PHP

Overview

The sample code is intended to demonstrate how you can utilise PHP to call the operations within API-NG and extract the desired output, it is very much a cut down sample and is not intended to be used in a production environment.

The code follows a simple workflow of finding the next horse racing market, displaying prices for the runners and then placing a bet with an invalid stake to trigger an error.

This documentation refers to the code available at https://github.com/betfair/API-NG-sample-code/tree/master/php.

Prerequisites

To run the sample code from the command line you must have a php5 cli installed along with the curl module enabled.

Debian linux Installation

In a Debian linux distro you can use the following commands to install the pre-requisites:

```
sudo apt-get update
sudo apt-get install php5-cli
sudo apt-get install php5-curl
```

Run the scripts

JSON-RPC

```
php -f jsonrpc.php <appkey> <sessiontoken>
```

Rescript

```
php -f rescript.php <appkey> <sessiontoken>
```

Code Snippets

Dealing with SSL in PHP

If you have errors relating to SSL certificate issues then you must do one of the following:

1) Quick fix for testing applications, should not be used in production as it may leave you exposed to man in the middle type attacks:

Add the following two lines to the sportsApingRequest function after the curl_init:

```
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, 0);
```

2) Correct fix for production applications:

You will need to make use of the CURLOPT_CAINFO option, and point it to the Betfair PEM formatted certificate (which you can export from your browser). The details of exporting the cert and using this option are beyond the scope of this document but can be found elsewhere online.

Calling API-NG with JSON-RPC protocol

Method and params values need to be change based on the required service operation. You can call batch multiple service operations together and correlate the responses with value of the id field.

```
function sportsApingRequest($appKey, $sessionToken, $operation, $params)
{
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, "https://api.betfair.com/exchange/betting/json-rpc/v1");
    curl_setopt($ch, CURLOPT_POST, 1);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($ch, CURLOPT_HTTPHEADER, array('Expect:',
        'X-Application: ' . $appKey,
        'X-Authentication: ' . $sessionToken,
        'Accept: application/json',
        'Content-Type: application/json'
    ));

    $postData =
        '[{ "jsonrpc": "2.0", "method": "SportsAPING/v1.0/' . $operation . '", "params" :' . $params . ', "id":
1}]';


    curl_setopt($ch, CURLOPT_POSTFIELDS, $postData);

    $response = json_decode(curl_exec($ch));
    curl_close($ch);

    if (isset($response[0]->error)) {
        echo 'Call to api-ng failed: ' . "\n";
        echo  'Response: ' . json_encode($response);
        exit(-1);
    } else {
        return $response;
    }

}
```

Calling API-NG with Rescript protocol

```
function sportsApingRequest($appKey, $sessionToken, $operation, $params)
{
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, "https://api.betfair.com/rest/v1/$operation/");
    curl_setopt($ch, CURLOPT_POST, 1);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($ch, CURLOPT_HTTPHEADER, array('Expect:',
        'X-Application: ' . $appKey,
        'X-Authentication: ' . $sessionToken,
        'Accept: application/json',
        'Content-Type: application/json'
    ));

    curl_setopt($ch, CURLOPT_POSTFIELDS, $params);

    $response = json_decode(curl_exec($ch));

    $http_status = curl_getinfo($ch, CURLINFO_HTTP_CODE);
    curl_close($ch);

    if ($http_status == 200) {
        return $response;
    } else {
        echo 'Call to api-ng failed: ' . "\n";
        echo  'Response: ' . json_encode($response);
        exit(-1);
    }
}
```

Calling listEventTypes to obtain and extract Horse Racing Event Type ID - JSON-RPC

```
echo extractHorseRacingEventTypeId(getAllEventTypes($appKey, $sessionToken));

function getAllEventTypes($appKey, $sessionToken)
{

    $jsonResponse = sportsApingRequest($appKey, $sessionToken, 'listEventTypes', '{"filter":{}}');

    return $jsonResponse[0]->result;
}

function extractHorseRacingEventTypeId($allEventTypes)
{
    foreach ($allEventTypes as $eventType) {
        if ($eventType->eventType->name == 'Horse Racing') {
            return $eventType->eventType->id;
        }
    }
}
```

Calling listEventTypes to obtain and extract Horse Racing Event Type ID - Rescript

```
echo extractHorseRacingEventTypeId(getAllEventTypes($appKey, $sessionToken));

function getAllEventTypes($appKey, $sessionToken)
{

    $jsonResponse = sportsApingRequest($appKey, $sessionToken, 'listEventTypes', '{"filter":{}}');

    return $jsonResponse;
}

function extractHorseRacingEventTypeId($allEventTypes)
{
    foreach ($allEventTypes as $eventType) {
        if ($eventType->eventType->name == 'Horse Racing') {
            return $eventType->eventType->id;
        }
    }
}
```

Calling listMarketCatalogue to get next UK horse racing market and print the marketId and runners - JSON-RPC

```
printMarketIdAndRunners(getNextUkHorseRacingMarket($appKey, $sessionToken, $horseRacingEventTypeId);

function getNextUkHorseRacingMarket($appKey, $sessionToken, $horseRacingEventTypeId)
{

    $params = '{"filter":{"eventTypeIds":["' . $horseRacingEventTypeId . '"],
              "marketCountries":["GB"],
              "marketTypeCodes":["WIN"],
              "marketStartTime":{"from":"' . date('c') . '"}},
              "sort":"FIRST_TO_START",
              "maxResults":"1",
              "marketProjection":["RUNNER_DESCRIPTION"]}';

    $jsonResponse = sportsApingRequest($appKey, $sessionToken, 'listMarketCatalogue', $params);

    return $jsonResponse[0]->result[0];
}

function printMarketIdAndRunners($nextHorseRacingMarket)
{

    echo "MarketId: " . $nextHorseRacingMarket->marketId . "\n";
    echo "MarketName: " . $nextHorseRacingMarket->marketName . "\n\n";

    foreach ($nextHorseRacingMarket->runners as $runner) {
        echo "SelectionId: " . $runner->selectionId . " RunnerName: " . $runner->runnerName . "\n";
    }

}
```

Calling listMarketCatalogue to get next UK horse racing market and print the marketId and runners - Rescript

```
printMarketIdAndRunners(getNextUkHorseRacingMarket($appKey, $sessionToken, $horseRacingEventTypeId);

function getNextUkHorseRacingMarket($appKey, $sessionToken, $horseRacingEventTypeId)
{

    $params = '{"filter":{"eventTypeIds":["' . $horseRacingEventTypeId . '"],
              "marketCountries":["GB"],
              "marketTypeCodes":["WIN"],
              "marketStartTime":{"from":"' . date('c') . '"}},
              "sort":"FIRST_TO_START",
              "maxResults":"1",
              "marketProjection":["RUNNER_DESCRIPTION"]}';

    $jsonResponse = sportsApingRequest($appKey, $sessionToken, 'listMarketCatalogue', $params);

    return $jsonResponse[0];
}

function printMarketIdAndRunners($nextHorseRacingMarket)
{

    echo "MarketId: " . $nextHorseRacingMarket->marketId . "\n";
    echo "MarketName: " . $nextHorseRacingMarket->marketName . "\n\n";

    foreach ($nextHorseRacingMarket->runners as $runner) {
        echo "SelectionId: " . $runner->selectionId . " RunnerName: " . $runner->runnerName . "\n";
    }

}
```

Calling listMarketBook to get volatile price data and print the marketId and runners with best available prices - JSON-RPC

```
printMarketIdAndRunnersAndPrices($nextHorseRacingMarket, getMarketBook($appKey, $sessionToken, $marketId));

function getMarketBook($appKey, $sessionToken, $marketId)
{
    $params = '{"marketIds":["' . $marketId . '"], "priceProjection":{"priceData":["EX_BEST_OFFERS"]}}';

    $jsonResponse = sportsApingRequest($appKey, $sessionToken, 'listMarketBook', $params);

    return $jsonResponse[0]->result[0];
}

function printMarketIdRunnersAndPrices($nextHorseRacingMarket, $marketBook)
{

    function printAvailablePrices($selectionId, $marketBook)
    {

        // Get selection
        foreach ($marketBook->runners as $runner)
            if ($runner->selectionId == $selectionId) break;

        echo "\nAvailable to Back: \n";
        foreach ($runner->ex->availableToBack as $availableToBack)
            echo $availableToBack->size . "@" . $availableToBack->price . " | ";

        echo "\n\nAvailable to Lay: \n";
        foreach ($runner->ex->availableToLay as $availableToLay)
            echo $availableToLay->size . "@" . $availableToLay->price . " | ";

    }


    echo "MarketId: " . $nextHorseRacingMarket->marketId . "\n";
    echo "MarketName: " . $nextHorseRacingMarket->marketName;

    foreach ($nextHorseRacingMarket->runners as $runner) {
        echo "\n\n\n================================================================================\n";

        echo "SelectionId: " . $runner->selectionId . " RunnerName: " . $runner->runnerName . "\n";
        echo printAvailablePrices($runner->selectionId, $marketBook);
    }
}
```

Calling listMarketBook to get volatile price data and print the marketId and runners with best available prices - Rescript

```
printMarketIdAndRunnersAndPrices($nextHorseRacingMarket, getMarketBook($appKey, $sessionToken, $marketId));

function getMarketBook($appKey, $sessionToken, $marketId)
{
    $params = '{"marketIds":["' . $marketId . '"], "priceProjection":{"priceData":["EX_BEST_OFFERS"]}}';

    $jsonResponse = sportsApingRequest($appKey, $sessionToken, 'listMarketBook', $params);

    return $jsonResponse[0];
}

function printMarketIdRunnersAndPrices($nextHorseRacingMarket, $marketBook)
{

    function printAvailablePrices($selectionId, $marketBook)
    {

        // Get selection
        foreach ($marketBook->runners as $runner)
            if ($runner->selectionId == $selectionId) break;

        echo "\nAvailable to Back: \n";
        foreach ($runner->ex->availableToBack as $availableToBack)
            echo $availableToBack->size . "@" . $availableToBack->price . " | ";

        echo "\n\nAvailable to Lay: \n";
        foreach ($runner->ex->availableToLay as $availableToLay)
            echo $availableToLay->size . "@" . $availableToLay->price . " | ";

    }


    echo "MarketId: " . $nextHorseRacingMarket->marketId . "\n";
    echo "MarketName: " . $nextHorseRacingMarket->marketName;

    foreach ($nextHorseRacingMarket->runners as $runner) {
        echo "\n\n\n================================================================================\n";

        echo "SelectionId: " . $runner->selectionId . " RunnerName: " . $runner->runnerName . "\n";
        echo printAvailablePrices($runner->selectionId, $marketBook);
    }
}
```

Place bet on first runner of the market. Stake is below minimum to prevent actual bet placement - JSON-RPC

```
printBetResult(placeBet($appKey, $sessionToken, $marketId, $selectionId));

function placeBet($appKey, $sessionToken, $marketId, $selectionId)
{

    $params = '{"marketId":"' . $marketId . '",
                "instructions":
                     [{"selectionId":"' . $selectionId . '",
                       "handicap":"0",
                       "side":"BACK",
                       "orderType":
                       "LIMIT",
                       "limitOrder":{"size":"1",
                                     "price":"1000",
                                     "persistenceType":"LAPSE"}
                       }], "customerRef":"fsdf"}';

    $jsonResponse = sportsApingRequest($appKey, $sessionToken, 'placeOrders', $params);

    return $jsonResponse[0]->result;

}

function printBetResult($betResult)
{
    echo "Status: " . $betResult->status;

    if ($betResult->status == 'FAILURE') {
        echo "\nErrorCode: " . $betResult->errorCode;
        echo "\n\nInstruction Status: " . $betResult->instructionReports[0]->status;
        echo "\nInstruction ErrorCode: " . $betResult->instructionReports[0]->errorCode;
    } else
        echo "Warning!!! Bet placement succeeded !!!";
}
```

Place bet on first runner of the market. Stake is below minimum to prevent actual bet placement - Rescript

```
printBetResult(placeBet($appKey, $sessionToken, $marketId, $selectionId));

function placeBet($appKey, $sessionToken, $marketId, $selectionId)
{

    $params = '{"marketId":"' . $marketId . '",
                "instructions":
                     [{"selectionId":"' . $selectionId . '",
                       "handicap":"0",
                       "side":"BACK",
                       "orderType":
                       "LIMIT",
                       "limitOrder":{"size":"1",
                                     "price":"1000",
                                     "persistenceType":"LAPSE"}
                       }], "customerRef":"fsdf"}';

    $jsonResponse = sportsApingRequest($appKey, $sessionToken, 'placeOrders', $params);

    return $jsonResponse;

}

function printBetResult($betResult)
{
    echo "Status: " . $betResult->status;

    if ($betResult->status == 'FAILURE') {
        echo "\nErrorCode: " . $betResult->errorCode;
        echo "\n\nInstruction Status: " . $betResult->instructionReports[0]->status;
        echo "\nInstruction ErrorCode: " . $betResult->instructionReports[0]->errorCode;
    } else
        echo "Warning!!! Bet placement succeeded !!!";
}
```