Non-merchant Gateway Methods

Table of Contents

Required Methods

> As of version 3.1 of Blesta, several of these required methods can be defined in a gateway configuration file instead.

The following methods are required to be implemented for each non-merchant gateway.

getName()

The getName() method simply returns the name of the gateway. It's always best to define any language in your gateway using language files (see Translating Blesta for more information).

> This method can be defined in a gateway configuration file instead.

```
class MyGateway extends NonmerchantGateway {
...
	public function getName() {
		return Language::_("MyGateway.name", true);
	}
...
}
```

getVersion()

This method must return the current version number of the gateway. Upon installation or upgrade, Blesta records the current code version so that it can tell when an upgrade/downgrade is available. The version number should be compatible with PHP's version_compare() function.

> This method can be defined in a gateway configuration file instead.

```
class MyGateway extends NonmerchantGateway {
	const VERSION = "1.0.0";
...
	public function getVersion() {
		return self::VERSION;
	}
...

}
```

getAuthors()

This method returns an array containing information about the authors of the gateway.

> This method can be defined in a gateway configuration file instead.

```
class MyGateway extends NonmerchantGateway {
    private static $authors = array(array('name' => "Phillips Data, Inc.", 'url' =>"http://www.blesta.com"));
...
	public function getAuthors() {
		return self::$authors;
	}
...
}
```

getCurrencies()

This method returns an array containing a set of all currencies supported by the gateway. Each currency should follow the ISO 4217 format for 3-character currency codes.

> This method can be defined in a gateway configuration file instead.

```
class MyGateway extends NonmerchantGateway {
...
	public function getCurrencies() {
		return array("AUD", "GBP", "USD");
	}
...
}
```

setCurrency()

This method sets the currency to be used for subsequent payments.

```
class MyGateway extends NonmerchantGateway {
...
	public function setCurrency() {
		$this->currency = $currency;
	}
...
}
```

getSettings(array $meta=null)

This method returns HTML content for the gateway's management page. The management page usually consists of form fields where API credentials can be entered. A set of existing *$meta* data may be given in order to pre-populate the form fields with data already saved for the gateway.

```
class MyGateway extends NonmerchantGateway {
...
	public function getSettings(array $meta=null) {
		$this->currency = $currency;
	}
...
}
```

editSettings(array $meta)

This method validates and returns data to be saved for the gateway. This data usually consists of API credentials necessary for processing payments through the gateway. The $meta fields to validate consist of those defined from *getSettings()*.

```
class MyGateway extends NonmerchantGateway {
...
	public function editSettings(array $meta) {
		// Define any rules necessary to validate the meta data given is valid and usable
		$rules = array(
			'account_email' => array(
				'valid' => array(
					'rule' => "isEmail",
					'message' => Language::_("MyGateway.!error.account_email.valid", true)
				)
			),
			'password' => array(
				'empty' => array(
					'rule' => "isEmpty",
					'negate' => true,
					'message' => Language::_("MyGateway.!error.password.empty", true)
				)
			)
		);

		// Set the rules to validate the $meta data against
		$this->Input->setRules($rules);

		// Validate whether the data given is valid
		$this->Input->validates($meta);

		// Return the meta data given
		return $meta;
	}
...
}
```

encryptableFields()

This method returns a list of field names that should be stored encrypted from the meta data returned from *editSettings()*.

```
class MyGateway extends NonmerchantGateway {
...
	public function encryptableFields() {
		return array("password");
	}
...
}
```

setMeta()

This method sets the meta data configured for the gateway for subsequent payments.

```
class MyGateway extends NonmerchantGateway {
...
	public function setMeta(array $meta=null) {
		$this->meta = $meta;
	}
...
}
```

validate(array $get, array $post)

This method validates the response from the gateway after a payment was made. Information on the payment must be returned in order to record the transaction in Blesta.

```
class MyGateway extends NonmerchantGateway {
...
	public function validate(array $get, array $post) {
		// Log the request received
		$this->log($this->ifSet($_SERVER['REQUEST_URI']), serialize($post), "output", true);

		#
		# TODO: verify the payment information received from the gateway is valid
		#
		$status = "approved";

		// Return the payment information
		return array(
			'client_id' => $this->ifSet($get['client_id']),
			'amount' => $this->ifSet($post['amount']),
			'currency' => $this->ifSet($post['currency']),
			'status' => $status,
			'reference_id' => null,
			'transaction_id' => $this->ifSet($post['transaction_id']),
			'parent_transaction_id' => null,
			'invoices' => array()
		);
	 }
...
}
```

success(array $get, array $post)

This method returns data for a successful transaction, and is used when a client returns to Blesta from the non-merchant gateway's website.

```
class MyGateway extends NonmerchantGateway {
...
	public function success(array $get, array $post) {
		// Return the payment information
		return array(
			'client_id' => $this->ifSet($get['client_id']),
			'amount' => $this->ifSet($post['amount']),
			'currency' => $this->ifSet($post['currency']),
			'status' => "approved",
			'reference_id' => null,
			'transaction_id' => $this->ifSet($post['transaction_id']),
			'parent_transaction_id' => null,
			'invoices' => array()
		);
	 }
...
}
```

Optional Methods

buildProcess(array $contact_info, $amount, array $invoice_amounts=null, array $options=null)

This method returns HTML content for the payment page from which a client can process a payment. This HTML usually consists of a form button that takes the client off-site to make a payment.

```
class MyGateway extends NonmerchantGateway {
...
        public function buildProcess(array $contact_info, $amount, array $invoice_amounts=null, array
$options=null) {
                // Fetch the view template that renders the fields from the gateway's "process.pdt" template
                $this->view = $this->makeView("process", "default", str_replace(ROOTWEBDIR, "", dirname
(__FILE__) . DS));

        // Load the helpers required for this view
        Loader::loadHelpers($this, array("Form", "Html"));

        $fields = array();
        $post_to = "";

        #
        # TODO: Define all form fields and the $post_to fields
        #
        $this->view->set("post_to", $post_to);
        $this->view->set("fields", $fields);

                return $this->view->fetch();
        }
...
}
```

buildAuthorize(array $contact_info, $amount, array $invoice_amounts=null, array $options=null)

This method is very similar to *buildProcess()*. It returns HTML content for the payment page from which a client can authorize a payment. Authorized payments are not completed until they are captured later.

capture($reference_id, $transaction_id, $amount, array $invoice_amounts=null)

This method is currently unused.

```
class MyGateway extends NonmerchantGateway {
...
        public function capture($reference_id, $transaction_id, $amount, array $invoice_amounts=null) {
                // Method is unsupported
                if (isset($this->Input))
            $this->Input->setErrors($this->getCommonError("unsupported"));
        }
...
}
```

void($reference_id, $transaction_id, $notes=null)

This method voids a payment transaction with the gateway. If voiding a payment is not supported by the gateway, and is not defined, the common "unsupported" error is set instead.

```
class MyGateway extends NonmerchantGateway {
...
        public function void($reference_id, $transaction_id, $notes=null) {
                // Method is unsupported
                if (isset($this->Input))
            $this->Input->setErrors($this->getCommonError("unsupported"));
        }
...
}
```

refund($reference_id, $transaction_id, $amount, $notes=null)

This method refunds a payment transaction with the gateway. If refunding a payment is not supported by the gateway, and is not defined, the common "unsupported" error is set instead.

```
class MyGateway extends NonmerchantGateway {
...
	public function refund($reference_id, $transaction_id, $amount, $notes=null) {
		// Method is unsupported
		if (isset($this->Input))
	    $this->Input->setErrors($this->getCommonError("unsupported"));
	}
...
}
```

install()

This method is invoked when the gateway is installed to perform any installation logic. For example, if the gateway requires any dependencies, you could check that those dependencies are met before the gateway is installed. If the gateway should not be installed, set Input errors.

```
class MyGateway extends NonmerchantGateway {
...
	public function install() {
		// Nothing to do
	}
...
}
```

upgrade($current_version)

This method is invoked when the gateway is upgraded from one version to another. Any gateway upgrade logic between versions can be performed here.

```
class MyGateway extends NonmerchantGateway {
...
	public function upgrade($current_version) {
		// Nothing to do
	}
...
}
```

uninstall($gateway_id, $last_instance)

This method is invoked when the gateway is uninstalled from a company. Any gateway cleanup logic can be performed here.

```
class MyGateway extends NonmerchantGateway {
...
	public function uninstall($gateway_id, $last_instance) {
		// Nothing to do
	}
...
}
```

getDescription()

The getDescription() method simply returns the description of the gateway. It's always best to define any language in your gateway using language files (see Translating Blesta for more information).

> This method was added in Blesta v4.9.0 and can be defined in a gateway configuration file instead.

```
class MyGateway extends NonmerchantGateway {
...
	public function getDescription() {
		return Language::_("MyGateway.description", true);
	}
...
}
```

getLogo()

This method returns the relative path from the gateway's directory to the logo image. If not defined, the logo set in the configuration file will be used, or the logo file at *views/default/images/logo.png*.

> This method can be defined in a gateway configuration file instead.

```
class MyGateway extends NonmerchantGateway {
...
	public function getLogo() {
		return "views/default/images/mygateway.png";
	}
...
}
```

getSignupUrl()

This method returns the URL to signup for an account with the gateway. If not defined, the signup URL set in the configuration file will be used.

> This method can be defined in a gateway configuration file instead.

```
class MyGateway extends NonmerchantGateway {
...
	public function getSignupUrl() {
		return "https://domain.com/signup/";
	}
...
}
```