

# Helping functions and methods

Method	Return	Description
<b>ResursBankModuleData::getPrefix</b> (\$extra)	A prefix	Returns the default prefix that is used by the plugin internally. Adding an extra variable, will return "the_prefix_\$extra". For example, all configuration data are using "admin" as an extra parameter, so currently the returned data is <b>trbwc_admin</b> . Getting configuration data in this way, for example if you want to determine if the plugin is enabled the complete option key in the database looks like <b>trbwc_admin_enabled</b> .
<b>ResursBankModuleData::getImage</b> (\$imageName)	URL	\$imageName is the short name (i.e. "logotype") of an image that usually resides in /images in the plugin structure. <i>Loading will occur with an autodetected file extension which limits the risks of loading something not allowed.</i>  Example: <b>getImage('author-photo')</b> Returns: <b>https://url.com/plugin/url/author-photo.[jpg png gif]</b>
<b>ResursBankModuleData::getGatewayPath</b> (\$subDirectory)	Absolute path to plugin structure.	By entering a subdirectory name it also returns <i>/full/path/to/subDirectory</i> . Example: <b>getGatewayPath('images')</b> Returns <b>/full/path/to/plugin/images</b>
<b>ResursBankModuleData::applyFilters</b> (\$filterName, \$value [, \$args])	Whatever that is applied.	This is actually a standard <code>apply_filters</code> , but with a helper that always adds a proper prefix to every filters applied.  Example: <b>applyFilters('checkout')</b> Will apply: <b>rbwc_checkout</b>  <b>Note: Compare to <code>applyFiltersDeprecated</code> that instead generates "resurs_bank_checkout", to comply with prior plugin releases.</b>
<b>ResursBankModuleData::getVersionByComposer</b>		Always returns version number that lies within composer.json.
<b>ResursBankModuleData::getCurrentVersion</b>		Always returns version number that lies within the initializer.
<b>ResursBankModuleData::getValidatedVersion</b>	boolean	Returns true if both composer-version and internal version is the same. If not, the plugin should warn inside wp-admin that someone forgot to update this data.
<b>ResursBankModuleData::getGenericClass</b>	Generic::class	Returns a <code>Generic::class</code> with properly configured template path pointing to plugin-directory/templates - adding templates to that directory in the extension .html makes it possible to call the class like this:  <pre>\$content .= self::getGenericClass()-&gt;getTemplate(     'plugin_information',     [         'required_drivers' =&gt; self::getSpecialString('required_drivers'),         'render' =&gt; \$renderData,     ] );</pre> The variables added to the second argument in the <code>getTemplate</code> -method could then be used like this: <pre>Required drivers: &lt;?php echo \$required_drivers ?&gt;</pre>
<b>ResursBankModuleData::getResursOption</b>	A value or null	Using <code>getResursOption</code> directly will give you the opportunity to just send in a saved option key in the method. The default parameter keys are prefixed <code>trbwc_admin</code> and is added automatically. It also checks values against "boolean-like" valued strings like <code>yes/true/no/false</code> (via <code>getTruth</code> below).  As the prefix COULD possibly change, it is recommended to always use those internals.
<b>ResursBankModuleData::getTruth</b> (\$value)	boolean or null	If you are using <code>get_options</code> directly, you can use this function to determine if the returned option should be considered a boolean or string (on the values <code>yes/true</code> and <code>false/no</code> that is the default boolean value returned from a woocommerce setup). If the returned value is null, then you can use it as a string. This is entirely handled by <code>Data::getResursOption</code> if you decide to use that instead.
<b>ResursBankModuleData::getFormFields</b> (\$section)	\$formFieldsArray	Form fields for admin based on section.

**ResursBankModule**  
`Data::setLogInternal($type, $message)`

-

To send logs to Woocommerce natural logger, `setLogInternal` can be used. The datatypes in the first argument can be found in the `Data` class and are defined as below. This first introduced an action, but was later converted to a static method.

In this plugin, the types are just used to mark severity, but uses respectively method in `WC_Logger` anyway:

Constant Type/Name	Description	WC_Logger()	Suggested when
<code>Data::LOG_DEBUG</code>	Debug information.	<code>debug()</code>	Developer notes.
<code>Data::LOG_NOTICE</code>	Notice only (default).	<code>notice()</code>	Notices to merchant and less important data.
<code>Data::LOG_CRITICAL</code>	Critical errors.	<code>critical()</code>	Critical errors, I.E. when API is down.
<code>Data::LOG_ERROR</code>	Not critical but an error.	<code>debug()</code>	
<code>Data::LOG_WARNING</code>	Not an error. Not critical. A warning.	<code>debug()</code>	A headsup to merchants/developers. I.E. when the plugin mismatches the required woocommerce version.

**Example on how to log data:**

```
Data::setLogInternal(  
    Data::LOG_NOTICE,  
    sprintf('Customer %s has an address located outside country.', $customer)  
);
```