ERA 6.x ServerAPI Tutorial in C Language

Version 6.3.12.0

ESET spol. s r.o.
January 21st, 2016

1 Usage

1. Load library
2. Get pointer to function, which can initialize library
3. Initialize library
4. Get pointer to function, which can send request and receive response
5. Get pointer to function, which can free response
6. Send request and receive response + free response
7. Get pointer to function, which can deinitialize library
8. Deinitialize library
9. Free library

1.1 Load library

```
HMODULE hMod = LoadLibrary(L"ServerApi.dll");
if (!hMod)
{
//error
}
```

Free library:

```
FreeLibrary(hMod);
```

1.2 Get pointer to function, which can initialize library

```
typedef int (*era_init_lib)();
era_init_lib init_lib = (era_init_lib)GetProcAddress(hMod,"era_init_lib");
if(!init_lib)
{
//error
}
```

1.3 Initialize library

```
int res = init_lib();

Return value:
ERA_SUCCESS                    0

ERA_INVALID_ARGUMENT           1
ERA_NOT_INITIALIZED            3
```

1.4 Get pointer to function, which can deinitialize library

```
typedef void (*era_deinit_lib)();
era_deinit_lib deinit_lib = (era_deinit_lib)GetProcAddress(hMod,"era_deinit_lib");
if(!deinit_lib)
{
//error
```

```
}
```

1.5 Get pointer to function, which can send request and receive response

```
typedef int (*era_process_request)(const char* request, char** response);
era_process_request send_request =
(era_process_request)GetProcAddress(hMod,"era_process_request");
if(!send_request)
{
//error
}
```

1.6 Get pointer to function, which can free response

```
typedef void (*era_free)(char* s);
era_free free_response = (era_free)GetProcAddress(hMod,"era_free");
if(!era_free free_response)
{
//error
}
```

1.7 Send request and receive response

```
std::string request;
char* szRes = NULL;
int res = send_request(request.c_str(),&szRes);

request.c_str() – JSON request, [in] const char*
&szRes – JSON response, [out] char** response

Return value:
ERA_SUCCESS                    0
ERA_UNSATISFIED_DEPENDENCY     2
```

Free response:

```
free_response(szRes);

szRes – [in] const char*
```

1.8 Deinitialize library

```
deinit_lib();
```

2 Example of usage

```
#include <windows.h>
#include <iostream>
#include <string>

typedef int (*era_process_request)(const char* request, char** response);
typedef void (*era_free)(char* s);
typedef int (*era_init_lib)();
typedef void (*era_deinit_lib)();


int main(int argc, char** argv)
{
    HMODULE hMod = LoadLibrary(L"ServerApi.dll"); //1
    if (!hMod)
    {
        std::cout<<  "Cannot load library error. Last error is:" << GetLastError()
<< std::endl;
        return 1;
    }

    era_init_lib init_lib = (era_init_lib)GetProcAddress(hMod,"era_init_lib"); //2
    if (!init_lib )
    {
        std::cout<<"Cannot init library" << std::endl;
        FreeLibrary(hMod); //9
        return 1;
    }

    era_deinit_lib deinit_lib =
(era_deinit_lib)GetProcAddress(hMod,"era_deinit_lib"); //7
    if (!deinit_lib )
    {
        std::cout<<"Cannot deinit libraries" << std::endl;
        FreeLibrary(hMod); //9
        return 1;
    }

    int res = init_lib(); //3
    if(res)
    {
        std::cout<<"Init lib result:" << res << std::endl;
        FreeLibrary(hMod); //9
        return 1;
    }

    era_process_request send_request =
(era_process_request)GetProcAddress(hMod,"era_process_request");//4
    if (!send_request)
    {
        std::cout<<"Cannot load  era_process_request" << std::endl;
        deinit_lib();  //8
```

```
        FreeLibrary(hMod); //9
        return 1;
    }

    era_free free_response = (era_free)GetProcAddress(hMod,"era_free");//5
    if (!free_response)
    {
        std::cout<<"Cannot load  era_free" << std::endl;
        deinit_lib();  //8
        FreeLibrary(hMod); //9
        return 1;
    }

    while (true)
    {
        std::string request;// = "{\"Era.ServerApi.StartRequest\":{}}";
        std::cout<<">";
        std::getline(std::cin, request);

        if (request == "quit")
        {;
            break;
        }

        std::cout<<"Executing json ..." << std::endl;
        char* szRes = NULL;
        int res = send_request(request.c_str(),&szRes); //6
        if (szRes)
            std::cout<<szRes<<std::endl;
        free_response(szRes); //6
    }

    std::cout<<"Exiting ... " << std::endl;

    deinit_lib();  //8

    FreeLibrary(hMod); //9

    std::cout<<"Library freed ... " << std::endl;

    return 0;
}
```

3 Examples of JSON messages

And some of their responses (some fields are described)

Start ServerApi:
Request: {"Era.ServerApi.StartRequest":{}}

Response: {"Era.ServerApi.SimpleResponse": {"result":true} }
result : true – it is ok

Create connection:
Request: {"Era.ServerApi.CreateConnectionRequest":{"host":"127.0.0.1","port":2223}}

hostname: identifies host i.e: localhost, 127.0.0.1, wayne.hq.eset.com
port: port for ConsoleApi - default: 2223

Response: {"Era.ServerApi.VerifyUserRequest":{"result":"11111111111111111111111asdsaas"} }
result: certificate chain

Verify certificate chain:
Request: {"Era.ServerApi.VerifyUserResponse":{"VerifyResult":true}}
VerifyResult: true means I accept certificate chain

Response: {"Era.ServerApi.SimpleResponse": {"result":true} }

Authentification:
Request: {"Era.Common.NetworkMessage.ConsoleApi.SessionManagement.RpcAuthLoginRequest" :
{"username":"Administrator", "password":"secret", "isDomainUser":false, "locale":"en-US"}}
isDomainUser: if username is domain user- true else false
locale: localization

Response: {"Era.Common.NetworkMessage.ConsoleApi.SessionManagement.RpcAuthLoginResponse":
{"userUuid":{"uuid":"00000000-0000-0000-7002-000000000002"}} }

Create computers:
Request:
{"Era.Common.NetworkMessage.ConsoleApi.Groups.RpcCreateComputerRequest":{"parentGroupUuid":{"uuid":"000
00000-0000-0000-7001-
000000000002"},"computerNames":["computer1"],"commonDescription":"first","collisionsHandling":1}}
parentGroupUuid: is uuid of parent group. In this case it is lost&found. Uuid for some other group “group1” can be obtained, when group1 is created.
collisionsHandling: default is 1, do not change

Response: {"Era.Common.NetworkMessage.ConsoleApi.Groups.RpcCreateComputerResponse":
{"results":[{"result":1,"requestIndex":0,"staticObjectIdentification":{"uuid":{"uuid":"0a7042bf-2cca-4568-9093-cd9c26cc842b"},"versionGuard":129},"staticObjectData":{"name":"computer1","description":"first"}}]} }
staticObjectIdentification: this is uuid assigned to computer. It can be used to delete computer.

Modify computer:
Request:
{"Era.Common.NetworkMessage.ConsoleApi.Groups.RpcModifyComputerRequest":{"staticObjectIdentification":{"uui
d":{"uuid":"0a7042bf-2cca-4568-9093-
cd9c26cc842b"},"versionGuard":129},"staticObjectData":{"name":"computer1","description":"modified"}}}
staticObjectIdentification: uuid is uuid of created computer
versionGuard: received from creating response

Response:
{"Era.Common.NetworkMessage.ConsoleApi.Groups.RpcModifyComputerResponse":
{"staticObjectIdentification":{"uuid":{"uuid":"0a7042bf-2cca-4568-9093-cd9c26cc842b"
},"versionGuard":133}} }

Remove computer:
Request:
{"Era.Common.NetworkMessage.ConsoleApi.Groups.RpcRemoveComputerRequest":{"staticObjectIdentification":[{"u
uid":{"uuid":"0a7042bf-2cca-4568-9093-cd9c26cc842b"},"versionGuard":133}]} }
staticObjectIdentification: uuid is uuid of modified computer
versionGuard: received from modifying response

{"Era.Common.NetworkMessage.ConsoleApi.Groups.RpcRemoveComputerResponse": {} }

Crate group:
Request:
{"Era.Common.NetworkMessage.ConsoleApi.Groups.RpcCreateStaticGroupRequest":{"staticObjectData":{"name":"scenarioCreateAndRemoveStaticGroup","description":"description"},"parentGroupUuid":{"uuid":"00000000-0000-0000-7001-000000000001"}}}
Fields have similar meaning as in the create computers request

Response:
{"Era.Common.NetworkMessage.ConsoleApi.Groups.RpcCreateStaticGroupResponse":{"staticObjectIdentification":{"uuid":{"uuid":"1c7f092c-0853-45ad-badd-6a6b949d8163"},"versionGuard":169},"staticObjectData":{"name":"scenarioCreateAndRemoveStaticGroup","description":"description"},"staticGroupRelations":{"parentGroup":{"uuid":"00000000-0000-0000-7001-000000000001"}}} }
Fields have similar meaning as in the create computers response

Move computer:
Request:
{"Era.Common.NetworkMessage.ConsoleApi.Groups.RpcMoveComputerRequest":{"staticObjectIdentification":[{"uuid":{"uuid":"0d767e6e-b35b-4d31-b411-51ceecb605b6"},"versionGuard":170}],"newParentGroupUuid":{"uuid":"507c5dba-791a-4b30-9940-9f5e7ad28a19"}}}
Response:
{"Era.Common.NetworkMessage.ConsoleApi.Groups.RpcMoveComputerResponse":
{"staticObjectIdentification":{"uuid":{"uuid":"0d767e6e-b35b-4d31-b411-51ceecb605b6"},"versionGuard":174}} }

Remove group:
Request:
{"Era.Common.NetworkMessage.ConsoleApi.Groups.RpcRemoveStaticGroupRequest":{"staticObjectIdentification":{"uuid":{"uuid":"507c5dba-791a-4b30-9940-9f5e7ad28a19"},"versionGuard":1000}}}
Fields have similar meaning as in the remove computers request

Response:
{"Era.Common.NetworkMessage.ConsoleApi.Groups.RpcRemoveStaticGroupResponse": {} }

Create policy:
Request:
{"Era.Common.NetworkMessage.ConsoleApi.Policies.RpcCreatePolicyRequest":{"staticObjectData":{"name":"scenarioCreatePolicyModifyAndRemoveIt","description":"Desciption"},"policyData":{"data":"","product":"agent"}}}

Response{"Era.Common.NetworkMessage.ConsoleApi.Policies.RpcCreatePolicyResponse":
{"staticObjectIdentification":{"uuid":{"uuid":"a86078af-480f-4848-a7ee-f521d05acaee"},"versionGuard":177}} }

Modify policy:
Request:
{"Era.Common.NetworkMessage.ConsoleApi.Policies.RpcModifyPolicyRequest":{"staticObjectIdentification":{"uuid":{"uuid":" a86078af-480f-4848-a7ee-f521d05acaee"},"versionGuard":177},"staticObjectData":{"name":"scenarioCreatePolicyModifyAndRemoveIt","description":"Modified"},"policyData":{"data":"","product":"Agent"}}}}
Response:
{"Era.Common.NetworkMessage.ConsoleApi.Policies.RpcModifyPolicyResponse":
{"staticObjectIdentification":{"uuid":{"uuid":"a86078af-480f-4848-a7ee-f521d05acaee"},"versionGuard":178}} }

Remove policy:
Request:
{"Era.Common.NetworkMessage.ConsoleApi.Policies.RpcRemovePolicyRequest":{"staticObjectIdentification":{"uuid":{"uuid":" a86078af-480f-4848-a7ee-f521d05acaee "},"versionGuard":178}}}

Response: {"Era.Common.NetworkMessage.ConsoleApi.Policies.RpcRemovePolicyResponse": {} }

Assign policy to computer:
Request:
{"Era.Common.NetworkMessage.ConsoleApi.Policies.RpcSetComputerPoliciesRequest":{"computerUuid":{"uuid":"e8e50609-b3bf-4a36-a736-7e411391d2a4"},"versionGuard":0,"policiesUuids":[{"uuid":"880b523b-7fa8-48f2-83eb-2f3dbf01e5b2"}]}}

Response:
{"Era.Common.NetworkMessage.ConsoleApi.Policies.RpcSetComputerPoliciesResponse": {"versionGuard":195} }

Create update client task:
Request:
{"Era.Common.NetworkMessage.ConsoleApi.TasksTriggers.RpcCreateClientTaskRequest":{"staticObjectData":{"name":"scenarioUpdateClient","description":"description"},"clientTaskConfiguration":{"taskType":5,"taskUpdate":{}}}}

Response: {"Era.Common.NetworkMessage.ConsoleApi.TasksTriggers.RpcCreateClientTaskResponse":{"staticObjectIdentification":{"uuid":{"uuid":"6e7fc02b-d76c-414d-9f83-9e952c8d449c"},"versionGuard":196}} }

Create scan client task:
Request:
{"Era.Common.NetworkMessage.ConsoleApi.TasksTriggers.RpcCreateClientTaskRequest":{"staticObjectData":{"name":"scenarioScanClient","description":"description"},"clientTaskConfiguration":{"taskType":1,"taskOnDemandScan":{"scanProfile":0,"customProfileName":"","scanTargets":["eset:\/\/AllTargets"],"cleaningEnabled":false}}}}

Response: {"Era.Common.NetworkMessage.ConsoleApi.TasksTriggers.RpcCreateClientTaskResponse":{"staticObjectIdentification":{"uuid":{"uuid":"8a45cf4f-b548-47f5-966c-53cccfe9b181"},"versionGuard":198}} }

Add client task now:
Request:
{"Era.Common.NetworkMessage.ConsoleApi.TasksTriggers.RpcAddClientTaskTargetRequest":{"clientTaskUuid" : {"uuid" : "8a45cf4f-b548-47f5-966c-53cccfe9b181"},"targetUuid" : {"uuid" : "e8e50609-b3bf-4a36-a736-7e411391d2a4"},"clientTriggerConfiguration" : {"triggerType" : 4,"triggerThrottle" : {},"asapTriggerCfg" : {}}}}

Response: {"Era.Common.NetworkMessage.ConsoleApi.TasksTriggers.RpcAddClientTaskTargetResponse":{"clientTriggerStaticObjectIdentification":{"uuid":{"uuid":"d18bc6ab-14f9-4b05-bc5d-7354bd248ae3"},"versionGuard":212}} }

Remove client task target:
Request:
{"Era.Common.NetworkMessage.ConsoleApi.TasksTriggers.RpcRemoveClientTaskTargetRequest":{"staticObjectIdentification":{"uuid":{"uuid":"726e84b5-de89-46e5-b006-f47cf0612074"},"versionGuard":10000}}}

Response: {"Era.Common.NetworkMessage.ConsoleApi.TasksTriggers.RpcRemoveClientTaskTargetResponse": {} }

Remove task:
Request:
{"Era.Common.NetworkMessage.ConsoleApi.TasksTriggers.RpcRemoveClientTaskRequest":{"staticObjectIdentification":{"uuid":{"uuid":"4ed41225-bd0c-4002-94b5-6dab2a01d239"},"versionGuard":10000}}}

Response: {"Era.Common.NetworkMessage.ConsoleApi.TasksTriggers.RpcRemoveClientTaskResponse": {} }

Create report template category:
Request: {"Era.Common.NetworkMessage.ConsoleApi.Reports.RpcCreateReportTemplateCategoryRequest" : {"staticObjectData" : {"name" : "TestReportTemplateCategory","description" : "description"}}}

Response: {"Era.Common.NetworkMessage.ConsoleApi.Reports.RpcCreateReportTemplateCategoryResponse":{"staticObjectIdentification":{"uuid":{"uuid":"edc28ace-f8d2-4c62-b50f-cc18fd53f0b5"},"versionGuard":223}} }

Create report template:
Request:
{"Era.Common.NetworkMessage.ConsoleApi.Reports.RpcCreateReportTemplateRequest" : {"staticObjectData" : {"name" : "scenarioGenerateReport","description" : "description"},"reportTemplate" : {"data" : {"used_symbol" : [{"column_id" : 0,"symbol_id" : 644,"aggregation_parameter" : {}}],"query_usage_definition_id" : 22},"rendering" : {"draw_chart" : false,"draw_table" : true,"table" : {"type_id" : 101,"columns" : [{"column_id" : 0,"order" : 0,"width" : 1,"label" : {"type" : 1,"res_id" : 508906757892866218,"literal" : "Computer name"}}]}}},"categoryUuid" : {"uuid" : "edc28ace-f8d2-4c62-b50f-cc18fd53f0b5"}}}

Response: {"Era.Common.NetworkMessage.ConsoleApi.Reports.RpcCreateReportTemplateResponse": {"staticObjectIdentification":{"uuid":{"uuid":"5064a4c1-d7ac-4ab6-b6f1-473d277b17a2"},"versionGuard":224}} }

Generate report from report template uuid:
Request:
{"Era.Common.NetworkMessage.ConsoleApi.Reports.RpcGenerateReportRequest" : {"reportTemplateUuid" : {"uuid" : "5064a4c1-d7ac-4ab6-b6f1-473d277b17a2"}}}

Response:
{"Era.ServerApi.ReportCSVResponse": {"reportCSV":"Computer name\nera-latest\nzmazma\ncomputer1"} }

Generate report from report template:
Request:

{"Era.Common.NetworkMessage.ConsoleApi.Reports.RpcGenerateReportRequest" : {"reportTemplate" : {"data" : {"used_symbol" : [{"column_id" : 0,"symbol_id" : 644,"aggregation_parameter" : {}}],"query_usage_definition_id" : 22},"rendering" : {"draw_chart" : false,"draw_table" : true,"table" : {"type_id" : 101,"columns" : [{"column_id" : 0,"order" : 0,"width" : 1,"label" : {"type" : 1,"res_id" : 508906757892866218,"literal" : "Computer name"}}]}}}}}

Response:
{"Era.ServerApi.ReportCSVResponse": {"reportCSV":"Computer name\nera-latest\nzmazma\ncomputer1"} }

Remove report template:
Request: {"Era.Common.NetworkMessage.ConsoleApi.Reports.RpcRemoveReportTemplateRequest" : {"staticObjectIdentification" : {"uuid" : {"uuid" : "5064a4c1-d7ac-4ab6-b6f1-473d277b17a2"},"versionGuard" : 224}}}

Response: {"Era.Common.NetworkMessage.ConsoleApi.Reports.RpcRemoveReportTemplateResponse": {} }

Remove report template category:
Request: {"Era.Common.NetworkMessage.ConsoleApi.Reports.RpcRemoveReportTemplateCategoryRequest" : {"staticObjectIdentification" : {"uuid" : {"uuid" : "edc28ace-f8d2-4c62-b50f-cc18fd53f0b5"},"versionGuard" : 223}}}

Response: {"Era.Common.NetworkMessage.ConsoleApi.Reports.RpcRemoveReportTemplateCategoryResponse": {} }

Close connection:
Request:
{"Era.ServerApi.CloseConnectionRequest":{}}

Response:
{"Era.ServerApi.SimpleResponse": {"result":true} }

Stop ServerApi:
Request:
{"Era.ServerApi.StopRequest":{}}

Response:
{"Era.ServerApi.SimpleResponse": {"result":true} }

Corresponding JSON messages to SPIKE operations:

Changing a client: **Modify computer**
Updating computer: **Create update client task**,**Add client task now**
Scan a client: **Create scan client task, Add client task now**
Changing configuration of client: **Assign policy to computer**
Creating a group: **Crate group**
Deleting a group: **Remove group**
Adding a client to a group: **Create computers, Move computer**
Removing a client from a group: **Remove computer**
Creating policies: **Create policy**
Updating policies: **Modify policy**
Deleting policies: **Remove policy**
Generate report to CSV: **Create report template category, Create report template, Generate report from report template uuid, Generate report from report template**

3.1 Correct sequence of messages

1. **Start ServerApi**
2. **Create connection**
3. **Verify certificate chain**
4. **Authentification**
5. Now you can create computers, policies, groups, tasks etc.
6. **Close connection**
7. **Stop ServerApi**

4 Example of this sequence

(for parsing JSON using boost/property_tree):

```
/// testconsoleapilib.cpp : Defines the entry point for the console application.
//


#include <windows.h>
#include <iostream>
#include <string>
#include <boost/property_tree/json_parser.hpp>
#include <boost/property_tree/ptree.hpp>
#include <sstream>
#include <boost/foreach.hpp>
#include <boost/scope_exit.hpp>

typedef int (*era_process_request)(const char* request, char** response);
typedef void (*era_free)(char* s);
typedef int (*era_init_lib)();
typedef void (*era_deinit_lib)();


int main(int argc, char** argv)
{
    HMODULE hMod = LoadLibrary(L"ServerApi.dll");
    BOOST_SCOPE_EXIT(&hMod) {
        if(hMod)
        {
            FreeLibrary(hMod);
        }
    } BOOST_SCOPE_EXIT_END
    if (!hMod)
```

```
    {
        std::cout<<  "Cannot load library error. Last error is:" << GetLastError()
<< std::endl;
        return 1;
    }

    era_init_lib init_lib = (era_init_lib)GetProcAddress(hMod,"era_init_lib");
    era_deinit_lib deinit_lib =
(era_deinit_lib)GetProcAddress(hMod,"era_deinit_lib");

    if (!init_lib )
    {
        std::cout<<"Cannot init library" << std::endl;
        return 1;
    }
    if (!deinit_lib )
    {
        std::cout<<"Cannot deinit libraries" << std::endl;
        return 1;
    }

    int res = init_lib();

    BOOST_SCOPE_EXIT(&deinit_lib,&res) {
        if(!res)
        {
            if(deinit_lib)
            {
                deinit_lib();
            }
        }
    } BOOST_SCOPE_EXIT_END

    if(res)
    {
        std::cout<<"Init lib result:" << res << std::endl;
        return 1;
    }

    era_process_request send_request =
(era_process_request)GetProcAddress(hMod,"era_process_request");
    if (!send_request)
    {
        std::cout<<"Cannot load  era_process_request" << std::endl;
        return 1;
    }

    era_free free_response = (era_free)GetProcAddress(hMod,"era_free");
    if (!free_response)
    {
        std::cout<<"Cannot load  era_free" << std::endl;
        return 1;
    }


    //Start ServerApi 1
    std::string request("{\"Era.ServerApi.StartRequest\":{}}");
    std::cout<<"Executing json " << request << " ..." << std::endl;
    char* szRes = NULL;
```

```
    res = send_request(request.c_str(),&szRes);
    if (!res)
    {
        std::string response(szRes);

        std::cout<<"Response is: " << response << std::endl;
        boost::property_tree::ptree pt2;
        std::istringstream is (response);
        read_json (is, pt2);

        bool resultMessageHasTrueResult = pt2.get
(boost::property_tree::ptree::path_type("Era.ServerApi.SimpleResponse/result",
'/'),false);

        if(!resultMessageHasTrueResult)
        {
            std::cout << "ServerApi was not started properly." << std::endl;
            return 1;
        }

    }
    else
    {
        std::cout << "Error in ServerApi: " << res << std::endl;
        free_response(szRes);
        return 1;

    }
    BOOST_SCOPE_EXIT(&send_request,&free_response){
        //Stop ServerApi 7
        std::string request="{\"Era.ServerApi.StopRequest\":{}}";
        std::cout<<"Executing json " << request << " ..." << std::endl;
        char* szRes = NULL;
        int res = send_request(request.c_str(),&szRes);
        if (!res)
        {
            std::string response(szRes);

            std::cout<<"Response is: " << response << std::endl;
            boost::property_tree::ptree pt2;
            std::istringstream is (response);
            read_json (is, pt2);

            bool resultMessageHasTrueResult = pt2.get
(boost::property_tree::ptree::path_type("Era.ServerApi.SimpleResponse/result",
'/'),false);

            if(!resultMessageHasTrueResult)
            {
                std::cout << "Stop request failed." << std::endl;
                free_response(szRes);
            }

        }
        else
        {
            std::cout << "Error in ServerApi: " << res << std::endl;
            free_response(szRes);
        }
```

```
        free_response(szRes);
    } BOOST_SCOPE_EXIT_END
    free_response(szRes);

    //CreateConfiguration 2
    request =
"{\"Era.ServerApi.CreateConnectionRequest\":{\"host\":\"127.0.0.1\",\"port\":2223}
}";
    std::cout<<"Executing json " << request << " ..." << std::endl;
    szRes = NULL;
    res = send_request(request.c_str(),&szRes);
    if (!res)
    {
        std::string response(szRes);

        std::cout<<"Response is: " << response << std::endl;
        boost::property_tree::ptree pt2;
        std::istringstream is (response);
        read_json (is, pt2);

        try{
            std::string resultMessageHasTrueResult = pt2.get<std::string>
(boost::property_tree::ptree::path_type("Era.ServerApi.VerifyUserRequest/result",
'/'));
        }
        catch(const std::exception& e)
        {
            std::cout << "ServerApi can not connect." << std::endl;
            free_response(szRes);
            return 1;
        }

    }
    else
    {
        std::cout << "Error in ServerApi: " << res << std::endl;
        free_response(szRes);
        return 1;
    }
    BOOST_SCOPE_EXIT(&send_request,&free_response){
        //CloseConnection 6
        std::string request="{\"Era.ServerApi.CloseConnectionRequest\":{}}";
        std::cout<<"Executing json " << request << " ..." << std::endl;
        char* szRes = NULL;
        int res = send_request(request.c_str(),&szRes);
        if (!res)
        {
            std::string response(szRes);

            std::cout<<"Response is: " << response << std::endl;
            boost::property_tree::ptree pt2;
            std::istringstream is (response);
            read_json (is, pt2);

            bool resultMessageHasTrueResult = pt2.get
(boost::property_tree::ptree::path_type("Era.ServerApi.SimpleResponse/result",
'/'),false);

            if(!resultMessageHasTrueResult)
```

```
            {
                std::cout << "Connection was not closed properly." << std::endl;
                free_response(szRes);
            }

        }
        else
        {
            std::cout << "Error in ServerApi: " << res << std::endl;
            free_response(szRes);
        }
        free_response(szRes);
    } BOOST_SCOPE_EXIT_END
    free_response(szRes);

    //VerifyUserResponse 3
    request="{\"Era.ServerApi.VerifyUserResponse\":{\"VerifyResult\":true}}";
    std::cout<<"Executing json " << request << " ..." << std::endl;
    szRes = NULL;
    res = send_request(request.c_str(),&szRes);
    if (!res)
    {
        std::string response(szRes);

        std::cout<<"Response is: " << response << std::endl;
        boost::property_tree::ptree pt2;
        std::istringstream is (response);
        read_json (is, pt2);

        bool resultMessageHasTrueResult = pt2.get
(boost::property_tree::ptree::path_type("Era.ServerApi.SimpleResponse/result",
'/'),false);

        if(!resultMessageHasTrueResult)
        {
            std::cout << "ServerApi was not started properly." << std::endl;
            free_response(szRes);
            return 1;
        }

    }
    else
    {
        std::cout << "Error in ServerApi: " << res << std::endl;
        free_response(szRes);
        return 1;
    }
    free_response(szRes);

    //Authentificate 4

request="{\"Era.Common.NetworkMessage.ConsoleApi.SessionManagement.RpcAuthLoginReq
uest\" : {\"username\":\"Administrator\", \"password\":\"secret\",
\"isDomainUser\":false, \"locale\":\"en-US\"}}";
    std::cout<<"Executing json " << request << " ..." << std::endl;
    szRes = NULL;
    res = send_request(request.c_str(),&szRes);
    if (!res)
    {
```

```
        std::string response(szRes);

        std::cout<<"Response is: " << response << std::endl;
        boost::property_tree::ptree pt2;
        std::istringstream is (response);
        read_json (is, pt2);

        try{
            std::string resultMessageHasTrueResult = pt2.get<std::string>
(boost::property_tree::ptree::path_type("Era.Common.NetworkMessage.ConsoleApi.Sess
ionManagement.RpcAuthLoginResponse/userUuid/uuid", '/'));
        }
        catch(const std::exception& e)
        {
            std::cout << "ServerApi can not authentificate. Error is: " <<
e.what() << std::endl;
            free_response(szRes);
            return 1;
        }

    }
    else
    {
        std::cout << "Error in ServerApi: " << res << std::endl;
        free_response(szRes);
        return 1;
    }
    free_response(szRes);

    //CreateStaticGroup 5

request="{\"Era.Common.NetworkMessage.ConsoleApi.Groups.RpcCreateStaticGroupReques
t\":{\"staticObjectData\":{\"name\":\"group1\",\"description\":\"description\"},\"
parentGroupUuid\":{\"uuid\":\"00000000-0000-0000-7001-000000000001\"}}}";
    std::cout<<"Executing json " << request << " ..." << std::endl;
    szRes = NULL;
    res = send_request(request.c_str(),&szRes);
    std::string uuidOfGroup = "";
    if (!res)
    {
        std::string response(szRes);

        std::cout<<"Response is: " << response << std::endl;
        boost::property_tree::ptree pt2;
        std::istringstream is (response);
        read_json (is, pt2);


        try{
            uuidOfGroup = pt2.get<std::string>
(boost::property_tree::ptree::path_type("Era.Common.NetworkMessage.ConsoleApi.Grou
ps.RpcCreateStaticGroupResponse/staticObjectIdentification/uuid/uuid", '/'));
            int version = pt2.get<int>
(boost::property_tree::ptree::path_type("Era.Common.NetworkMessage.ConsoleApi.Grou
ps.RpcCreateStaticGroupResponse/staticObjectIdentification/versionGuard", '/'));
            std::string name = pt2.get<std::string>
(boost::property_tree::ptree::path_type("Era.Common.NetworkMessage.ConsoleApi.Grou
ps.RpcCreateStaticGroupResponse/staticObjectData/name", '/'));
```

```
            std::string description = pt2.get<std::string>
(boost::property_tree::ptree::path_type("Era.Common.NetworkMessage.ConsoleApi.Grou
ps.RpcCreateStaticGroupResponse/staticObjectData/description", '/'));
            std::string parentuuid = pt2.get<std::string>
(boost::property_tree::ptree::path_type("Era.Common.NetworkMessage.ConsoleApi.Grou
ps.RpcCreateStaticGroupResponse/staticGroupRelations/parentGroup/uuid", '/'));
            if(parentuuid != "00000000-0000-0000-7001-000000000001" || name
!="group1" || description!="description")
               {
                   throw std::runtime_error("Result is not expected");
               }
        }
        catch(const std::exception& e)
        {
            std::cout << "ServerApi can not create group. Error is: " << e.what()
<< std::endl;
            free_response(szRes);
            return 1;
        }



    }
    else
    {
        std::cout << "Error in ServerApi: " << res << std::endl;
        free_response(szRes);
        return 1;
    }
    free_response(szRes);

    //CreateComputer 5

request="{\"Era.Common.NetworkMessage.ConsoleApi.Groups.RpcCreateComputerRequest\"
:{\"parentGroupUuid\":{\"uuid\":\"" + uuidOfGroup +
"\"},\"computerNames\":[\"computer1\"],\"commonDescription\":\"first\",\"collision
sHandling\":1}}";
    std::cout<<"Executing json " << request << " ..." << std::endl;
    szRes = NULL;
    res = send_request(request.c_str(),&szRes);
    if (!res)
    {
        std::string response(szRes);

        std::cout<<"Response is: " << response << std::endl;
        boost::property_tree::ptree pt2;
        std::istringstream is (response);
        read_json (is, pt2);


        try{
           boost::property_tree::ptree& pos =
pt2.get_child(boost::property_tree::ptree::path_type("Era.Common.NetworkMessage.Co
nsoleApi.Groups.RpcCreateComputerResponse/results", '/'));
            int size = 0 ;
            BOOST_FOREACH (const boost::property_tree::ptree::value_type&
resultCreateComputer, pos) {
                if(size == 1)
                {
```

```
                    throw std::runtime_error("Result in create compures has to have
one result");
                }

                size++;
                int resultCreateComp = resultCreateComputer.second.get<int>
(boost::property_tree::ptree::path_type("result", '/'));
                int requestIndex = resultCreateComputer.second.get<int>
(boost::property_tree::ptree::path_type("requestIndex", '/'));
                std::string uuid =
resultCreateComputer.second.get<std::string>(boost::property_tree::ptree::path_typ
e("staticObjectIdentification/uuid/uuid", '/'));
                std::string version =
resultCreateComputer.second.get<std::string>(boost::property_tree::ptree::path_typ
e("staticObjectIdentification/versionGuard", '/'));
                std::string name =
resultCreateComputer.second.get<std::string>(boost::property_tree::ptree::path_typ
e("staticObjectData/name", '/'));
                std::string description =
resultCreateComputer.second.get<std::string>(boost::property_tree::ptree::path_typ
e("staticObjectData/description", '/'));
                if(resultCreateComp!=1 || requestIndex != 0 || name !="computer1"
|| description!="first")
                {
                    throw std::runtime_error("Result is not expected");
                }

            }


        }
        catch(const std::exception& e)
        {
            std::cout << "ServerApi can not create computer. Error is: " <<
e.what() << std::endl;
            free_response(szRes);
            return 1;
        }



    }
    else
    {
        std::cout << "Error in ServerApi: " << res << std::endl;
        free_response(szRes);
        return 1;
    }
    free_response(szRes);



    return 0;
}
```