**UNIVERZITET U NOVOM SADU**
**FAKULTET TEHNIČKIH NAUKA**

# STANDARDI I MODELIRANJE ELEKTROENERGETSKIH SISTEMA

## PREDMETNI PROJEKAT

Kandidat:
Tiodorovic Milica
br. indeksa: E358/2013

Novi Sad, decembar 2016. godine

# 1. PROJEKTNI ZADATAK

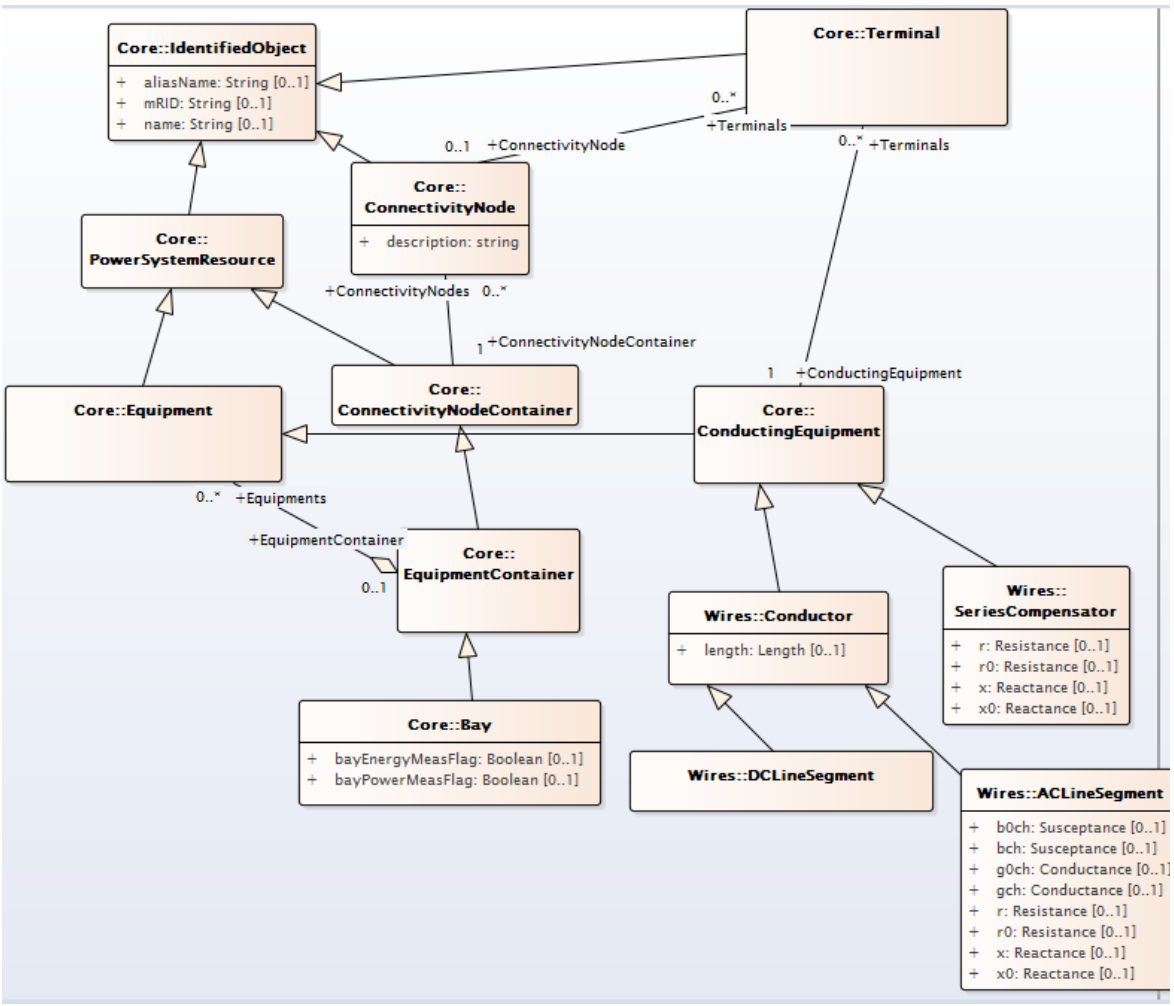
Implementirati Model Server koji omogućava modelovanje klasa i njihovih međusobnih referenci definisanih u modelu iec61970cim16v15_iec61968cim12v05_iec62325cim02v05_IES.eap. Klase koje se koriste u projektnom zadatku broj 47 su prikazane u tabeli 1.1.

| ***Paket*** | ***Klasa*** |
|---|---|
| *Core* | *IdentifiedObject* |
| | *Terminal* |
| | *ConnectivityNode* |
| | *PowerSystemResource* |
| | *ConnectivityNodeContainer* |
| | *Equipment* |
| | *EquipmentContainer* |
| | *ConductingEquipment* |
| | *Bay* |
| *Wires* | *ACLineSegment* |
| | *DCLineSegment* |
| | *SeriesCompensator* |
| | *Conductor* |

*Tabela 1.1. Spisak klasa sadržanih u projektnom zadatku broj 47*

## 2. KLASNI DIJAGRAM

Na slici 2.1 dat je prikaz klasnog dijagrama koji sadrži sve klase, atribute pomenutih klasa i reference među njima koje su od interesa za kontekstni profil definisan projektnim zadatkom broj 47. Dijagram klasa je modelovan upotrebom CIM (Common Information Model) apstraktnog UML (Unified Modelling Language) modela kojim su predstavljeni elementi elektroenergetskog sistema. Modelovani elementi pripadaju IEC 61970 standardu koji propisuje pravila i standard modelovanja fizičkih karakteristika elektroenergetske mreže.



*Slika 1.1. Klasni dijagram projektnog zadatka 47.*

Klase prikazane na dijagramu pripadaju paketima *Core* i *Wires*. Paket *Core* sadrži bazne, najčešće apstraktne, klase koje predstavljaju model fizičkih karakteristika energetskih mreža. Klase iz ovog paketa takođe pružaju opis mogućnosti grupisanja elemenata energetskih mreža. *Wires* paket predstavlja proširenje *Core* paketa i definiše klase koje opisuju komponente prenosnih i distributivnih energetskih mreža.

## 3. DEFINISANJE CIM PROFILA

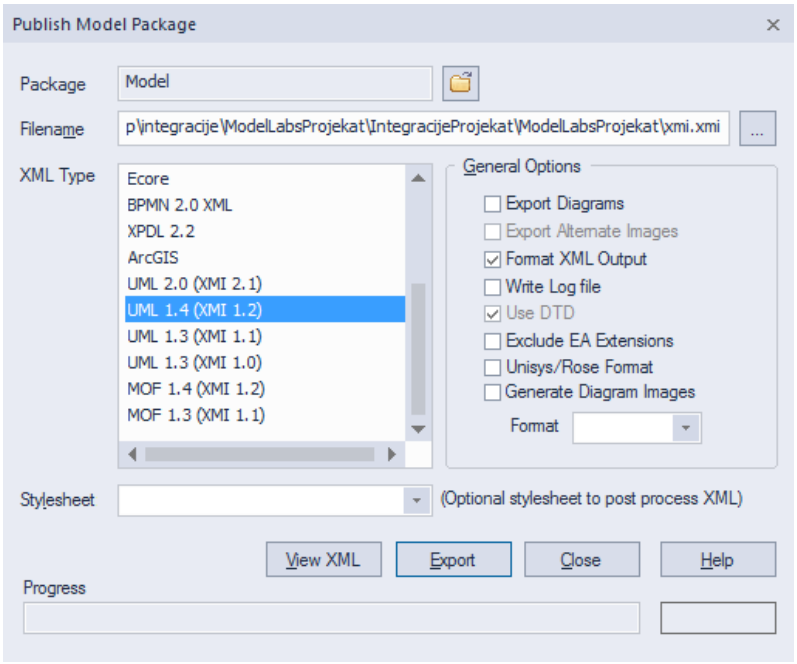
*CIM* profil je opis modela podataka i čini ga skup definicija klasa, njihovih atributa i veza između ovih klasa. U prvoj iteraciji cilj je unapred pripremljen *UML* model iz *Enterprise Architect* exportovati u *XMI* fajl. *XMI* file se koristi u daljem procesu modeliranja. Definisani profil je podskup polaznog modela i njegova svrha je definisanje domenskih i kontekstno zavisnih modela.Na osnovu ulazne *XMI* datoteke, *CIMTool* omogućava odabir klasa koje će ući u *CIM* profil, kao i čuvanje pofila u nekom od sledecih formata *html, rdf, txt*.

Proces kreiranja *XMI* fajla sastoji se od sledecih koraka:

1.Unutar *Enterprise Architect* alata, u okviru *Project Browser* prozora, nalazi se *Model*. Desnim klikom na *Model*, otvorice se padajuci meni i pojavice se opcija *Export Model to XMI.*

*2.*Izborom ove opcije, otvara se novi prozor. Klikom na polje *Publish* otvorice se podesavanja prikazana na slici 2.1

3. Zatim oznacimo *UML 1.4(XMI 1.2)* kao *XML Type*, a onda i *Format XML Output*. Klikom na *Export* pokrenuli smo generisanje *XMI* fajla koji ce se naci u okviru lokacije koje smo odradili.

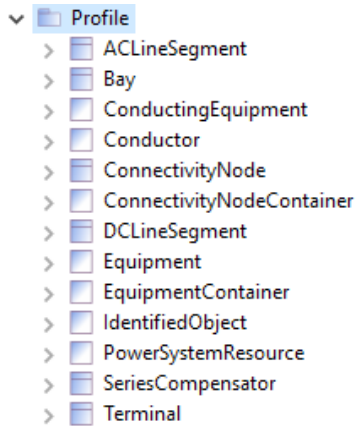



*Slika 2.1 Export xml fajla*

## 4. Kreiranje RDFS fajla

Kada smo to obavili prelazimo na sledecu tacku u izradnji projekta, a to je kreiranje profila. Za ovu svrhu potreban nam je drugi alat koji se zove *CIM Tool*. Ovaj alat se koristi kako bi *XMI* fajl, koji nosi podatke o modelu, prebacili u *RDFS* fajl, na osnovu kojeg dizajniramo *CIM* profil na osnovu *UML* sheme modela.

Dakle prvobitno importujemo nas *XMI* fajl, zatim dodajemo klase koje su nam potrebne, kao i njihova polja, reference itd. Izgled profila I njegov klasni sadrzaj dat je prikazan je na slici 3.1.




*Slika 3.1 Klase sadrzane u profilu za projekat 47*

Kada je zavrsen ovaj korak,  generisan je RDFS fajl koji predstavlja prosirenje modela

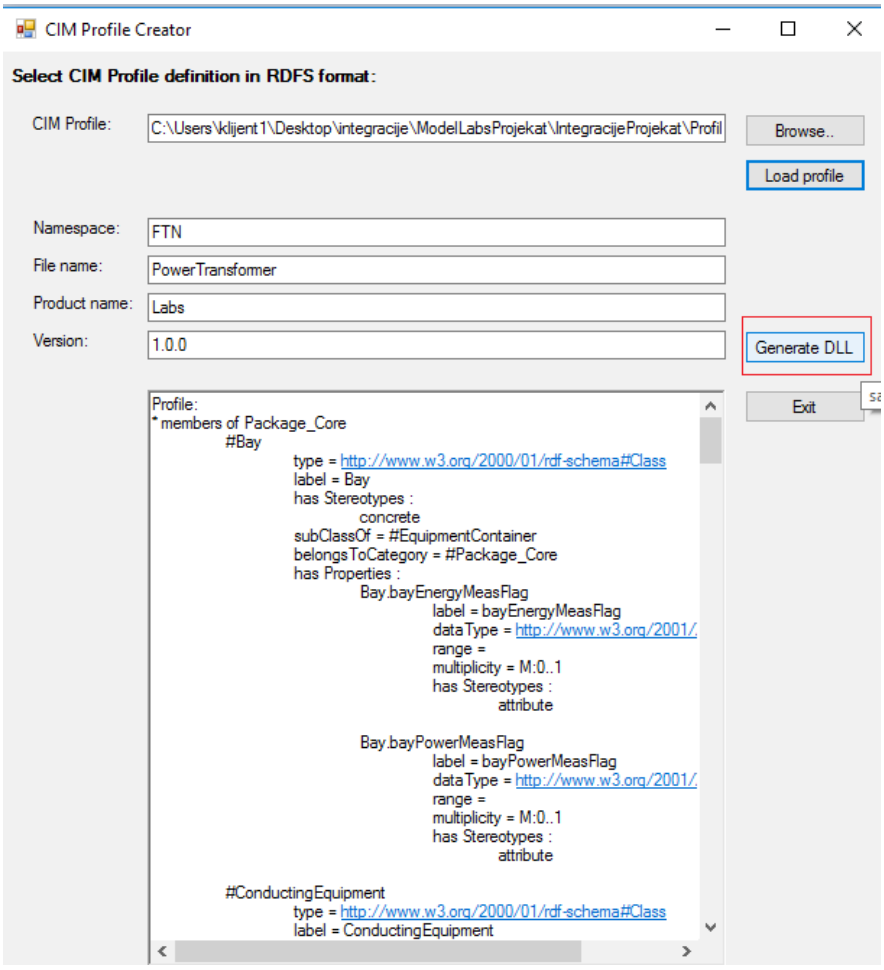
masinski citljivih meta-podataka. On predstavlja jedan od nekoliko zapisa CIM profila.

## 5. Generisanje .dll fajla

Menjamo okruzenje prelaskom u *VisualStudio.* Kada pokrenemo projekat *CIMProfileCreator* izacice nam Forma koja nudi generisnje *.dll* na osnovu prethodno generisanog rdfs fajla. Dakle, nadjemo prethodno generisani rdfs fajl I pritiskom na dugme *Generate DLL* nas .dll fajl bice generisan.

Nakon ovoga kreirani .dll treba ukljuciti u projekat gde je potreban, a to je u ovom projektu *CIMAdapter* u folder *Lib.* Razlog je taj sto *CIMAdapter* predstavlja spregu izmedju *CIM* profila i objektnog modela., odnosno za konverziju podataka iz *CIM/XML* datoteke u format čitljiv

*Network Model* servisu.




*Slika 4.1 Generisanje .dll fajla*

## 7. DEFINISANJE MODEL KODOVA

Na osnovu kreirane DLL biblioteke, potrebno je preslikati dobijeni model u aplikaciju. Veoma vazan segment u izradi projekta je upavo pisanje model kodova. Svaki tip resursa u modelu se jednoznačno identifikuje odgovarajucim model kodom, svaka klasa i njen atribut. Svaka cifra u model kodu ima svoj razlog i smisao. Odredjena semanticka pravila nas usmeravaju kako treba da definisemo model kodove, i veoma je vazno da se drzimo tih pravila, jer u kasnijem delu izrade projekta to moze prouzrokovati velike probleme usled neprepoznavanja resursa.

Pre definisanja model kodova, potrebno je jedinstveno obeležiti sve konkretne klase u

modelu. Ovo se postiže dodelom *DMSType* oznake svakoj konkretnoj klasi u modelu. *DMSType* predstavlja enumeraciju tipa *short*. U tabeli 2.1. je prikazan spisak *DMSType* oznaka za sve klase projektnog zadatka.

| ***DMSType*** | ***ModelCode*** |
|---|---|
| Bay | 0x0001 |
| SeriesCompensator | 0x0002 |
| ACLineSegment | 0x0003 |
| DCLineSegment | 0x0004 |
| ConnectivityNode | 0x0005 |
| Terminal | 0x0006 |

*Tabela 2.1. Spisak DMSType oznaka za sve konkretne klase projektnog zadatka 47.*

Nakon definisanja *DMSType* oznaka moguće je pristupiti definisanju model kodova za sve klase i njihove atribute. Model kodovi su predstavljeni ModelCode enumeracijom tipa long dužine 64 bita.

| ***Klasa*** | ***Naziv Model Code-a*** | ***ModelCode*** |
|---|---|---|
| *IdentifiedObject* | IDOBJ | 0x1000000000000000 |
| | IDOBJ_GID | 0x1000000000000104 |
| | IDOBJ_ALIAS | 0x1000000000000207 |
| | IDOBJ_MRID | 0x1000000000000307 |
| | IDOBJ_NAME | 0x1000000000000407 |
| *PowerSystemResource* | PSR | 0x1100000000000000 |
| *ConnectivityNode* | CNODE | 0X1200000000050000 |
| | CNODE_DESC | 0X1200000000050107 |

| | | |
|---|---|---|
| | CNODE_TERMINALS | 0X1200000000050219 |
| | CNODE_CNC | 0X1200000000050309 |
| *Equipment* | EQUIPMENT | 0X1110000000000000 |
| | EQUIPMENT_EC | 0X1110000000000109 |
| | EQUIPMENT_AGGREGATE | 0X1110000000000201 |
| | EQUIPMENT_NORMSVC | 0X1110000000000301 |
| *ConnectivityNodeContainer* | CNCONTAINER | 0X1120000000000000 |
| | CNCONTAINER_CNS | 0X1120000000000119 |
| *ConductingEquipment* | CONDEQUIPMENT | 0X1111000000000000 |
| | CONDEQUIP_TERMS | 0X1111000000000119 |
| *SeriesCompensator* | SERIESCOMPENSATOR | 0X1111100000020000 |
| | SERIESCOMPENSATOR_R | 0X1111100000020105 |
| | SERIESCOMPENSATOR_R0 | 0X1111100000020205 |
| | SERIESCOMPENSATOR_X | 0X1111100000020305 |
| | SERIESCOMPENSATOR_X0 | 0X1111100000020405 |
| *Conductor* | CONDUCTOR | 0X1111200000000000 |
| | CONDUCTOR_LENGTH | 0X1111200000000105 |
| *EquipmentContainer* | EQUIPMENTCONTAINER | 0X1121000000000000 |
| | EQUIPMENTCON_EQUIPS | 0X1121000000000119 |
| *Bay* | BAY | 0X1121100000010000 |
| | BAY_ENERGYFLAG | 0X1121100000010101 |
| | BAY_POWERFLAG | 0X1121100000010201 |
| *DCLineSegment* | DCLINESEGMENT | 0X1111210000040000 |
| | DCLINES_INDUTANCE | 0X1111210000040105 |
| | DCLINES_RESISTANCE | 0X1111210000040205 |

| | | |
|---|---|---|
| *ACLineSegmennt* | ACLINESEGMENT | 0X1111220000030000 |
| | ACLINESEGMENT_B0CH | 0X1111220000030105 |
| | ACLINESEGMENT_BCH | 0X1111220000030205 |
| | ACLINESEGMENT_G0CH | 0X1111220000030305 |
| | ACLINESEGMENT_GCH | 0X1111220000030405 |
| | ACLINESEGMENT_R | 0X1111220000030505 |
| | ACLINESEGMENT_R0 | 0X1111220000030605 |
| | ACLINESEGMENT_X | 0X1111220000030705 |
| *Terminal* | TERMINAL | 0X1300000000060000 |
| | TERMINAL_CE | 0X1300000000060109 |
| | TERMINAL_CN | 0X1300000000060209 |
| | TERMINAL_CONNECTED | 0X1300000000060301 |
| | TERMINAL_PHASES | 0X130000000006040A |
| | TERMINAL_SEQNUM | 0X1300000000060503 |

*Tabela 2.2 Spisak model kodova klasa i atributa definisanih u sklopu izrade projektnog zadatka*

# 8. RAZVOJ IMPORTERA NA OSNOVU DEFINISANE RDFS ŠEME I TESTIRANJE SA TESTNIM CIM/XML FAJLOM

Nakon uspešnog kreiranja *DLL* biblioteke koja nosi podatke o objektnom modelu, ispunjeni su svi uslovi za razvoj importera, odnosno adaptera. Podaci između adaptera i servisa se razmenjuju putem delta objekata. Delta objekat sadrži kolekcije operacija dodavanja, izmene i brisanja elemenata. Nakon kreiranja delta objekta, kreirani objekat se prosleđuje servisu preko *Generic Data Access (GDA)* interfejsa. *GDA* standard omogućava pristup podacima bez ikakvog znanja o logičkoj strukturi podatka. Ovim standardom je propisano da se bilo koja klasa i njeni atributi iz internog modela predstavljaju u formi resursa.
Resursi u ovom projektnom zadatku su modelovani klasom *ResurceDescription* koja sadrži jedinstveni identifikator. Svaki resurs može biti dodatno opisan atributima. Atributi resursa koji se opisuje je u ovom projektnom zadatku modelovan klasom Property koja se sastoji iz dva dela, identifikatora i vrednosti. Svaki property je jednoznačno određen preko model koda dok

vrednost sadrži samu vrednost atributa.

Ulazna CIM/XML datoteka za CIM importer aplikaciju je formatirana u skladu sa Resource Description Framework-om. U ovoj datoteci su RDF sintaksom definisane konkretne instance objekata klasa definisanih specifikacijom predmetnog zadatka. Jedinstveni identifikator resursa je obeležen rdf:ID atributom i naveden je prvi. Nakon identifikatora slede ostali atributi. Ukoliko je neki atribut referenca na drugi objekat, obeležen je atributom rdf:resource koji kao vrednost sadrži identifikator referenciranog elementa. Na slici 5.1 je prikazan deo CIM/XML fajla sa testnim podacima.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:cim="http://iec.ch/TC57/2010/CIM-schema-cim15#"
    xmlns:ftn="http://www.ftnydro.com/CIM15/2010/extension#">
    <!--bay-->
    <cim:Bay rdf:ID="BAY_01">
        <cim:Bay.bayEnergyMeasFlag>true</cim:Bay.bayEnergyMeasFlag>
        <cim:Bay.bayPowerMeasFlag>true</cim:Bay.bayPowerMeasFlag>
        <cim:IdentifiedObject.aliasName>bay1</cim:IdentifiedObject.aliasName>
        <cim:IdentifiedObject.mRID>BAY_01</cim:IdentifiedObject.mRID>
        <cim:IdentifiedObject.name>name_bay1</cim:IdentifiedObject.name>
    </cim:Bay>
    <cim:Bay rdf:ID="BAY_02">
        <cim:Bay.bayEnergyMeasFlag>false</cim:Bay.bayEnergyMeasFlag>
        <cim:Bay.bayPowerMeasFlag>true</cim:Bay.bayPowerMeasFlag>
        <cim:IdentifiedObject.aliasName>bay1</cim:IdentifiedObject.aliasName>
        <cim:IdentifiedObject.mRID>BAY_02</cim:IdentifiedObject.mRID>
        <cim:IdentifiedObject.name>name_bay2</cim:IdentifiedObject.name>
    </cim:Bay>
<!--seriesCompensator-->
    <cim:SeriesCompensator rdf:ID="SC_01">
        <cim:SeriesCompensator.r>33.2</cim:SeriesCompensator.r>
        <cim:SeriesCompensator.r0>11.2</cim:SeriesCompensator.r0>
        <cim:SeriesCompensator.x>32.321</cim:SeriesCompensator.x>
        <cim:SeriesCompensator.x0>76.3</cim:SeriesCompensator.x0>
        <cim:Equipment.aggregate>true</cim:Equipment.aggregate>
        <cim:Equipment.EquipmentContainer rdf:resource="#BAY_01"/>
        <cim:Equipment.normallyInService>true</cim:Equipment.normallyInService>
        <cim:IdentifiedObject.aliasName>sc_01</cim:IdentifiedObject.aliasName>
        <cim:IdentifiedObject.mRID>SC_01</cim:IdentifiedObject.mRID>
        <cim:IdentifiedObject.name>name_sc01</cim:IdentifiedObject.name>
        </cim:SeriesCompensator>
```

Slika 5.1. CIM/XML fajl sa testnim podacima

CIM importer aplikacija parsira učitanu CIM/XML datoteku i dobijene vrednosti mapira na

*ConcreteModel* instance klasa prethodno generisane *DLL* biblioteke. Kada se podaci učitaju i kada se kreiraju *ConcreteModel* instance, podatke je potrebno transformisati u oblik pogodan za slanje *Network Model* servisu. Transformacija podataka se vrši u klasi PowerTransformerConverter izvršavanjem metode *CreateNMSDelta*, koja poziva metodu *ConvertModelAndPopulateDelta* koja je prikazana na slici 5.2. U ovoj metodi se pozivaju metode za sve konkretne kalse koje preuzimaju sve objekte odgovarajućeg tipa iz modela i od svakog objekta kreira instancu klase *ResourceDescription* pozivom metode *CreateResourceDesctiption* koja kao ulazne parametre prima *DMSType* oznaku i objekat od kog treba kreirati instancu klase *ResourceDescription*. Za svaki resurs je potrebno obezbediti postojanje jedinstvenog globalnog identifikatora (GID – Global ID) koji se kreira pozivanjem odgovarajuće statičke metode klase *ModelCodeHelper*. Metodom *DefineIDMaping* klase *ImportHelper* se vrši mapiranje identifikatora iz ulazne CIM/XML datoteke na novi globalni identifikator. Poslednji korak konverzije je poziv metode *PopulateProperties* klase PowerTransformerConverter. Svrha ove metode je preslikavanje svih propertija iz CIM/XML datoteke na model kodove. Za svaku konkretnu klasu se prvo pozivaju metode za preslikavanje propertija roditeljske klase dok se ne stigne do vrha hijerarhije odnosno poziva metode *PopulateIdentifiedObjectProperties*.

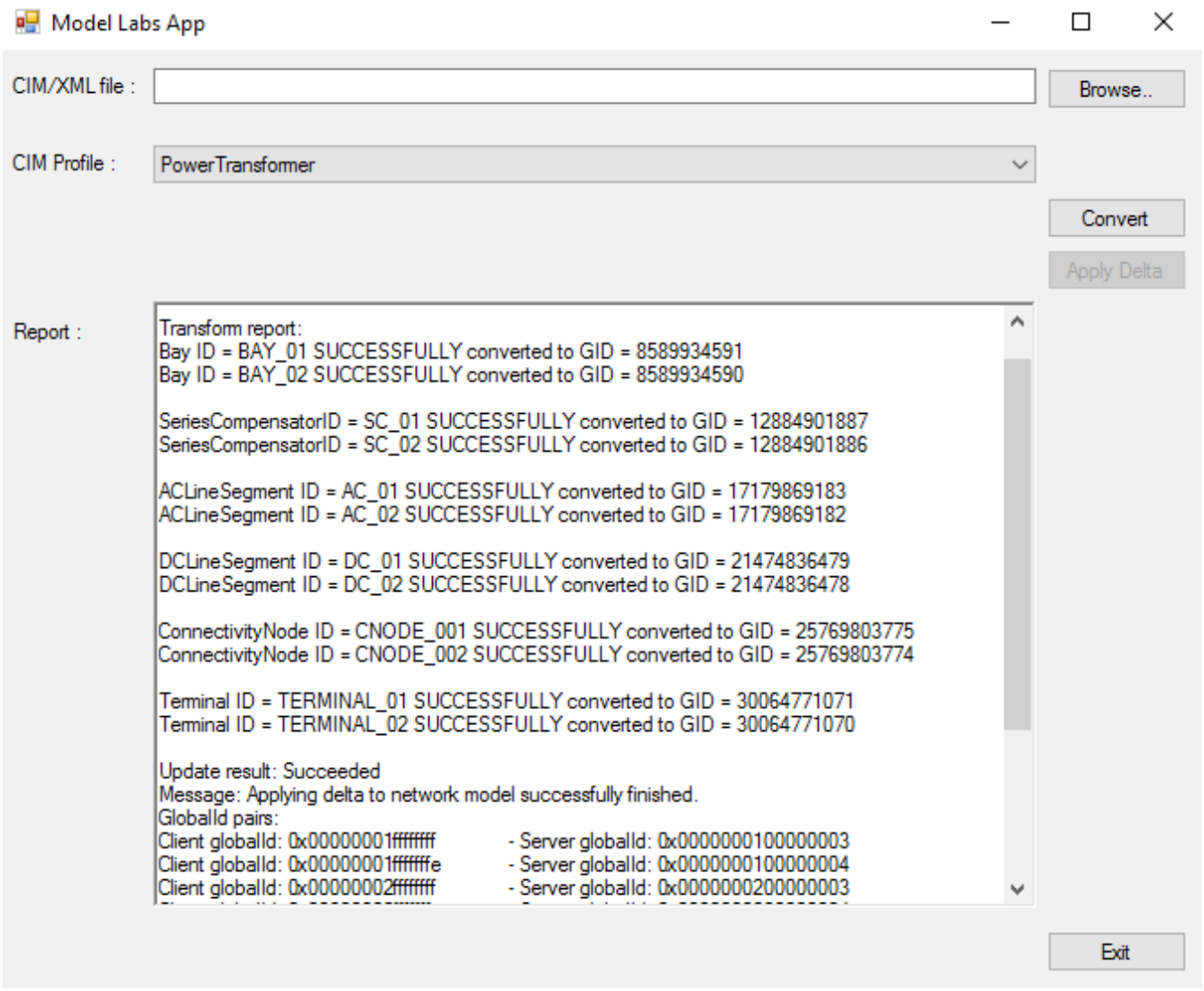
```
private void ConvertModelAndPopulateDelta()
{
    LogManager.Log("Loading elements and creating delta...", LogLevel.Info);

    //// import all concrete model types (DMSType enum)//DA LI MORA REDOM?-mora
    ImportBay();
    ImportSeriesCompensator();
    ImportACLineSegment();
    ImportDCLineSegment();
    ImportConnectivityNode();
    ImportTerminal();


    LogManager.Log("Loading elements and creating delta completed.", LogLevel.Info);
}
```

*Slika 5.2. ConvertModelAndPopulateDelta metoda*

Testiranje konverzije podataka i primene delte na *Network Model Service* se vrši pomoću *ModelLabsApp* aplikacije. Na slici 5.3 je prikazan rezultat uspešnog izvršavanja *ModelLabsApp* aplikacije. Bira se putanja do *CIM/XML* fajla i zatim se poziva konverzija i primena delte na *Network Model*. U aplikaciji se ispisuju rezultati konverzije podataka iz *CIM/XML* fajla u *ConcreteModel*, odnosno konverzije *rdf:ID u GID*..



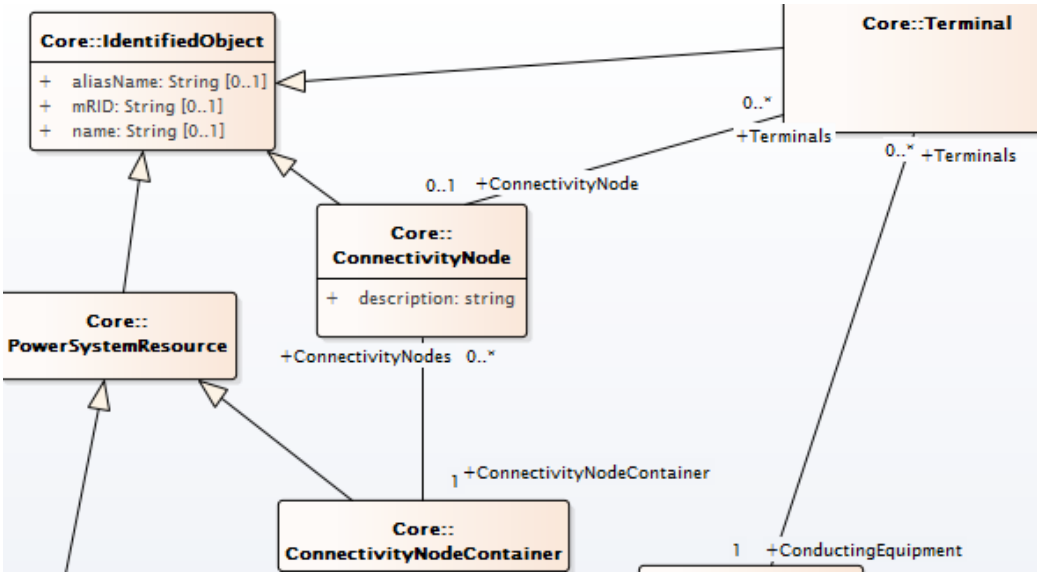

*Slika 5.3. Uspešna konverzija i primena delte*

# 9. RAZVOJ MODEL SERVERA

Razvoj model servera obuhvata nekoliko koraka. Prvi korak razvoja predstavlja proširenje *Common* komponente sistema koja je referencirana od ostalih komponenti. U klasi *ModelDefines* potrebno je definisati sve model kodove i *DMS* tipove, a u klasi *Enums* sve potrebne enumeracije. Nakon unesenih model kodova, *DMS* tipova i enumeracija, potrebno je u klasi *ModelResourceDesc* potrebno je dopuniti listu atributa koje klijentu nije dozvoljeno da

menja u metodi *InitializeNotSettablePropertyIds* i potrebno je definisati redosled kojim se elementi dodaju u model podataka u metodi *InitializeTypeIdsInInsertOrder*. Kada se izmeni *Common* komponenta sistema, može se pristupiti implementaciji odgovarajućih klasa *DMS* modela na *Network Model* servisu. Postupak implementacije klasa DMS modela će biti objašnjen na primeru klase *ConnectivityNode.*

Za prikaz je zgodno koristiti UML dizajnirani model prikazan na slici 6.1, kako bi olaksali sebi dalji rad iz razloga sto je definisanje klasa na osnovu pomatranja model kodova veoma nerazumno i zbunjujuce.




*Slika 6.1 Prikaz klase ConnectivityNode*

Sa slike 6.1 se vidi da je ovo konkretna klasa koja nasledjuje *IdentifiedObject* klasu, ima atribut *description* koji je tipa *string ,* referencu na klasu *ConnectivityNodeContainer* i atribute nasledjene klase. “Citanje” ovih informacija iz model kodova je veoma nezgodno za programera.

Postupak implementacije klasa *DMS* modela će biti objašnjen na primeru klase *ConnectivityNode*. Na slici 6.2 je prikazana lista atributa i referenci koje ova klasa sadrži, a definisani su projektnim zadatkom, kao i konstruktor koji prima *GID* elementa. Atributi koji

predstavljaju reference su tipa long i sadrže *GID*-ove referenciranih objekata.

```
public class ConnectivityNode : IdentifiedObject
{

    private List<long> terminals1 = new List<long>();

    private string description = string.Empty;

    private long connNodeContainer = 0;



    public ConnectivityNode(long globalId)
        : base(globalId)
    {
    }
```

*Slika 6.2 Atributi i konstruktor klase ConnectivityNode*

Da bi rad sa atributima klase bio moguć potrebno je implementirati *HasProperty*, *GetProperty* i *SetProperty* metode koje su detaljnije opisane u tabeli 3.1.

| **Metoda** | **Opis** |
|---|---|
| *HasProperty* (slika 6.3) | Proverava da li je prosleđeni *ModelCode* odgovara nekom od atributa klase. Prvo se proveravaju atributi tekuće klase, a zatim se poziva provera atributa roditeljske klase. |
| *GetProperty* (slika 6.4) | Konvertuje atribute klase u *Property* objekte koje zahteva *GDA* standard. Proverava da li prosleđeni *ModelCode* odgovara nekom od atributa tekuće klase, i ukoliko odgovara pozivom metode *SetValue* smešta njegovu vrednost u *Property*. Ukoliko nije u pitanju neki od atributa tekuće klase poziva se *GetProperty* metoda roditeljske klase. |
| *SetProperty* (slika 6.5) | Postavlja vrednosti atributa klase na osnovu *Property* objekata koje zahteva *GDA* standard. Proverava da li prosleđeni *ModelCode property*-ja odgovara nekom od atributa tekuće klase, i ukoliko odgovara smešta vrednost iz *Property*-ja u atribut. Ukoliko nije u pitanju neki od atributa tekuće klase poziva se *SetProperty* metoda roditeljske klase. Važno je napomenuti da ove metoda ne sme da dozvoli postavljanje vrednosti atributa definisanih u klasi *ModelResourcesDescs* u skupu *NotSettablePropertyIds*. |

*Tabela 3.1. Spisak metoda za konverziju atributa klase u Property objekte i obrnuto*

```
public override bool HasProperty(ModelCode t)
{
    switch (t)
    {
        case ModelCode.CNODE_CNC:
        case ModelCode.CNODE_DESC:
        case ModelCode.CNODE_TERMINALS:

            return true;

        default:
            return base.HasProperty(t);

    }
}
```

*Slika 6.3  Implementacija HasProperty metode u klasi ConnectivityNode*

```
public override void GetProperty(Property property)
{
    switch (property.Id)
    {
        case ModelCode.CNODE_CNC:
            property.SetValue(connNodeContainer);
            break;

        case ModelCode.CNODE_DESC:
            property.SetValue(description);
            break;

        case ModelCode.CNODE_TERMINALS:
            property.SetValue(terminals1);
            break;



        default:
            base.GetProperty(property);
            break;
    }
}
```

*Slika 6.4 Implementacija GetProperty metode u klasi ConnectivityNode*

```
public override void SetProperty(Property property)
{
    switch (property.Id)
    {
        case ModelCode.CNODE_CNC:
            connNodeContainer = property.AsReference();
            break;

        case ModelCode.CNODE_DESC:
            description = property.AsString();
            break;


        default:
            base.SetProperty(property);
            break;
    }
}
```

*Slika 6.5 Implementacija SetProperty metode u klasi ConnectivityNode*

Da bi rad sa referencama bio moguć u klasi je potrebno implementirati sledeće metode: *IsReferenced*, *GetReferences*, *AddReference* i *RemoveReference*. Detaljniji opis ovih metoda je dat u tabeli 3.2.

| **Metoda** | **Opis** |
|---|---|
| *IsReferenced* (slika 6.6) | Proverava da li postoji neki entitet koji referencira ovu instancu klase. Ovu metodu implementiraju samo klase koje imaju listu referenci (*target*-a) među atributima. Povratna vrednost ove metode je indikacija da li neka od listi referenci među |

| | |
|---|---|
| | atributima sadrži bar jedan *GID*. Server poziva ovu metodu kada dobije zahtev za brisanjem ovog objekta, i ukoliko je referenciran od strane nekog drugog entiteta u sistemu zahtev za brisanje se odbija. |
| *GetReferences* (slika 6.7) | Služi za čitanje referenci objekta. Vraća mapu čiji ključ je *ModelCode* odgovarajuće reference, a vrednosti u mapi su liste *GID*-ova referenciranih elemenata. |
| *AddReference* (slika 6.8) | Ovu metodu implementiraju samo klase koje imaju listu referenci (*target*-a) među atributima. Metodu koristi server kako bi dodao odgovarajuću referencu u listu referenci (*target*-a). |
| *RemoveReference* (slika 6.9) | Ovu metodu implementiraju samo klase koje imaju listu referenci (*target*-a) među atributima. Metodu koristi server kako bi uklonio odgovarajuću referencu iz liste referenci (*target*-a). |

*Tabela 3.2. Spisak metoda za rad sa referencama.*

```
public override bool IsReferenced
{
    get
    {
        return terminals1.Count != 0 || base.IsReferenced;
    }
}
```

*Slika 6.6 Implementacija metode IsReferenced za klasu ConnectivityNode*

```
public override void GetReferences(Dictionary<ModelCode, List<long>> references, TypeOfReference refType)
{
    if (connNodeContainer != 0 && (refType == TypeOfReference.Reference || refType == TypeOfReference.Both))
    {
        references[ModelCode.CNODE_CNC] = new List<long>();
        references[ModelCode.CNODE_CNC].Add(connNodeContainer);
    }

    if (terminals1 != null && terminals1.Count != 0 && (refType == TypeOfReference.Target || refType == TypeOfReference.Both))
    {
        references[ModelCode.CNODE_TERMINALS] = terminals1.GetRange(0, terminals1.Count);
    }

    base.GetReferences(references, refType);
}
```

*Slika 6.7 Implementacija metode GetReferences za klasu ConnectivityNode*

```
public override void AddReference(ModelCode referenceId, long globalId)
{
    switch (referenceId)
    {
        case ModelCode.TERMINAL_CN:
            terminals1.Add(globalId);
            break;


        default:
            base.AddReference(referenceId, globalId);
            break;
    }
}
```

*Slika 6.8 Implementacija metode AddReference za klasu ConnectivityNode*

```
public override void RemoveReference(ModelCode referenceId, long globalId)
{
    switch (referenceId)
    {
        case ModelCode.TERMINAL_CN:

            if (terminals1.Contains(globalId))
            {
                terminals1.Remove(globalId);
            }
            else
            {
                CommonTrace.WriteTrace(CommonTrace.TraceWarning, "Entity (GID = 0x{0:x16}) doesn't contain reference 0x{1:x16}.", t
            }

            break;

        default:
            base.RemoveReference(referenceId, globalId);
            break;
    }
}
```

*Slika 6.9 Implementacija metode RemoveReference za klasu ConnectivityNode*

Na kraju implementacije klase *ConnectivityNode* potrebno je redefinisati *Equals* metodu preklapanjem tako da je njena povratna vrednost *true* ako su objektima koji se porede globalni identifikatori i svi propertiji isti. Implementacija ove metode je prikazana na slici 6.10.

```
public override bool Equals(object obj)
{
    if (base.Equals(obj))
    {
        ConnectivityNode x = (ConnectivityNode)obj;
        return (x.Description == this.Description && x.ConnNodeContainer == this.ConnNodeContainer &&
                CompareHelper.CompareLists(x.Terminals1, this.Terminals1));
    }
    else
    {
        return false;
    }
}
```

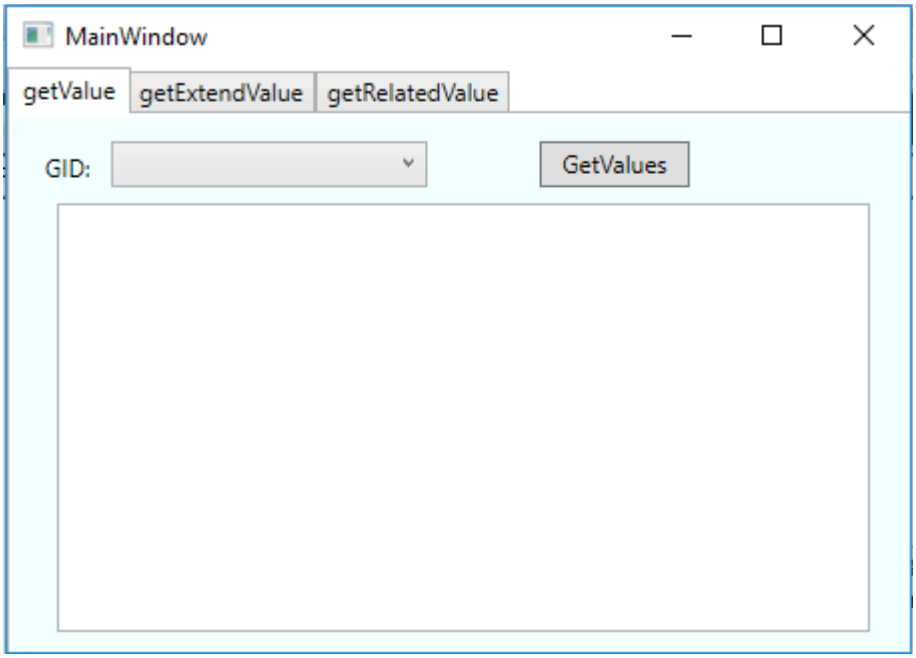
*Slika 6.10 Impementacija Equals metode*

Navedeni postupak je ponovljen pri implementaciji svih klasa *DMS* modela, sa odgovarajućim atributima i model kodovima uz poštovanje gorepomenutih pravila o

manipulaciji referencama.

# 10. RAZVOJ KLIJENTSKE APLIKACIJE

Nakon upesne implementacije prethodno opisanih delova predmetnog projekta moguće je pristupiti testiranju sistema, odnosno njegovom korišćenju pomoću jednostavne klijentske aplikacije *UI* koja je prikazana na slici 7.1.



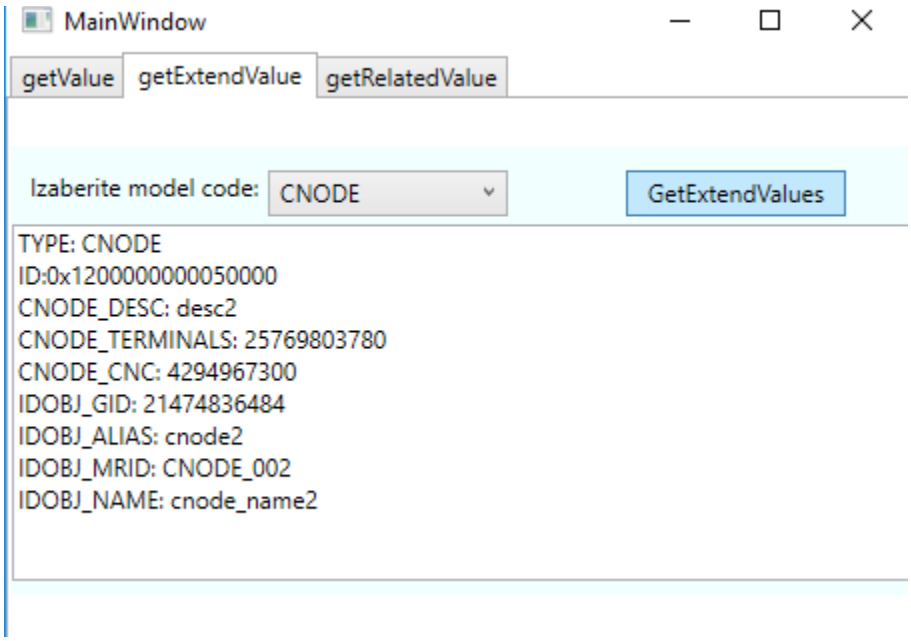

*Slika 7.1 Klijentska aplikacija UI*

Pri pokretanju *UI* aplikacije pokreće se *Network Model* server – aplikacija *NetworkModelServiceSelfHost*. *UI* putem *WCF*-a (*Windows Communication Foundation*) omogućava klijent-server *TCP* (*Transmission Control Protocol*) komunikaciju ove dve aplikacije. Klijentska aplikacija nudi opcije konvezije podataka *CIM/XML* datoteke u *DMS* model, kao I opcije za kreiranje i slanje pomenutih *GDA* upita *Network Model* serveru, a rezultati ovih upita se čuvaju kao *XML* datoteke. U tabeli 4.1. je prikazana i detaljnije opisana lista operacija koje nudi *INetworkModelGDAContract* interfejs.

| Metoda | Opis |
|---|---|
| *ApplyUpdate* | Primenjuje delta objekte na model podataka. Poziva se iz *CIM* adaptera prilikom primene kreiranih delti. Vraća informaciju o rezultatu primene delte. |
| *GetValues* | Pruža mogućnost preuzimanja *ResourceDescription*-a elementa sa željenim globalnim identifikatorom i dobavljanje *property*-ja tog elementa. |
| *GetExtentValues* | Pruža mogućnost pristupa listi *ResourceDescription*-a elemenata željenog tipa.i dobavljanja *property*-ja za svaki od elemenata. Vraća identifikator upita na osnovu koga klijent može da pristupi rezultatima upita. |
| *GetRelatedValues* | Omogućava pristup informacijama o referenciranom resursu. Prima *source* (*GID* elementa koji referencira željene elemente), listu željenih *property*-ja rezultujećeg resursa i *Association* objekat koji predstavlja vezu između *source* elementa i rezultujućih resursa. Vraća identifikator upita na osnovu koga klijent može da pristupi rezultatima upita. |
| *GetDescendantValues* | Dobavlja informacije o nizu referenciranih resursa. Prihvata niz *source* elemenata, listu željenih *property*-ja rezultujećih resursa i niz *Association* objekata koji predstavljaju vezu između *source* elemenata i rezultujućih resursa. Vraća identifikator upita na osnovu koga klijent može da pristupi rezultatima upita. |

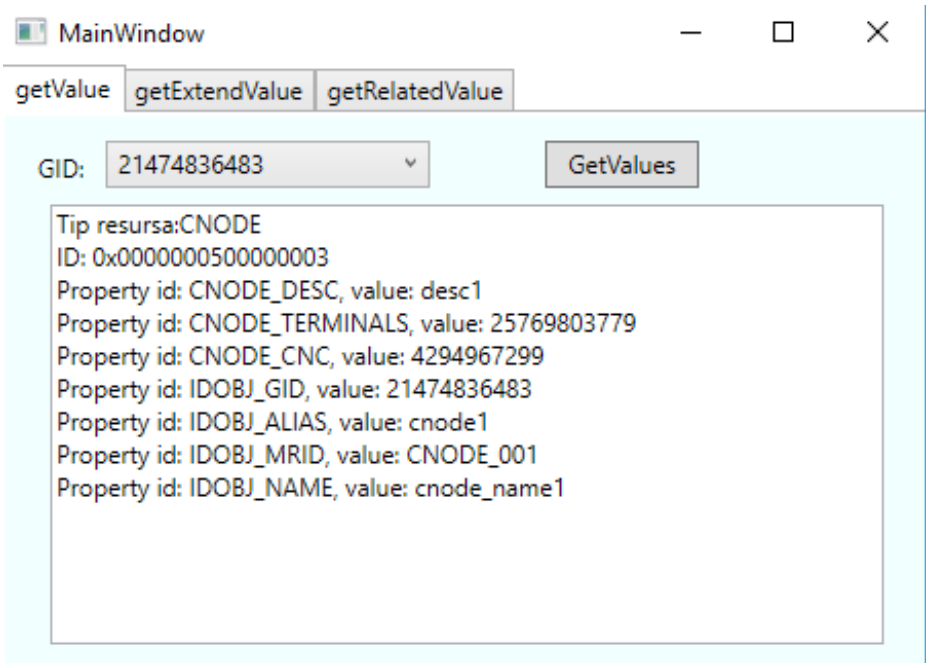
*Tabela 4.1. Spisak operacija INetworkModelGDAContract interfejsa.*

Na slici 7.2. je prikazan rezultat *GetExtentValues* upita kojem je kao parameter prosleđen *ModelCode* CNODE. Rezultat upita predstavljaju podaci o svim instancama klase *ConnectivityNode* u sistemu i *XML* fajl.




*Slika 7.2. Rezultati GetVExtendalues upita*

Pomoću metode *GetValues* čitaju podaci o resursu sa zeljenim *GIDom*. U prikazanom primeru je kao argument ovoj funkciji prosleđena vrednost globalnog identifikatora **0x0000000500000003**, a kao rezultat je dobijen željeni objekat klase *ConnectivityNode* sa svim svojim atributima (slika 7.3).




*Slika 7.3. Rezultati GetValues upita*