

thinknode™ Examples

These examples provide a starting point for issuing http connections and requests to the dosimetry app on the thinknode™ framework. They are provided as is, and are written in python and c++. Any further dependencies are listed along with the provided scripts.

C++

A simple C++ project that handles posting immutable objects and calculation requests to thinknode™ framework. The *main()* function toggles on which task to perform. Below are the defined functions of the project as well as a link to download the file in its entirety.

Dependencies:

- [libcurl](#)
- [jsoncpp](#)

thinknode.cpp

- [thinknode.cpp](#)

```
// Copyright (c) 2015 .decimal, Inc. All rights reserved.
// Desc:      Worker to perform tasks on thinknode framework

#include "stdafx.h"
#include <stdio.h>
#include <string.h>
#include <iostream>
#include <fstream>
#include <sstream>
#include <json/json.h>

#define CURL_STATICLIB
#include <curl/curl.h>

using namespace std;

// API configuration
string basic_user = "ABCDEFGHIJKLMNOPQRSTUVWXYZ123456"; // Base64 encoded
username:password
string api_url = "https://api.thinknode.com/v1.0"; // thinknode url
string app_name = "Dosimetry"; // app name
string app_version = "1.0.0"; // app version
```

```
// Curl get request call back
static size_t WriteCallback(void *contents, size_t size, size_t nmemb, void
*userp)
...

// Select a specific json tag into string
string get_json_value(string json, string id, int num = 0)
...

// Curllib get http request
string do_curl_get(string auth, string url)
...

// Curllib post http request
string do_curl_post(string auth, string json, string url)
...

// Handles http request to get the user ID from the basic_user
string get_user_token()
...

// Handles http request for realm id
string get_realm_id(string token)
...

// Handles http request for context id
string get_context_id(std::map<string, string> config)
...

// API Authentication
std::map<string, string> authenticate()
...

// Grab and post the specified calc request
void post_calc_request()
...

// Grab and post sepcified object to the ISS
void post_immutable_object()
...

int main(void)
...
```

C++: Immutable Storage

Posting an object to Immutable Storage

The below example is a function in the [thinknode.cpp](#) class to post an immutable object to the dosimetry app on the thinknode™ framework. This example can be used for any immutable storage post using any datatype by replacing the json iss file.

Dependencies:

- [compute_aperture_creation_params.json](#)

```
void post_immutable_object()
{
    // Immutable info
    string path = "C:\\\\"; // Path of folder
    json file is located in
    string sjon_iss_file = "aperture_creation_params.json"; // local json
    object file
    string obj_name = "aperture_creation_params"; // app named_type

    std::map<string, string> iam = authenticate();

    // Read local immutable json file
    std::ifstream json_file((path + sjon_iss_file).c_str());
    string str((std::istreambuf_iterator<char>(json_file)),
    std::istreambuf_iterator<char>());

    // Post object
    std::cout << "Posting Object to ISS..." << std::endl;
    string authentication_string = "Authorization : Bearer " +
iam["user_token"];
    string res = do_curl_post(
        authentication_string,
        str,
        api_url + "/iss/named/" + app_name + "/" + obj_name + "?context=" +
iam["context_id"]);
    std::cout << "Immuntable ID: " << res << std::endl;
}
```

Returns:

1. The ID (in json) of the object stored in Immutable Storage.

C++: Calculation Request

The below example is a function in the [thinknode.cpp](#) class to post a calculation request to dosimetry. This example can be used for any calculation request using any datatype by replacing the calculation request json file. This request will post a calculation request, check the status using long polling with a specified timeout, and return the calculation result.

Dependencies:

- [compute_aperture.json](#)

```
void post_calc_request()
{
    // Request info
    string path = "C:\\\\"; // Path of folder json
    file is located in
    string sjson_iss_file = "compute_aperture.json"; // local json calc
    request

    std::map<string, string> iam = authenticate();

    std::ifstream json_file((path + sjson_iss_file).c_str());
    string str((std::istreambuf_iterator<char>(json_file),
    std::istreambuf_iterator<char>()));

    string authentication_string = "Authorization : Bearer " +
    iam["user_token"];
    // Get calculation id
    std::cout << "Sending Calculation..." << std::endl;
    string calculation_id = get_json_value(
        do_curl_post(authentication_string,
        str, api_url + "/calc/?context=" + iam["context_id"],
        "id");

    // Get calculation Status - using long polling
    std::cout << "Checking Calculation Status..." << std::endl;
    string calculation_status = get_json_value(
        do_curl_get(authentication_string,
        api_url + "/calc/" + calculation_id +
        "/status/?status=completed&progress=1&timeout=5"),
        "type");
    if (calculation_status.find("failed") != string::npos)
    {
        std::cout << "Server Responded: " << calculation_status << std::endl;
        return;
    }
}
```

```
// Get calculation Result
std::cout << "Fetching Calculation Result..." << std::endl;
string calculation_result = do_curl_get(
    authentication_string,
    api_url + "/calc/" + calculation_id + "/result/?context=" +
iam["context_id"]);

std::cout << "Calculation Result: " << calculation_result << std::endl;
}
```

Returns:

1. The calculation result (in json) of the API function called.

Python

This python example is a simple python project that handles posting immutable objects and calculation requests to thinknode™ framework. The *post_calc_request* and *post_immutable_object* are separate project files which handle the tasks to perform. Each calls the [thinknode.cfg](#) file to configure the api settings and the [thinknode.py](#) file handles all the http requests. Below are the defined functions of the file as well as a link to download the file in its entirety.

Dependencies:

- [Python Requests](#)
- [thinknode.py](#)
- [thinknode.cfg](#)

thinknode.cfg

Configuration file for connecting to thinknode™. *basic_user* being a base64 encoded username and password and *api_url* being the connection string to the thinknode™ framework.

[thinknode.cfg](#)

```
{
  "basic_user": "<Base64 encoded username:password>",
  "api_url": "https://api.thinknode.com/v1.0"
}
```

thinknode.py

The [thinknode.py](#) file performs worker tasks to the thinknode™ framework. Below are the defined functions of the file as well as a link to download the file in its entirety.

- [thinknode.py](#)

```
# Copyright (c) 2015 .decimal, Inc. All rights reserved.
# Desc:      Worker to perform tasks on thinknode framework

# Check that the response returned a successful code
def assert_success(res):
    ...

# Read the thinknode config file
def read_config(path):
    ...

# Authenticate with thinknode and store necessary ids
# param config: connection settings (url and unique basic user
authentication)
def authenticate(config, app_name, app_version):
    ...

# Send calculation request to thinknode api
def do_calculation(config, json_data):
    ...

# Post immutable object to ISS
def post_immutable(config, json_data, app_name, obj_name):
    ...
```

Python: Immutable Storage

Posting an object to Immutable Storage

Basic example using the [thinknode.py](#) class to post an immutable object to the dosimetry app on the thinknode™ framework. This example can be used for any immutable storage post using any datatype by replacing the json iss file.

Dependencies:

- [thinknode.py](#)
- [compute_aperture_creation_params.json](#)

[post_immutable_object.py](#)

```
# Copyright (c) 2015 .decimal, Inc. All rights reserved.
# Desc:      Post an immutable json object to the thinknode framework

import thinknode
import json

json_iss_file = "aperture_creation_params.json" #local json object file
app_name = "dosimetry" #app name for thinknode
and local folder
app_version = "1.0.0" #app version for thinknode
url
obj_name = "aperture_creation_params" #app named_type

# Get unique user_id and api_url
config = thinknode.read_config('thinknode.cfg')
# Get IAM ids
iam = thinknode.authenticate(config, app_name, app_version)

# App object to post to iss
with open(app_name + '/objects/' + json_iss_file) as data_file:
    json_data = json.load(data_file)

# Post immutable object to ISS
res = thinknode.post_immutable(iam, json_data, app_name, obj_name)
print("Immutable id: " + res.text)
```

Returns:

1. The ID (in json) of the object stored in Immutable Storage.

Python: Calculation Request

Basic example using the [thinknode.py](#) class to post a calculation request to dosimetry. This example can be used for any calculation request using any datatype by replacing the calculation request json file. This request will post a calculation request, check the status using long polling with a specified timeout, and return the calculation result.

Dependencies:

- [thinknode.py](#)
- [compute_aperture.json](#)

[post_calc_request.py](#)

```
# Copyright (c) 2015 .decimal, Inc. All rights reserved.
# Desc:      Post a json calculation request to the thinknode framework

import thinknode
import json

json_calc_file = "compute_aperture.json"#local json calc request
app_name = "dosimetry"                #app name for thinknode and local
folder
app_version = "1.0.0"                 #app version for thinknode url

# Get unique user_id and api_url
config = thinknode.read_config('thinknode.cfg')
# Get IAM ids
iam = thinknode.authenticate(config, app_name, app_version)

# App calculation request
with open(app_name + '/' + json_calc_file) as data_file:
    json_data = json.load(data_file)

# Send calc request and wait for answer
res = thinknode.do_calculation(iam, json_data)
print("Calculation Result: " + res.text)
```

Returns:

1. The calculation result (in json) of the API function called.

Node.js

The following section contains examples using node.js and (if applicable) the specified modules. These examples are for a high level approach to encoding and decoding the blob data that is part of the Dosimetry App calculation request.

Base64 Blob Format

The blob returned by a calculation request is formatted as such:

```
// value_type enum definitions
// Nil = 0;
// Boolean = 1;
// Number = 2;
// String = 3;
// Blob = 4;
```

```
// List = 5;
// Record = 6;

// For value_types nil, boolean, number, string
// <uint32 value_type enum (4 bytes)><data>

// For value_types blob, list, record
// <uint31 value_type enum (4 bytes)><size of each data (8 bytes)><data>
```

Node: Decrypt Base64 Blob Data

The following example shows , using node.js, how to decode the base64 encoded data returned by a calculation request.

```
// The below base64string is a blob array with the values [ -25, -25, 25, -25,
25, 25, -25, 25 ]
var b64string = "CF2Hl0z_eJxjYWBgcGBABpYH0GgHHHwMcQDM7AYN";

// The below base64string is a number set to the value 25.1
//var b64string = "NztmHgz_eJxjYmBgmDUTCCQtHQATFwOT";

var buf = new Buffer(b64string, 'base64');

var zlib = require('zlib');

function read_base_255_number(buf, offset) {
  var n = 0;
  var s = 0;
  while (offset < buf.length) {
    var digit = buf[offset];
    var value = buf.readUInt8(offset);
    offset++;
    s++;
    if (digit.toString(16) === 'ff') {
      break;
    }
    n = n * 255;
    n += value;
  }
  return [s, n];
}

var size = read_base_255_number(buf, 4);

zlib.unzip(buf.slice(4 + size[0]), function (err, data) {
  if (err) {
```

```
        throw err
    }
    var value_type = data.readUInt32LE(0);

    // Number
    if (value_type === 2) {
        console.log("DOUBLE", data.readDoubleLE(4));
    }
    // Blob
    else if (value_type === 4) {
        // Read size here
        var values = [];
        for (var i = 12; i < data.length; i+=8) {
            values.push(data.readDoubleLE(i));
        }
        console.log(values); // Outputs: [ -25, -25, 25, -25, 25, 25, -25, 25
    ]
    }
});
```

From:
<http://apps.dotdecimal.com/> - **decimal App Documentation**

Permanent link:
<http://apps.dotdecimal.com/doku.php?id=userguide:thinknode&rev=1424272230>

Last update: **2021/07/29 18:19**

