
Subject: Re: Flatbuffers package
Posted by [Xemuth](#) on Tue, 15 Dec 2020 11:51:27 GMT
[View Forum Message](#) <> [Reply to Message](#)

client flatbuffers example:

```
#include <Core/Core.h>
```

```
#include "flatbuffer-command/command_generated.h"
```

```
using namespace Upp;
```

```
//a simple Command struct
```

```
struct Command{
```

```
public:
```

```
int id;
```

```
String name;
```

```
Command(int _id, String _name){id = _id; name = _name;}
```

```
String ToString()const{
```

```
String toStr;
```

```
toStr << "Id: " << id << ", Name: " << name ;
```

```
return toStr;
```

```
}
```

```
};
```

```
struct CommandSerializer{
```

```
public:
```

```
CommandSerializer(const Command& cmd){
```

```
flatbuffers::Offset<flatbuffers::String> str1 = builder.CreateString(cmd.name.ToStd()); //First, we  
create a special flatbuffers object that handle string
```

```
TcplpServer::CommandBuilder commandBuilder(builder); //Then we create a commandBuilder  
(a special object that will construct our Command (the fbs file))
```

```
commandBuilder.add_id(cmd.id); //We define value of each parameters of our command
```

```
commandBuilder.add_name(str1); //We define value of each parameters of our command
```

```
flatbuffers::Offset<TcplpServer::Command> freshCommand = commandBuilder.Finish(); //we  
retrieve our command by telling our commandBuilder we are done with it
```

```
builder.Finish(freshCommand); //we end the builder of our newly command (named  
freshCommand)
```

```
}
```

```
int GetSize(){return builder.GetSize();} //Return a ptr to our serialized object
```

```
uint8_t* GetDatasPtr(){return builder.GetBufferPointer();} //Return size of our serialized object
```

```
private:
```

```
flatbuffers::FlatBufferBuilder builder; //Builder is used to create everything we want with  
flatbuffers
```

```
};
```

```

Command CommandDeserializer(const uint8_t* datas, int size){
    flatbuffers::Verifier verifier(datas,size); //Verifier is used to verify our buffer of data is a valid
command
    if(TcpIpServer::VerifyCommandBuffer(verfier)){
        const TcpIpServer::Command* serializedCmd = TcpIpServer::GetCommand(datas); //If our
buffer is valid then we retrieve a ptr to the command within this buffer
        return Command(serializedCmd->id(), String(serializedCmd->name()->c_str())); //then we use
our buffer to build our command
    }
    throw Exc("Invalide command");
}

```

```

Command SendCommand(const Command& cmd){
    TcpSocket s;
    s.Timeout(600);
    if(!s.Connect("127.0.0.1", 3214)){
        throw Exc("Unable to connect to server!");
    }
    CommandSerializer serializer(cmd);

    dword sizeToSend = serializer.GetSize();
    if(s.Put(&sizeToSend,sizeof(sizeToSend)))
        s.Timeout(5000).Put(serializer.GetDatasPtr(),serializer.GetSize());

```

```

    String datas;
    dword sizeReceived;
    if(s.Get(&sizeReceived, sizeof(sizeReceived)))
        datas = s.Timeout(5000).Get(sizeReceived);

    return CommandDeserializer((const uint8_t*)datas.ToStd().c_str(),datas.GetLength());
}

```

```

CONSOLE_APP_MAIN{
    StdLogSetup(LOG_COUT | LOG_FILE);
    try{
        LOG(SendCommand(Command(3,"Exemple")));
        LOG(SendCommand(Command(3,"Exemple2")));
        LOG(SendCommand(Command(3,"Exemple3")));
    }catch(Exc& exception){
        LOG(exception);
    }
}

```

File Attachments

1) [ClientFlatbuffers.7z](#), downloaded 348 times
