**Custom Code Hooks Starter Guide**

The Custom Code Hooks add-on provides the ability to have your own custom PHP code executed inside of NolaPro® when users perform various actions in the system. You can simply write a function that you want to perform some additional task and then assign it to a predefined hook (also called an event or trigger point). When the hook is reached in NolaPro it will call the function(s) that you registered with the event. A trigger point might be when a customer record is modified, an order added, or payroll hours entered.

Next, we'll walk through an example, so you can see how this is done. We will create our own function that responds to the customer update trigger inside NolaPro. Let's say we have a customer relationship management (CRM) program that we want to keep up to date with any NolaPro changes.

1. Create a folder called ***custom*** inside your nolapro folder. All of the files that you want included and executed should go inside this folder. Only files directly under the custom folder will be automatically included. You can create subdirectories inside the custom folder if you'd like to create files that won't be automatically included inside NolaPro, but that may be referenced by your hook functions.

2. Create a file inside custom called ***customer_updater.php*** (it can have any filename that you'd like actually).

3. Open the customer_updater.php file in an editor.

4. Start a function definition called ***custom_customer_updater***. (We recommend the convention of beginning your custom functions with the word custom to avoid conflicts with any NolaPro functions.) Add a parameter called $params which is an associative array and will contain relevant data from the event.

```
function custom_customer_updater($params) {

}
```

5. Add the line var_dump($params) to get an idea of the data that will be passed in.

```
function custom_customer_updater($params) {
     var_dump($params);
}
```

6. At the top of the customer_updater.php page, above the custom_customer_updater function, add a line to register your new function with the customer update event. At the end of this document is a list of all of the triggers that are available for tying custom functions to. The customer update one that we'll need is called ***ar.customer.update***.

```
register_custom('ar.customer.update', 'custom_customer_updater');
```

So the formula is register_custom('event.name', 'my_function'). One event can have several

functions attached and one function can apply to several events. You could also have a function that emails you when a customer record is changed. You'd register it alongside the one above with:

```
register_custom('ar.customer.update', 'custom_email_me');
```

7. Open up NolaPro and go to Orders -> Customer Add/Update and update a customer. You should see the result of var_dump which displays the contents of the $params array. It may look something like the following:

   array(7) { ["id"]=> string(4) "1209" ["table"]=> string(8) "customer" ["customerid"]=> string(4) "1209" ["name"]=> string(16) "Noguska Test" ["event"]=> string(18) "ar.customer.update" ["userid"]=> string(2) "32" ["companyid"]=> string(1) "1" }

   For the majority of events the array elements ***id*** and ***table*** will be set. These point to the key database record that was added, updated or deleted during the event. Also with each event the userid and companyid are passed. You can access these values in your custom function by using $params['id'], $params['name'], etc.

8. Just displaying these values isn't helpful of course, so the next thing you might want to do is to retrieve the entire customer record from the NolaPro database so that you can send the info to your other system. Nolapro handles database interaction by utilizing the ADOdb database abstraction layer. You can view documentation on this here: http://phplens.com/adodb/. To access the database inside your function you'll first need to make available the connection that has already been created.

```
function custom_customer_updater($params) {
     global $conn; // Make the db connection available here
}
```

9. Write a query to get the customer record we are interested in and for now just send the data to the screen (from this point you can modify it to update your system instead).

```
function custom_customer_updater($params) {
 global $conn; // Make the db connection available here
 $sql = "select * from customer where id='$params[id]'";
 $rs = $conn->execute($sql);
 if (!$rs) {
     // Add your own error checking here
     // Some error with the query
     return;
 }
 if ($rs->EOF) {
     // Record not found
     return;
 }
 // Access the fields by using $rs->fields['id'], etc.
 // or by doing a $r = $rs->fetchrow(); and then
 // $r['id'], etc.
 echo "Customer: " . $rs->fields['companyname'] ."<br>";
```

```
 echo "Address: " . $rs->fields['address1'] ."<br>";
}
```

Here's an easy way to loop through records if you need to at some point in your function:

```
while($r = $rs->fetchrow()) {
 echo "ID: $r[id], Company Name: $r[companyname]<br>";
}
```

10. That's really all there is to getting your own code to run in response to activities within NolaPro.

**List of Available NolaPro Triggers**

NolaPro event names are produced by taking the module name, the section within the module and the action being performed and concatenating them together with a period. So if the module is AR, the section is Order and the action is Update, then when you register your function the event name to use would be ar.order.update.

| Module | Section | Action |
|---|---|---|
| admin | company | add |
| admin | company | delete |
| admin | company | update |
| admin | emailcenter | send |
| admin | user | add |
| admin | user | delete |
| admin | user | update |
| ap | bill | add |
| ap | bill | delete |
| ap | bill | update |
| ap | check | cashed |
| ap | check | uncashed |
| ap | check | void |
| ap | check | write |
| ap | commissioncheck | write |
| ap | manualcheck | write |
| ap | po | add |
| ap | po | delete |
| ap | po | passtoap |
| ap | po | receive |
| ap | po | update |

| Module | Section | Action |
|---|---|---|
| ap | preapprovedcheck | write |
| ap | vendor | add |
| ap | vendor | delete |
| ap | vendor | update |
| ap | withoutpo | receive |
| ar | bankdeposit | add |
| ar | bankdeposit | delete |
| ar | carrier | add |
| ar | carrier | delete |
| ar | carrier | update |
| ar | carriermethod | add |
| ar | carriermethod | delete |
| ar | carriermethod | update |
| ar | creditcheck | write |
| ar | customer | add |
| ar | customer | delete |
| ar | customer | update |
| ar | invoice | add |
| ar | invoice | delete |
| ar | invoice | update |
| ar | order | add |
| ar | order | delete |
| ar | order | invoice |
| ar | order | quote |
| ar | order | serviceticketpdf |
| ar | order | shipment |
| ar | order | update |
| ar | payment | add |
| ar | paymentplan | add |
| ar | paymentplan | delete |
| ar | paymentplan | post |
| ar | paymentplan | update |
| ar | rma | add |
| ar | rma | delete |
| ar | rma | update |
| ar | salescategory | add |
| ar | salescategory | delete |

| Module | Section | Action |
|---|---|---|
| ar | salescategory | update |
| ar | salesperson | add |
| ar | salesperson | delete |
| ar | salesperson | update |
| ar | salesterritory | edit |
| ar | shipto | add |
| ar | shipto | delete |
| ar | shipto | update |
| b2b | customer | update |
| b2b | order | add |
| b2b | payment | add |
| gl | account | add |
| gl | account | delete |
| gl | account | reactivate |
| gl | account | update |
| gl | bank | reconcile |
| gl | voucher | add |
| gl | voucher | delete |
| gl | voucher | post |
| gl | voucher | postreverse |
| gl | voucher | update |
| gl | year | close |
| inv | inventorylocation | add |
| inv | inventorylocation | delete |
| inv | inventorylocation | update |
| inv | item | add |
| inv | item | adjustment |
| inv | item | delete |
| inv | item | transfer |
| inv | item | update |
| inv | itemlocation | add |
| inv | itemlocation | delete |
| inv | itemlocation | update |
| inv | itemvendor | add |
| inv | itemvendor | delete |
| inv | itemvendor | update |
| pr | check | edit |

| Module | Section | Action |
|---|---|---|
| pr | check | write |
| pr | companycontribution | add |
| pr | companycontribution | delete |
| pr | companycontribution | update |
| pr | employee | add |
| pr | employee | update |
| pr | employeededuction | add |
| pr | employeededuction | delete |
| pr | employeededuction | update |
| pr | generalbenefit | add |
| pr | generalbenefit | delete |
| pr | generalbenefit | update |
| pr | generaldeduction | add |
| pr | generaldeduction | delete |
| pr | generaldeduction | update |
| pr | generalfactors | update |
| pr | hours | add |
| pr | pension | add |
| pr | pension | delete |
| pr | pension | update |
| pr | period | calculate |