Fulfillment Example

This article has been generated from the online version of the documentation and might be out of date. Please, make sure to always refer to the online version of the documentation for the up-to-date information.

Auto-generated at October 27, 2021

This example demonstrates a script that will retrieve all requests in the status pending and process them based on their type (purchase, change, cancel, suspend or resume)

```
<?php

require_once "vendor/autoload.php";

class ProductRequests extends \Connect\FulfillmentAutomation
{
    public function processRequest($request)
    {
        $this->logger->info("Processing Request: " . $request->id . " for asset: " . $request->asset->id);
        switch ($request->type) {
            case "purchase":
                //Get value of a parameter with id "email"
                $email = $request->asset->getParameterByID('email');
                if($email->value == ""){
                    throw new \Connect\Inquire(array(
                        $request->asset->params['email']->error("Email address has not been provided,
please provide one")
                    ));
                }
                //Get value for a concrete item using MPN
                $itemX = $request->asset->getItemByMPN('itemX');
                foreach ($request->asset->items as $item) {
                    if ($item->quantity > 1000000) {
                        $this->logger->info("Is Not possible to purchase product " . $item->id . " more
than 1000000 time, requested: " . $item->quantity);
                        throw new \Connect\Fail("Is Not possible to purchase product " . $item->id . "
more than 1000000 time, requested: " . $item->quantity);
                    }
                    else {
                        //Do some provisioning operation
                        //Update the parameters to store data
                        $paramsUpdate[] = new \Connect\Param('ActivationKey', 'somevalue');
                        $request->requestProcessor->fulfillment->updateParameters($request,
$paramsUpdate);
                        //Potential actions to be done with a request:
                        // Set a parameter that requires changes and move request to inquire
                        if($requiresChanges){
                            throw new \Connect\Inquire([
                                new \Connect\Param([
                                     "id" => "email",
                                     "value_error" => "Invalid email"
                                ])
```

```
                            ]);
                        }
                        //Fail request
                        throw new \Connect\Fail("Request Can't be processed");
                        //Approve a template
                        //We may use a template defined on vendor portal as activation response, this will
be what customer sees on panel
                        return new \Connect\ActivationTemplateResponse("TL-497-535-242");
                        // We may use arbitrary output to be returned as approval, this will be seen on
customer panel. Please see that output must be in markup format
                        return new \Connect\ActivationTileResponse('\n# Welcome to Fallball!\n\nYes, you
decided to have an account in our amazing service!\n\n');
                        // If we return empty, is approved with default message
                        return;
                    }
                }
            case "cancel":
                //Handle cancellation request
            case "change":
                //Handle change request
                //get added items:
                $newItems = $request->getNewItems();
                // get removed items:
                $removed = $request->getRemovedItems();
                // Get changed items, in other words the ones that quantity has been modified
                $changed = $request->getChangedItems();
            default:
                throw new \Connect\Fail("Operation not supported:".$request->type);
        }
    }

    public function processTierConfigRequest($tierConfigRequest){
        //This method allows processing Tier Requests, in same manner as simple requests.
        // Is required to be implemented since v15
    }
}

//Main Code Block

try {
    //In case Config is not passed into constructor, configuration from config.json is used
    $requests = new ProductRequests(new \Connect\Config([
        'apiKey' => 'Key_Available_in_ui',
        'apiEndpoint' => 'https://api.connect.cloud.im/public/v1',
        'products' => 'CN-631-322-641' #Optional value
    ]));
```

```
    $requests->process();
} catch (Exception $e) {
    print "Error processing requests:" . $e->getMessage();
}
```