X gRPC-

:

-
- proto-
-

gRPC- Python.

gRPC- proto-.

:

1. Python IDE.
2. pip :

```
pip>=21.1.2
grpcio-tools>=1.38.0
googleapis-common-protos
pyOpenSSL==19.1.0
```

proto-

proto- :

1. proto-.
2. py-: script.py.
3. grpc-proto-files. ITV Google proto-.
4. .

ITV proto-, gRPC-.

gRPC- . C:\ProgramData\ITV\IntellectX\Tickets.

.

ConfigurationService.ListUnits .

```
import grpc

from OpenSSL import crypto
from grpc._channel import _InactiveRpcError

from axxonsoft.bl.config.ConfigurationService_pb2 import ListUnitsRequest
from axxonsoft.bl.config.ConfigurationService_pb2_grpc import ConfigurationServiceStub
from axxonsoft.bl.auth.Authentication_pb2 import AuthenticateRequest
from axxonsoft.bl.auth.Authentication_pb2_grpc import AuthenticationServiceStub


def get_channel_credentials(cert_path):
    with open(cert_path, 'rb') as f:
        certificate = f.read()

    creds = grpc.ssl_channel_credentials(root_certificates=certificate)

    cert = crypto.load_certificate(crypto.FILETYPE_PEM, certificate)
    common_name = cert.get_subject().CN

    return creds, common_name


def get_ssl_channel(server, channel_creds, override_cn, auth_creds=None):
    channel_creds = grpc.composite_channel_credentials(channel_creds, auth_creds) if auth_creds else
channel_creds
    return grpc.secure_channel(server, channel_creds, options=(('grpc.ssl_target_name_override', override_cn),))


def get_auth_credentials(simple_channel, username, password):
    client = AuthenticationServiceStub(simple_channel)
    auth_request = AuthenticateRequest(user_name=username, password=password)
    response = client.Authenticate(auth_request)
    auth_header = (response.token_name, response.token_value)
    auth_creds = grpc.metadata_call_credentials(
        lambda _, cb: cb([auth_header], None))
    return auth_creds


def get_authorized_channel(certificate_path, ip="127.0.0.1", port=20109, username="root", password="root"):
    server = f"{ip}:{port}"
    channel_creds, cert_common_name = get_channel_credentials(certificate_path)
    try:
        simple_channel = get_ssl_channel(server, channel_creds, cert_common_name)
        auth_creds = get_auth_credentials(simple_channel, username, password)
        return get_ssl_channel(server, channel_creds, cert_common_name, auth_creds)
    except _InactiveRpcError as ex:
        print(f"Unable to connect to server. Details:\n{ex.details()}")


if __name__ == '__main__':
    print('This script need to provide a path to the certificate')
    path = r"C:\ProgramData\ITV\IntellectX\Tickets\Node.crt"
    channel = get_authorized_channel(path)
    config_service = ConfigurationServiceStub(channel)
    request = ListUnitsRequest(unit_uids=["root"])
    response = config_service.ListUnits(request)
    print(f"Found {len(response.units)} units:\n{response.units}")
```

get_authorized_channel :

1. certificate_path – ;
2. ip – IP- ("127.0.0.1");
3. port – gRPC API (20109);
4. username – ("root");
5. password – ("root").

proto- ITV .