Code Examples

Developers please use https://api-stage.bimplus.net/v2 (stage version of Bimplus API) and http://portal-stage.bimplus.net/ (stage version of Shop /Portal) for testing purposes. The production version of the portal is located under https://portal.bimplus.net/ and the base url of the API is different for the production version.

Postman examples

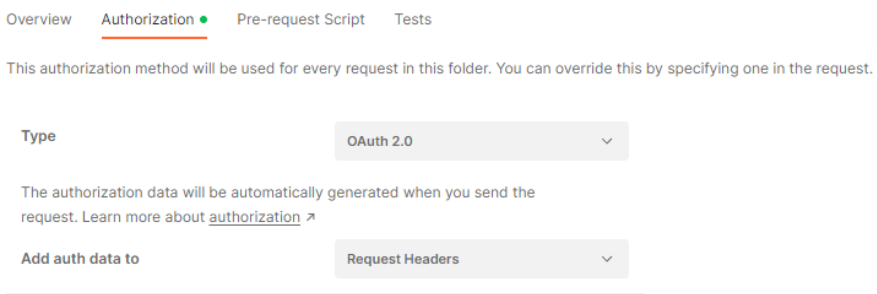
1. **Please install Postman and import attached postman collections**
   BimPlus.postman_collection.json

   postman_globals.json
   Production.postman_environment.json
   Stage.postman_environment.json

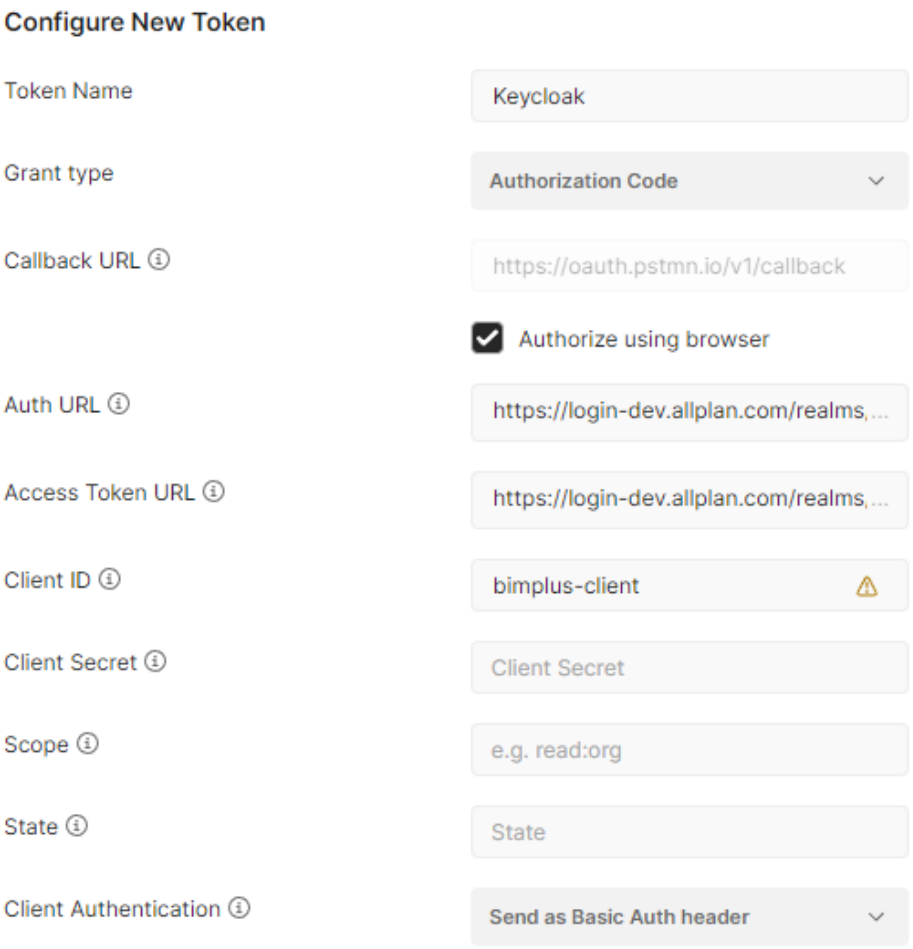
   if you don't know how it works show this video

2. **Set Postman authenication direct to OIDC server, avoiding authentication request on start.**
   Postman supports OpenID authentication and can receive tokens directly from OIDC server.
   Open request group (or even down to single request), select Authorization page, set authorisation type 'OAuth 2.0' and configure OIDC BIm+ server.

   Then create Configure new token. Take server names from answer on discovery URL.
   Preferrably use variables to specify different servers for different environments.



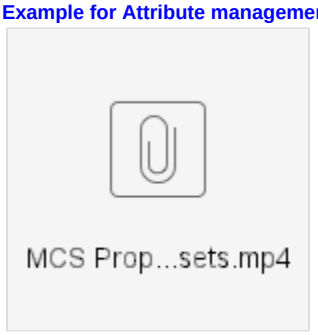
Click "Get New Access Token". Token should be loaded in token field.
Set "Auto-refresh Token" above to ON.

3. **Example for Ifc Import, creation of new revisions and final revision compare**

4. **Example for creating Modeldata**
by using Geometry templates controller, objects controller and geometry types
install additional postman collection

Some mp4, which shows usage of geometry templates, csg and mesh data

5. **Example for Attribute management and PSet definitions**



6. **Example for creating Alignments with REST API**

7. **Example for creating StructuralAnalysis Modeldata with REST API**
   you can find the same example as .NET Implementation using BIF assemblies under
   https://github.com/Bimplus/bimplus-dotnet-demo

y

C# demo projects

BimPlusCube.zip --> It explains how to create project data (nodes, geometry etc)

BimPlusDemo.zip --> It explains how to create project data (nodes, geometry etc)It explains how to authenticate, get your teams, get your project, how to create a project, how to get the project_id of your created project, create a model under the same project and upload an ifc file.

PHP Rest Client

BimplusRestClient.php (PHP REST client)

```
// Example 1: Hello function (no accessToken needed)
// ------------------------------------------------------------------------
$url = 'https://api-stage.bimplus.net/v2/hello';
$verb = 'GET';

// Create object
$request = new BimplusRestClient(
    $url,
    $verb
);

// Execute
$request->execute();
$response = $request->getResponse();

echo '<pre>';
print_r(json_decode($response));
echo '</pre>';




// Example 2: Get user data
// ------------------------------------------------------------------------
$userId = 'xxxxxxxxxxxxxxxxxxxxxxxxxxx'; // Bimplus userID
$url = 'https://api-stage.bimplus.net/v2/users/' . $userId;
$verb = 'GET';
$accessToken = 'xxxxxxxxxxxxxxxxxxxxxxxxxxx'; // accessToken

// Create object
$request = new BimplusRestClient(
    $url,
    $verb,
```

```
    $accessToken
);

// Execute
$request->execute();
$response = $request->getResponse();

echo '<pre>';
print_r(json_decode($response));
echo '</pre>';



// Example 3: Update user data
// ------------------------------------------------------------------------
$userId = 'xxxxxxxxxxxxxxxxxxxxxxxxxxx'; // Bimplus userID
$url = 'https://api-stage.bimplus.net/v2/users/' . $userId;
$verb = 'PUT';
$accessToken = 'xxxxxxxxxxxxxxxxxxxxxxxxxxx'; // accessToken
$requestBody = array(
    'email' => 'new.email@allplan.com'
);

// Create object
$request = new BimplusRestClient(
    $url,
    $verb,
    $accessToken,
    $requestBody
);

// Execute
$request->execute();
$response = $request->getResponse();

echo '<pre>';
print_r(json_decode($response));
echo '</pre>';
```

A Simple Javascript example

```
jQuery(document).ready(function() {
	jQuery("#datagrid").jqGrid({
		datatype : 'json',
		type : "GET",
		ajaxGridOptions : {
			contentType : "application/json"
		},
		loadBeforeSend : function(xhr) {
			xhr.setRequestHeader('Authorization', 'BimPlus 199c55110e2044b88e21a0c1cbb02fe3')
		},
		url : 'http://api-dev.bimplus.net/v2/teams',
		colNames : ['ID', 'Name', 'Slug', 'Status'],
		colModel : [{
			name : 'id',
			width : 200,
			align : "center",
			sortable : true
		}, {
			name : 'name',
			width : 200,
			align : "center",
			sortable : true
		}, {
			name : 'slug',
			width : 200,
			align : "center",
			sortable : true
		}, {
			name : 'status',
			width : 200,
			align : "center",
			sortable : true
		}],
		jsonReader : {
			repeatitems : false,
			root : function(obj) {
				return obj;
			},
			page : function(obj) {
				return 1;
			},
			total : function(obj) {
				return 1;
			},
			records : function(obj) {
				return obj.length;
			}
		},
		rowNum : 10,
		rowList : [10, 20, 30],
		pager : '#pager10',
		viewrecords : true,
		caption : "Bimplus Team Details",
	}).navGrid('#navGrid');
})
window.setTimeout(refreshGrid, 5000);
function refreshGrid() {
	var grid = jQuery("#datagrid");
	grid.trigger("reloadGrid");
	window.setTimeout(refreshGrid, 5000);
}
```

ObjectiveC (iPad) Examples

Authentication

ObjectiveC

JSON

```
CommunicationSingleton *CSS = [CommunicationSingleton sharedSingleton];

    // building the json object
    if (([emailString length] != 0) && ([passwordString length] != 0))
    {

        NSMutableURLRequest *request = [[NSMutableURLRequest alloc] init];
        NSString *requestURLString = [NSString stringWithFormat:@"%@authorize",[CSS getDefaultAPIURL]];
        [request setURL:[NSURL URLWithString:requestURLString]];
        [request setHTTPMethod:@"POST"];
        [request setValue:@"application/json" forHTTPHeaderField:@"Content-Type"];

        NSString *requestBodyString = [NSString stringWithFormat:@"{\"user_id\": \"%@\", \"password\": \"%@\",
\"client_id\": \"%@\"}",emailString,passwordString,[[NSUserDefaults standardUserDefaults] objectForKey:@"
ClientIdentificationForAuthorizationPurposes"]];
        [request setHTTPBody:[requestBodyString dataUsingEncoding:NSUTF8StringEncoding]];
        [request setTimeoutInterval:60];

        __block BOOL loginFailed = NO;

        __block BOOL invalidUser = NO;

        dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_HIGH, 0), ^(void){

            //build an info object and convert to json
            NSError *requestError;
            NSHTTPURLResponse *responseHTTP;

            NSData *response = [NSURLConnection sendSynchronousRequest:request returningResponse:&responseHTTP
error:&requestError];

            if ([responseHTTP statusCode] == 200)
            {
                NSDictionary *tempdict = [NSJSONSerialization JSONObjectWithData:response options:
NSJSONReadingMutableContainers error:&requestError];
                [CSS setUpAuthenticationForType:[tempdict valueForKey:@"token_type"] withID:[tempdict
valueForKey:@"access_token"]];

                // authorization was successfull

            } else {

                loginFailed = YES;

                if (requestError.code == -1012) {

                    // Failed because it's an invalid user

                    invalidUser = YES;

                } else {

                    // check if project list stored on device.
                    // if yes, start offline mode
                }
            }
        });

    }
```

Find out how many teams I am part of

ObjectiveC

JSON

```
CommunicationSingleton *CSS = [CommunicationSingleton sharedSingleton];

    //build an info object and convert to json
    NSError *requestError;
    NSHTTPURLResponse *responseHTTP;

    NSMutableURLRequest *requestForSlug = [[NSMutableURLRequest alloc] init];

    NSString *teamRequestString = [NSString stringWithFormat:@"%@teams",[CSS getDefaultAPIURL]];
    [requestForSlug setURL:[NSURL URLWithString:teamRequestString]];
    [requestForSlug setHTTPMethod:@"GET"];
    [requestForSlug setValue:@"application/json" forHTTPHeaderField:@"Content-Type"];
    [requestForSlug setValue:[CSS getAuthorization] forHTTPHeaderField:@"Authorization"];

    NSData *responseForSlug = [NSURLConnection sendSynchronousRequest:requestForSlug returningResponse:
&responseHTTP error:&requestError];
    [requestForSlug release];

    if ([responseHTTP statusCode] == 200) {
        // Success. Team names array.
        NSArray *tempdictForSlug = [NSJSONSerialization JSONObjectWithData:responseForSlug options:
NSJSONReadingMutableContainers error:&requestError];
        NSLog(@"How many teams I am part of: %d", [tempdictForSlug count]);
    }else{
        // Login Failed.
    }
```

Get team project list

ObjectiveC

JSON

```
CommunicationSingleton *CSS = [CommunicationSingleton sharedSingleton];

    //build an info object and convert to json
    NSError *requestError;
    NSHTTPURLResponse *responseHTTP;

    //get projectList
    NSMutableURLRequest *requestForProjectList = [[NSMutableURLRequest alloc] init];

    NSString *projectListRequestString = [NSString stringWithFormat:@"%@%@/projects",[CSS getDefaultAPIURL],
[CSS getSlug]];
    [requestForProjectList setURL:[NSURL URLWithString:projectListRequestString]];
    [requestForProjectList setHTTPMethod:@"GET"];
    [requestForProjectList setValue:@"application/json" forHTTPHeaderField:@"Content-Type"];
    [requestForProjectList setValue:[CSS getAuthorization] forHTTPHeaderField:@"Authorization"];

    NSData *responseForProjectList = [NSURLConnection sendSynchronousRequest:requestForProjectList
returningResponse:&responseHTTP error:&requestError];

    [requestForProjectList release];

    if (responseForProjectList == NULL) {

            // Login Failed
    }
    else {

        NSError *error;
        NSArray *projectList = [NSJSONSerialization JSONObjectWithData:responseForProjectList options:
NSJSONReadingMutableContainers error:&error];

        BOOL success = [[ProjectManager sharedProjectManager] loadProjectsFromData:projectList];

        if (!success){

            // Failed to Load Projects List

        }
    }
```

Get project details

ObjectiveC

JSON

```
//build an info object and convert to json
        NSError *requestError;
        NSHTTPURLResponse *responseHTTP;
        NSError *dictError;

        //Get project details

        NSMutableURLRequest *requestForProjectDetails;
        requestForProjectDetails = [[NSMutableURLRequest alloc] init];

        [requestForProjectDetails setURL:[NSURL URLWithString:[NSString stringWithFormat:@"%@%@/projects/%@",
[CSS getDefaultAPIURL],[CSS getSlug],proj.mID]]];
        [requestForProjectDetails setHTTPMethod:@"GET"];
        [requestForProjectDetails setValue:@"application/json" forHTTPHeaderField:@"Content-Type"];
        [requestForProjectDetails setValue:[CSS getAuthorization] forHTTPHeaderField:@"Authorization"];

        NSData *responseForProjectDetails = [NSURLConnection sendSynchronousRequest:requestForProjectDetails
returningResponse:&responseHTTP error:&requestError];
        [requestForProjectDetails release];

        if (responseHTTP.statusCode == 200) {

            // Success, we have a project detail dictionary
            NSDictionary *projectDetails = [NSJSONSerialization JSONObjectWithData:responseForProjectDetails
options:NULL error:&dictError];

            // Time to process this project details as you want/need
            [proj processProjectDetails:projectDetails];

        }else{
            // Download Failed
            return;
        }
```

Get project topology

ObjectiveC

JSON

```
//build an info object and convert to json
        NSError *requestError;
        NSHTTPURLResponse *responseHTTP;

        NSString* requestString = [[NSString alloc] initWithFormat:@"%@%@/projects/%@/topology",[CSS
getDefaultAPIURL],[CSS getSlug], mProjectID];

        NSMutableURLRequest *requestForProjectsTree = [[NSMutableURLRequest alloc] init];
        [requestForProjectsTree setURL:[NSURL URLWithString:requestString]];
        [requestString release];

        [requestForProjectsTree setHTTPMethod:@"GET"];
        [requestForProjectsTree setValue:@"application/json" forHTTPHeaderField:@"Content-Type"];
        [requestForProjectsTree setValue:[[CommunicationSingleton sharedSingleton] getAuthorization]
forHTTPHeaderField:@"Authorization"];

        NSData *response = [NSURLConnection sendSynchronousRequest:requestForProjectsTree returningResponse:
&responseHTTP error:&requestError];
        [requestForProjectsTree release];

        NSString *path = [[proj getProjectFolderPath] stringByAppendingPathComponent:[NSString stringWithFormat:
@"%@.topology", mProjectID]];

        if ([responseHTTP statusCode] == 200)
        {
            // Save project topology locally.
            [response writeToFile:path atomically:NO];

            // Load project topology to memory.
            [proj loadTopologyFromData:response];

        } else {

            // If Failed

            // Check if we have the last used topology for this project is available offline (rembember that
this is could not be the last version)
            if ([[NSFileManager defaultManager] fileExistsAtPath:path]) {
                // Loading project topology offline

                NSData* data = [NSData dataWithContentsOfFile:path];
                [proj loadTopologyFromData:data];

            } else {
                // Download Failed + Local Storage not available
            }

        }
```

Get project disciplines

ObjectiveC

JSON

```
    Project* proj = [[ProjectManager sharedProjectManager] getProjectbyID:mProjectID];

    maxNodes = [[proj getAllNodesWithoutParents] count];

    // Don't download nodes data if we want to download them dynamically
    // we are still going to need the parents
```

```
    if ( ![[[NSUserDefaults standardUserDefaults] valueForKey:kDownloadNodesDynamically] boolValue] ) {

        dispatch_async(dispatch_get_main_queue(), ^(void){

            mDownloadManager = [[DownloadManager alloc] init];
            mDownloadManager.delegate = self;

            //get disciplines
            NSArray *disciplineIds = proj.projectsDisciplineIDs;
            for (int i = 0; i < [disciplineIds count]; i++) {
                [mDownloadManager downloadFiles:[proj getAllNodesWithoutParents] forProject:mProjectID
forDiscipline:[disciplineIds objectAtIndex:i]];
            }

            if ([disciplineIds count] < 1) {
                [mDownloadManager downloadFiles:[proj getAllNodesWithoutParents] forProject:mProjectID
forDiscipline:NULL];
            }

        });
    }
```

In "downloadFiles" method there are some logic and data storage operations and for each of the discipline you can call this snippet:

```
        NodeHeaderDatas *nodeDatas = [mFilesToProcess objectAtIndex:0];
        NSString *nodeidForAPICall = nodeDatas.nodeID;

        CommunicationSingleton *CSS = [CommunicationSingleton sharedSingleton];
        NSMutableURLRequest *requestForNodeData = [[NSMutableURLRequest alloc] init];
        NSString *nodesURL;
        if (nodeDatas.disciplineForDownloading != NULL) {
            nodesURL = [NSString stringWithFormat:
                        @"%@%@/objects/%@/disciplines/%@/geometries/threejs",
                        [CSS getDefaultAPIURL],
                        [CSS getSlug],
                        nodeidForAPICall,
                        nodeDatas.disciplineForDownloading];
        }else{
            nodesURL = [NSString stringWithFormat:
                        @"%@%@/objects/%@/geometries/threejs",
                        [CSS getDefaultAPIURL],
                        [CSS getSlug],
                        nodeidForAPICall];

        }
        [requestForNodeData setURL:[NSURL URLWithString:nodesURL]];
        [requestForNodeData setHTTPMethod:@"GET"];
        [requestForNodeData setValue:@"application/json" forHTTPHeaderField:@"Content-Type"];
        [requestForNodeData setValue:[CSS getAuthorization] forHTTPHeaderField:@"Authorization"];

        if(requestForNodeData){
            mConnection = [[NSURLConnection alloc] initWithRequest:requestForNodeData delegate:self
startImmediately:YES];
        }
if (!mConnection)
        {
            // Connection Failed
        }
        else
        {
            [mConnection scheduleInRunLoop:[NSRunLoop currentRunLoop] forMode:NSRunLoopCommonModes];
            [mConnection start];
        }
        [requestForNodeData release];
```

Upload project thumbnail

ObjectiveC

JSON

```
CommunicationSingleton *CSS = [CommunicationSingleton sharedSingleton];
NSString *urlString = [[CSS getDefaultAPIURL] stringByAppendingFormat:@"%@/projects/%@/thumbnail",[CSS getSlug],
currentProject.mID];
MultIPartFormCommunication *MPC = [MultIPartFormCommunication sharedSingleton];
if (![MPC uploadImage:imagePlaceholder.image withExtension:@"jpeg" withFileName:@"thumbnail" ToURLString:
urlString]){
[[ProjectManager sharedProjectManager] addUnUploadedThumbnailsProjectIdToList:currentProject.mID];
}
```

See the bottom of the page to see the uploadImage method

Create an issue in the project

ObjectiveC

JSON

```
CommunicationSingleton *CSS = [CommunicationSingleton sharedSingleton];
//get its dictionary:
    NSMutableDictionary *currentIssueData = [self getIssueInformationForIssueId:ISSUE_ID];

    //create nsdata for uploading:
    NSError *error;
    NSData *dataForCurrentIssue = [NSJSONSerialization dataWithJSONObject:currentIssueData options:
NSJSONWritingPrettyPrinted error:&error];

    //create issue request:
    NSMutableURLRequest *uploadRequestForCurrentIssue;
    uploadRequestForCurrentIssue = [[NSMutableURLRequest alloc] init];

    NSString *urlStringForIssue = [NSString stringWithFormat:@"%@%@/projects/%@/issues",
                                   [CSS getDefaultAPIURL],
                                   [CSS getSlug],
                                   self.mID];

    [uploadRequestForCurrentIssue setURL:[NSURL URLWithString:urlStringForIssue]];
    [uploadRequestForCurrentIssue setHTTPMethod:@"POST"];
    [uploadRequestForCurrentIssue setValue:@"application/json" forHTTPHeaderField:@"Content-Type"];
    [uploadRequestForCurrentIssue setValue:[CSS getAuthorization] forHTTPHeaderField:@"Authorization"];
    [uploadRequestForCurrentIssue setHTTPBody:dataForCurrentIssue];

    NSHTTPURLResponse *responseHTTPForCurrentIssue;

    NSData *responseForCurrentIssue = [NSURLConnection sendSynchronousRequest:uploadRequestForCurrentIssue
returningResponse:&responseHTTPForCurrentIssue error:&error];
    [uploadRequestForCurrentIssue release];

    if ([responseHTTPForCurrentIssue statusCode] == 201) {
        // Success, issue uploaded
    }else{
        // Error, upload failed
    }
```

Get the issue list from the project

ObjectiveC

JSON

```
CommunicationSingleton *CSS = [CommunicationSingleton sharedSingleton];
    //create issue request:
    NSMutableURLRequest *issueListRequest;
    issueListRequest = [[NSMutableURLRequest alloc] init];

    NSString *urlStringForIssue = [NSString stringWithFormat:@"%@%@/projects/%@/issues",
                                   [CSS getDefaultAPIURL],
                                   [CSS getSlug],
                                   self.mID];
    DLog(@"%@",urlStringForIssue);
    [issueListRequest setURL:[NSURL URLWithString:urlStringForIssue]];
    [issueListRequest setHTTPMethod:@"GET"];
    [issueListRequest setValue:@"application/json" forHTTPHeaderField:@"Content-Type"];
    [issueListRequest setValue:[CSS getAuthorization] forHTTPHeaderField:@"Authorization"];

    NSHTTPURLResponse *responseHTTPForIssueList;
    NSError *error;

    NSData *rawIssueList = [NSURLConnection sendSynchronousRequest:issueListRequest returningResponse:
&responseHTTPForIssueList error:&error];
    [issueListRequest release];

    if ([responseHTTPForIssueList statusCode] == 200) {
        NSMutableArray *issueList = [NSJSONSerialization JSONObjectWithData:rawIssueList options:
NSJSONReadingMutableContainers error:&error];
        return  issueList;
    }else{
        return NULL;
    }
```

Delete all the issues from the project

ObjectiveC

JSON

```
//delete it form the server
        CommunicationSingleton *CSS = [CommunicationSingleton sharedSingleton];
        //create issue request:
        NSMutableURLRequest *deleteRequest = [[NSMutableURLRequest alloc] init];

        NSString *urlStringForIssue = [NSString stringWithFormat:@"%@%@/issues/%@",
                                       [CSS getDefaultAPIURL],
                                       [CSS getSlug],
                                       issueid];

        [deleteRequest setURL:[NSURL URLWithString:urlStringForIssue]];
        [deleteRequest setHTTPMethod:@"DELETE"];
        [deleteRequest setValue:@"application/json" forHTTPHeaderField:@"Content-Type"];
        [deleteRequest setValue:[CSS getAuthorization] forHTTPHeaderField:@"Authorization"];

        NSHTTPURLResponse *responseHTTPForDeletion;
        [NSURLConnection sendSynchronousRequest:deleteRequest returningResponse:&responseHTTPForDeletion error:
&error];
        [deleteRequest release];

        if ([responseHTTPForDeletion statusCode] == 200) {

                //successful deletion
                return 0;

        } else {

        //add it to a deletion list
        [self addIssueToUnDeletedIssueList:issueid];

        return 1;

        }
```

Create a new pin for the issue

ObjectiveC

JSON

```
CommunicationSingleton *CSS = [CommunicationSingleton sharedSingleton];

        NSMutableDictionary *pinData = [self getPinDataForPinId:PIN_ID inIssue:ISSUE_ID];

        //remove the id and the objectsparent id since they are unrequired data for the server:
        NSString *oldObjectsParentID = [pinData objectForKey:@"objectsParentId"];
        [pinData removeObjectForKey:@"objectsParentId"];
        NSString *oldPinId = [pinData objectForKey:@"id"];
        [pinData removeObjectForKey:@"id"];

        //set the new issue id:
        [pinData setObject:newCurrentIssueId forKey:@"issueId"];

        //create nsdata for uploading:
        NSData *dataForCurrentPin = [NSJSONSerialization dataWithJSONObject:pinData options:
NSJSONWritingPrettyPrinted error:&error];

        //create pin request:
        NSMutableURLRequest *uploadRequestForCurrentPin;
        uploadRequestForCurrentPin = [[NSMutableURLRequest alloc] init];

        NSString *urlStringForPin = [NSString stringWithFormat:@"%@%@/issues/%@/pins",
                                     [CSS getDefaultAPIURL],
                                     [CSS getSlug],
                                     self.newCurrentIssueID];

        [uploadRequestForCurrentPin setURL:[NSURL URLWithString:urlStringForPin]];
        [uploadRequestForCurrentPin setHTTPMethod:@"POST"];
        [uploadRequestForCurrentPin setValue:@"application/json" forHTTPHeaderField:@"Content-Type"];
        [uploadRequestForCurrentPin setValue:[CSS getAuthorization] forHTTPHeaderField:@"Authorization"];
        [uploadRequestForCurrentPin setHTTPBody:dataForCurrentPin];
NSHTTPURLResponse *responseHTTPForCurrentPin;

        NSData *responseForCurrentPin = [NSURLConnection sendSynchronousRequest:uploadRequestForCurrentPin
returningResponse:&responseHTTPForCurrentPin error:&error];
        [uploadRequestForCurrentPin release];

        if ([responseHTTPForCurrentPin statusCode] == 201) {
                // success
        } else {
                // Upload Failed
        }
```

Get all the pins of the issue

ObjectiveC

JSON

```
CommunicationSingleton *CSS = [CommunicationSingleton sharedSingleton];
    //create issue request:
    NSMutableURLRequest *pinListRequest;
    pinListRequest = [[NSMutableURLRequest alloc] init];

    NSString *urlStringForIssue = [NSString stringWithFormat:@"%@%@/issues/%@/pins",
                                   [CSS getDefaultAPIURL],
                                   [CSS getSlug],
                                   issueid];
    DLog(@"%@",urlStringForIssue);
    [pinListRequest setURL:[NSURL URLWithString:urlStringForIssue]];
    [pinListRequest setHTTPMethod:@"GET"];
    [pinListRequest setValue:@"application/json" forHTTPHeaderField:@"Content-Type"];
    [pinListRequest setValue:[CSS getAuthorization] forHTTPHeaderField:@"Authorization"];

    NSHTTPURLResponse *responseHTTPForPinList;
    NSError *error;

    NSData *rawPinList = [NSURLConnection sendSynchronousRequest:pinListRequest returningResponse:
&responseHTTPForPinList error:&error];
    [pinListRequest release];

    if ([responseHTTPForPinList statusCode] == 200) {
        NSMutableArray *pinList = [NSJSONSerialization JSONObjectWithData:rawPinList options:
NSJSONReadingMutableContainers error:&error];
        return  pinList;
    }else{
        return NULL;
    }
```

Create/Upload a new attachment for the issue

ObjectiveC

JSON

```
NSString *attachmentURL = [[CSS getDefaultAPIURL] stringByAppendingFormat:@"%@/issues/%@/attachments", [CSS
getSlug],currentIssueId];
        MultIPartFormCommunication *MPC = [MultIPartFormCommunication sharedSingleton];
        NSString *imageName = [NSString stringWithFormat:@"issueImage%f",[[NSDate date] timeIntervalSince1970]];
        if ([MPC uploadImage:issueImage withExtension:@"jpeg" withFileName:imageName ToURLString:attachmentURL])
{
                //succesful uploading
            }else{
                //unsuccesful uploading store the issue image in a list:
            }
```

See the bottom of the page to see the uploadImage method

Get the attachment list from the issue

ObjectiveC

JSON

```
CommunicationSingleton *CSS = [CommunicationSingleton sharedSingleton];
    NSMutableURLRequest *attachmentsRequest;
    attachmentsRequest = [[NSMutableURLRequest alloc] init];

    NSString *urlStringForIssue = [NSString stringWithFormat:@"%@%@/issues/%@/attachments",
                                   [CSS getDefaultAPIURL],
                                   [CSS getSlug],
                                   issueid];
    DLog(@"%@",urlStringForIssue);
    [attachmentsRequest setURL:[NSURL URLWithString:urlStringForIssue]];
    [attachmentsRequest setHTTPMethod:@"GET"];
    [attachmentsRequest setValue:@"application/json" forHTTPHeaderField:@"Content-Type"];
    [attachmentsRequest setValue:[CSS getAuthorization] forHTTPHeaderField:@"Authorization"];

    NSHTTPURLResponse *responseHTTPForAttachments;
    NSError *error;

    NSData *rawAttachmentsList = [NSURLConnection sendSynchronousRequest:attachmentsRequest returningResponse:
&responseHTTPForAttachments error:&error];
    [attachmentsRequest release];

    if ([responseHTTPForAttachments statusCode] == 200) {
        NSMutableArray *attachmentsList = [NSJSONSerialization JSONObjectWithData:rawAttachmentsList options:
NSJSONReadingMutableContainers error:&error];
        if ([attachmentsList count] > 0) {
            return attachmentsList;
        }else{
            return NULL;
        }
    }else{
        return NULL;
    }
```

Upload image utility method for ObjectiveC

ObjectiveC

JSON

```
- (BOOL)uploadImage:(UIImage*)image withExtension:(NSString*)ext withFileName:(NSString*)fileName ToURLString:
(NSString*)urlString{

    CommunicationSingleton *CSS = [CommunicationSingleton sharedSingleton];
    NSURL *url = [NSURL URLWithString:urlString];
    NSMutableURLRequest *request = [NSMutableURLRequest requestWithURL:url];
    // Method
    [request setHTTPMethod:@"POST"];

    // Set headers
    NSString *boundary = @"AaB03x";
    NSString *contentType = [NSString stringWithFormat:@"multipart/form-data; boundary=%@", boundary];

    [request setValue:[CSS getAuthorization] forHTTPHeaderField:@"Authorization"];
    [request addValue:contentType forHTTPHeaderField:@"Content-Type"];

    // FORM
    NSData *dataForTestImage = UIImageJPEGRepresentation(image, 1.0);
    NSMutableData *body = [NSMutableData data];

    [body appendData:[[NSString stringWithFormat:@"--%@--\r\n", boundary] dataUsingEncoding:
NSUTF8StringEncoding]];
```

```
    [body appendData:[[NSString stringWithFormat:@"Content-Disposition: form-data; name=\"file\"; filename=\"%
@.%@\"\r\n", fileName,ext] dataUsingEncoding:NSUTF8StringEncoding]];
    [body appendData:[@"Content-Type: application/octet-stream\r\n\r\n" dataUsingEncoding:
NSUTF8StringEncoding]];
    [body appendData:dataForTestImage];

    // fileName
    [body appendData:[[NSString stringWithFormat:@"\r\n--%@--\r\n", boundary] dataUsingEncoding:
NSUTF8StringEncoding]];
    [body appendData:[[NSString stringWithFormat:@"Content-Disposition: form-data; name=\"fileName\"\r\n\r\n%@.%
@", fileName,ext] dataUsingEncoding:NSUTF8StringEncoding]];

    // type
    [body appendData:[[NSString stringWithFormat:@"\r\n--%@--\r\n", boundary] dataUsingEncoding:
NSUTF8StringEncoding]];
    [body appendData:[[NSString stringWithFormat:@"Content-Disposition: form-data; name=\"type\"\r\n\r\nimage/%
@", ext] dataUsingEncoding:NSUTF8StringEncoding]];

    // size
    [body appendData:[[NSString stringWithFormat:@"\r\n--%@--\r\n", boundary] dataUsingEncoding:
NSUTF8StringEncoding]];
    [body appendData:[[NSString stringWithFormat:@"Content-Disposition: form-data; name=\"size\"\r\n\r\n%d",
[dataForTestImage length]] dataUsingEncoding:NSUTF8StringEncoding]];


    // objectIds
    [body appendData:[[NSString stringWithFormat:@"\r\n--%@--\r\n", boundary] dataUsingEncoding:
NSUTF8StringEncoding]];
    [body appendData:[[NSString stringWithFormat:@"Content-Disposition: form-data; name=\"objectIds\"\r\n\r\n%
@", @""]
dataUsingEncoding:NSUTF8StringEncoding]];

    // End
    [body appendData:[[NSString stringWithFormat:@"\r\n--%@--\r\n", boundary] dataUsingEncoding:
NSUTF8StringEncoding]];

    [request setHTTPBody:body];

    NSHTTPURLResponse* response;
    NSError* err;
    [NSURLConnection sendSynchronousRequest:request
                          returningResponse:&response
                                      error:&err];

    if ([response statusCode] == 201) {
        return true;
    }else{
        return false;
    }
}
```