

Python

- [Prerequisites:](#)
- [A note on Python3:](#)
- [Installation:](#)
- [Run the scripts](#)
- [Calling API-NG with JSON-RPC protocol](#)
- [Calling API-NG with Rescript protocol](#)
- [Get next available horse racing market and runner information using listMarketCatalogue](#)
- [Get available price for the next horse racing market using listMarketBook](#)
- [Placing a bet on first active runner from next horse racing market using placeOrders](#)

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

Prerequisites:

1. python v 2.7.2 - <http://www.python.org/getit/releases/2.7.2/>
2. urllib2 python module - <http://docs.python.org/2/library/urllib2.html>
3. json python module - <http://docs.python.org/2/library/json.html>
4. datetime python module - <http://docs.python.org/2/library/datetime.html>
5. sys python module - <http://docs.python.org/2/library/sys.html>

A note on Python3:

We have added a python3 version of the json-rpc script, which is in the [python subdirectory](#) of the github sample code repo named ApiNgDemoJsonRpc-python3.py. This functions exactly the same way as the python 2.7X sample, but with compatibility tweaks for Python 3. The documentation below reflects the python 2.7X code, but the

Installation:

You only need to clone or download the repository linked to above. If you do not have a valid Python 2.7.X installation already then please follow the download and installation instructions from the [python wiki](#).

Run the scripts

Change to the directory where you cloned the repository to and run the sample of your choice as follows:

JSON-RPC

```
python ApiNgDemoJsonRpc.py <appkey> <sessiontoken>
```

Rescript

```
python ApiNgDemoRescript.py <appkey> <sessiontoken>
```



Note: If the command line arguments for application key and session token are not provided then the script will prompt for application key and session token

Calling API-NG with JSON-RPC protocol

Method and param values need to be changed based on the required service operation. You can execute multiple service operation together with a single call using batch json-rpc call where you can correlate the responses with value of the id.

```

URL = url = "https://api.betfair.com/exchange/betting/json-rpc/v1"
jsonrpc_req = '{"jsonrpc": "2.0", "method": "SportsAPING/v1.0/listEventTypes", "params": {"filter":{ }}, "id": 1}'
headers = {'X-Application': appKey, 'X-Authentication': sessionToken, 'content-type': 'application/json'}

def callAping(jsonrpc_req):
    try:
        req = urllib2.Request(url, jsonrpc_req, headers)
        response = urllib2.urlopen(req)
        jsonResponse = response.read()
        return jsonResponse
    except urllib2.URLError:
        print 'Oops no service available at ' + str(url)
        exit()
    except urllib2.HTTPError:
        print 'Oops not a valid operation from the service ' + str(url)
        exit()

```

Calling API-NG with Rescript protocol

```

url = 'https://api.betfair.com/rest/v1.0/${operationName}/'
headers = {'X-Application': appKey, 'X-Authentication': sessionToken, 'content-type': 'application/json',
'accept': 'application/json'}
request = '{"filter":{"eventTypeIds":["7"],"marketCountries":["GB"],"marketStartTime":{"from":"2013-05-21T00:00:00Z"},"sort":"FIRST_TO_START","maxResults":"1","marketProjection":["RUNNER_METADATA"]}'

def callAping(url, request):
    try:
        req = urllib2.Request(url, request, headers)
        response = urllib2.urlopen(req)
        jsonResponse = response.read()
        return jsonResponse

    except urllib2.URLError:
        print 'Oops there is some issue with the request'
        exit()
    except urllib2.HTTPError:
        print 'Oops there is some issue with the request' + urllib2.HTTPError.getcode()
        exit()

```

Get next available horse racing market and runner information using listMarketCatalogue

JSON-RPC

```

def getMarketCatalogueForNextGBWin(eventTypeID):
    if (eventTypeID is not None):
        print 'Calling listMarketCatalogue Operation to get MarketID and selectionId'
        now = datetime.datetime.now().strftime('%Y-%m-%dT%H:%M:%SZ')
        market_catalogue_req = '{"jsonrpc": "2.0", "method": "SportsAPING/v1.0/listMarketCatalogue", "params":
{"filter":{"eventTypeId":["' + eventTypeID + '"],"marketCountries":["GB"],"marketTypeCodes":["WIN"],'\
'"marketStartTime":{"from":"' + now + '"},"sort":"FIRST_TO_START","maxResults":"1","marketProjection":
["RUNNER_METADATA"]}, "id": 1}'
        """
        print market_catalogue_req
        """
        market_catalogue_response = callAping(market_catalogue_req)
        """
        print market_catalogue_response
        """
        market_catalogue_loads = json.loads(market_catalogue_response)
        try:
            market_catalogue_results = market_catalogue_loads['result']
            return market_catalogue_results
        except:
            print 'Exception from API-NG' + str(market_catalogue_results['error'])
            exit()

def getMarketId(marketCatalogueResult):
    if( marketCatalogueResult is not None):
        for market in marketCatalogueResult:
            return market['marketId']

def getSelectionId(marketCatalogueResult):
    if(marketCatalogueResult is not None):
        for market in marketCatalogueResult:
            return market['runners'][0]['selectionId']

marketCatalogueResult = getMarketCatalogueForNextGBWin(horseRacingEventTypeID)
marketid = getMarketId(marketCatalogueResult)
runnerId = getSelectionId(marketCatalogueResult)

```

Rescript

```

def getMarketCatalogue(eventTypeID):
    if(eventTypeID is not None):
        print 'Calling listMarketCatalogue Operation to get MarketID and selectionId'
        endPoint = 'https://api.betfair.com/rest/v1.0/listMarketCatalogue/'
        now = datetime.datetime.now().strftime('%Y-%m-%dT%H:%M:%SZ')
        market_catalogue_req = '{"filter":{"eventTypes":["' + eventTypeID + '"],"marketCountries":["GB"],"
marketStartTime":{"from":"' + now + '"},"sort":"FIRST_TO_START","maxResults":"1","marketProjection":
["RUNNER_METADATA"]}'

        market_catalogue_response = callApi(endPoint, market_catalogue_req)

        market_catalogue_loads = json.loads(market_catalogue_response)
        return market_catalogue_loads

def getMarketId(marketCatalogueResult):
    if(marketCatalogueResult is not None):
        for market in marketCatalogueResult:
            return market['marketId']

def getSelectionId(marketCatalogueResult):
    if(marketCatalogueResult is not None):
        for market in marketCatalogueResult:
            return market['runners'][0]['selectionId']

marketCatalogueResult = getMarketCatalogue(horseRacingEventTypeID)
marketid = getMarketId(marketCatalogueResult)
runnerId = getSelectionId(marketCatalogueResult)

```

Get available price for the next horse racing market using listMarketBook

JSON-RPC

```

def getMarketBookBestOffers(marketId):
    print 'Calling listMarketBook to read prices for the Market with ID : ' + marketId
    market_book_req = '{"jsonrpc": "2.0", "method": "SportsAPING/v1.0/listMarketBook", "params": {"marketIds":
[" + marketId + "], "priceProjection": {"priceData": ["EX_BEST_OFFERS"]}}, "id": 1}'
    """
    print market_book_req
    """
    market_book_response = callAping(market_book_req)
    """
    print market_book_response
    """
    market_book_loads = json.loads(market_book_response)
    try:
        market_book_result = market_book_loads['result']
        return market_book_result
    except:
        print 'Exception from API-NG' + str(market_book_result['error'])
        exit()

def printPriceInfo(market_book_result):
    if(market_book_result is not None):
        print 'Please find Best three available prices for the runners'
        for marketBook in market_book_result:
            runners = marketBook['runners']
            for runner in runners:
                print 'Selection id is ' + str(runner['selectionId'])
                if (runner['status'] == 'ACTIVE'):
                    print 'Available to back price : ' + str(runner['ex']['availableToBack'])
                    print 'Available to lay price : ' + str(runner['ex']['availableToLay'])
                else:
                    print 'This runner is not active'

market_book_result = getMarketBookBestOffers(marketid)
printPriceInfo(market_book_result)

```

Rescript

```

def getMarketBook(marketId):
    if( marketId is not None):
        print 'Calling listMarketBook to read prices for the Market with ID :' + marketId
        market_book_req = '{"marketIds":["' + marketId + '"],"priceProjection":{"priceData":
["EX_BEST_OFFERS"]}}'

        endPoint = 'https://api.betfair.com/rest/v1.0/listMarketBook/'

        market_book_response = callAping(endPoint, market_book_req)

        market_book_loads = json.loads(market_book_response)
        return market_book_loads

def printPriceInfo(market_book_result):
    print 'Please find Best three available prices for the runners'
    for marketBook in market_book_result:
        try:
            runners = marketBook['runners']
            for runner in runners:
                print 'Selection id is ' + str(runner['selectionId'])
                if (runner['status'] == 'ACTIVE'):
                    print 'Available to back price :' + str(runner['ex']['availableToBack'])
                    print 'Available to lay price :' + str(runner['ex']['availableToLay'])
                else:
                    print 'This runner is not active'
        except:
            print 'No runners available for this market'

market_book_result = getMarketBook(marketid)
printPriceInfo(market_book_result)

```

Placing a bet on first active runner from next horse racing market using placeOrders

JSON-RPC

```

def placeFailingBet(marketId, selectionId):
    if( marketId is not None and selectionId is not None):
        print 'Calling placeOrder for marketId :' + marketId + ' with selection id :' + str(selectionId)
        place_order_Req = '{"jsonrpc": "2.0", "method": "SportsAPING/v1.0/placeOrders", "params":
{"marketId":"' + marketId + '","instructions":\

'[{ "selectionId":"' + str(
        selectionId) + '","handicap":0,"side":"BACK","orderType":"LIMIT","limitOrder":{"size":"0.01",
price":1.50,"persistenceType":"LAPSE"}}, {"customerRef":"test12121212121"}, {"id": 1}]'
        """
        print place_order_Req
        """
        place_order_Response = callAping(place_order_Req)
        place_order_load = json.loads(place_order_Response)
        try:
            place_order_result = place_order_load['result']
            print 'Place order status is ' + place_order_result['status']
            """
            print 'Place order error status is ' + place_order_result['errorCode']
            """
            print 'Reason for Place order failure is ' + place_order_result['instructionReports'][0]
['errorCode']
        except:
            print 'Exception from API-NG' + str(place_order_result['error'])

placeBet(marketid, runnerId)

```

Rescript

```

def placeBet(marketId, selectionId):
    if( marketId is not None and selectionId is not None):
        print 'Calling placeOrder for marketId :' + marketId + ' with selection id :' + str(selectionId)
        place_order_Req = '{"marketId":"' + marketId + '", "instructions":'\
            '{"selectionId":"' + str(
                selectionId) + '", "handicap": "0", "side": "BACK", "orderType": "LIMIT", "limitOrder": {"size": "1.01", "
price": "1.50", "persistenceType": "LAPSE"} }', "customerRef": "test12121212121"}'
        endPoint = 'https://api.betfair.com/rest/v1.0/placeOrders/'

        place_order_Response = callApi(endPoint, place_order_Req)
        place_order_load = json.loads(place_order_Response)
        print 'Place order status is ' + place_order_load['status']

        print 'Reason for Place order failure is ' + place_order_load['instructionReports'][0]['errorCode']

placeBet(marketid, runnerId)

```

- [Prerequisites:](#)
- [A note on Python3:](#)
- [Installation:](#)
- [Run the scripts](#)
- [Calling API-NG with JSON-RPC protocol](#)
- [Calling API-NG with Rescript protocol](#)
- [Get next available horse racing market and runner information using listMarketCatalogue](#)
- [Get available price for the next horse racing market using listMarketBook](#)
- [Placing a bet on first active runner from next horse racing market using placeOrders](#)