomLine1()

{

return $this->container['custom\_line1'];

}

public function setCustomLine1($custom\_line1)

{

$this->container['custom\_line1'] = $custom\_line1;

return $this;

}

public function getCustomLine2()

{

return $this->container['custom\_line2'];

}

public function setCustomLine2($custom\_line2)

{

$this->container['custom\_line2'] = $custom\_line2;

return $this;

}

public function getCustomLine3()

{

return $this->container['custom\_line3'];

}

public function setCustomLine3($custom\_line3)

{

$this->container['custom\_line3'] = $custom\_line3;

return $this;

}

public function getCustomLine4()

{

return $this->container['custom\_line4'];

}

public function setCustomLine4($custom\_line4)

{

$this->container['custom\_line4'] = $custom\_line4;

return $this;

}

public function offsetExists($offset)

{

return isset($this->container[$offset]);

}

public function offsetGet($offset)

{

return isset($this->container[$offset]) ? $this->container[$offset] : null;

}

public function offsetSet($offset, $value)

{

if (is\_null($offset)) {

$this->container[] = $value;

} else {

$this->container[$offset] = $value;

}

}

public function offsetUnset($offset)

{

unset($this->container[$offset]);

}

public function \_\_toString()

{

if (defined('JSON\_PRETTY\_PRINT')) { // use JSON pretty print

return json\_encode(

ObjectSerializer::sanitizeForSerialization($this),

JSON\_PRETTY\_PRINT

);

}

return json\_encode(ObjectSerializer::sanitizeForSerialization($this));

}

}

# Record

<?php

namespace Swagger\Client\Model;

use \ArrayAccess;

use \Swagger\Client\ObjectSerializer;

class Record implements ModelInterface, ArrayAccess

{

const DISCRIMINATOR = null;

protected static $swaggerModelName = 'Record';

protected static $swaggerTypes = [

'response' => '\Swagger\Client\Model\Response',

'id' => 'int',

'reference' => 'string',

'business' => '\Swagger\Client\Model\Business',

'contact' => '\Swagger\Client\Model\Contact',

'po' => 'string',

'invoice' => 'string',

'customer' => 'string',

'tax\_exempt' => 'bool',

'note' => 'string',

'credit\_cards' => '\Swagger\Client\Model\VaultCreditCard[]',

'checks' => '\Swagger\Client\Model\VaultCheck[]'

];

protected static $swaggerFormats = [

'response' => null,

'id' => 'int32',

'reference' => null,

'business' => null,

'contact' => null,

'po' => null,

'invoice' => null,

'customer' => null,

'tax\_exempt' => null,

'note' => null,

'credit\_cards' => null,

'checks' => null

];

public static function swaggerTypes()

{

return self::$swaggerTypes;

}

public static function swaggerFormats()

{

return self::$swaggerFormats;

}

protected static $attributeMap = [

'response' => 'Response',

'id' => 'ID',

'reference' => 'Reference',

'business' => 'Business',

'contact' => 'Contact',

'po' => 'PO',

'invoice' => 'Invoice',

'customer' => 'Customer',

'tax\_exempt' => 'TaxExempt',

'note' => 'Note',

'credit\_cards' => 'CreditCards',

'checks' => 'Checks'

];

protected static $setters = [

'response' => 'setResponse',

'id' => 'setId',

'reference' => 'setReference',

'business' => 'setBusiness',

'contact' => 'setContact',

'po' => 'setPo',

'invoice' => 'setInvoice',

'customer' => 'setCustomer',

'tax\_exempt' => 'setTaxExempt',

'note' => 'setNote',

'credit\_cards' => 'setCreditCards',

'checks' => 'setChecks'

];

protected static $getters = [

'response' => 'getResponse',

'id' => 'getId',

'reference' => 'getReference',

'business' => 'getBusiness',

'contact' => 'getContact',

'po' => 'getPo',

'invoice' => 'getInvoice',

'customer' => 'getCustomer',

'tax\_exempt' => 'getTaxExempt',

'note' => 'getNote',

'credit\_cards' => 'getCreditCards',

'checks' => 'getChecks'

];

public static function attributeMap()

{

return self::$attributeMap;

}

public static function setters()

{

return self::$setters;

}

public static function getters()

{

return self::$getters;

}

public function getModelName()

{

return self::$swaggerModelName;

}

protected $container = [];

public function \_\_construct(array $data = null)

{

$this->container['response'] = isset($data['response']) ? $data['response'] : null;

$this->container['id'] = isset($data['id']) ? $data['id'] : null;

$this->container['reference'] = isset($data['reference']) ? $data['reference'] : null;

$this->container['business'] = isset($data['business']) ? $data['business'] : null;

$this->container['contact'] = isset($data['contact']) ? $data['contact'] : null;

$this->container['po'] = isset($data['po']) ? $data['po'] : null;

$this->container['invoice'] = isset($data['invoice']) ? $data['invoice'] : null;

$this->container['customer'] = isset($data['customer']) ? $data['customer'] : null;

$this->container['tax\_exempt'] = isset($data['tax\_exempt']) ? $data['tax\_exempt'] : null;

$this->container['note'] = isset($data['note']) ? $data['note'] : null;

$this->container['credit\_cards'] = isset($data['credit\_cards']) ? $data['credit\_cards'] : null;

$this->container['checks'] = isset($data['checks']) ? $data['checks'] : null;

}

public function listInvalidProperties()

{

$invalidProperties = [];

return $invalidProperties;

}

public function valid()

{

return count($this->listInvalidProperties()) === 0;

}

public function getResponse()

{

return $this->container['response'];

}

public function setResponse($response)

{

$this->container['response'] = $response;

return $this;

}

public function getId()

{

return $this->container['id'];

}

public function setId($id)

{

$this->container['id'] = $id;

return $this;

}

public function getReference()

{

return $this->container['reference'];

}

public function setReference($reference)

{

$this->container['reference'] = $reference;

return $this;

}

public function getBusiness()

{

return $this->container['business'];

}

public function setBusiness($business)

{

$this->container['business'] = $business;

return $this;

}

public function getContact()

{

return $this->container['contact'];

}

public function setContact($contact)

{

$this->container['contact'] = $contact;

return $this;

}

public function getPo()

{

return $this->container['po'];

}

public function setPo($po)

{

$this->container['po'] = $po;

return $this;

}

public function getInvoice()

{

return $this->container['invoice'];

}

public function setInvoice($invoice)

{

$this->container['invoice'] = $invoice;

return $this;

}

public function getCustomer()

{

return $this->container['customer'];

}

public function setCustomer($customer)

{

$this->container['customer'] = $customer;

return $this;

}

public function getTaxExempt()

{

return $this->container['tax\_exempt'];

}

public function setTaxExempt($tax\_exempt)

{

$this->container['tax\_exempt'] = $tax\_exempt;

return $this;

}

public function getNote()

{

return $this->container['note'];

}

public function setNote($note)

{

$this->container['note'] = $note;

return $this;

}

public function getCreditCards()

{

return $this->container['credit\_cards'];

}

public function setCreditCards($credit\_cards)

{

$this->container['credit\_cards'] = $credit\_cards;

return $this;

}

public function getChecks()

{

return $this->container['checks'];

}

public function setChecks($checks)

{

$this->container['checks'] = $checks;

return $this;

}

public function offsetExists($offset)

{

return isset($this->container[$offset]);

}

public function offsetGet($offset)

{

return isset($this->container[$offset]) ? $this->container[$offset] : null;

}

public function offsetSet($offset, $value)

{

if (is\_null($offset)) {

$this->container[] = $value;

} else {

$this->container[$offset] = $value;

}

}

public function offsetUnset($offset)

{

unset($this->container[$offset]);

}

public function \_\_toString()

{

if (defined('JSON\_PRETTY\_PRINT')) { // use JSON pretty print

return json\_encode(

ObjectSerializer::sanitizeForSerialization($this),

JSON\_PRETTY\_PRINT

);

}

return json\_encode(ObjectSerializer::sanitizeForSerialization($this));

}

}

# Response

<?php

namespace Swagger\Client\Model;

use \ArrayAccess;

use \Swagger\Client\ObjectSerializer;

class Response implements ModelInterface, ArrayAccess

{

const DISCRIMINATOR = null;

protected static $swaggerModelName = 'Response';

protected static $swaggerTypes = [

'error' => 'bool',

'message' => 'string'

];

protected static $swaggerFormats = [

'error' => null,

'message' => null

];

public static function swaggerTypes()

{

return self::$swaggerTypes;

}

public static function swaggerFormats()

{

return self::$swaggerFormats;

}

protected static $attributeMap = [

'error' => 'Error',

'message' => 'Message'

];

protected static $setters = [

'error' => 'setError',

'message' => 'setMessage'

];

protected static $getters = [

'error' => 'getError',

'message' => 'getMessage'

];

public static function attributeMap()

{

return self::$attributeMap;

}

public static function setters()

{

return self::$setters;

}

public static function getters()

{

return self::$getters;

}

public function getModelName()

{

return self::$swaggerModelName;

}

protected $container = [];

public function \_\_construct(array $data = null)

{

$this->container['error'] = isset($data['error']) ? $data['error'] : null;

$this->container['message'] = isset($data['message']) ? $data['message'] : null;

}

public function listInvalidProperties()

{

$invalidProperties = [];

return $invalidProperties;

}

public function valid()

{

return count($this->listInvalidProperties()) === 0;

}

public function getError()

{

return $this->container['error'];

}

public function setError($error)

{

$this->container['error'] = $error;

return $this;

}

public function getMessage()

{

return $this->container['message'];

}

public function setMessage($message)

{

$this->container['message'] = $message;

return $this;

}

public function offsetExists($offset)

{

return isset($this->container[$offset]);

}

public function offsetGet($offset)

{

return isset($this->container[$offset]) ? $this->container[$offset] : null;

}

public function offsetSet($offset, $value)

{

if (is\_null($offset)) {

$this->container[] = $value;

} else {

$this->container[$offset] = $value;

}

}

public function offsetUnset($offset)

{

unset($this->container[$offset]);

}

public function \_\_toString()

{

if (defined('JSON\_PRETTY\_PRINT')) { // use JSON pretty print

return json\_encode(

ObjectSerializer::sanitizeForSerialization($this),

JSON\_PRETTY\_PRINT

);

}

return json\_encode(ObjectSerializer::sanitizeForSerialization($this));

}

}

# SearchVault

<?php

namespace Swagger\Client\Model;

use \ArrayAccess;

use \Swagger\Client\ObjectSerializer;

class SearchVault implements ModelInterface, ArrayAccess

{

const DISCRIMINATOR = null;

protected static $swaggerModelName = 'SearchVault';

protected static $swaggerTypes = [

'reference' => 'string',

'business\_name' => 'string',

'first\_name' => 'string',

'last\_name' => 'string'

];

protected static $swaggerFormats = [

'reference' => null,

'business\_name' => null,

'first\_name' => null,

'last\_name' => null

];

public static function swaggerTypes()

{

return self::$swaggerTypes;

}

public static function swaggerFormats()

{

return self::$swaggerFormats;

}

protected static $attributeMap = [

'reference' => 'Reference',

'business\_name' => 'BusinessName',

'first\_name' => 'FirstName',

'last\_name' => 'LastName'

];

protected static $setters = [

'reference' => 'setReference',

'business\_name' => 'setBusinessName',

'first\_name' => 'setFirstName',

'last\_name' => 'setLastName'

];

protected static $getters = [

'reference' => 'getReference',

'business\_name' => 'getBusinessName',

'first\_name' => 'getFirstName',

'last\_name' => 'getLastName'

];

public static function attributeMap()

{

return self::$attributeMap;

}

public static function setters()

{

return self::$setters;

}

public static function getters()

{

return self::$getters;

}

public function getModelName()

{

return self::$swaggerModelName;

}

protected $container = [];

public function \_\_construct(array $data = null)

{

$this->container['reference'] = isset($data['reference']) ? $data['reference'] : null;

$this->container['business\_name'] = isset($data['business\_name']) ? $data['business\_name'] : null;

$this->container['first\_name'] = isset($data['first\_name']) ? $data['first\_name'] : null;

$this->container['last\_name'] = isset($data['last\_name']) ? $data['last\_name'] : null;

}

public function listInvalidProperties()

{

$invalidProperties = [];

return $invalidProperties;

}

public function valid()

{

return count($this->listInvalidProperties()) === 0;

}

public function getReference()

{

return $this->container['reference'];

}

public function setReference($reference)

{

$this->container['reference'] = $reference;

return $this;

}

public function getBusinessName()

{

return $this->container['business\_name'];

}

public function setBusinessName($business\_name)

{

$this->container['business\_name'] = $business\_name;

return $this;

}

public function getFirstName()

{

return $this->container['first\_name'];

}

public function setFirstName($first\_name)

{

$this->container['first\_name'] = $first\_name;

return $this;

}

public function getLastName()

{

return $this->container['last\_name'];

}

public function setLastName($last\_name)

{

$this->container['last\_name'] = $last\_name;

return $this;

}

public function offsetExists($offset)

{

return isset($this->container[$offset]);

}

public function offsetGet($offset)

{

return isset($this->container[$offset]) ? $this->container[$offset] : null;

}

public function offsetSet($offset, $value)

{

if (is\_null($offset)) {

$this->container[] = $value;

} else {

$this->container[$offset] = $value;

}

}

public function offsetUnset($offset)

{

unset($this->container[$offset]);

}

public function \_\_toString()

{

if (defined('JSON\_PRETTY\_PRINT')) { // use JSON pretty print

return json\_encode(

ObjectSerializer::sanitizeForSerialization($this),

JSON\_PRETTY\_PRINT

);

}

return json\_encode(ObjectSerializer::sanitizeForSerialization($this));

}

}

# Settings

<?php

namespace Swagger\Client\Model;

use \ArrayAccess;

use \Swagger\Client\ObjectSerializer;

class Settings implements ModelInterface, ArrayAccess

{

const DISCRIMINATOR = null;

protected static $swaggerModelName = 'Settings';

protected static $swaggerTypes = [

'credit\_card\_transaction\_settings' => '\Swagger\Client\Model\CreditCardTransactionSettings',

'check\_transaction\_settings' => '\Swagger\Client\Model\CheckTransactionSettings',

'business' => '\Swagger\Client\Model\Business',

'main\_contact' => '\Swagger\Client\Model\Contact',

'receipt40' => '\Swagger\Client\Model\Receipt',

'receipt80' => '\Swagger\Client\Model\Receipt',

'print\_signature\_line' => 'bool',

'response' => '\Swagger\Client\Model\Response',

'account\_id' => 'string',

'copy\_email\_receipt' => '\Swagger\Client\Model\FieldInfo'

];

protected static $swaggerFormats = [

'credit\_card\_transaction\_settings' => null,

'check\_transaction\_settings' => null,

'business' => null,

'main\_contact' => null,

'receipt40' => null,

'receipt80' => null,

'print\_signature\_line' => null,

'response' => null,

'account\_id' => null,

'copy\_email\_receipt' => null

];

public static function swaggerTypes()

{

return self::$swaggerTypes;

}

public static function swaggerFormats()

{

return self::$swaggerFormats;

}

protected static $attributeMap = [

'credit\_card\_transaction\_settings' => 'CreditCardTransactionSettings',

'check\_transaction\_settings' => 'CheckTransactionSettings',

'business' => 'Business',

'main\_contact' => 'MainContact',

'receipt40' => 'Receipt40',

'receipt80' => 'Receipt80',

'print\_signature\_line' => 'PrintSignatureLine',

'response' => 'Response',

'account\_id' => 'AccountID',

'copy\_email\_receipt' => 'CopyEmailReceipt'

];

protected static $setters = [

'credit\_card\_transaction\_settings' => 'setCreditCardTransactionSettings',

'check\_transaction\_settings' => 'setCheckTransactionSettings',

'business' => 'setBusiness',

'main\_contact' => 'setMainContact',

'receipt40' => 'setReceipt40',

'receipt80' => 'setReceipt80',

'print\_signature\_line' => 'setPrintSignatureLine',

'response' => 'setResponse',

'account\_id' => 'setAccountId',

'copy\_email\_receipt' => 'setCopyEmailReceipt'

];

protected static $getters = [

'credit\_card\_transaction\_settings' => 'getCreditCardTransactionSettings',

'check\_transaction\_settings' => 'getCheckTransactionSettings',

'business' => 'getBusiness',

'main\_contact' => 'getMainContact',

'receipt40' => 'getReceipt40',

'receipt80' => 'getReceipt80',

'print\_signature\_line' => 'getPrintSignatureLine',

'response' => 'getResponse',

'account\_id' => 'getAccountId',

'copy\_email\_receipt' => 'getCopyEmailReceipt'

];

public static function attributeMap()

{

return self::$attributeMap;

}

public static function setters()

{

return self::$setters;

}

public static function getters()

{

return self::$getters;

}

public function getModelName()

{

return self::$swaggerModelName;

}

protected $container = [];

public function \_\_construct(array $data = null)

{

$this->container['credit\_card\_transaction\_settings'] = isset($data['credit\_card\_transaction\_settings']) ? $data['credit\_card\_transaction\_settings'] : null;

$this->container['check\_transaction\_settings'] = isset($data['check\_transaction\_settings']) ? $data['check\_transaction\_settings'] : null;

$this->container['business'] = isset($data['business']) ? $data['business'] : null;

$this->container['main\_contact'] = isset($data['main\_contact']) ? $data['main\_contact'] : null;

$this->container['receipt40'] = isset($data['receipt40']) ? $data['receipt40'] : null;

$this->container['receipt80'] = isset($data['receipt80']) ? $data['receipt80'] : null;

$this->container['print\_signature\_line'] = isset($data['print\_signature\_line']) ? $data['print\_signature\_line'] : null;

$this->container['response'] = isset($data['response']) ? $data['response'] : null;

$this->container['account\_id'] = isset($data['account\_id']) ? $data['account\_id'] : null;

$this->container['copy\_email\_receipt'] = isset($data['copy\_email\_receipt']) ? $data['copy\_email\_receipt'] : null;

}

public function listInvalidProperties()

{

$invalidProperties = [];

return $invalidProperties;

}

public function valid()

{

return count($this->listInvalidProperties()) === 0;

}

public function getCreditCardTransactionSettings()

{

return $this->container['credit\_card\_transaction\_settings'];

}

public function setCreditCardTransactionSettings($credit\_card\_transaction\_settings)

{

$this->container['credit\_card\_transaction\_settings'] = $credit\_card\_transaction\_settings;

return $this;

}

public function getCheckTransactionSettings()

{

return $this->container['check\_transaction\_settings'];

}

public function setCheckTransactionSettings($check\_transaction\_settings)

{

$this->container['check\_transaction\_settings'] = $check\_transaction\_settings;

return $this;

}

public function getBusiness()

{

return $this->container['business'];

}

public function setBusiness($business)

{

$this->container['business'] = $business;

return $this;

}

public function getMainContact()

{

return $this->container['main\_contact'];

}

public function setMainContact($main\_contact)

{

$this->container['main\_contact'] = $main\_contact;

return $this;

}

public function getReceipt40()

{

return $this->container['receipt40'];

}

public function setReceipt40($receipt40)

{

$this->container['receipt40'] = $receipt40;

return $this;

}

public function getReceipt80()

{

return $this->container['receipt80'];

}

public function setReceipt80($receipt80)

{

$this->container['receipt80'] = $receipt80;

return $this;

}

public function getPrintSignatureLine()

{

return $this->container['print\_signature\_line'];

}

public function setPrintSignatureLine($print\_signature\_line)

{

$this->container['print\_signature\_line'] = $print\_signature\_line;

return $this;

}

public function getResponse()

{

return $this->container['response'];

}

public function setResponse($response)

{

$this->container['response'] = $response;

return $this;

}

public function getAccountId()

{

return $this->container['account\_id'];

}

public function setAccountId($account\_id)

{

$this->container['account\_id'] = $account\_id;

return $this;

}

public function getCopyEmailReceipt()

{

return $this->container['copy\_email\_receipt'];

}

public function setCopyEmailReceipt($copy\_email\_receipt)

{

$this->container['copy\_email\_receipt'] = $copy\_email\_receipt;

return $this;

}

public function offsetExists($offset)

{

return isset($this->container[$offset]);

}

public function offsetGet($offset)

{

return isset($this->container[$offset]) ? $this->container[$offset] : null;

}

public function offsetSet($offset, $value)

{

if (is\_null($offset)) {

$this->container[] = $value;

} else {

$this->container[$offset] = $value;

}

}

public function offsetUnset($offset)

{

unset($this->container[$offset]);

}

public function \_\_toString()

{

if (defined('JSON\_PRETTY\_PRINT')) { // use JSON pretty print

return json\_encode(

ObjectSerializer::sanitizeForSerialization($this),

JSON\_PRETTY\_PRINT

);

}

return json\_encode(ObjectSerializer::sanitizeForSerialization($this));

}

}

# TerminalSettings

<?php

namespace Swagger\Client\Model;

use \ArrayAccess;

use \Swagger\Client\ObjectSerializer;

class TerminalSettings implements ModelInterface, ArrayAccess

{

const DISCRIMINATOR = null;

protected static $swaggerModelName = 'TerminalSettings';

protected static $swaggerTypes = [

'id' => 'int',

'terminal\_id' => 'int',

'sec\_code' => 'string',

'gateway\_terminal' => 'bool',

'consumer\_credits' => 'bool',

'dl\_required' => 'bool',

'check\_verify' => 'bool',

'identity\_verify' => 'bool'

];

protected static $swaggerFormats = [

'id' => 'int32',

'terminal\_id' => 'int32',

'sec\_code' => null,

'gateway\_terminal' => null,

'consumer\_credits' => null,

'dl\_required' => null,

'check\_verify' => null,

'identity\_verify' => null

];

public static function swaggerTypes()

{

return self::$swaggerTypes;

}

public static function swaggerFormats()

{

return self::$swaggerFormats;

}

protected static $attributeMap = [

'id' => 'ID',

'terminal\_id' => 'Terminal\_ID',

'sec\_code' => 'SEC\_Code',

'gateway\_terminal' => 'Gateway\_Terminal',

'consumer\_credits' => 'Consumer\_Credits',

'dl\_required' => 'DL\_Required',

'check\_verify' => 'Check\_Verify',

'identity\_verify' => 'Identity\_Verify'

];

protected static $setters = [

'id' => 'setId',

'terminal\_id' => 'setTerminalId',

'sec\_code' => 'setSecCode',

'gateway\_terminal' => 'setGatewayTerminal',

'consumer\_credits' => 'setConsumerCredits',

'dl\_required' => 'setDlRequired',

'check\_verify' => 'setCheckVerify',

'identity\_verify' => 'setIdentityVerify'

];

protected static $getters = [

'id' => 'getId',

'terminal\_id' => 'getTerminalId',

'sec\_code' => 'getSecCode',

'gateway\_terminal' => 'getGatewayTerminal',

'consumer\_credits' => 'getConsumerCredits',

'dl\_required' => 'getDlRequired',

'check\_verify' => 'getCheckVerify',

'identity\_verify' => 'getIdentityVerify'

];

public static function attributeMap()

{

return self::$attributeMap;

}

public static function setters()

{

return self::$setters;

}

public static function getters()

{

return self::$getters;

}

public function getModelName()

{

return self::$swaggerModelName;

}

protected $container = [];

public function \_\_construct(array $data = null)

{

$this->container['id'] = isset($data['id']) ? $data['id'] : null;

$this->container['terminal\_id'] = isset($data['terminal\_id']) ? $data['terminal\_id'] : null;

$this->container['sec\_code'] = isset($data['sec\_code']) ? $data['sec\_code'] : null;

$this->container['gateway\_terminal'] = isset($data['gateway\_terminal']) ? $data['gateway\_terminal'] : null;

$this->container['consumer\_credits'] = isset($data['consumer\_credits']) ? $data['consumer\_credits'] : null;

$this->container['dl\_required'] = isset($data['dl\_required']) ? $data['dl\_required'] : null;

$this->container['check\_verify'] = isset($data['check\_verify']) ? $data['check\_verify'] : null;

$this->container['identity\_verify'] = isset($data['identity\_verify']) ? $data['identity\_verify'] : null;

}

public function listInvalidProperties()

{

$invalidProperties = [];

return $invalidProperties;

}

public function valid()

{

return count($this->listInvalidProperties()) === 0;

}

public function getId()

{

return $this->container['id'];

}

public function setId($id)

{

$this->container['id'] = $id;

return $this;

}

public function getTerminalId()

{

return $this->container['terminal\_id'];

}

public function setTerminalId($terminal\_id)

{

$this->container['terminal\_id'] = $terminal\_id;

return $this;

}

public function getSecCode()

{

return $this->container['sec\_code'];

}

public function setSecCode($sec\_code)

{

$this->container['sec\_code'] = $sec\_code;

return $this;

}

public function getGatewayTerminal()

{

return $this->container['gateway\_terminal'];

}

public function setGatewayTerminal($gateway\_terminal)

{

$this->container['gateway\_terminal'] = $gateway\_terminal;

return $this;

}

public function getConsumerCredits()

{

return $this->container['consumer\_credits'];

}

public function setConsumerCredits($consumer\_credits)

{

$this->container['consumer\_credits'] = $consumer\_credits;

return $this;

}

public function getDlRequired()

{

return $this->container['dl\_required'];

}

public function setDlRequired($dl\_required)

{

$this->container['dl\_required'] = $dl\_required;

return $this;

}

public function getCheckVerify()

{

return $this->container['check\_verify'];

}

public function setCheckVerify($check\_verify)

{

$this->container['check\_verify'] = $check\_verify;

return $this;

}

public function getIdentityVerify()

{

return $this->container['identity\_verify'];

}

public function setIdentityVerify($identity\_verify)

{

$this->container['identity\_verify'] = $identity\_verify;

return $this;

}

public function offsetExists($offset)

{

return isset($this->container[$offset]);

}

public function offsetGet($offset)

{

return isset($this->container[$offset]) ? $this->container[$offset] : null;

}

public function offsetSet($offset, $value)

{

if (is\_null($offset)) {

$this->container[] = $value;

} else {

$this->container[$offset] = $value;

}

}

public function offsetUnset($offset)

{

unset($this->container[$offset]);

}

public function \_\_toString()

{

if (defined('JSON\_PRETTY\_PRINT')) { // use JSON pretty print

return json\_encode(

ObjectSerializer::sanitizeForSerialization($this),

JSON\_PRETTY\_PRINT

);

}

return json\_encode(ObjectSerializer::sanitizeForSerialization($this));

}

}

# Transaction

<?php

namespace Swagger\Client\Model;

use \ArrayAccess;

use \Swagger\Client\ObjectSerializer;

class Transaction implements ModelInterface, ArrayAccess

{

const DISCRIMINATOR = null;

protected static $swaggerModelName = 'Transaction';

protected static $swaggerTypes = [

'submit\_time' => '\DateTime',

'tran\_type' => 'string',

'tran\_amt' => 'double',

'tran\_tax' => 'double',

'tran\_inv' => 'string',

'tran\_po' => 'string',

'tran\_c\_num' => 'string',

'tran\_note' => 'string',

'card\_num' => 'string',

'card\_name' => 'string',

'card\_exp' => 'string',

'card\_type' => 'string',

'avs\_street' => 'string',

'avs\_zip' => 'string',

'cvv\_num' => 'string',

'proc\_id' => 'string',

'proc\_resp' => 'string',

'proc\_code' => 'string',

'proc\_mess' => 'string',

'avs\_code' => 'string',

'avs\_resp' => 'string',

'cvv\_code' => 'string',

'cvv\_resp' => 'string',

'items' => '\Swagger\Client\Model\TransactionItem[]',

'business\_name' => 'string',

'business\_address' => 'string',

'business\_address2' => 'string',

'business\_city' => 'string',

'business\_state' => 'string',

'business\_zip' => 'string',

'business\_country' => 'string',

'business\_website' => 'string',

'contact\_first\_name' => 'string',

'contact\_last\_name' => 'string',

'contact\_phone' => 'string',

'contact\_fax' => 'string',

'contact\_email' => 'string',

'contact\_title' => 'string',

'ip' => 'string',

'tax\_exempt' => 'bool'

];

protected static $swaggerFormats = [

'submit\_time' => 'date-time',

'tran\_type' => null,

'tran\_amt' => 'double',

'tran\_tax' => 'double',

'tran\_inv' => null,

'tran\_po' => null,

'tran\_c\_num' => null,

'tran\_note' => null,

'card\_num' => null,

'card\_name' => null,

'card\_exp' => null,

'card\_type' => null,

'avs\_street' => null,

'avs\_zip' => null,

'cvv\_num' => null,

'proc\_id' => null,

'proc\_resp' => null,

'proc\_code' => null,

'proc\_mess' => null,

'avs\_code' => null,

'avs\_resp' => null,

'cvv\_code' => null,

'cvv\_resp' => null,

'items' => null,

'business\_name' => null,

'business\_address' => null,

'business\_address2' => null,

'business\_city' => null,

'business\_state' => null,

'business\_zip' => null,

'business\_country' => null,

'business\_website' => null,

'contact\_first\_name' => null,

'contact\_last\_name' => null,

'contact\_phone' => null,

'contact\_fax' => null,

'contact\_email' => null,

'contact\_title' => null,

'ip' => null,

'tax\_exempt' => null

];

public static function swaggerTypes()

{

return self::$swaggerTypes;

}

public static function swaggerFormats()

{

return self::$swaggerFormats;

}

protected static $attributeMap = [

'submit\_time' => 'Submit\_Time',

'tran\_type' => 'Tran\_Type',

'tran\_amt' => 'Tran\_Amt',

'tran\_tax' => 'Tran\_Tax',

'tran\_inv' => 'Tran\_Inv',

'tran\_po' => 'Tran\_PO',

'tran\_c\_num' => 'Tran\_CNum',

'tran\_note' => 'Tran\_Note',

'card\_num' => 'Card\_Num',

'card\_name' => 'Card\_Name',

'card\_exp' => 'Card\_Exp',

'card\_type' => 'Card\_Type',

'avs\_street' => 'AVS\_Street',

'avs\_zip' => 'AVS\_Zip',

'cvv\_num' => 'CVV\_Num',

'proc\_id' => 'Proc\_ID',

'proc\_resp' => 'Proc\_Resp',

'proc\_code' => 'Proc\_Code',

'proc\_mess' => 'Proc\_Mess',

'avs\_code' => 'AVS\_Code',

'avs\_resp' => 'AVS\_Resp',

'cvv\_code' => 'CVV\_Code',

'cvv\_resp' => 'CVV\_Resp',

'items' => 'Items',

'business\_name' => 'Business\_Name',

'business\_address' => 'Business\_Address',

'business\_address2' => 'Business\_Address2',

'business\_city' => 'Business\_City',

'business\_state' => 'Business\_State',

'business\_zip' => 'Business\_Zip',

'business\_country' => 'Business\_Country',

'business\_website' => 'Business\_Website',

'contact\_first\_name' => 'Contact\_FirstName',

'contact\_last\_name' => 'Contact\_LastName',

'contact\_phone' => 'Contact\_Phone',

'contact\_fax' => 'Contact\_Fax',

'contact\_email' => 'Contact\_Email',

'contact\_title' => 'Contact\_Title',

'ip' => 'IP',

'tax\_exempt' => 'TaxExempt'

];

protected static $setters = [

'submit\_time' => 'setSubmitTime',

'tran\_type' => 'setTranType',

'tran\_amt' => 'setTranAmt',

'tran\_tax' => 'setTranTax',

'tran\_inv' => 'setTranInv',

'tran\_po' => 'setTranPo',

'tran\_c\_num' => 'setTranCNum',

'tran\_note' => 'setTranNote',

'card\_num' => 'setCardNum',

'card\_name' => 'setCardName',

'card\_exp' => 'setCardExp',

'card\_type' => 'setCardType',

'avs\_street' => 'setAvsStreet',

'avs\_zip' => 'setAvsZip',

'cvv\_num' => 'setCvvNum',

'proc\_id' => 'setProcId',

'proc\_resp' => 'setProcResp',

'proc\_code' => 'setProcCode',

'proc\_mess' => 'setProcMess',

'avs\_code' => 'setAvsCode',

'avs\_resp' => 'setAvsResp',

'cvv\_code' => 'setCvvCode',

'cvv\_resp' => 'setCvvResp',

'items' => 'setItems',

'business\_name' => 'setBusinessName',

'business\_address' => 'setBusinessAddress',

'business\_address2' => 'setBusinessAddress2',

'business\_city' => 'setBusinessCity',

'business\_state' => 'setBusinessState',

'business\_zip' => 'setBusinessZip',

'business\_country' => 'setBusinessCountry',

'business\_website' => 'setBusinessWebsite',

'contact\_first\_name' => 'setContactFirstName',

'contact\_last\_name' => 'setContactLastName',

'contact\_phone' => 'setContactPhone',

'contact\_fax' => 'setContactFax',

'contact\_email' => 'setContactEmail',

'contact\_title' => 'setContactTitle',

'ip' => 'setIp',

'tax\_exempt' => 'setTaxExempt'

];

protected static $getters = [

'submit\_time' => 'getSubmitTime',

'tran\_type' => 'getTranType',

'tran\_amt' => 'getTranAmt',

'tran\_tax' => 'getTranTax',

'tran\_inv' => 'getTranInv',

'tran\_po' => 'getTranPo',

'tran\_c\_num' => 'getTranCNum',

'tran\_note' => 'getTranNote',

'card\_num' => 'getCardNum',

'card\_name' => 'getCardName',

'card\_exp' => 'getCardExp',

'card\_type' => 'getCardType',

'avs\_street' => 'getAvsStreet',

'avs\_zip' => 'getAvsZip',

'cvv\_num' => 'getCvvNum',

'proc\_id' => 'getProcId',

'proc\_resp' => 'getProcResp',

'proc\_code' => 'getProcCode',

'proc\_mess' => 'getProcMess',

'avs\_code' => 'getAvsCode',

'avs\_resp' => 'getAvsResp',

'cvv\_code' => 'getCvvCode',

'cvv\_resp' => 'getCvvResp',

'items' => 'getItems',

'business\_name' => 'getBusinessName',

'business\_address' => 'getBusinessAddress',

'business\_address2' => 'getBusinessAddress2',

'business\_city' => 'getBusinessCity',

'business\_state' => 'getBusinessState',

'business\_zip' => 'getBusinessZip',

'business\_country' => 'getBusinessCountry',

'business\_website' => 'getBusinessWebsite',

'contact\_first\_name' => 'getContactFirstName',

'contact\_last\_name' => 'getContactLastName',

'contact\_phone' => 'getContactPhone',

'contact\_fax' => 'getContactFax',

'contact\_email' => 'getContactEmail',

'contact\_title' => 'getContactTitle',

'ip' => 'getIp',

'tax\_exempt' => 'getTaxExempt'

];

public static function attributeMap()

{

return self::$attributeMap;

}

public static function setters()

{

return self::$setters;

}

public static function getters()

{

return self::$getters;

}

public function getModelName()

{

return self::$swaggerModelName;

}

protected $container = [];

public function \_\_construct(array $data = null)

{

$this->container['submit\_time'] = isset($data['submit\_time']) ? $data['submit\_time'] : null;

$this->container['tran\_type'] = isset($data['tran\_type']) ? $data['tran\_type'] : null;

$this->container['tran\_amt'] = isset($data['tran\_amt']) ? $data['tran\_amt'] : null;

$this->container['tran\_tax'] = isset($data['tran\_tax']) ? $data['tran\_tax'] : null;

$this->container['tran\_inv'] = isset($data['tran\_inv']) ? $data['tran\_inv'] : null;

$this->container['tran\_po'] = isset($data['tran\_po']) ? $data['tran\_po'] : null;

$this->container['tran\_c\_num'] = isset($data['tran\_c\_num']) ? $data['tran\_c\_num'] : null;

$this->container['tran\_note'] = isset($data['tran\_note']) ? $data['tran\_note'] : null;

$this->container['card\_num'] = isset($data['card\_num']) ? $data['card\_num'] : null;

$this->container['card\_name'] = isset($data['card\_name']) ? $data['card\_name'] : null;

$this->container['card\_exp'] = isset($data['card\_exp']) ? $data['card\_exp'] : null;

$this->container['card\_type'] = isset($data['card\_type']) ? $data['card\_type'] : null;

$this->container['avs\_street'] = isset($data['avs\_street']) ? $data['avs\_street'] : null;

$this->container['avs\_zip'] = isset($data['avs\_zip']) ? $data['avs\_zip'] : null;

$this->container['cvv\_num'] = isset($data['cvv\_num']) ? $data['cvv\_num'] : null;

$this->container['proc\_id'] = isset($data['proc\_id']) ? $data['proc\_id'] : null;

$this->container['proc\_resp'] = isset($data['proc\_resp']) ? $data['proc\_resp'] : null;

$this->container['proc\_code'] = isset($data['proc\_code']) ? $data['proc\_code'] : null;

$this->container['proc\_mess'] = isset($data['proc\_mess']) ? $data['proc\_mess'] : null;

$this->container['avs\_code'] = isset($data['avs\_code']) ? $data['avs\_code'] : null;

$this->container['avs\_resp'] = isset($data['avs\_resp']) ? $data['avs\_resp'] : null;

$this->container['cvv\_code'] = isset($data['cvv\_code']) ? $data['cvv\_code'] : null;

$this->container['cvv\_resp'] = isset($data['cvv\_resp']) ? $data['cvv\_resp'] : null;

$this->container['items'] = isset($data['items']) ? $data['items'] : null;

$this->container['business\_name'] = isset($data['business\_name']) ? $data['business\_name'] : null;

$this->container['business\_address'] = isset($data['business\_address']) ? $data['business\_address'] : null;

$this->container['business\_address2'] = isset($data['business\_address2']) ? $data['business\_address2'] : null;

$this->container['business\_city'] = isset($data['business\_city']) ? $data['business\_city'] : null;

$this->container['business\_state'] = isset($data['business\_state']) ? $data['business\_state'] : null;

$this->container['business\_zip'] = isset($data['business\_zip']) ? $data['business\_zip'] : null;

$this->container['business\_country'] = isset($data['business\_country']) ? $data['business\_country'] : null;

$this->container['business\_website'] = isset($data['business\_website']) ? $data['business\_website'] : null;

$this->container['contact\_first\_name'] = isset($data['contact\_first\_name']) ? $data['contact\_first\_name'] : null;

$this->container['contact\_last\_name'] = isset($data['contact\_last\_name']) ? $data['contact\_last\_name'] : null;

$this->container['contact\_phone'] = isset($data['contact\_phone']) ? $data['contact\_phone'] : null;

$this->container['contact\_fax'] = isset($data['contact\_fax']) ? $data['contact\_fax'] : null;

$this->container['contact\_email'] = isset($data['contact\_email']) ? $data['contact\_email'] : null;

$this->container['contact\_title'] = isset($data['contact\_title']) ? $data['contact\_title'] : null;

$this->container['ip'] = isset($data['ip']) ? $data['ip'] : null;

$this->container['tax\_exempt'] = isset($data['tax\_exempt']) ? $data['tax\_exempt'] : null;

}

public function listInvalidProperties()

{

$invalidProperties = [];

return $invalidProperties;

}

public function valid()

{

return count($this->listInvalidProperties()) === 0;

}

public function getSubmitTime()

{

return $this->container['submit\_time'];

}

public function setSubmitTime($submit\_time)

{

$this->container['submit\_time'] = $submit\_time;

return $this;

}

public function getTranType()

{

return $this->container['tran\_type'];

}

public function setTranType($tran\_type)

{

$this->container['tran\_type'] = $tran\_type;

return $this;

}

public function getTranAmt()

{

return $this->container['tran\_amt'];

}

public function setTranAmt($tran\_amt)

{

$this->container['tran\_amt'] = $tran\_amt;

return $this;

}

public function getTranTax()

{

return $this->container['tran\_tax'];

}

public function setTranTax($tran\_tax)

{

$this->container['tran\_tax'] = $tran\_tax;

return $this;

}

public function getTranInv()

{

return $this->container['tran\_inv'];

}

public function setTranInv($tran\_inv)

{

$this->container['tran\_inv'] = $tran\_inv;

return $this;

}

public function getTranPo()

{

return $this->container['tran\_po'];

}

public function setTranPo($tran\_po)

{

$this->container['tran\_po'] = $tran\_po;

return $this;

}

public function getTranCNum()

{

return $this->container['tran\_c\_num'];

}

public function setTranCNum($tran\_c\_num)

{

$this->container['tran\_c\_num'] = $tran\_c\_num;

return $this;

}

public function getTranNote()

{

return $this->container['tran\_note'];

}

public function setTranNote($tran\_note)

{

$this->container['tran\_note'] = $tran\_note;

return $this;

}

public function getCardNum()

{

return $this->container['card\_num'];

}

public function setCardNum($card\_num)

{

$this->container['card\_num'] = $card\_num;

return $this;

}

public function getCardName()

{

return $this->container['card\_name'];

}

public function setCardName($card\_name)

{

$this->container['card\_name'] = $card\_name;

return $this;

}

public function getCardExp()

{

return $this->container['card\_exp'];

}

public function setCardExp($card\_exp)

{

$this->container['card\_exp'] = $card\_exp;

return $this;

}

public function getCardType()

{

return $this->container['card\_type'];

}

public function setCardType($card\_type)

{

$this->container['card\_type'] = $card\_type;

return $this;

}

public function getAvsStreet()

{

return $this->container['avs\_street'];

}

public function setAvsStreet($avs\_street)

{

$this->container['avs\_street'] = $avs\_street;

return $this;

}

public function getAvsZip()

{

return $this->container['avs\_zip'];

}

public function setAvsZip($avs\_zip)

{

$this->container['avs\_zip'] = $avs\_zip;

return $this;

}

public function getCvvNum()

{

return $this->container['cvv\_num'];

}

public function setCvvNum($cvv\_num)

{

$this->container['cvv\_num'] = $cvv\_num;

return $this;

}

public function getProcId()

{

return $this->container['proc\_id'];

}

public function setProcId($proc\_id)

{

$this->container['proc\_id'] = $proc\_id;

return $this;

}

public function getProcResp()

{

return $this->container['proc\_resp'];

}

public function setProcResp($proc\_resp)

{

$this->container['proc\_resp'] = $proc\_resp;

return $this;

}

public function getProcCode()

{

return $this->container['proc\_code'];

}

public function setProcCode($proc\_code)

{

$this->container['proc\_code'] = $proc\_code;

return $this;

}

public function getProcMess()

{

return $this->container['proc\_mess'];

}

public function setProcMess($proc\_mess)

{

$this->container['proc\_mess'] = $proc\_mess;

return $this;

}

public function getAvsCode()

{

return $this->container['avs\_code'];

}

public function setAvsCode($avs\_code)

{

$this->container['avs\_code'] = $avs\_code;

return $this;

}

public function getAvsResp()

{

return $this->container['avs\_resp'];

}

public function setAvsResp($avs\_resp)

{

$this->container['avs\_resp'] = $avs\_resp;

return $this;

}

public function getCvvCode()

{

return $this->container['cvv\_code'];

}

public function setCvvCode($cvv\_code)

{

$this->container['cvv\_code'] = $cvv\_code;

return $this;

}

public function getCvvResp()

{

return $this->container['cvv\_resp'];

}

public function setCvvResp($cvv\_resp)

{

$this->container['cvv\_resp'] = $cvv\_resp;

return $this;

}

public function getItems()

{

return $this->container['items'];

}

public function setItems($items)

{

$this->container['items'] = $items;

return $this;

}

public function getBusinessName()

{

return $this->container['business\_name'];

}

public function setBusinessName($business\_name)

{

$this->container['business\_name'] = $business\_name;

return $this;

}

public function getBusinessAddress()

{

return $this->container['business\_address'];

}

public function setBusinessAddress($business\_address)

{

$this->container['business\_address'] = $business\_address;

return $this;

}

public function getBusinessAddress2()

{

return $this->container['business\_address2'];

}

public function setBusinessAddress2($business\_address2)

{

$this->container['business\_address2'] = $business\_address2;

return $this;

}

public function getBusinessCity()

{

return $this->container['business\_city'];

}

public function setBusinessCity($business\_city)

{

$this->container['business\_city'] = $business\_city;

return $this;

}

public function getBusinessState()

{

return $this->container['business\_state'];

}

public function setBusinessState($business\_state)

{

$this->container['business\_state'] = $business\_state;

return $this;

}

public function getBusinessZip()

{

return $this->container['business\_zip'];

}

public function setBusinessZip($business\_zip)

{

$this->container['business\_zip'] = $business\_zip;

return $this;

}

public function getBusinessCountry()

{

return $this->container['business\_country'];

}

public function setBusinessCountry($business\_country)

{

$this->container['business\_country'] = $business\_country;

return $this;

}

public function getBusinessWebsite()

{

return $this->container['business\_website'];

}

public function setBusinessWebsite($business\_website)

{

$this->container['business\_website'] = $business\_website;

return $this;

}

public function getContactFirstName()

{

return $this->container['contact\_first\_name'];

}

public function setContactFirstName($contact\_first\_name)

{

$this->container['contact\_first\_name'] = $contact\_first\_name;

return $this;

}

public function getContactLastName()

{

return $this->container['contact\_last\_name'];

}

public function setContactLastName($contact\_last\_name)

{

$this->container['contact\_last\_name'] = $contact\_last\_name;

return $this;

}

public function getContactPhone()

{

return $this->container['contact\_phone'];

}

public function setContactPhone($contact\_phone)

{

$this->container['contact\_phone'] = $contact\_phone;

return $this;

}

public function getContactFax()

{

return $this->container['contact\_fax'];

}

public function setContactFax($contact\_fax)

{

$this->container['contact\_fax'] = $contact\_fax;

return $this;

}

public function getContactEmail()

{

return $this->container['contact\_email'];

}

public function setContactEmail($contact\_email)

{

$this->container['contact\_email'] = $contact\_email;

return $this;

}

public function getContactTitle()

{

return $this->container['contact\_title'];

}

public function setContactTitle($contact\_title)

{

$this->container['contact\_title'] = $contact\_title;

return $this;

}

public function getIp()

{

return $this->container['ip'];

}

public function setIp($ip)

{

$this->container['ip'] = $ip;

return $this;

}

public function getTaxExempt()

{

return $this->container['tax\_exempt'];

}

public function setTaxExempt($tax\_exempt)

{

$this->container['tax\_exempt'] = $tax\_exempt;

return $this;

}

public function offsetExists($offset)

{

return isset($this->container[$offset]);

}

public function offsetGet($offset)

{

return isset($this->container[$offset]) ? $this->container[$offset] : null;

}

public function offsetSet($offset, $value)

{

if (is\_null($offset)) {

$this->container[] = $value;

} else {

$this->container[$offset] = $value;

}

}

public function offsetUnset($offset)

{

unset($this->container[$offset]);

}

public function \_\_toString()

{

if (defined('JSON\_PRETTY\_PRINT')) { // use JSON pretty print

return json\_encode(

ObjectSerializer::sanitizeForSerialization($this),

JSON\_PRETTY\_PRINT

);

}

return json\_encode(ObjectSerializer::sanitizeForSerialization($this));

}

}

# TransactionItem

<?php

namespace Swagger\Client\Model;

use \ArrayAccess;

use \Swagger\Client\ObjectSerializer;

class TransactionItem implements ModelInterface, ArrayAccess

{

const DISCRIMINATOR = null;

protected static $swaggerModelName = 'TransactionItem';

protected static $swaggerTypes = [

'tax\_included' => 'bool',

'credit' => 'bool',

'extended\_amount' => 'double',

'item\_number' => 'string',

'unit\_cost' => 'double',

'quantity' => 'int',

'description' => 'string',

'discount' => 'double',

'tax' => 'double',

'commodity\_code' => 'string',

'unit\_of\_measure' => 'string'

];

protected static $swaggerFormats = [

'tax\_included' => null,

'credit' => null,

'extended\_amount' => 'double',

'item\_number' => null,

'unit\_cost' => 'double',

'quantity' => 'int32',

'description' => null,

'discount' => 'double',

'tax' => 'double',

'commodity\_code' => null,

'unit\_of\_measure' => null

];

public static function swaggerTypes()

{

return self::$swaggerTypes;

}

public static function swaggerFormats()

{

return self::$swaggerFormats;

}

protected static $attributeMap = [

'tax\_included' => 'TaxIncluded',

'credit' => 'Credit',

'extended\_amount' => 'ExtendedAmount',

'item\_number' => 'ItemNumber',

'unit\_cost' => 'UnitCost',

'quantity' => 'Quantity',

'description' => 'Description',

'discount' => 'Discount',

'tax' => 'Tax',

'commodity\_code' => 'CommodityCode',

'unit\_of\_measure' => 'UnitOfMeasure'

];

protected static $setters = [

'tax\_included' => 'setTaxIncluded',

'credit' => 'setCredit',

'extended\_amount' => 'setExtendedAmount',

'item\_number' => 'setItemNumber',

'unit\_cost' => 'setUnitCost',

'quantity' => 'setQuantity',

'description' => 'setDescription',

'discount' => 'setDiscount',

'tax' => 'setTax',

'commodity\_code' => 'setCommodityCode',

'unit\_of\_measure' => 'setUnitOfMeasure'

];

protected static $getters = [

'tax\_included' => 'getTaxIncluded',

'credit' => 'getCredit',

'extended\_amount' => 'getExtendedAmount',

'item\_number' => 'getItemNumber',

'unit\_cost' => 'getUnitCost',

'quantity' => 'getQuantity',

'description' => 'getDescription',

'discount' => 'getDiscount',

'tax' => 'getTax',

'commodity\_code' => 'getCommodityCode',

'unit\_of\_measure' => 'getUnitOfMeasure'

];

public static function attributeMap()

{

return self::$attributeMap;

}

public static function setters()

{

return self::$setters;

}

public static function getters()

{

return self::$getters;

}

public function getModelName()

{

return self::$swaggerModelName;

}

protected $container = [];

public function \_\_construct(array $data = null)

{

$this->container['tax\_included'] = isset($data['tax\_included']) ? $data['tax\_included'] : null;

$this->container['credit'] = isset($data['credit']) ? $data['credit'] : null;

$this->container['extended\_amount'] = isset($data['extended\_amount']) ? $data['extended\_amount'] : null;

$this->container['item\_number'] = isset($data['item\_number']) ? $data['item\_number'] : null;

$this->container['unit\_cost'] = isset($data['unit\_cost']) ? $data['unit\_cost'] : null;

$this->container['quantity'] = isset($data['quantity']) ? $data['quantity'] : null;

$this->container['description'] = isset($data['description']) ? $data['description'] : null;

$this->container['discount'] = isset($data['discount']) ? $data['discount'] : null;

$this->container['tax'] = isset($data['tax']) ? $data['tax'] : null;

$this->container['commodity\_code'] = isset($data['commodity\_code']) ? $data['commodity\_code'] : null;

$this->container['unit\_of\_measure'] = isset($data['unit\_of\_measure']) ? $data['unit\_of\_measure'] : null;

}

public function listInvalidProperties()

{

$invalidProperties = [];

return $invalidProperties;

}

public function valid()

{

return count($this->listInvalidProperties()) === 0;

}

public function getTaxIncluded()

{

return $this->container['tax\_included'];

}

public function setTaxIncluded($tax\_included)

{

$this->container['tax\_included'] = $tax\_included;

return $this;

}

public function getCredit()

{

return $this->container['credit'];

}

public function setCredit($credit)

{

$this->container['credit'] = $credit;

return $this;

}

public function getExtendedAmount()

{

return $this->container['extended\_amount'];

}

public function setExtendedAmount($extended\_amount)

{

$this->container['extended\_amount'] = $extended\_amount;

return $this;

}

public function getItemNumber()

{

return $this->container['item\_number'];

}

public function setItemNumber($item\_number)

{

$this->container['item\_number'] = $item\_number;

return $this;

}

public function getUnitCost()

{

return $this->container['unit\_cost'];

}

public function setUnitCost($unit\_cost)

{

$this->container['unit\_cost'] = $unit\_cost;

return $this;

}

public function getQuantity()

{

return $this->container['quantity'];

}

public function setQuantity($quantity)

{

$this->container['quantity'] = $quantity;

return $this;

}

public function getDescription()

{

return $this->container['description'];

}

public function setDescription($description)

{

$this->container['description'] = $description;

return $this;

}

public function getDiscount()

{

return $this->container['discount'];

}

public function setDiscount($discount)

{

$this->container['discount'] = $discount;

return $this;

}

public function getTax()

{

return $this->container['tax'];

}

public function setTax($tax)

{

$this->container['tax'] = $tax;

return $this;

}

public function getCommodityCode()

{

return $this->container['commodity\_code'];

}

public function setCommodityCode($commodity\_code)

{

$this->container['commodity\_code'] = $commodity\_code;

return $this;

}

public function getUnitOfMeasure()

{

return $this->container['unit\_of\_measure'];

}

public function setUnitOfMeasure($unit\_of\_measure)

{

$this->container['unit\_of\_measure'] = $unit\_of\_measure;

return $this;

}

public function offsetExists($offset)

{

return isset($this->container[$offset]);

}

public function offsetGet($offset)

{

return isset($this->container[$offset]) ? $this->container[$offset] : null;

}

public function offsetSet($offset, $value)

{

if (is\_null($offset)) {

$this->container[] = $value;

} else {

$this->container[$offset] = $value;

}

}

public function offsetUnset($offset)

{

unset($this->container[$offset]);

}

public function \_\_toString()

{

if (defined('JSON\_PRETTY\_PRINT')) { // use JSON pretty print

return json\_encode(

ObjectSerializer::sanitizeForSerialization($this),

JSON\_PRETTY\_PRINT

);

}

return json\_encode(ObjectSerializer::sanitizeForSerialization($this));

}

}

# TransactionType

<?php

namespace Swagger\Client\Model;

use \Swagger\Client\ObjectSerializer;

class TransactionType

{

const SALE = 'Sale';

const CREDIT = 'Credit';

const AUTHORIZATION = 'Authorization';

const PRE\_AUTH = 'PreAuth';

const POST\_AUTH = 'PostAuth';

const FORCE\_AUTH = 'ForceAuth';

const VOID = 'Void';

const QUERY = 'Query';

const VERIFY = 'Verify';

const MARK = 'Mark';

const \_RETURN = 'Return';

const \_USE = 'Use';

public static function getAllowableEnumValues()

{

return [

self::SALE,

self::CREDIT,

self::AUTHORIZATION,

self::PRE\_AUTH,

self::POST\_AUTH,

self::FORCE\_AUTH,

self::VOID,

self::QUERY,

self::VERIFY,

self::MARK,

self::\_RETURN,

self::\_USE,

];

}

}

# UpdateTransactionInfo

<?php

namespace Swagger\Client\Model;

use \ArrayAccess;

use \Swagger\Client\ObjectSerializer;

class UpdateTransactionInfo implements ModelInterface, ArrayAccess

{

const DISCRIMINATOR = null;

protected static $swaggerModelName = 'UpdateTransactionInfo';

protected static $swaggerTypes = [

'invoice\_number' => 'string',

'customer\_number' => 'string',

'po\_number' => 'string',

'note' => 'string'

];

protected static $swaggerFormats = [

'invoice\_number' => null,

'customer\_number' => null,

'po\_number' => null,

'note' => null

];

public static function swaggerTypes()

{

return self::$swaggerTypes;

}

public static function swaggerFormats()

{

return self::$swaggerFormats;

}

protected static $attributeMap = [

'invoice\_number' => 'InvoiceNumber',

'customer\_number' => 'CustomerNumber',

'po\_number' => 'PONumber',

'note' => 'Note'

];

protected static $setters = [

'invoice\_number' => 'setInvoiceNumber',

'customer\_number' => 'setCustomerNumber',

'po\_number' => 'setPoNumber',

'note' => 'setNote'

];

protected static $getters = [

'invoice\_number' => 'getInvoiceNumber',

'customer\_number' => 'getCustomerNumber',

'po\_number' => 'getPoNumber',

'note' => 'getNote'

];

public static function attributeMap()

{

return self::$attributeMap;

}

public static function setters()

{

return self::$setters;

}

public static function getters()

{

return self::$getters;

}

public function getModelName()

{

return self::$swaggerModelName;

}

protected $container = [];

public function \_\_construct(array $data = null)

{

$this->container['invoice\_number'] = isset($data['invoice\_number']) ? $data['invoice\_number'] : null;

$this->container['customer\_number'] = isset($data['customer\_number']) ? $data['customer\_number'] : null;

$this->container['po\_number'] = isset($data['po\_number']) ? $data['po\_number'] : null;

$this->container['note'] = isset($data['note']) ? $data['note'] : null;

}

public function listInvalidProperties()

{

$invalidProperties = [];

return $invalidProperties;

}

public function valid()

{

return count($this->listInvalidProperties()) === 0;

}

public function getInvoiceNumber()

{

return $this->container['invoice\_number'];

}

public function setInvoiceNumber($invoice\_number)

{

$this->container['invoice\_number'] = $invoice\_number;

return $this;

}

public function getCustomerNumber()

{

return $this->container['customer\_number'];

}

public function setCustomerNumber($customer\_number)

{

$this->container['customer\_number'] = $customer\_number;

return $this;

}

public function getPoNumber()

{

return $this->container['po\_number'];

}

public function setPoNumber($po\_number)

{

$this->container['po\_number'] = $po\_number;

return $this;

}

public function getNote()

{

return $this->container['note'];

}

public function setNote($note)

{

$this->container['note'] = $note;

return $this;

}

public function offsetExists($offset)

{

return isset($this->container[$offset]);

}

public function offsetGet($offset)

{

return isset($this->container[$offset]) ? $this->container[$offset] : null;

}

public function offsetSet($offset, $value)

{

if (is\_null($offset)) {

$this->container[] = $value;

} else {

$this->container[$offset] = $value;

}

}

public function offsetUnset($offset)

{

unset($this->container[$offset]);

}

public function \_\_toString()

{

if (defined('JSON\_PRETTY\_PRINT')) { // use JSON pretty print

return json\_encode(

ObjectSerializer::sanitizeForSerialization($this),

JSON\_PRETTY\_PRINT

);

}

return json\_encode(ObjectSerializer::sanitizeForSerialization($this));

}

}

# VaultCheck

<?php

namespace Swagger\Client\Model;

use \ArrayAccess;

use \Swagger\Client\ObjectSerializer;

class VaultCheck implements ModelInterface, ArrayAccess

{

const DISCRIMINATOR = null;

protected static $swaggerModelName = 'VaultCheck';

protected static $swaggerTypes = [

'response' => '\Swagger\Client\Model\Response',

'check\_id' => 'int',

'main' => 'bool',

'check\_routing' => 'string',

'check\_account' => 'string',

'check\_number' => 'string',

'check\_first\_name' => 'string',

'check\_last\_name' => 'string',

'note' => 'string',

'masked\_routing' => 'string',

'masked\_account' => 'string',

'account\_type' => 'string'

];

protected static $swaggerFormats = [

'response' => null,

'check\_id' => 'int32',

'main' => null,

'check\_routing' => null,

'check\_account' => null,

'check\_number' => null,

'check\_first\_name' => null,

'check\_last\_name' => null,

'note' => null,

'masked\_routing' => null,

'masked\_account' => null,

'account\_type' => null

];

public static function swaggerTypes()

{

return self::$swaggerTypes;

}

public static function swaggerFormats()

{

return self::$swaggerFormats;

}

protected static $attributeMap = [

'response' => 'Response',

'check\_id' => 'CheckID',

'main' => 'Main',

'check\_routing' => 'CheckRouting',

'check\_account' => 'CheckAccount',

'check\_number' => 'CheckNumber',

'check\_first\_name' => 'CheckFirstName',

'check\_last\_name' => 'CheckLastName',

'note' => 'Note',

'masked\_routing' => 'MaskedRouting',

'masked\_account' => 'MaskedAccount',

'account\_type' => 'AccountType'

];

protected static $setters = [

'response' => 'setResponse',

'check\_id' => 'setCheckId',

'main' => 'setMain',

'check\_routing' => 'setCheckRouting',

'check\_account' => 'setCheckAccount',

'check\_number' => 'setCheckNumber',

'check\_first\_name' => 'setCheckFirstName',

'check\_last\_name' => 'setCheckLastName',

'note' => 'setNote',

'masked\_routing' => 'setMaskedRouting',

'masked\_account' => 'setMaskedAccount',

'account\_type' => 'setAccountType'

];

protected static $getters = [

'response' => 'getResponse',

'check\_id' => 'getCheckId',

'main' => 'getMain',

'check\_routing' => 'getCheckRouting',

'check\_account' => 'getCheckAccount',

'check\_number' => 'getCheckNumber',

'check\_first\_name' => 'getCheckFirstName',

'check\_last\_name' => 'getCheckLastName',

'note' => 'getNote',

'masked\_routing' => 'getMaskedRouting',

'masked\_account' => 'getMaskedAccount',

'account\_type' => 'getAccountType'

];

public static function attributeMap()

{

return self::$attributeMap;

}

public static function setters()

{

return self::$setters;

}

public static function getters()

{

return self::$getters;

}

public function getModelName()

{

return self::$swaggerModelName;

}

protected $container = [];

public function \_\_construct(array $data = null)

{

$this->container['response'] = isset($data['response']) ? $data['response'] : null;

$this->container['check\_id'] = isset($data['check\_id']) ? $data['check\_id'] : null;

$this->container['main'] = isset($data['main']) ? $data['main'] : null;

$this->container['check\_routing'] = isset($data['check\_routing']) ? $data['check\_routing'] : null;

$this->container['check\_account'] = isset($data['check\_account']) ? $data['check\_account'] : null;

$this->container['check\_number'] = isset($data['check\_number']) ? $data['check\_number'] : null;

$this->container['check\_first\_name'] = isset($data['check\_first\_name']) ? $data['check\_first\_name'] : null;

$this->container['check\_last\_name'] = isset($data['check\_last\_name']) ? $data['check\_last\_name'] : null;

$this->container['note'] = isset($data['note']) ? $data['note'] : null;

$this->container['masked\_routing'] = isset($data['masked\_routing']) ? $data['masked\_routing'] : null;

$this->container['masked\_account'] = isset($data['masked\_account']) ? $data['masked\_account'] : null;

$this->container['account\_type'] = isset($data['account\_type']) ? $data['account\_type'] : null;

}

public function listInvalidProperties()

{

$invalidProperties = [];

return $invalidProperties;

}

public function valid()

{

return count($this->listInvalidProperties()) === 0;

}

public function getResponse()

{

return $this->container['response'];

}

public function setResponse($response)

{

$this->container['response'] = $response;

return $this;

}

public function getCheckId()

{

return $this->container['check\_id'];

}

public function setCheckId($check\_id)

{

$this->container['check\_id'] = $check\_id;

return $this;

}

public function getMain()

{

return $this->container['main'];

}

public function setMain($main)

{

$this->container['main'] = $main;

return $this;

}

public function getCheckRouting()

{

return $this->container['check\_routing'];

}

public function setCheckRouting($check\_routing)

{

$this->container['check\_routing'] = $check\_routing;

return $this;

}

public function getCheckAccount()

{

return $this->container['check\_account'];

}

public function setCheckAccount($check\_account)

{

$this->container['check\_account'] = $check\_account;

return $this;

}

public function getCheckNumber()

{

return $this->container['check\_number'];

}

public function setCheckNumber($check\_number)

{

$this->container['check\_number'] = $check\_number;

return $this;

}

public function getCheckFirstName()

{

return $this->container['check\_first\_name'];

}

public function setCheckFirstName($check\_first\_name)

{

$this->container['check\_first\_name'] = $check\_first\_name;

return $this;

}

public function getCheckLastName()

{

return $this->container['check\_last\_name'];

}

public function setCheckLastName($check\_last\_name)

{

$this->container['check\_last\_name'] = $check\_last\_name;

return $this;

}

public function getNote()

{

return $this->container['note'];

}

public function setNote($note)

{

$this->container['note'] = $note;

return $this;

}

public function getMaskedRouting()

{

return $this->container['masked\_routing'];

}

public function setMaskedRouting($masked\_routing)

{

$this->container['masked\_routing'] = $masked\_routing;

return $this;

}

public function getMaskedAccount()

{

return $this->container['masked\_account'];

}

public function setMaskedAccount($masked\_account)

{

$this->container['masked\_account'] = $masked\_account;

return $this;

}

public function getAccountType()

{

return $this->container['account\_type'];

}

public function setAccountType($account\_type)

{

$this->container['account\_type'] = $account\_type;

return $this;

}

public function offsetExists($offset)

{

return isset($this->container[$offset]);

}

public function offsetGet($offset)

{

return isset($this->container[$offset]) ? $this->container[$offset] : null;

}

public function offsetSet($offset, $value)

{

if (is\_null($offset)) {

$this->container[] = $value;

} else {

$this->container[$offset] = $value;

}

}

public function offsetUnset($offset)

{

unset($this->container[$offset]);

}

public function \_\_toString()

{

if (defined('JSON\_PRETTY\_PRINT')) { // use JSON pretty print

return json\_encode(

ObjectSerializer::sanitizeForSerialization($this),

JSON\_PRETTY\_PRINT

);

}

return json\_encode(ObjectSerializer::sanitizeForSerialization($this));

}

}

# VaultCreditCard

<?php

namespace Swagger\Client\Model;

use \ArrayAccess;

use \Swagger\Client\ObjectSerializer;

class VaultCreditCard implements ModelInterface, ArrayAccess

{

const DISCRIMINATOR = null;

protected static $swaggerModelName = 'VaultCreditCard';

protected static $swaggerTypes = [

'magnetic\_tracks' => 'string',

'track1' => 'string',

'track2' => 'string',

'number' => 'string',

'masked\_number' => 'string',

'holder' => 'string',

'exp\_date' => 'string',

'cvv' => 'string',

'address' => 'string',

'city' => 'string',

'state' => 'string',

'zip' => 'string',

'swiped' => 'bool',

'type\_string' => 'string',

'cvv\_type' => '\Swagger\Client\Model\CVVPresenceType',

'not\_present' => 'bool',

'response' => '\Swagger\Client\Model\Response',

'card\_id' => 'int',

'note' => 'string',

'main' => 'bool'

];

protected static $swaggerFormats = [

'magnetic\_tracks' => null,

'track1' => null,

'track2' => null,

'number' => null,

'masked\_number' => null,

'holder' => null,

'exp\_date' => null,

'cvv' => null,

'address' => null,

'city' => null,

'state' => null,

'zip' => null,

'swiped' => null,

'type\_string' => null,

'cvv\_type' => null,

'not\_present' => null,

'response' => null,

'card\_id' => 'int32',

'note' => null,

'main' => null

];

public static function swaggerTypes()

{

return self::$swaggerTypes;

}

public static function swaggerFormats()

{

return self::$swaggerFormats;

}

protected static $attributeMap = [

'magnetic\_tracks' => 'MagneticTracks',

'track1' => 'Track1',

'track2' => 'Track2',

'number' => 'Number',

'masked\_number' => 'MaskedNumber',

'holder' => 'Holder',

'exp\_date' => 'ExpDate',

'cvv' => 'CVV',

'address' => 'Address',

'city' => 'City',

'state' => 'State',

'zip' => 'ZIP',

'swiped' => 'Swiped',

'type\_string' => 'TypeString',

'cvv\_type' => 'CVVType',

'not\_present' => 'NotPresent',

'response' => 'Response',

'card\_id' => 'CardID',

'note' => 'Note',

'main' => 'Main'

];

protected static $setters = [

'magnetic\_tracks' => 'setMagneticTracks',

'track1' => 'setTrack1',

'track2' => 'setTrack2',

'number' => 'setNumber',

'masked\_number' => 'setMaskedNumber',

'holder' => 'setHolder',

'exp\_date' => 'setExpDate',

'cvv' => 'setCvv',

'address' => 'setAddress',

'city' => 'setCity',

'state' => 'setState',

'zip' => 'setZip',

'swiped' => 'setSwiped',

'type\_string' => 'setTypeString',

'cvv\_type' => 'setCvvType',

'not\_present' => 'setNotPresent',

'response' => 'setResponse',

'card\_id' => 'setCardId',

'note' => 'setNote',

'main' => 'setMain'

];

protected static $getters = [

'magnetic\_tracks' => 'getMagneticTracks',

'track1' => 'getTrack1',

'track2' => 'getTrack2',

'number' => 'getNumber',

'masked\_number' => 'getMaskedNumber',

'holder' => 'getHolder',

'exp\_date' => 'getExpDate',

'cvv' => 'getCvv',

'address' => 'getAddress',

'city' => 'getCity',

'state' => 'getState',

'zip' => 'getZip',

'swiped' => 'getSwiped',

'type\_string' => 'getTypeString',

'cvv\_type' => 'getCvvType',

'not\_present' => 'getNotPresent',

'response' => 'getResponse',

'card\_id' => 'getCardId',

'note' => 'getNote',

'main' => 'getMain'

];

public static function attributeMap()

{

return self::$attributeMap;

}

public static function setters()

{

return self::$setters;

}

public static function getters()

{

return self::$getters;

}

public function getModelName()

{

return self::$swaggerModelName;

}

protected $container = [];

public function \_\_construct(array $data = null)

{

$this->container['magnetic\_tracks'] = isset($data['magnetic\_tracks']) ? $data['magnetic\_tracks'] : null;

$this->container['track1'] = isset($data['track1']) ? $data['track1'] : null;

$this->container['track2'] = isset($data['track2']) ? $data['track2'] : null;

$this->container['number'] = isset($data['number']) ? $data['number'] : null;

$this->container['masked\_number'] = isset($data['masked\_number']) ? $data['masked\_number'] : null;

$this->container['holder'] = isset($data['holder']) ? $data['holder'] : null;

$this->container['exp\_date'] = isset($data['exp\_date']) ? $data['exp\_date'] : null;

$this->container['cvv'] = isset($data['cvv']) ? $data['cvv'] : null;

$this->container['address'] = isset($data['address']) ? $data['address'] : null;

$this->container['city'] = isset($data['city']) ? $data['city'] : null;

$this->container['state'] = isset($data['state']) ? $data['state'] : null;

$this->container['zip'] = isset($data['zip']) ? $data['zip'] : null;

$this->container['swiped'] = isset($data['swiped']) ? $data['swiped'] : null;

$this->container['type\_string'] = isset($data['type\_string']) ? $data['type\_string'] : null;

$this->container['cvv\_type'] = isset($data['cvv\_type']) ? $data['cvv\_type'] : null;

$this->container['not\_present'] = isset($data['not\_present']) ? $data['not\_present'] : null;

$this->container['response'] = isset($data['response']) ? $data['response'] : null;

$this->container['card\_id'] = isset($data['card\_id']) ? $data['card\_id'] : null;

$this->container['note'] = isset($data['note']) ? $data['note'] : null;

$this->container['main'] = isset($data['main']) ? $data['main'] : null;

}

public function listInvalidProperties()

{

$invalidProperties = [];

return $invalidProperties;

}

public function valid()

{

return count($this->listInvalidProperties()) === 0;

}

public function getMagneticTracks()

{

return $this->container['magnetic\_tracks'];

}

public function setMagneticTracks($magnetic\_tracks)

{

$this->container['magnetic\_tracks'] = $magnetic\_tracks;

return $this;

}

public function getTrack1()

{

return $this->container['track1'];

}

public function setTrack1($track1)

{

$this->container['track1'] = $track1;

return $this;

}

public function getTrack2()

{

return $this->container['track2'];

}

public function setTrack2($track2)

{

$this->container['track2'] = $track2;

return $this;

}

public function getNumber()

{

return $this->container['number'];

}

public function setNumber($number)

{

$this->container['number'] = $number;

return $this;

}

public function getMaskedNumber()

{

return $this->container['masked\_number'];

}

public function setMaskedNumber($masked\_number)

{

$this->container['masked\_number'] = $masked\_number;

return $this;

}

public function getHolder()

{

return $this->container['holder'];

}

public function setHolder($holder)

{

$this->container['holder'] = $holder;

return $this;

}

public function getExpDate()

{

return $this->container['exp\_date'];

}

public function setExpDate($exp\_date)

{

$this->container['exp\_date'] = $exp\_date;

return $this;

}

public function getCvv()

{

return $this->container['cvv'];

}

public function setCvv($cvv)

{

$this->container['cvv'] = $cvv;

return $this;

}

public function getAddress()

{

return $this->container['address'];

}

public function setAddress($address)

{

$this->container['address'] = $address;

return $this;

}

public function getCity()

{

return $this->container['city'];

}

public function setCity($city)

{

$this->container['city'] = $city;

return $this;

}

public function getState()

{

return $this->container['state'];

}

public function setState($state)

{

$this->container['state'] = $state;

return $this;

}

public function getZip()

{

return $this->container['zip'];

}

public function setZip($zip)

{

$this->container['zip'] = $zip;

return $this;

}

public function getSwiped()

{

return $this->container['swiped'];

}

public function setSwiped($swiped)

{

$this->container['swiped'] = $swiped;

return $this;

}

public function getTypeString()

{

return $this->container['type\_string'];

}

public function setTypeString($type\_string)

{

$this->container['type\_string'] = $type\_string;

return $this;

}

public function getCvvType()

{

return $this->container['cvv\_type'];

}

public function setCvvType($cvv\_type)

{

$this->container['cvv\_type'] = $cvv\_type;

return $this;

}

public function getNotPresent()

{

return $this->container['not\_present'];

}

public function setNotPresent($not\_present)

{

$this->container['not\_present'] = $not\_present;

return $this;

}

public function getResponse()

{

return $this->container['response'];

}

public function setResponse($response)

{

$this->container['response'] = $response;

return $this;

}

public function getCardId()

{

return $this->container['card\_id'];

}

public function setCardId($card\_id)

{

$this->container['card\_id'] = $card\_id;

return $this;

}

public function getNote()

{

return $this->container['note'];

}

public function setNote($note)

{

$this->container['note'] = $note;

return $this;

}

public function getMain()

{

return $this->container['main'];

}

public function setMain($main)

{

$this->container['main'] = $main;

return $this;

}

public function offsetExists($offset)

{

return isset($this->container[$offset]);

}

public function offsetGet($offset)

{

return isset($this->container[$offset]) ? $this->container[$offset] : null;

}

public function offsetSet($offset, $value)

{

if (is\_null($offset)) {

$this->container[] = $value;

} else {

$this->container[$offset] = $value;

}

}

public function offsetUnset($offset)

{

unset($this->container[$offset]);

}

public function \_\_toString()

{

if (defined('JSON\_PRETTY\_PRINT')) { // use JSON pretty print

return json\_encode(

ObjectSerializer::sanitizeForSerialization($this),

JSON\_PRETTY\_PRINT

);

}

return json\_encode(ObjectSerializer::sanitizeForSerialization($this));

}

}