Elektrotehnički Fakultet Sarajevo
Odsjek za Računarstvo i Informatiku

Specijalna poglavlja softwareskih sistema

Septembar 2004

OPC SPECIFIKACIJE

UVOD

OPC (**O**bject **L**inking and **E**mbeding -**OLE** for Process Control) , tj Microsoft mehanizam implementiran u Windows Operativnim sistemima za razmjenu podataka izmedju aplikacija , definira standardne objekte , metode, i osobine za zadovoljenje zahtjeva interoperabilnosti tj. da mogu medjusobno komunicirati sa mnogo različitih tipova hardwarea koristeći isti OPC klijent software, u mnogim real-time procesnim aplikacijama.

OPC klijent djeluje takodjer kao drajver, i to omogućava procesnoj bazi podataka HMI sistema da dobije podatke od bilo kojeg OPC servera. Ova mogućnost dozvoljava HMI softwareu da komunicira sa bilo kojim tipom procesnog hardwarea za koji postoji OPC server i čini korisnikov izbor hardwarea sa kojim će raditi skoro neograničenim.

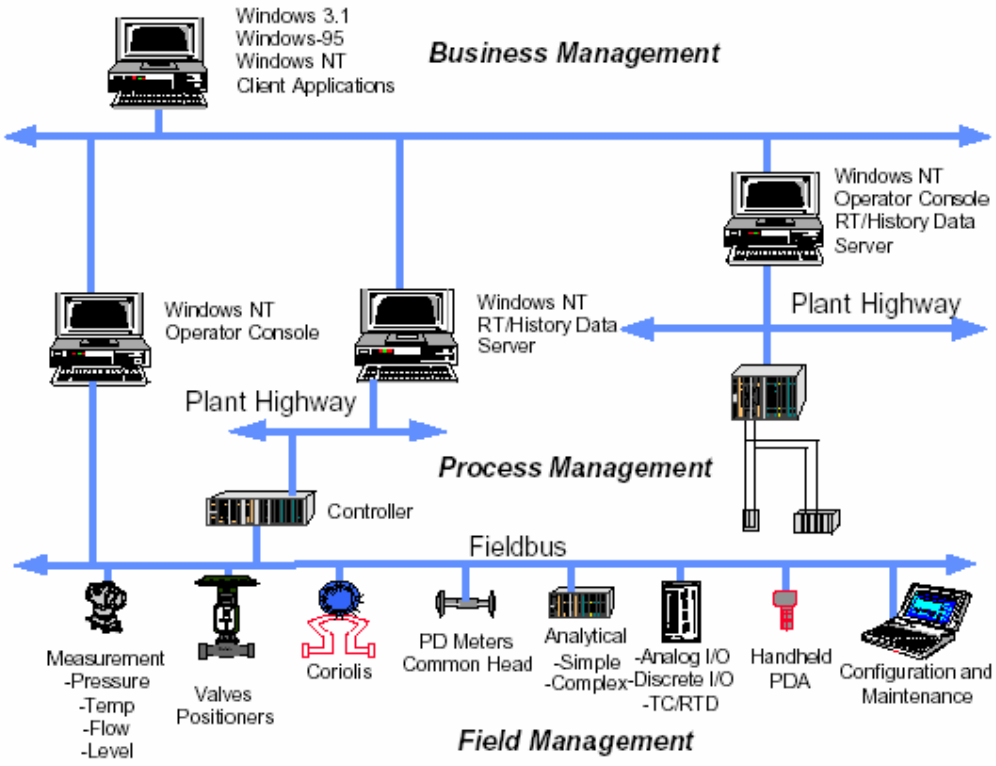
OPC - STANDARD ZA RAZMJENU PODATAKA UNUTAR SISTEMA UPRAVLJANJA I VODJENJA PROCESA i POSLOVNIH OBRADA

Rijetko kada bilo je toliko mnogo uzbudjenja u oblasti računarskog vodjenja u upravljanja procesima kao kada se pojavio OPC kao komunikacioni standard.

OPC je akronim za " **O**LE for **P**rocess **C**ontrol", tj. Microsoft Windows OLE (object linking and embedding) tehnologija (kasnije evoluirala u ActiveX), prilagodjena za procesno vodjenje i nadzor.

Standardni mehanizam komunikacije medju brojnim izvorima podataka, bilo uradjaja rasutih diljem proizvodnih pogona jedne fabrike, ili bazama podataka u računarima u kontrolnoj sobi ili u ofisima je osnovna motivacija za OPC.

Informaciona arhitektura u procesnoj industriji pokazana je na slijedećoj slici i uključuje slijedeće nivoe:



Arhitektura informacionog sistema upravljanja

Management u postrojenju (field management)

Sa uvodjenjem inteligentnih (smart) uredjaja u postrojenjima kao dijelova sistema vodjenja nadzora i upravljanja (inteligentni transmiteri, aktuatori, itd), pojavljuje se obilje informacija o tim uredjajima i o postrojenju gdje su ugradjeni, koji nisu prije bili raspoloživi. Ove informacije obezbjedjuju podatke o zdravlju uredjaja , njegovim konfiguracionim parametrima, konstrukcionim materijalima, itd. Sve ove informacije moraju se prikazati i korisniku, kao i bilo kojoj aplikaciji koja ih koristi, na konzistentan način.

Management procesa

Instaliranje distribuiranih sistema upravljanja (DCS) i SCADA sistema da nadziru i upravljaju procesima čine ove podatke raspoložive i u elektronskoj formi, za razliku od ranijih sistema kada su mnogi od njih bili ručno prikupljani i zapisivani.

Poslovni management

Beneficije u poslovnom vodjenju sistema su ogromne nakon instaliranja savremenog sistema upravljanja. Ovo se postiže integracijom informacija prikupljenih iz procesa u poslovne sisteme koji upravljaju finansijskim aspektima proizvodnih procesa.

Obezbjedjivanje ovih informacija na konzistentan način klijent aplikacijama , minimizira napor koji je potrebno uložiti da se obezbjedi ova integracija.

Da bi se efektivno relizovali ovi ciljevi, proizvodjači trebaju biti u stanju da koriste gotove (off the shelf) alate kao što su SCADA paketi, baze podataka, spreadsheetovi, itd.
Ključ ovoga je u otvorenoj i efikasnoj komunikacionoj arhitekturi koja se koncentrira na pristup podatcima, a ne na tipovima podataka.

Da bi ovi sistemi radili zajedno morali su se riješiti mnogi problemi koji su bili ozbiljniji nego što je to problem povezivanja mreža, rad sa različitim operativnim sistemima i povezivanje "*open systems*" koji su se u praksi pokazali da nisu baš tako otvoreni i koji su trebali svojom otvorenom arhitekturom da olakšaju to povezivanje.

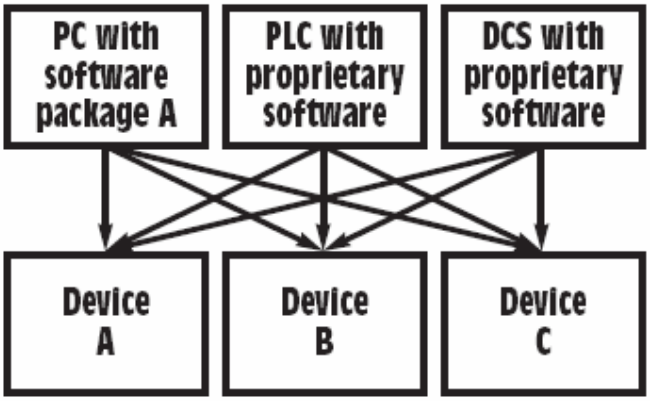
Osnovni razlog zašto su računarski sistemi vodjenja i nadzora procesa imali i svoje dodatne specifićne teškoće je u tome da interfejsi izmedju procesnih uredjaja i računarskih sistema nisu bili standardizirani.

Specifični za svakog proizvodjača (proprietary) sistemi koji medjusobno ne komuniciraju su bili standardna praksa sve do nedavno u mnogim industrijskim postrojenjima.

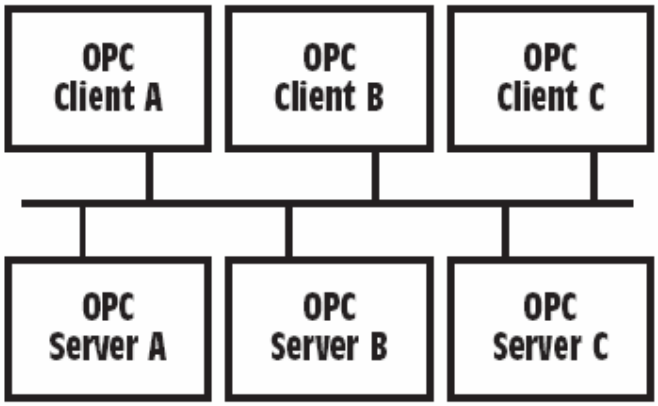
Usljed odsustva standarda u ovoj oblasti , proizvodjači (Vendors) su razvili specifična hardwaresko softwareska rješenja. Zbog toga mnogi sistemi procesnog i poslovnog upravljanja i vodjenja su imali svoje specifične tehnike, interfejse, API module , da bi se pristupalo informacijama u takvim sistemima.

Cijena integracije takvih sistema kao i troškovi softwareskog održavanja i računarska podrška su bili značajni.

Pojavljivanje drajvera za svaki specifični tip hardwarea i njihove izmjene pri svakoj modifikaciji hardwarea bile su glavobolja za sve integratore i održavaoce ovakvih sistema.
U takvim okolnostima , za tipičan sistem procesnog i poslovnog nadzora i vodjenja, 25-30% vremena IT inženjera je trošeno na razvoj i debagiranje drajverskih modula softwarea. Ovo stanje je ilustrirano i na slijedećoj slici :



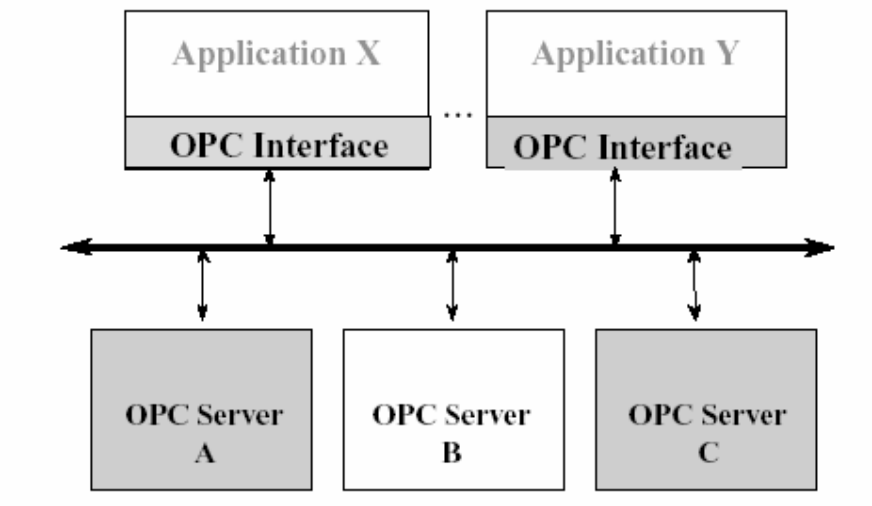
Rješenje za sve ovo je bio da se razvije standard koji će obezbjediti istinsku "plug-and-play" softwaresku tehnologiju za sisteme upravljanja i vodjenja procesa, gdje će svaki sistem, svaki uredjaj i svaki drajver moći se slobodno povezati i komunicirati medjusobno. Ova ideja je ilustrirana na narednoj slici :



Namjena

Ono što je potrebno je zajednički način pristupa podatcima od strane aplikacija ka bilo kojem izvoru podataka , bilo da je to neki uredjaj ili baza podataka.

Slijedeća slika pokazuje ovu ideju :



Aplikacije koje rade sa mnogo OPC Servera

OPC Server na ovoj slici i u slijedećim poglavljima je korišten kao sinonim za bilo koji server koji obezbjedjuje OPC interfejse , kao napr.

- OPC Server za pristup podatcima (OPC **D**ata**A**ccess Server)
- OPC Server za Alarme i dogadjaje (OPC **A**larm&**E**vent Server)
- OPC Server za historijske podatke (OPC **H**istorical**D**ata **A**ccess Server)

Ranija Arhitektura klijent aplikacija

Postoje mnoge klijent aplikacije koje su bile razvijene i koje zahtjevaju podatke iz izvora podataka i pristupaju tim podatcima nezavisno, razvijajući "drajvere" za svoje potrebe.

Ovo je vodilo ka nizu problema , kao :

- **Dupliranje napora** od mnogo nezavisnih proizvodjača softwarea, jer je svako za svoj software pisao svoj drajver za specifični tip hardwarea.
- **Nekonzistentnost izmedju drajvera** različitih Vendora (softwareskih kuća)
- **Promjena u hardwareu** je vodila najčešće do toga da se morao pisati novi drajverski software
- **Konflikti pristupa** Dva softwarea nisu mogli istovremeno pristupiti nekom hardwareu pošto su obadva sadržavala nezavisne drajvere.

Proizvodjači hardwarea su pokušavali da riješe ove probleme razvijajući drajvere za svoj hardware, ali su bili suočeni sa problemom različitih Klijent protokola.

OLE (**O**bject **L**inking and **E**mbedding) za upravljanje procesima (**OPC**), je povukao ovu liniju izmedju proizvodjača hadwarea i onih koji su razvijali software.

OLE je bio kasnije restruktuiran od objektno orijentirane na objektno baziranu arhitekturu i Microsoft mu je promjenio naziv u **ActiveX.**

OPC obezbjedjuje mehanizam da osigura podatke iz izvora podataka i prenosi ih ka bilo kojem klijentu na standardan način.

Proizvodjač hardwarea je mogao sada da razvije višekoristivi , visoko optimiziran Server, koji komunicira sa izvorom podataka, i održava mehanizam da efikasno pristupi podatcima iz uredjaja – izvora podataka.

Dakle, ako obezbjedimo Server sa OPC interfejsom, onda će bilo koji klijent moći pristupiti tom uredjaju i njegovim podatcima.

Kastomizirana aplikaciona arhitektura

Sve veći broj aplikacija se razvija u okružajima kao što su : VB i VBA (Visual basic for application), Delphi, Power Builder, Visual C ++. OPC takodjer mora da uzme u obzir ovaj trend u obzir. Microsoft je uočio ovaj trend i dizajnirao je **OLE/COM** da bi omogućio komponentama (pisanim u C i C ++) da se mogu koristiti od strane korisnikovog prilagodjenog (kastomiziranog) programa (pisanog u VB, VBA ili Delphi).

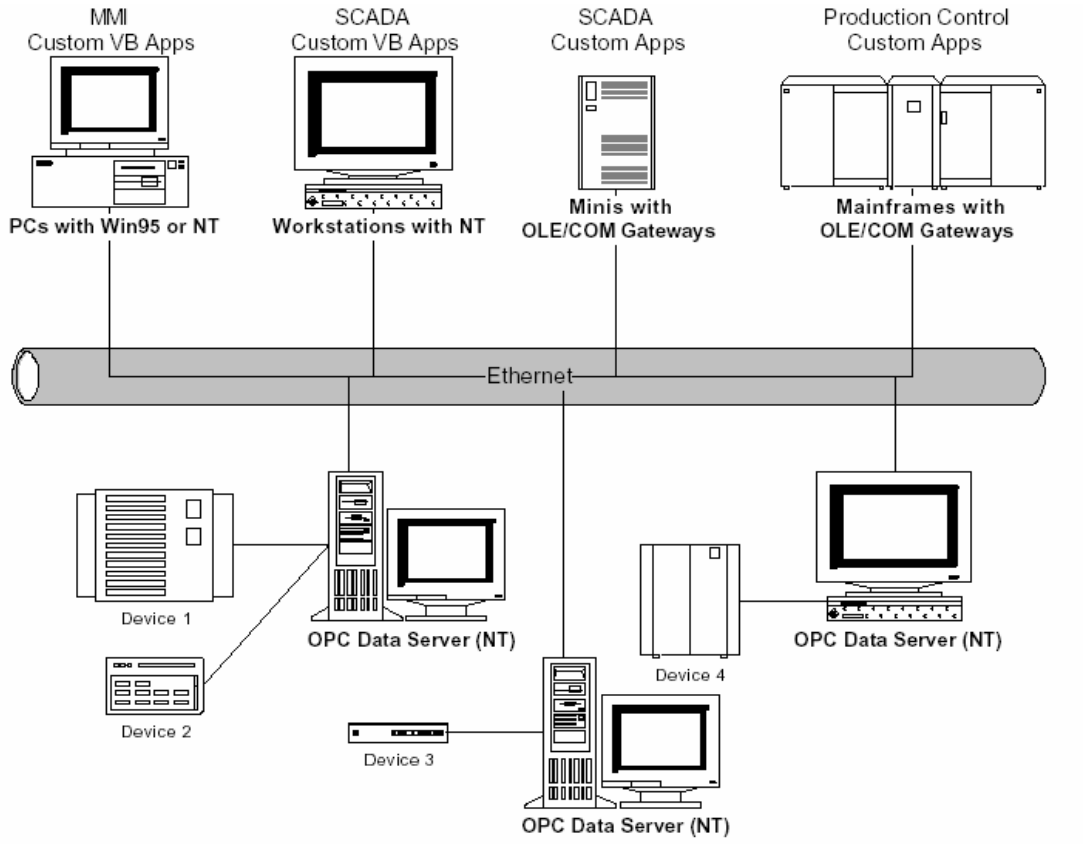
Dakle namjena svih standardnih specifikacija je da olakšaju razvoj OPC Servera u C i C ++ , a da olakšaju razvoj OPC klijent aplikacija pisanih u bilo kojem jeziku koji korisnik izabere.

OPŠTE

OPC je dizajniran da omogući klijentskim aplikacijama pristup procesnim podatcima na konzistentan način. OPC pruža slijedeće prednosti:

- Proizvodjači hardwarea treba samo da urade jedan set softwareskih komponenti koje će korisnici koristiti u njihovim aplikacijama.
- Proizvodjači softwareskih programa neće morati da uvjek ponovo pišu drajvere zato što je došlo do promjena u novoj verziji hardwarea.
- Korisnici će imati na raspolaganju više opcija hardwarea kod sistemske integracije svojih sistema.

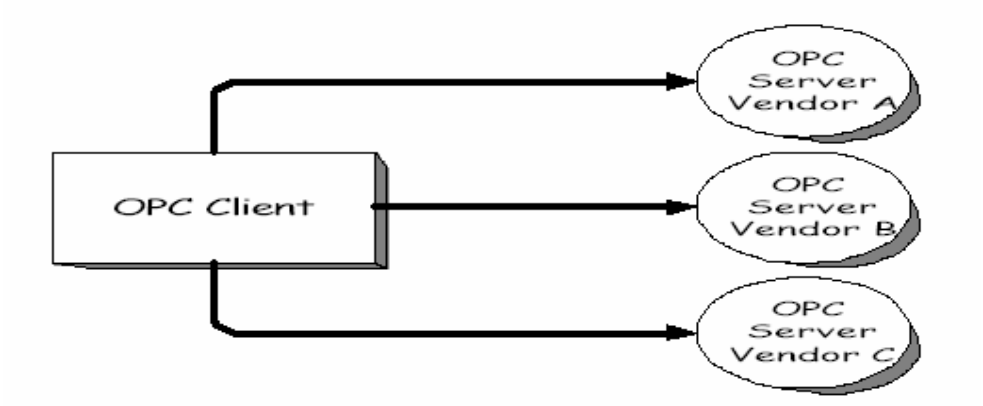
Sa OPC , integracija sistema u okružaju heterogenog kompjuterskog hardwarea će postati jednostavnija.



Heterogeni kompjuterski oktužaj

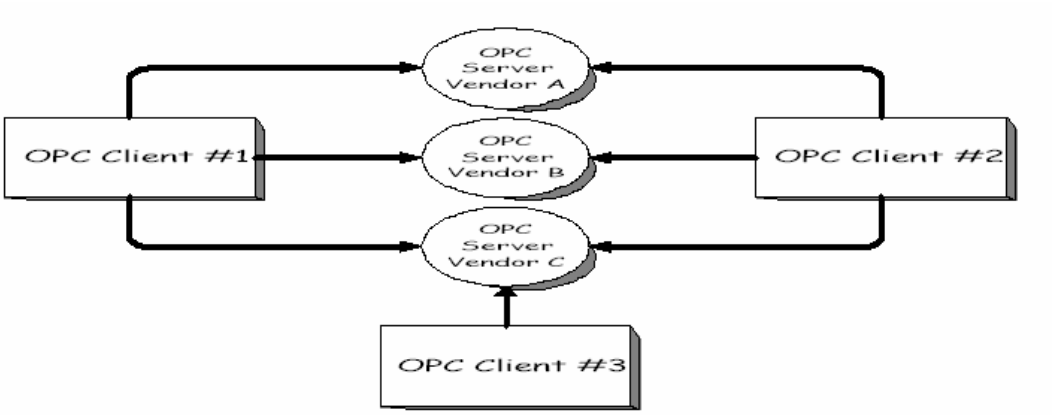
OPC objekti i interfejsi

Daćemo opis OPC COM objekata i njihovog interfejsa implementiranog sa OPC Serverima. OPC klijent se može povezati sa OPC Serverima različitih proizvodjača.



OPC Klijent

OPC Serveri mogu biti napisani od različitih proizvodjača hardwarea. Kôd kojeg isporučuje Vendor (proizvodjač hardwarea), odredjuje uredjaje i podatke do kojih Server ima pristup, kao i detalje kako Server fizički pristupa tim podatcima.



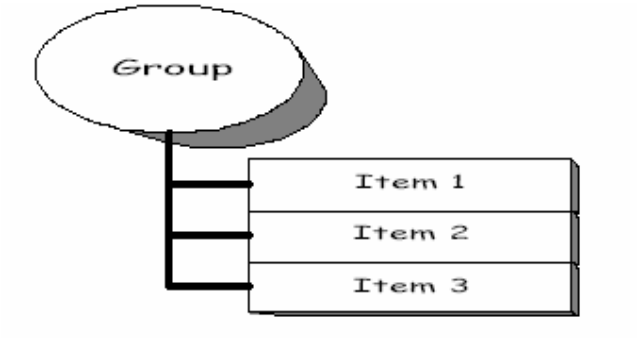
Relacija izmedju OPC Klijenta i Servera

Pregled OPC DataAccess funkcije

Na višem nivou, **OPC DataAccess** Server se sastoji od nekoliko objekata: , servera, grupe i detalja (item). Objekt OPC Server sadrži informaciju o serveru i služi kao kontejner za OPC grupne objekte. OPC group objekat sadrži informaciju o samom sebi i obezbjedjuje mehanizam za držanje i logičko organizovanje OPC itema.

OPC grupe obezbjedjuju način za klijente da organiziraju podatke. Naprimjer, grupa može predstavljati detalje (items) u nekom datom operatorskom prikazu ili izvještaju. Podatci mogu biti čitani ili upisivani. Konekcije bazirane na izuzeću (exception based), mogu se takodjer kreirati izmedju klijenta i detalja u grupi i mogu biti omogućene ili onemogućene. OPC klijent može konfigurirati brzinu sa kojom OPC Server treba obezbjedjivati promjene u podatcima za OPC klijenta.

Postoji dva tipa grupa, javne i lokalne (ili privatne). Javna (public) grupa se koristi ako se dijeli izmedju više klijenata, lokalna grupa je lokalna i samo za jednog klijenta.
Unutar svake Grupe , klijent može definirati jedan ili više OPC detalja (items).



Relacija izmedju Grupe i detalja

OPC detalji prestavljaju konekcije sa izvorima podataka unutar servera. OPC detalj, sa perspektive korisničkog interfejsa, nije pristupačan kao objekat od strane OPC klijenta. Zbog toga, nema vanjskih interfejsa definiranih za OPC Item. Svi pristupi ka OPC Itemima su preko OPC Group objekta, koji "sadrži" OPC item, ili jednostavno tamo gdje je OPC Item definiran.

Pridružena svakom Itemu je vrijednost (Value), kvalitet (quality) i vremenski otisak (Stamp) -[V,Q,T]. Vrijednost je u formi VARIANT, a kvalitet je sličnog tipa onome kako je ona specifirana i za Fieldbus (procesni bus koji povezuje u mrežu uredjaje u postrojenju).

Primjetimo da Itemi nisu izvori podataka, oni su samo konekcije sa njima. Naprimjer, tagovi u DCS sistemu postoje bez obzira da li OPC klijent im pristupa. Prema tome o OPC Itemu trebamo razmišljati kao samo o specifijeru adrese podatka, a ne kao stvarnom izvoru podatka, koji je referenciran adresom u Itemu.

Pregled *OPC A*larm i *E*vent handlera

Ovi interfejsi obezbjedjuju mehanizme za OPC klijente da budi izvješteni o pojavljivanju specificiranih dogadjaja i alarmnih uslova. Oni takodjer obezbjedjuju servise koji omogućavaju OPC klijentima da odrede dogadjaje i uslove koji su podržani od strane OPC Servera, i da dobiju njihov tekući status.

U okviru OPC, *alarm* je nenormalno stanje (*condition*) i zbog toga je specijalan slučaj stanja. Stanje (condition) je stanje (named state) u OPC Event Serveru, ili jednom od njegovih sadržavajućih objekata, koji je od interesa za OPC klijente. Naprimjer, tag FC101 može imati slijedeće uslove pridružene sa njim:

visoki alarm (high alarm), vrlo visoki alarm (highhigh alarm), niski alarm (low alarm), i vrlo niski alarm (lowlow alarm).

Sa druge strane, dogadjaj (event), je pojava koja je značajna za OPC Server, za uredjaj koji predstavlja, i za njegove OPC klijente. Dogadjaj (event) može ali i ne mora biti udružen sa uslovom. Naprimjer, prelasci u alarmna i normalna stanja su dogadjaji koji su udruženi sa stanjima. Sa druge strane,

promjene u konfiguraciji sistema, i sistemske greške su dogadjaji koji nisu vezani za specifične dogadjaje. OPC klijenti mogu se prijaviti da budu obavješteni od strane Servera o pojavi specificiranih dogadjaja.

IOPCEventServer interfejs obezbjedjuje metode koji omogućavaju OPC klijentu da:

- odredi tip dogadjaja koje OPC Server podržava
- Unese pretplate (subscription) klijenata na specifične dogadjaje, tako da OPC klijenti mogu primiti obavjesti kada se oni dese. Mogu se koristiti da se definišu podskupovi željenih dogadjaja.
- Uslovi pristupa i manipulacije koji su implementirani od strane OPC Servera

Pored **IOPCEventServer** interfejsa, OPC Event Server može podržavati opcione interfejse za uslove brousinga koji su implementirani od strane servera i za upravljanje javnim grupama.

Pregled OPC historijskog pristupa podatcima (OPC *H*istorical *D*ata *A*ccess)

.Mašine za prikupljanje historiskih podataka su dodatni izvor informacija koji moraju biti distribuirani korisnicima i softwareskim klijentima koji su zainteresirani za ovu informaciju. Ranije, većina historiskih sistema je koristila njihove vlastite interfejse za prikupljanje ovih podataka.

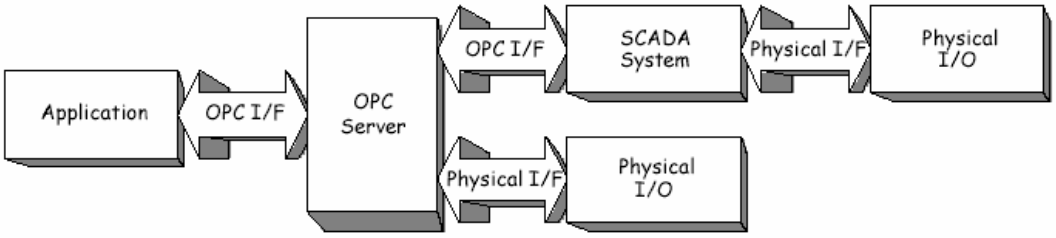
U cilju integracije podataka na svim nivoima poslovanja, historiska informacija može takodjer biti smatrana kao još jedan tip podataka.

Postoji nekoliko tipova servera historijskih podataka (Historian server). Neki od najvažnijih tipova koji su podržani ovom specifikacijom su:

- Jednostavni serveri trend podataka. Ovi serveri obezbjedjuju u suštini samo pohranjivanje sirovih podataka. Podatci su tipično onog tipa koji je raspoloživ od **OPC D**ata **A**ccess servera, obićno u formi trojke (triple) koja uključuje : [*Time , Value & Quality*], tj. Vrijeme, vrijednost i kvalitet podatka. [**V,Q,T**].
- Serveri sa kompleksnom kompresijom podataka i analizom. Ovi serveri obezbjedjuju kompresiju podataka kao i pohranjivanje sirovih podataka. Oni su u stanju da obezbjede sumarne podatke ili funkcije analize podataka, kao što su : srednje vrijednosti, minimumi, maksimumi, itd. Oni mogu podržati ažuriranje podataka kao i historiju ovih ažuriranja. Takodjer mogu podržavati i komentare zajedno sa historijskim podatcima.

Gdje se OPC uklapa

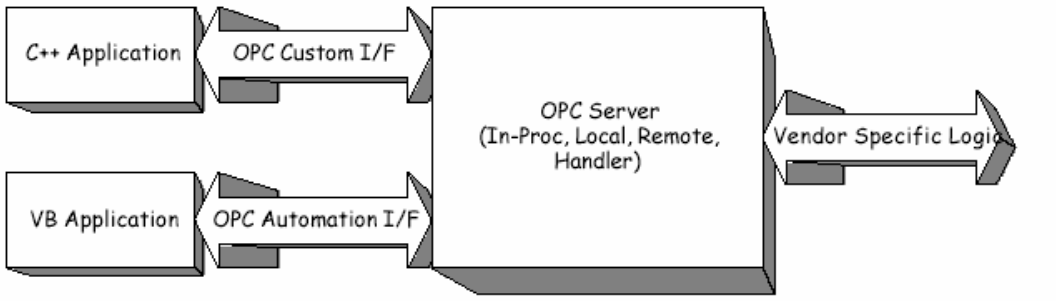
Mada je OPC primarno dizajniran za pristup podatcima sa mrežnog servera, OPC interfejsi se mogu koristiti na mnogim mjestima unutar aplikacije. Na najnižem nivou oni mogu dobijati sirove podatke od fizičkih uredjaja u SCADA ili DCS, ili iz SCADA ili DCS sistema u aplikaciju. Arhitektura i dizajn čine mogućim da se konstruiše OPC Server koji dozvoljava klijentu aplikacije da pristupi podatcima iz mnogih OPC Servera koje obezbjedjuju razni proizvodjači hardwarea priključeni na različitim čvorovima , putem jednog objekta.



OPC relacija Klijent-Server.

Opšta OPC arhitektura i komponente

OPC specifikacije uvjek sadrže dva seta interfejsa: Prilagodjeni (Custom) interfejsi i interfejsi nazvani Atomation I/F (automatizirajući interfejs) . Ovo je pokazano na slijedećoj slici:



OPC interfejsi

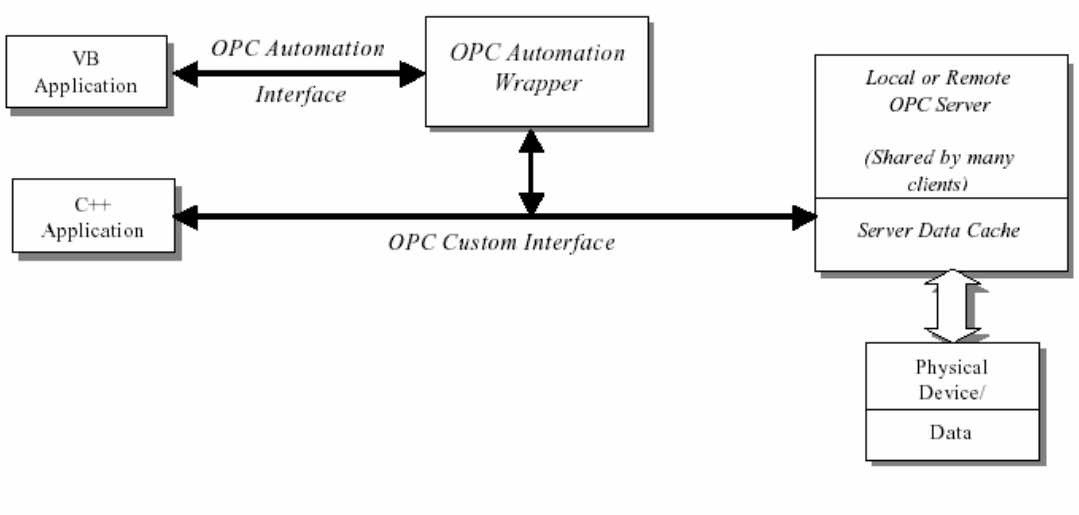
OPC specifikacija specificira COM interfejse (tj. šta su interfejsi i koje funkcije ostvaruju) , a ne njihovu implementaciju.

Dakle, ona specificira ponašanje koje se očekuje od interfejsa da obezbjede klijent aplikacije koje ih koriste.

Kao i COM implementacije, arhitektura OPC je klijent-server model, gdje OPC Server komponenta obezbjedjuje interfejs ka OPC objektima i upravlja sa njima.

Postoji nekoliko razmatranja u implementaciji OPC Servera. Glavno pitanje je frekvencija prenosa podataka preko nedjeljivih komunikacionih staza ka fizičkim uredjajima ili drugim basevima podataka. Dakle, mi očekujemo da OPC serveri će biti ili lokalni ili daljinski EXE fajlovi koji uključuju kōd koji je odgovoran za efikasno prikupljanje podataka od fizikalnog uredjaja ili baze podataka.

OPC klijent aplikacija komunicira sa OPC serverom preko specifičnih custom i automatizacionih interfejsa. OPC serveri moraju implementirati **custom** interfejse i opciono mogu implementirati **automatizacione** interfejse. U nekim slućajevima OPC Fondacija (tijelo koje promovira OPC standardizaciju), može obezbjediti omotnicu (wrapper) za standardni automatizacioni interfejs . Ovaj "wrapperDLL" može biti korišten za bilo koji Vendor specifični server, kao na slici :



Tipična OPC arhitektura

Lokalni i daljinski serveri

OPC Server Vendori imaju jedan od dva pristupa umreženju svojih proizvoda:

- Klijent treba uvjek da se konektira sa lokalnim serverom koji će onda koristiti već postojeće mehanizme za pristup mreži (proprietary). Ovaj pristup se koristi obićno od Vendora koji dodaju OPC mogućnost već postojećem distribuiranom proizvodu kao što je bio FIX kod Intellutiona.

- Oni mogu indicirati da se klijent konektira na željeni server na ciljnom čvoru i koriste DCOM arhitekturu da obezbjede ovo umreženje.

OPC Server Browser

Interfejs za OPC Server browser (**IOPCServerList**) je specificiran kao dio **OPCCommon** dokumenta

Postavka problema realizacije ovog browsera

Problem koji se postavlja je "kako da klijent program pokaže korisniku koji OPC Serveri su na raspolaganju na datoj mašini"?. OPC Serveri se registruju u sistemu preko kategorija komponenti (Component Categories).

Ovo dozvoljava Microsoftovom ICatInformation (IID_IcatInformation) Interface na StdComponentCategoriesMgr:
(CLSID_StdComponentCategoriesMgr),
da se koristi da odredi koji OPC serveri su instalirani na lokalnoj mašini.

Problem je da ovaj mehanizam ne radi za udaljene mašine pošto Component Categories Manager je DLL i IcatInformation interfejs radi samo "**in-process**", tj na istoj mašini.
Kao rezultat toga, ne postoji način za klijenta da dobije listu OPC Servera instaliranih na udaljenim mašinama.

Pregled mogućeg riješenja

Server Browser OPCENUM.EXE kojeg isporučuje OPC fondacija može biti rezidentan na bilo kojoj mašini i on će pristupiti lokalnom manageru Component kategorija (Component Categories Manager), i obezbjedjuje novi interfejs IOPCServerList koji se može distribuirati (marshaling) i biti korišten od strane udaljenih klijenata.

Ovaj server ima publikovani classid , i može se instalirati po jednaput na svakoj mašini koja ugošćuje (hosting) OPC servere.

Klijent još uvjek treba da zna ime noda ciljne mašine na kojoj traži OPC servere, modjutim on može sada kreirati ovaj objekat daljinski i koristiti njegov IOPCServerList interfejs da odredi koji tipovi i vrste servera su na raspolaganju na toj mašini.

Sumarni rezime

Baziran na Microsoftovoj OLE (sada ActiveX), COM (component object model) i DCOM (distributed component object model) tehnologijama, OPC se sastoji od standadnog seta interfejsa, osobina, i metoda za korištenje u

sistemima vodjenja i nadzora procesnih i proizvodnih sistema, i povezivanje ovih sa poslovnim sistemima.

ActiveX/COM tehnologije definišu kako individualne softwareske komponente mogu interaktirati i dijeliti podatke izmedju sebe. Bazirane na Microsoftovoj NT tehnologiji, OPC obezbjedjuje zajednički interfejs za komuniciranje izmedju raznorodnih procesnih uredjaja i I/O podsistema, bez obzira koji operativni sistemi i MMI software koriste.

Kao što je već napomenuto cilj standarda je bio : "plug- and-play", koncept koji je razvio Microsoft i niz drugih kompanija. Koristeći standardni način konfiguriranja hardwarea i softwareskog interfejsa, uredjaj će se lako povezivati sa ostalim djelom sistema i trenutačno nakon toga raditi, bez dugotrajne instalacione procedure i kompleksnog konfiguriranja.

Korisnik, umjesto da uči 100 i više alata za povezivanje od raznih proizvodjača, sada treba da nauči samo jedan set alata, jer će svi OPC drajveri raditi na isti način.

OPC standard zahtjeva od proizvodjača hardwarea da obezbjede ovu kolekciju i distribuciju podatka iz svog hardwarea. Oni su zasigurno najbliskiji hardwareu i znaju kako interno pristupiti podatcima. Ovi uredjaji sa OPC drajverom tada postaju OPC serveri, osiguravajući OPC klijent aplikacijama podatke na konzistentan i standardiziran način. IT inženjeri koji razvijaju aplikacije mogu to sada raditi u bilo kojem programskom jeziku koji im odgovara.

Šta je COM?

Component object model osigurava standardne interfejse i medjukomponentne komunikacije. Putem COM, jedna aplikacija može koristiti osobine bilo kojeg drugog aplikacionog objekta ili operativnog sistema, ili da omogući da se ta softwareska komponenta poboljša (upgrade) , bez da to utiče na rad ukupnog sistema i rješenja.
COM se može koristit od strane razvojnih IT inženjera i integratora sistema, da se kreiraju prilagodjena (customized) rješenja.

COM je binarni i generički standard, i nalazi se kao jezgro (core) i u DCOM, ActiveX i OLE tehnologiji.

Šta je OLE?

Objektno povezivanje i ugradjivanje (object linking and embedding) , se koristi da obezbjedi integraciju medju aplikacijama, omogućujući visok nivo aplikacione kompatibilnosti, čak i izmedju različitih tipova informacija. OLE tehnologija je bazirana na COM, i omogućava razvoj višekoristivih (reusable) plug-and-play objekata koji su medjusobno operativni čak i u okviru višestrukih aplikacija. OLE takodjer obezbjedjuje višekoristivi razvoj softwarea baziran na komponentama, pri čemu se softwareske komponente mogu pisati u bilo kojem jeziku, i isporučenom od bilo kojeg proizvodjača softwarea.

Šta je OLE automatizacija?

OLE atomatizacija i u njoj uključene COM tehnologije su razvijene od strane Microsofta da omoguće komponentama (pisanim u C i C++) , da budu korištene od strane prilagodjenog programa (pisanog u napr. VB ili Delphi). Ovaj model obezbjedjuje precizno zadovoljenje potreba upravljanja i vodjenja u procesnoj industriji, gdje firme koje razvijaju hardware pišu softwareske komponnete u C i C++, za omogućavanje pristupa podatcima unutar uredjaja. Kroz OPC, IT inženjeri koji razvijaju aplikacije mogu pisati kod u bilo kojem jeziku da zahtjevaju i koriste ove podatke.

Šta je DCOM?

Distribuirani komponentni objektni model proširuje COM na mreže (tj. daljinske objekte). Riječ je o novom vrlo optimiziranom protokolu, gdje se daljinske komponente pojavljuju kao da su lokalne. DCOM je prvo bio realizovan za Windows NT 4.0 u 1996 godini. Microsoft Java i VB Script podržavaju razvoj DCOM i ActiveX.

Šta je ActiveX?

ActiveX je kišobran (omotnica) za širok opseg tehnologija koje su ranije nazivane OLE Controls, a sve se baziraju na COM. Preimenovanje i restruktuiranje OLE Controls tehnologije je dovelo do toga da je postala ***objekt bazirana*** a ne ***objekt orijentirana.***

ActiveX je otvorena, integrirana platforma koja omogućava IT razvojnim inženjerima na Web stranicama da kreiraju portabilne aplikacije i interaktivne sadržaje za WWW aplikacije. Riječ je o otvorenoj kros platformi, i podržavanoj na Mac, Windows i Unix sistemima.

Sumirajući gore iznjeto možemo reći da Microsoftov COM je softwareska arhitektura koja dozvoljava da se grade aplikacije iz binarnih softwareskih komponenata. COM je bazna arhitektura koja omogućava nadgradnju više nivovskih softwareskih servisa kao što su oni koje obezbjedjuje ActiveX.
ActiveX i COM dozvoljavaju aplikacijama da dijele "objekte", kao naprimjer spreadsheetove koji su uronjeni (embedded) u dokumente za procesiranje teksta. Kada se pojave nove ažurirane vrijednosti u spreadsheetovima, ActiveX i COM će se pobrinuti da se ovi ažurirani podatci pojave i u tekstu dokumenta gdje su uronjeni.

Pristup podatcima je omogućen svakom u hijerarhiji

Još jedna bitna prednost primjene OPC standarda je da je omogućen pristup procesnim podatcima na svakom nivou u firmi. Procesni podatci nisu više ograničeni samo na Operatore i procesne inženjere i managere procesa.

Pristup kroz VB aplikacije koje koriste OPC Data Acess Specifikaciju dozvoljava IT inženjerima da pišu aplikacije koje koriste procesne podatke u poslovnim aplikacijama.

Nadalje, ove aplikacije mogu biti pisane sa malo ili nikakvog znanja o industrijskoj mreži i njenoj konfiguraciji kroz koju se prenose podatci iz procesa.

Široka primjena i usvajanje OPC standarda rezultira u većem dijeljenju informacija izmedju više aplikacija koje se simultano odvijaju. Kako je sve više aplikacija OPC omogućeno (OPC enabled), ista informacija se može distribuirati ka mnogim aplikacijama kao što je : održavanje, skladište i nabavka rezervnih dijelova, operatorski displeji, i menagment dokumenata, korištenjem kombinacije OPC i DCOMa , tako da se tehnološki procesi i poslovni procesi mogu simultano koordinirati.

Lakoća korištenja – Autokonfiguracija tagova

Efektivno dizajnirane OPC komponente se takodjer vrlo lako koriste, i zahtjevaju vrlo malo konfiguracije. OPC serveri ne zahtjevaju od korisnika da konfigurira tagove. Server može automitizirati ovo konfiguriranje, tako da je instaliranje OPC automatizovano rješenje.

Dodavanje i brisanje procesnih tačaka bez zaustavljanja sistema

Nove procesne tačke se mogu dodati ili izbrisati bez da se server mora zaustaviti. Ovo je značajno superiornije u odnosu na mnoge vlastite firmenske proizvode koji zahtjevaju da se drajver zaustavi prije nego što se nove procesne tačke mogu dodati. Primjer korištenja ove sposobnosti je naprimjer : tačke se mogu dodati putem interfejsa baze podataka, definišući tačke konzistentne sa sintaksom servera. Podatci će se nakon toga početi trenutačno pojavljivati u bazi sa OPC servera.

Sinhrono i asinhrono upisivanje na uredjaj

Sinhrono ili asinhrono upisivanje na uredjaj , sa potvrdjivanjem (acknowledgement), je superiorna karakteristika u odnosu na ranije DDE drajvere, što je uvjek predstavljalo veliki problem IT inženjerima koji su razvijali aplikaciju. Tako naprimjer kod nekih DDE drajvera, aplikacija je pokušavala da upiše podatak na PLC uredjaj. Medjutim, prije nego što bi vrijednost i dospjela do PLC-ja, vrijednost bi već bila prepisana od strane drajvera u polling ciklusu.

Sa uvodjenjem DCOM, kako OPC rješava problem kada se udajeni server isključi

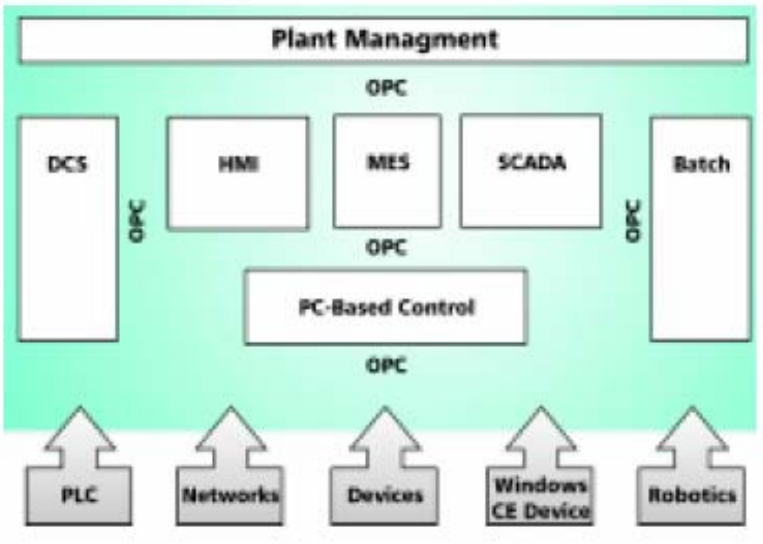
DCOM obezbjedjuje ugradjenu osobinu koja osigurava da OPC klijenti i serveri imaju pouzdan mehanizam da razmjenjuju podatke u realnom vremenu kroz mrežu. DCOM takodjer upravlja sa ponovnim pokušajima (retries) i isteklim vremenima (time-outs) izmedju OPC klijenta i udaljenog OPC servera , i pokušava da ponovno uspostavi izgubljenu komunikaciju.

Jedna od strogih osobina OPC je da on naslijedjuje sve standardne softwareske tehnologije kao što su: ActiveX, DCOM i WindowsNT. DCOM tehnologija čini distribuirani klijent/server mrežni protokol transparentnim za OPC aplikaciju. DCOM može slati OPC poruke koristeći niz transportnih protokola kao što su: UDP, TCP/IP, IPX, koristeći istu OPC aplikaciju koja koristi DCOM.

OPC i XML

OPC je važna komponenta Microsoft Distributed Network Architecture for Manufacturing (DNA-M). [Microsoftova Internet arhitektura za proizvodnju]. Ova arhitektura je bazirana na različitim MS proizvodima (tj. Operativnim sistemima, aplikacionim serverima, programskim okružajima), i tehnologijama (DCOM; XML, Internet), koje su dizajnirane da olakšaju integraciju aplikacija automatizacije : tj. ERP (Enterprise resource planning), DCS, SCADA, HMI, MES (Management engineering systems), PC bazirano upravljanje.

Jedna od ovih tehnologija je i BizTalk , MS programski okružaj za razmjenu podataka izmedju aplikacija koje su bazirane na XML (eXtensible Markup Language, jezik za opisivanje strukture podataka), šemi i na postojećim ili budućim industrijskim standardima za razmjenu podataka (vidjeti MS Biztalk Server 2002). Korištenje BizTalka pojednostavljuje integraciju izmedju aplikacija. Sa ovom tehnologijom, nije tako važno da se aplikacije usaglase na konvencijama pozivanja i da ih implementiraju, nego razmjena podataka se dešava pomoću samoobjašnjivih dokumenata.



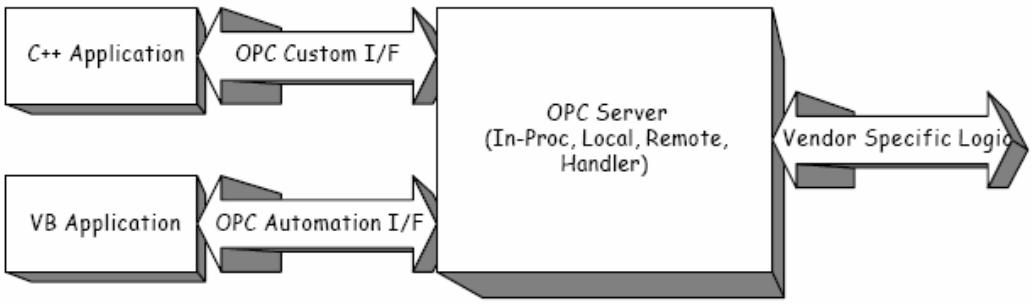
Integraciona arhitektura DNA-M

Ovo predstavlja novi izazov za OPC standard. Integracija sa BizTalkom će dovesti do porasta važnosti OPC u DNA-M, pošto će biti moguće koristiti OPC i van oblasti I/O interfejsa izmedju procesnih uredjaja i računara.

OSNOVE OPC DIZAJNA

Definicije interfejsa

OPC specifikacije uvjek sadrže dva skupa interfejsa : Prilagodjeni (custom) interfejs i automatizacioni interfejs. Ovo je pokazano i na slijedećoj slici:



OPC intefejsi

OPC klijent aplikacija komunicira sa OPC serverom kroz specifične prilagodjene i automatizacione interfejse. OPC serveri **moraju** implementirati kastomizirani interfejs i mogu opciono implementirati automatizacioni interfejs. U nekim slućajevima OPC Fondacija obezbjedjuje standardni automatizacioni interfejsni omotač (wrapper). Ovaj "wrapperDLL" se može koristiti za bilo koji Vendor specifični server.

OPC klijent komunicira sa OPC serverom pozivajući funkcije iz OPC **kastomiziranog** (custom) interfejsa.

Proizvodjači hardwarea koji za njega razvijaju OPC server mogu implementirati i funkcije **opcionog** (izbornog) interfejsa. Kada OPC server podržava opcioni interfejs, sve funkcije unutar tog interfejsa moraju biti implementirane, čak i u slućaju kada će funkcija samo vratiti **E_NOTIMPL**. OPC klijent koji želi da koristi funkcionalnosti opcionog interfejsa će pitati OPC servera za opcioni interfejs. OPC klijent ipak mora biti tako dizajniran da ne zahtjeva da ovaj opcioni interfejs i mora postojati kod OPC servera.

Općenito, OPC klijent programi koji se kreiraju koristeći skript bazirane programske jezike će koristiti automatizacioni interfejs. Klijent programi koji su kreirani u C++ će lakše koristiti kastomizirane interfejse za postizanje najbolje performanse.
OPC serveri moraju implementirati kastomizirani interfejs i opciono mogu implementirati automatizacioni interfejs.

Vlasništvo nad memorijom

Prema COM specifikaciji, klijenti moraju osloboditi svu memoriju pridruženu sa '**out**' ili '**in/out**' parametrima. Ovo uključuje i memoriju na koju su usmjereni pointeri elemenata unutar bilo koje strukture. Ovo je vrlo važno da shvate IT inženjeri koji pišu OPC klijent software, inače iskusiće curenje memorije (memory leakage) koje će biti teško otkriti. Preporućeni način da se ovo realizuje je da klijent kreira subrutinu koja će se koristiti za korektno oslobadjanje memorije iz svakog tipa strukture.

Nezavisno od uspjeha ili neuspjeha poziva od strane klijenta ka serveru, server mora uvjek vratiti dobro definirane vrijednosti za 'out' (izlazne) parametere. Oslobadjanje alociranih resursa za ove 'out' parametre je odgovornost klijenta.

Opaska : Ako je rezultat greške bilo koja *FAILED* greška kao naprimjer *E_OUT OF MEMORY*, OPC server treba vratiti *NULL* za sve 'out' pointere (ovo je standardno ponašanje COM modula).

Greške i povratni kodovi

OPC specifikacije opisuju interfejse i odgovarajuće ponašanje koje OPC server mora da implementira i od kojih OPC klijent aplikacija zavisi. Lista grešaka i povratnih kodova je sadržana u svakoj od tri OPC specifikacije. Za svaki opisani metod , lista svih mogućih kodova grešaka je uključena , zajedno sa najčešćim OLE kodovoma grešaka.

U svakom slućaju '**E**' kodovi grešaka (koje počinju sa slovom **E**) , će indicirati FAILED tip grešaka a '**S**' kodovi grešaka će indicirati barem djelomićni uspjeh.

Isključenje (Shutdown) OPC servera

Mogućnost da OPC server obavjesti klijente o svom padu je realizovana kroz zahtjev OPC servera da se svi klijenti odspoje od servera. Ova mogućnost je obezbjedjena za sve tipove OPC servera (**D**ata**A**ccess, **A**larm&**E**vent, **H**istorical **D**ata **A**ccess).
Funkcionalnost je raspoloživa putem konekcione tačke na Server objektu i korespondirajuće tome na strani klijenta je **IOPCShutdown** interfejs.

Klijenti trebaju koristiti ovu funkcionalnost da bi mogli ostvariti na kontrolirani način prestanak dolaska podataka sa servera koji je u shutdownu.

***IconnectionPointContainer* (na OPC serveru)**

Ovaj interfejs obezbjedjuje pristup ka konekcionoj tački za IOPCShutdown.

Klijent koji je povezan sa više OPC servera (naprimjer DataAccess , Alarm&Event server, od jednog ili više Vendora), treba da održava posebne shutdown callbacks za svaki objekat pošto svaki OPC server pojedinačno može da ode u shutdown.

IOPCCommon

Ovaj interfejs se koristi od strane svih OPC Server tipova (**D**ata**A**ccess, **A**larm&**E**vent, **H**istorical **D**ata **A**ccess). On obezbjedjuje mogućnost da se postavi i pošalje upit za LocaleID , koji će se koristiti za datu klijent/server sesiju.
To znači da akcije jednog klijenta ne utiču na bilo koje druge klijente.

Kao što je to slučaj i sa drugim interfejsima kao što je **Iunknown**, instanca ovog interfejsa za svaki server je jedinstvena. Dakle, *OPC DataAccess server object* i *OPC Alarms&Events server object* mogu obadva osigurati po jednu implementaciju od **IOPCCommon**. Klijent koji održava konekcije sa obadva servera će, kao i u slučaju bilo kojeg drugog interfejsa, koristiti interfejse za ova dva objekta nezavisno jedno od drugoga.

Pitanja instalacije i registracije OPC server softwarea

Svaki proizvodjač OPC server softwarea (Vendor) obezbjedjuje SETUP.EXE , da bi se instalirale potrebne komponente za njihove servere. Druga pitanja koja se tiču OLE softwarea je management Windows Registry i Component Categories.

Pitanja su:

- koji unosi trebaju biti uradjeni
- kako se oni mogu realizovati

Component Categories

Sa ogromnom količinom komponenti na svakom posebnom kompjuterskom sistemu, njihov management postaje sve teži i teži. OPC klijenti često trebaju da enumeriraju (prebroje) OPC servere koje žele da koriste u datom kontekstu. U svojoj prvoj verziji OPC je specificirao pod-ključ (sub-key), koji se zvao OPC , da bi tagirao sve OPC Server ulaze u registre. Klijenti su trebali da browsuju tražeći ovaj ključ.

Ovaj metod se pokazao kao neefikasan jer je zahtjevao da se browsuju svi ***CLSID*** ulazi u registre. Mogli su se pojaviti problemi sa kolizijom imena. Konačno pristup udaljenim registrima je ograničen u NT5.0 arhitekturi (WIN 2000, WinXP).

Za sve serverske aplikacije poslije DataAccess 1.0A, koriste se **Component Categories** kao način kategorizacije OPC Servera po njihovoj implementiranoj funkcionalnosti.

Klijenti mogu koristiti novi interfejs **IOPCServerList** da bi dobili listu servera sa zahtjevanom funkcionalnošću.

OPC definira " implemented categories" za svaku verziju svake ***OPC Interface*** specifikacije.

Očekuje se da će server prvo kreirati svaku kategoriju koju koristi i nakon toga će se registrirati za tu kategoriju. Pogledati ***IcatRegister*** dokumentaciju za dodatne informacije.
Jedan server može pripadati u više od jedne kategorije, naprimjer može podržavati DataAccess Verzije 1.0A i 2.0 i još Alarm&Event handling.

Registracija Component Categories

Za vrijeme procesa registracije, svaki OPC Server mora se registrovati sa **Managerom Component Categories**, koji je COM objekat koji isporučuje MS. OPC klijenti će slati upite ***Components Category Manageru*** da nabroji (enumerate) *CLSID* svih registriranih OPC servera.

Registracija Servera

Da bi se registrirao sa Component Categories Managerom, server treba prvo registrirati OPC definiranu Category ID (*CATID*) i OPC definiranu *Category Description* , pozivajući

IcatRegister :: RegisterCategories()

a zatim registrirajući svoj vlastiti CLSID kao implementaciju CATID sa pozivom ka

IcatRegister::RegisterClassImplCategories()

Da bi se dobio interfejs pointer na *IcatRegister*, treba pozvati *CoCreateInstance(),* kao što je to pokazano u narednom promjeru iz Alarm&Event Servera :

```
#include <comcat.h>

CoCreateInstance(CLSID_StdComponentCategoriesMgr, NULL,
CLSCTX_INPROC_SERVER, IID_ICatRegister, (void**)&pcr);
```

Prebrojavanje klijenta (Client Enumeration)

Klijenti će koristiti interfejs ***IOPCServerList*** da dobiju listu servera bilo lokalno ili na udaljenom hostu. Ovaj interfejs u osnovi obezbjedjuje funkcionalnost *Component Categories* Managera. Definiran je u standardu OPC, pošto pristup *Component Categories* Manageru nije moguć na udaljenim mašinama.

Unosi u registre za Proxy/Stub DLL

Proxy/stub DLL-ovi se koriste za organizovanje interfejsa ka LOCAL i REMOTE serverima. Generira se direktno iz IDL koda i treba biti isti za svaki OPC server. Općenito, Proxy/Stub koristi samo registraciju.

Testirani i gotovi proxy/stub DLL je na raspolaganju na OPC Foundation Web site-u tako da nema potrebe da Vendor razvija ovaj DLL.

Kreiranje ulaza u registre (Registry Entries)

COM definira mehanizam samoregistracije koji omogućava da se uključe (enkapsuliraju) potrebe za unose u registre u DLL ili EXE, i na taj način obezbedjujući za klijente i servere lagani način da se svaki modul u potpunosti i korektno registruje.

Nadalje, COM takodjer uključuje deregistraciju , tako da server može da ukloni sve svoje registrirajuče unose , kada se DLL ili EXE otkloni iz fajl sistema, tako da se registri očiste od beskorisnih unosa.

Kada se zatraži od servera da se samoregistruje, server mora kreirati sve unose za svaku komponentu koju podržava, uključujući sve ulaze za tipske biblioteke. Kada se zatraži od njega da se deregistruje ("un-register") , server mora ukloniti one unose koje je kreirao kod samoregistracije.

Za DLL server, ovi zahtjevi se realizuju putem call-ova ka izvezenim funkcijama ***DllRegisterserver*** i ***DllUnregisterServer***, koji moraju egzistirati unutar DLL pod tačno ovim imenima. Obadvije funkcije se pozivaju bez argumenata i vraćaju HRESULT da indiciraju rezultat.

Dva koda grešaka ako postoje su :

SELFREG_E_CLASS (neuspjeh da registruje/deregistruje CLSID informaciju) i
SELFREG_E_TYPELIB (neuspjeh da registruje/deregistruje TypeLib informaciju).

Ako je server pakovan u EXE modul, tada aplikacija koja želi da registruje server, lansira EXE server sa komandnim argumentom na liniji : */RegServer* ili *–RegServer* (neosjetljiv na velika/mala slova).
Ako aplikacija želi da deregistruje server, lansiraće EXE fajl sa argumentom na komandnoj liniji */UnregServer* ili *–UnregServer*. Samoregistrirajući EXE detektuje ove argumente na komandnoj liniji i poziva iste operacije kao što bi i DLL unutar **DllRegisterServer** i **DllUnregisterServer**, registrirajući svoju stazu pod LocalSever32 umjesto InprocServer32 ili InprocHandler32.

Server mora registrirati punu stazu do instalacione lokacije od DLL ili EXE modula, za njhove odgovarajuće *InprocServer32* , *InprocHandler32*, i *LocalServer32* ključeve u registrima.
Staza modula (*module path*) se može lako dobiti pomoću Win32 API funkcije *GetModuleFileName*.

Opaska: Server ne treba da registruje proxy/stub interfejse. Oni treba da budu registrirani od strane proxy/stub DLL kako je to već ranije pomenuto.

Registri ulazi za proxy interfejse mogu se lagano generirati kada se kompilira proxy dll. Jednostavno treba definirati konstantu REGISTER_PROXY_DLL za vrijeme kompilacije, i izvesti DllRegisterServer i DllUnregisterSrver za vrijeme linkovanja.

Sada je moguće unjeti podatke u registre izvršavajući regsvr32 i prenoseći proxy.dll ime kao argumenat.

OPC DATA ACCESS SPECIFIKACIJE

OPC Data Access specifikacije sadrže informacije za dizajn:

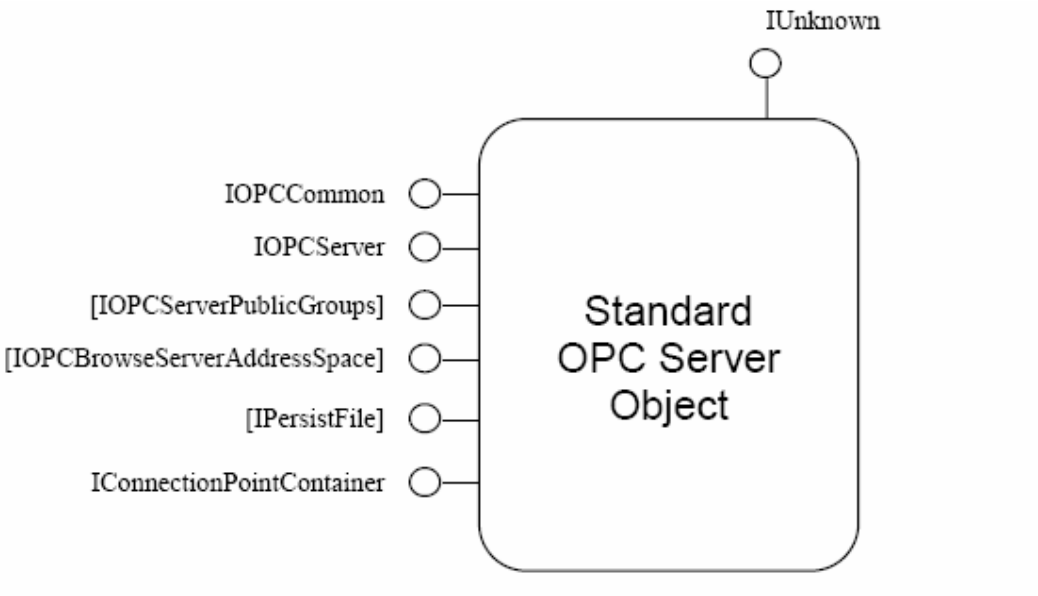
1. OPC DataAccess Custom Interface - Ovaj dokument opisuje Interfejse i Metode za OPC komponente i objekte (Components and objects).

2. **OPC data Access Automation Interface** – poseban dokument (Verzija 2.02) koji opisuje OPC automatizacioni interfejs koji olakšava korištenje Visual Basica, Delphija i drugih automation enabled softwareskih proizvoda da se interfejsiraju sa OPC Serverima.

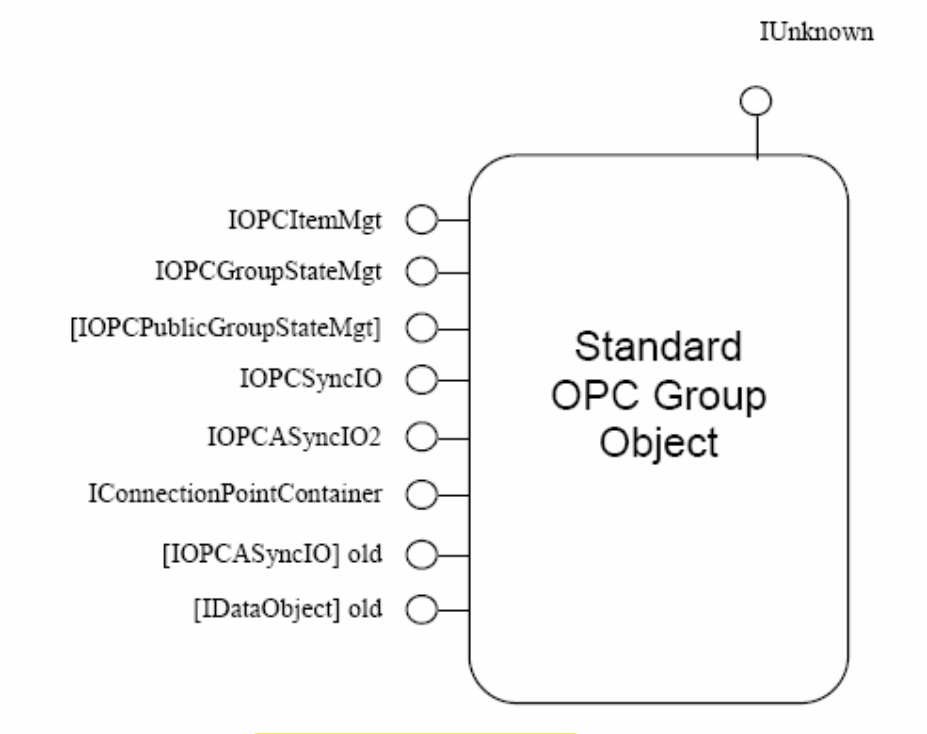
SPECIFIKACIJE ZA OPC DATAACCESS CUSTOM INTERFACE
STANDARD VERZIJA 2.04

OPC Server objekat obezbjedjuje način da se pristupi (read/write) ili komunicira sa skupom izvora podataka. Tipovi izvora koji su na raspolaganju su funkcija načina implementacije servera.
OPC klijent spaja se sa OPC serverom i komunicira sa njim putem interfejsa. OPC server objekt osigurava funkcionalnost OPC klijentu da kreira i manipulira OPC group objektima. Ove grupe dozvoljavaju klijentima da organiziraju podatke kojima žele pristupiti. Grupa se može aktivirati i deaktivirati kao jedinica. Grupa takodjer obezbjedjuje način za klijenta da se 'pretplati' (subscribe) na listu itema (detalja) tako da može biti obaviješten o njihovoj promjeni.

Opaska: Svim COM objektima se pristupa kroz interfejse. Klijent vidi samo interfejse, Dakle objekti koji su opisani u ovom standardu su 'logičke' predstave koje ne moraju imati ništa zajedničkog sa stvarnom internom implementaciojm servera. Naredna slika daje sumarnu predstavu OPC objekata i njihovih interfejsa. Neki od interfejsa su opcioni i označeni su u uglastim zagradama [ ].



Standardni OPC Server objekat



Standardni OPC Group objekat

OPCServer objekt je primarni objekt kojeg OPC server izlaže. Interfejsi koje ovaj objekt osigurava su:

- Iunknown
- IOPCServer
- IOPCServerPublicGroups (opciono)
- IOPCBrowseServerAddressSpace (opciono)
- IpersistFile (opciono)
- IOPCItemPropreties
- IconnectionPointContainer

IOPCServer je glavni interfejs za OPC server. OPC server se registruje kod operativnog sistema na način kako je to bilo opisano ranije.

IOPCItemProperties

Ovaj interfejs može biti korišten od strane klijenata da browsuju raspoložive osobine (takodjer nazivane i atributi ili parametri) pridruženi ITEMID i da čitaju tekuće vrijednosti ovih osobina. Na neki način ova funkcionalnost je slična onoj koju obezbjedjuje **BrowseServerAddressSpace**, putem **EnumItemAttributes** i **SyncIORead** funkcija. Medjutim i razlikuje se od ova dva interfejsa u dva važna aspekta:

a) namjenjen je da bude mnogo lakši za korištenje
b) nije optimiziran za efikasan pristup velikom obimu podataka.

Ona je prije svega namjenjena da dozvoli aplikaciji da lagano browsira i čita male količine dodatnih informacija specifičnih za dati ITEMID.

Dizajn ovog interfejsa je baziran na predpostavci da mnogi ITEMIDs su pridruženi sa drugim ITEMID koji predstavljaju vezane sa njima vrijednosti kao što je opseg inženjerskih jedinica , opis procesne tačke ili status alarma.

Naprimjer sistem može interno biti izgradjen od '*recorda*' koji predstavljaju kompleksne objekte kao PID regulatore, tajmere, brojače, analogne ulaze , itd.

Ovi elementi rekorda će imati svoje osobine kao što su:

> Tekuća vrijednost, zadata vrijednost (set point), visoka (hi) alarmna vrijednost, niska (low) alarmna vrijednost , opis , itd.)

Ovaj interfejs dozvoljava fleksibilan i pogodan način browsovanja , lociranja i čitanja ovih informacija bez da postavlja bilo kakva ograničenja na njihovu strukturu.

On takodjer dozvoljava da te informacije budu čitane bez potrebe kreiranja i manipuliranja sa OPCGroups.

U većini slučajeva sistem kao što je ovaj gore opisani (tj. Interno sastavljen od 'recorda') će takodjer posjedovati hijerarhijski adresni prostor za OPC u formi naprimjer:

A100 je grana ('branch'), a

A100.CV, A100.SP, A100.OUT, A100.DESC su listovi ('leafs').

Drugim riječima , osobine detalja (item) koji je u formi recorda će se mapirati u u ITEMIDS nižeg nivoa.

Tipično korištenje ovog interfejsa od strane klijenta biće da dobije ITEMID , bilo dobivši 'LEAF' putem interfejsa **BrowseServerAddress** ili putem direktnog unosa preko boksa za editiranje. ITEMID će biti prenesen do **QueryAvailablePropreties()**. Rezultirajuća lista bit će pokazana korisniku. On će selektirati osobine koje želi da vidi sa liste. Klijent će prenjeti ovaj set do **GetItemProperties()** da dobije 'snapshot' snimak podataka.

Opciono, klijent je mogao poslati set do **LookupItemIDs** i koristiti rezultirajući set *ITEMID*s da kreira *OPCGroup* da nju koristi da dobija podatke uzastopno u kontinuitetu.

Primjer

Tipični OPC ITEMID može biti FIC101.CV. Ovo će predstavljati tekuću vrijednost taga ili funkcionalni blok koji se zove FIC101. Ovaj funkcionalni blok ima i druge osobine koje su mu pridružene kao što su :

Inženjerske jedinice, opis konture , itd.

Nadalje, funkcionalni blok može takodjer uključivati granice alarma, i status alarma, parametre podešenja regulatora, kao i dokumentaciju kros referenci (tj. gdje se sve blok pojavljuje u sprezi), informacije za održavanje, help ekrane za pomoć u radu sa blokom i niz drugih osobina. Sve ove osobine su pridružene jedna drugoj putem njihove zajedničke asocijacije sa FIC101.

Ovaj interfejs obezbjedjuje pogodan shortcut da se brzo pristupi svim ovim osobinama.

HMI paket može koristiti ovaj interfejs da dozvoli korisniku da indicira da HI i LO vrijednosti za inženjersku jedinicu trebaju biti korištene kod skaliranja vrijednosti bargrafa.

Primjetimo da pošto ove asocijacije mogu biti tipa 'many to many' (tj. mnoge sa mnogima) , a mogu takodjer biti i cirkularne (prstenaste), klijent aplikacija neće ih automatski sve istraživati.

U mnogim slučajevima do ovih osobina korisnik može pristupiti i preko ItemIDs kao naprimjer. FIC101.HI_EU, FIC101.DESC, FIC101.ALMSTAT, itd. Ovi *ITEMIDs* se mogu naći i u *OPCGroup*. Ovaj interfejs omogućava alternativni način pristupa osobinama , kada je potrebno dobiti na efikasan način veliki broj informacija.

Ove osobine su podjeljene u tri skupa. OPC 'fixed' skup sadrži osobine koje su identične onima koje se dobiju natrag od poziva **OPCITEMATTRIBUTES**, preporučeni '*recommended*' skup je onaj koji je zajednički za sve tipove OPC servera, i 'Vendor specific' skup sadrži dodatne osobine koje su specifične za dati tip hardwareskog uredjaja.

Čitanje i Upisivanje podataka

Postoje tri načina da se klijent dobije podatke

* ***IOPCSync::Read*** (iz cache memorije ili iz registara fizičkog uredjaja)
* ***IOPCAsyncIO2::Read*** (sa uredjaja)
* ***IOPCCallback::OndataChange()*** , bazirano na izuzeću , koje može takodjer biti trigerovano sa *IOPCAsyncIO2::Refresh*

Nadalje postoje dva načina da se izbace podatci preko OPC servera na fizički izlaz:

- ***IOPCSyncIO::Write***
- ***IOPCAsyncIO2::AsyincWrite***

Opcenito o grupama

Svaka grupa ima ime. Za privatne grupe ime mora biti jedinstveno za sve privatne grupe koje pripadaju tom klijentu. Za javne grupe ime mora biti jedinstveno medju svim javnim grupama. Dok klijent može promjeniti ime privatne grupe, ime javne grupe ne može biti promjenjeno.

Privatna grupa i javna grupa mogu imati isto ime samo ako klijent nije povezan sa javnom grupom istog imena.

Imena grupa su *case sensitive*. To znači da Grupa1 je različita od grupa1.

Grupe i detalji (items) unutar grupa imaju flagove aktivnosti (active flag). Aktivno stanje grupe se održava nezavisno od aktivnog stanja detalja . Promjena stanja aktiviteta grupe ne mjenja stanje detalja.

Klijenti setuju i resetuju flagove aktiviteta za grupe i detalje kao efikasan način i alternativa dodavanju i uklanjanju grupa i detalja u njima.

Tako naprimjer ako je na HMI displeju prikaz minimiziran , onda detalji koji dolaze sa servera na taj prikaz se neće prikazivati i klijent može resetovati njihove flagove tako da mu ih server ne šalje i time rastereti komunikaciju.

Drugi način minimizacije ovog saobračaja je poziv ***OnDataChange*** unutar adresnog prostora klijenta, tako da kada se aktivni detalji u aktivnim grupama promjene (promjena je definirana kao promjena u vrijednosti (value) od posljednje prethodne koja je poslata klijentu) , ili promjena u kvalitetu (Q) te vrijednosti. Server će vratiti vrijednosti (V) i flagove kvaliteta (Q) za one detalje unutar aktivne grupe koji su se promjenili.

Javne grupe

IOPCServerPublicGroups (opcioni interfejs)

Ovaj opcioni interfejs dozvoljava management javnih grupa
Moguće je dizajnirati aplikaciju tako da iste grupe podataka se koriste od strane više klijenata. U ovakvim slućajevima opciona mogućnost servera sa

javnim grupama obezbjedjuje pogodan mehanizam za klijente i servere da dijele ove grupe.

Javne grupe mogu biti kreirane od strane servera ili može ih kreirati i klijent. Kada ih kreira klijent, one se prvo kreiraju kao privatne grupe a onda se konvertuju u javne grupe sa pozivom **MoveToPublic**.

Klijent može izbrojati (enumerate) raspoložive javne grupe po imenima koristeći :

IOPCServer::CreateGroupEnumerator. Može se spojiti ('*connect*') na javnu grupu pozivjući **GetPublicGroupByName**. Može ispitati sadržaj grupe putem : **IEnumOPCItemAttributes**. Može doznačiti klijentske handles i tipove podataka koji su poželjni za specifičnog klijenta koristeći razne **IOPCItemMgt** funkcije.

Kada se klijent spoji na javnu grupu, on se ponaša vrlo slično kao i kod spajanja na privatnu grupu. On može aktivirati ili deaktivirati grupu ili detalje (items) u grupi. On može postaviti klijent handles za grupu i detalje unutar grupe. On može postaviti zahtjevani tip podataka za detalje u grupi sa kojim mu server treba slati te podatke.
Medjutim sve ove operacije se odnose samo na tog specifićnog klijenta. One ne utiču na način kako su drugi klijenti povezani na tu javnu grupu i šta oni od nje traže.
Izuzetak od ovakvog ponašanja prema javnoj grupi u odnosu na privatnu grupu je da on ne može dodavati ili brisati detalje iz te javne grupe.

SPECIFIKACIJE ZA DATA ACCESS AUTOMATION INTERFACE STANDARD VERZIJA 2.02

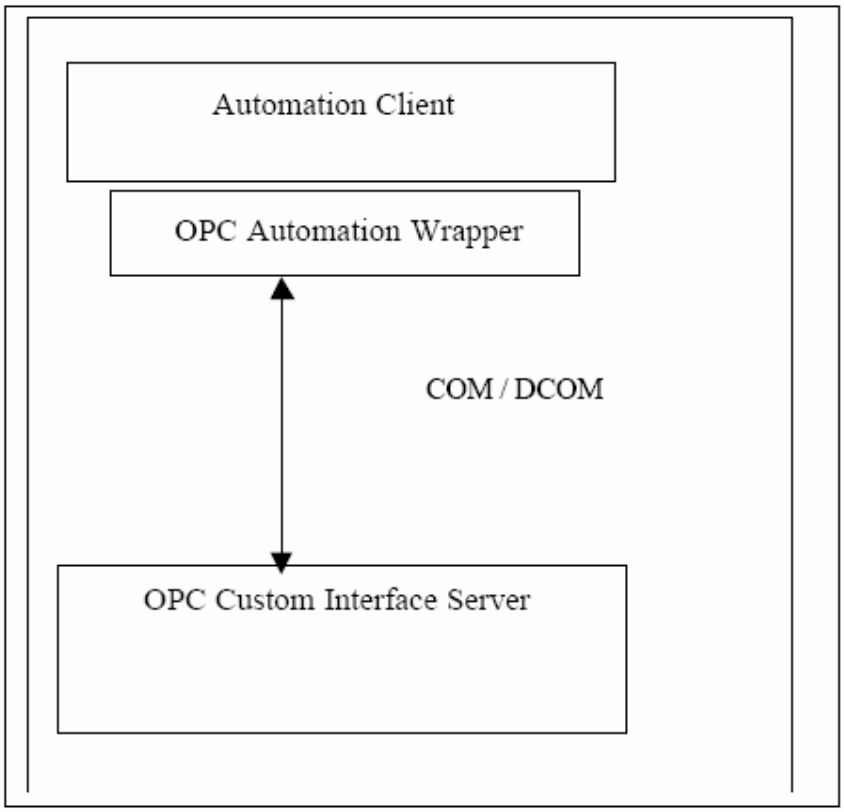
Namjena

Ono što ovaj standard želi da definiše je zajednički način na koji aplikacije u oblasti upravljanja i vodjenja procesa mogu pristupiti procesnim podatcima. Ovaj interfejs treba da obezbjedi istu funcionalnost kao i kastom (custom) interfejs, ali na način koji je blizak trendu u automatizaciji programiranja.

Kao što je već omogućeno da automatizacione aplikacije mogu pristupiti drugim softwareskim okružajima kao napr. RDBMS, MS Office aplikacijama, WWW objektima, ovaj interfejs je sačinjen tako da olakša razvoj aplikacija, bez žrtvovanja funkcionalnosti definirane prolagodjenim (custom) interfejsom.

Naredna slika pokazuje automatizacioni klijent koji poziva OPC Data Access Server koristeći 'wrapper' DLL. Ovaj omotač prevodi izmedju kastomiziranog interfejsa kojeg obezbjedjuje server i automatizacionog interfejsa kojeg želi klijent. Primjetimo da u opštem slučaju konekcija izmedju Automation klijenta

i Automation servera bit će 'In Process' dok konekcija izmedju Automation servera i Custom servera može biti bilo In process, lokalna ili udaljena.



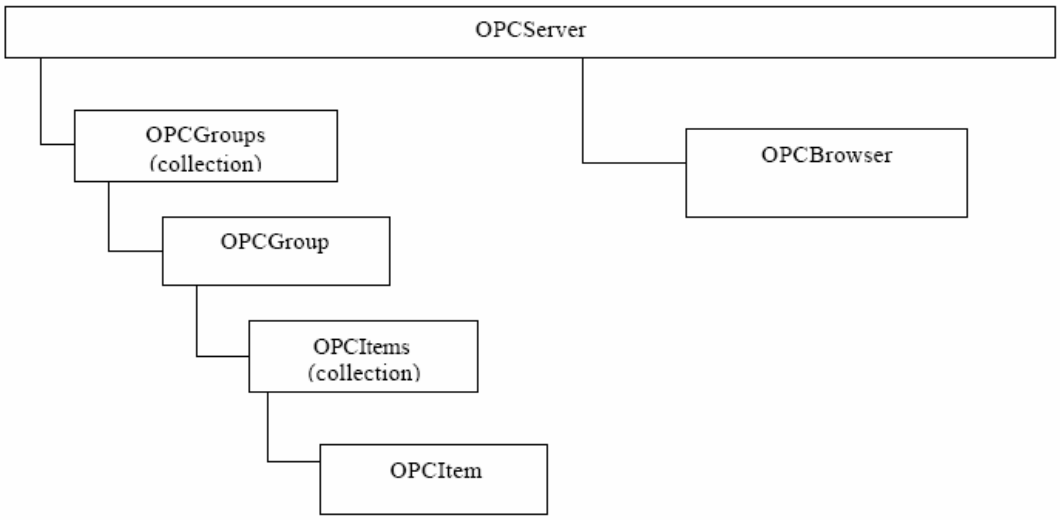
Custom i automation klijent aplikacije koje se povezuju sa OPC Serverima

Arhitektura

Osnovni cilj dizajna je da je ovaj interfejs namjenjen da radi kao 'wrapper' za postojeće OPC DataAccess Custom Interfejs servere, i da obezbjede jedan pogodan za korištenje mehanizam za funkcionalnost koju obezbjedjuje kastom interfejs.

Interfejsi su u potpunosti podržavani od VC++ i VB 5.0 i kasnije verzije. Oni dozvoljavaju svakoj aplikaciji koja ima OLE automation interfejs (napr. VB, VC++ ili VBA) da pristupe OPC interfejsu.

OPC Automatation Server Object model



Automation object hijerarhija

Objekt	Opis
OPCServer	.Instanca OPC Servera. Korisnik mora kreirati OPCServer objekat , prije nego se mogu dobiti reference za druge objekte. Sadrži *OPCGroups* kolekciju i kreira *OPCBrowser* objekte.
OPCGroups	Automation kolekcija koja sadrži sve OPCGroup objekte. Ovaj klijent je kreirao unutar opsega OPCServera sa kojim se Automation aplikacija povezala putem *OPCServer.Connect().*
OPCGroup	Instanca OPCGroup objekta. Namjena ovog objekta da održava informaciju o stanju i obezbjedi mehanizam da osigura servise akvizicije podataka za *OPCItem Collection object* koji OPCGroup object referencira.
OPCItems	Automation kolekcija koja sadrži sve OPCItem objekte koje je ovaj klijent kreirao unutar opsega OPCServer-a , i odgovarajućeg OPCGroup objekta koji je Automation aplikacija kreirala.
OPCItem	Automation objekt koji održava definiciju itema, tekuću vrijednost, statusnu informaciju i posljednje vrijeme ažuriranja (update). Primjetimo da Custom interfejs nema ovaj posebni *Item Object*.
OPCBrowser	Objekt koji browsuje imena itema u konfiguraciji servera. Postoji samo jedna instanca OPCBrowser objekta za instancu OPCServer objekta.

OPC Data Access Automation Object Model

OPCServer objekt obezbjedjuje način da se pristupi (za čitanje i pisanje, tj. Read/write) ili komunicira sa skupom izvora podataka. Tipovi izvora koji su na raspolaganju su funkcija implementacije servera.
OPC Automation klijent se povezuje sa OPC Automation Serverom koji komunicira sa izvorom podataka (t.j. sa Data Access Custom Serverima) , putem funcionalnosti koju obezbjedjuju automation objekti koji su opisani u ovom standardu.

OPCServer obezbjedjuje jedan *OPCGroups* objekat za automation kolekciju da bi održavao sakupljanje *OPCGroup* objekata.

OPCGroup objekat dozvoljava klijentima da organiziraju podatke kojima žele da pristupe. *OPCGroup* može biti aktivirana ili deaktivirana kao jedinica. *OPCGroup* takodjer obezbjedjuje način za klijenta da se 'pretplati' (subscribe) , na listu itema tako da mogu biti upoznati sa njihovim promjenama kada se dese.

OPCGroup objekat obezbjedjuje *OPCItems* kolekciju od OPCItema.

Sinhronizacija podataka

Postoje zahtjevi da VB klijent može da čita ili prima podatke kao što su vrijednost, kvalitet i vremensku informaciju (timestamp) u sinhronizmu.

Ako klijent dobije vrijednosti koristeći bilo koji od metoda čitanja treba da bude siguran da vrijednost (value), timestamp, i kvalitet (V,T,Q) će biti u sinhronizmu unutar njih samih.
Ako klijent dobija podatke registrirajući se za *DataChange events* (dogadjaje), tada V,T,Q trebaju biti u sinhronizmu unutar opsega EventHandler rutine.

Ako klijent mješa ova dva pristupa bit će nemoguće za klijenta da obezbjedi da su osobine itema tačno u sinhronizmu , pošto dogadjaj koji je promjenio osobine se mogao pojaviti izmedju vremena kada klijent pristupa različitim osobinama.

Izuzeća i dogadjaji (exceptions and events)

Većina metoda opisanih ovdje komunicira sa *OPC Custom* serverom. U OLE automation, nema lakog načina da se vrati informacija o grešci kada se pristupi nekoj osobini. Najbolji način da se ovo razriješi je da automation server generira izuzeće (exception) ako se takva greška pojavi u izvoru podataka. To znači da klijent treba da ima kodiranu logiku izuzeća da bi mogao da manipuliše ovim greškama.

Dogadjaji (events)

Automation interfejs podržava mehanizam notifikacije o dogadjaju koji je uveden u VB 5.0 i svim narednim verzijama.

Automation server trigeruje dogadjaje kao odziv na *Async Refresh*, *Async Read* i *Async Write Method* pozive (*calls*). Dodatno, Automation server trigeruje dogadjaje kada se podatci promjene u skladu sa specifikacijama klijenta.

Polja (arrays)

Po konvenciji, OPC Automation interfejs predpostavlja da su polja bazirana na indeksu koji počinje sa 1 . Ako se varijabla polja prenosi ka funkciji koja je veća od *Count* ili *NumItems* parametra, samo Count ili NumItems elementi će se koristiti , počevši od indeksa 1.
Da bi se izbjegle greške sugeriše se da VB kôd koristi "Option base 1".

Kolekcije (Collections)

OLE Automation kolekcije su objekti koji podržavaju Count, item, i skrivenu osobinu *_NewEnum*. Svaki objekt koji ima ove parametre kao dio interfejsa se može zvati kolekcija.

Svi string parametri za OPC interfejse su **UNICODE**, pošto su izvorni OLE APIji svi sa UNICODE-om.

DCOM I CORBA – DEFINICIJE

- DCOM – je distribuirano proširenje COM koji gradi sloj objekta daljinskog poziva procedure (ORPC) , nad DCE RPC da be podržao udaljene objekte. COM server može kreirati objektne instance višestrukih klasa objekta, COM objekat može podržati višestruke interfejse. Interfejs se sastoji od skupa funkcionalno povezanih metoda.
- COM klijent interaktira sa COM objektom dobijajući pointer na jedan od interfejsa objekta i pozivajući metode kroz taj pointer, kao da je objekat rezidentan u adresnom prostoru klijenta.

- **DCE** (DISTRIBUTED COMPUTING ENVIRONMENT) podržava konstrukciju i integraciju C-baziranih klijent/server aplikacija u heterogenom distribuiranom okruženju.
- **RPC** mehanizam izolira klijenta od detalja kao što su: gdje je server lociran, tipova hadvera, i platformi operativnog sistema na kojima se server izvršava, razlika i načinu predstavljanja podataka izmedju klijentskih i serverskih platformi, kao i od mrežnih transportnih mehanizama koji se koriste.

DCOM I CORBA – DEFINICIJE

- COM specificira da svaki interfejs mora slijediti standardni layout memorije, koji je isti kao kod C++ tabele virtuelne funkcije. Pošto je specifikacija na binarnom nivou, dozvoljava integraciju binarnih komponenti koje mogu biti napisane u različitim programskim jezicima kao C++, Java i VB.

- CORBA – je distribuirani objektni framework koji je predložen od OMG grupe (Object management group). Jezgro CORBA arhitekture je **Object Request Broker (ORB**) koji djeluje kao objektni bas preko kojeg objekti transparentno interaktiraju sa drugim objektima lociranim lokalno ili daljinski.

DCOM I CORBA – DEFINICIJE

- CORBA objekat je predstavljen prema vanjskom svijetu preko interfejsa sa skupom metoda. Specifična instanca objekta se identificira putem objektne reference. Klijent CORBA objekta dobija svoju objektnu referencu i koristi je kao handle da bi realizirao pozive metoda, kao da je objekat lociran u adresnom prostoru klijenta.

- ORB je odgovoran za sve mehanizme koji su potrebni da se nadje implementacija objekta, da ga pripremi da primi zahtjev, da mu prenese zahtjev , i da izvrši odgovor (reply) ako ga ima natrag ka klijentu. Implementacija objekta interaktira sa ORB putem ili OA (Object adapter) ili kroz ORB interfejs

DCOM I CORBA – TERMINOLOGIJA

- **INTERFACE** – je imenovana kolekcija abstraktnih operacija (metoda) koja predstavlja neku funkcionalnost.
- **OBJECT CLASS(** ili **CLASS)** – je imenovana konkretna implementacija jednog ili više interfejsa.
- **OBJECT (** ili **OBJECT INSTANCE)** – je instantinizacija neke objektne klase.
- **OBJECT SERVER** – Proces odgovoran za kreiranje i hosting objektnih instanci.
- **CLIENT** – je proces koji poziva metod nekog objekta .

DCOM I CORBA – KARAKTERISTIKE

- **DCOM i CORBA** framework obezbjedjuju klijent-server tip komunikacije. Da bi zahtjevao servis, klijent poziva metod koji je implementiran od strane udaljenog objekta, koji će djelovati kao server u ovom klijent-server modelu.
- Servis kojeg obezbjedjuje server je enkapsuliran kao objekat a interfejs za taj objekat je opisan u IDL-u (**Interface definition language**) .
- Interfejsi definirani u IDL fajlu služe kao ***Ugovor*** (contract) izmedju servera i njegovih klijenata.
- Klijenti interaktiraju sa serverom pozivajući metode koje su opisane u IDL-u. Stvarna implementacija objekta je sakrivena od klijenta.
- Neke objektno orijentirane programske karakteristike su prisutne na nivou IDL-a kao što je enkapsulacija podataka, polimorfizam i jednostruko naslijedjivanje

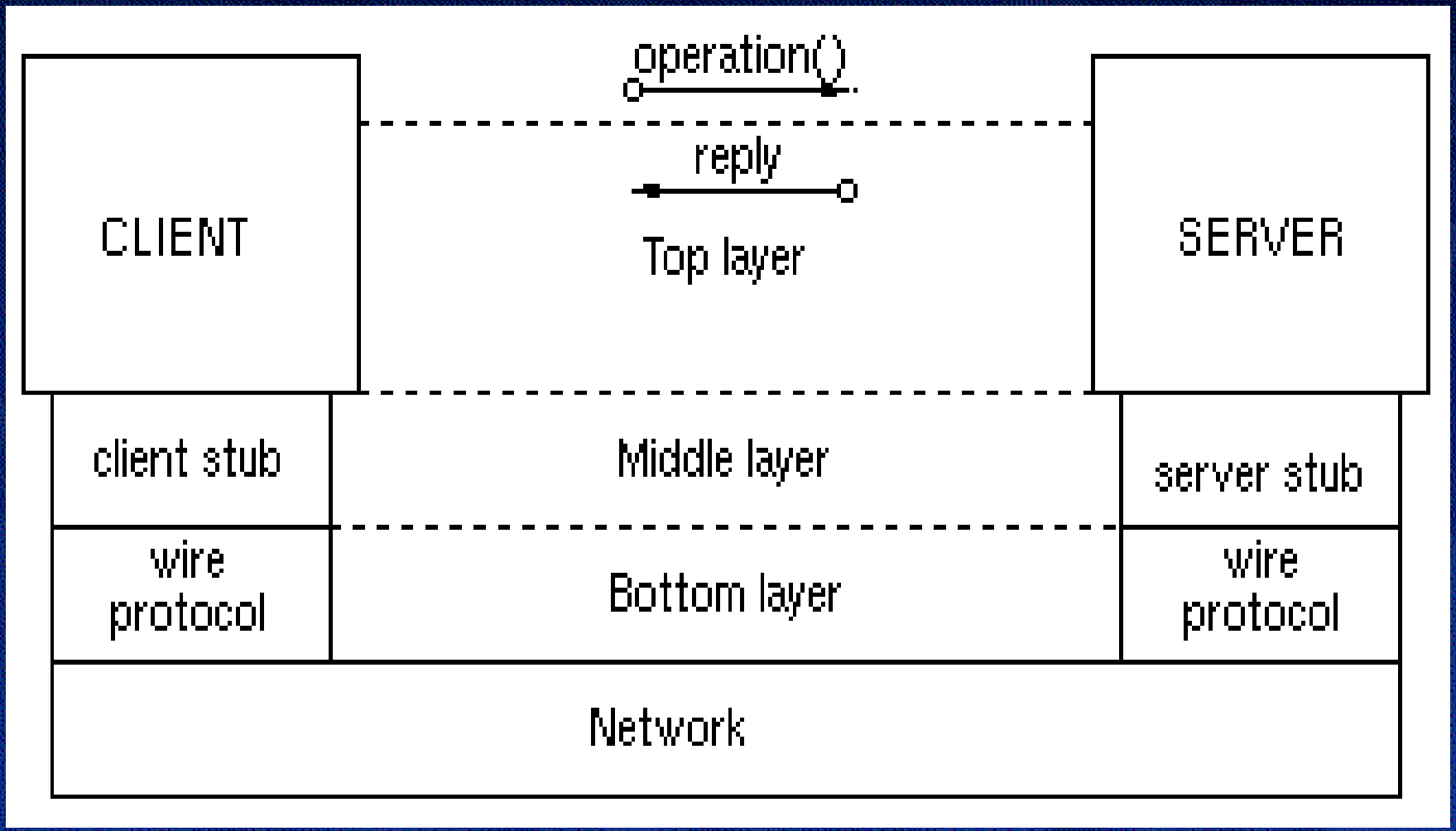
DCOM I CORBA – KARAKTERISTIKE

- **CORBA** takodjer podržava i višestruko naslijedjivanje na nivou IDL-a, dok DCOM ne podržava. Umjesto toga pojam objekta koji ima višestruke interfejse se koristi da se postignu slićne namjene i kod **DCOM**-a.
- **CORBA IDL** može takodjer specificirati i izuzeća.

- I kod DCOM I CORBA interakcije izmedju klijentskog procesa i objektnog servera se implementiraju kao objektno orijentirani RPC tip komunikacije. Naredna slika pokazuje tipičnu RPC komunikaciju.

- Da bi se pozvala udaljena funkcija, klijent šalje poziv ka klijentskom stabu (client stub). Stab pakuje parametre poziva u poruku zahtjeva , i poziva žičani (wire) protokol da bi poslao poruku do servera.

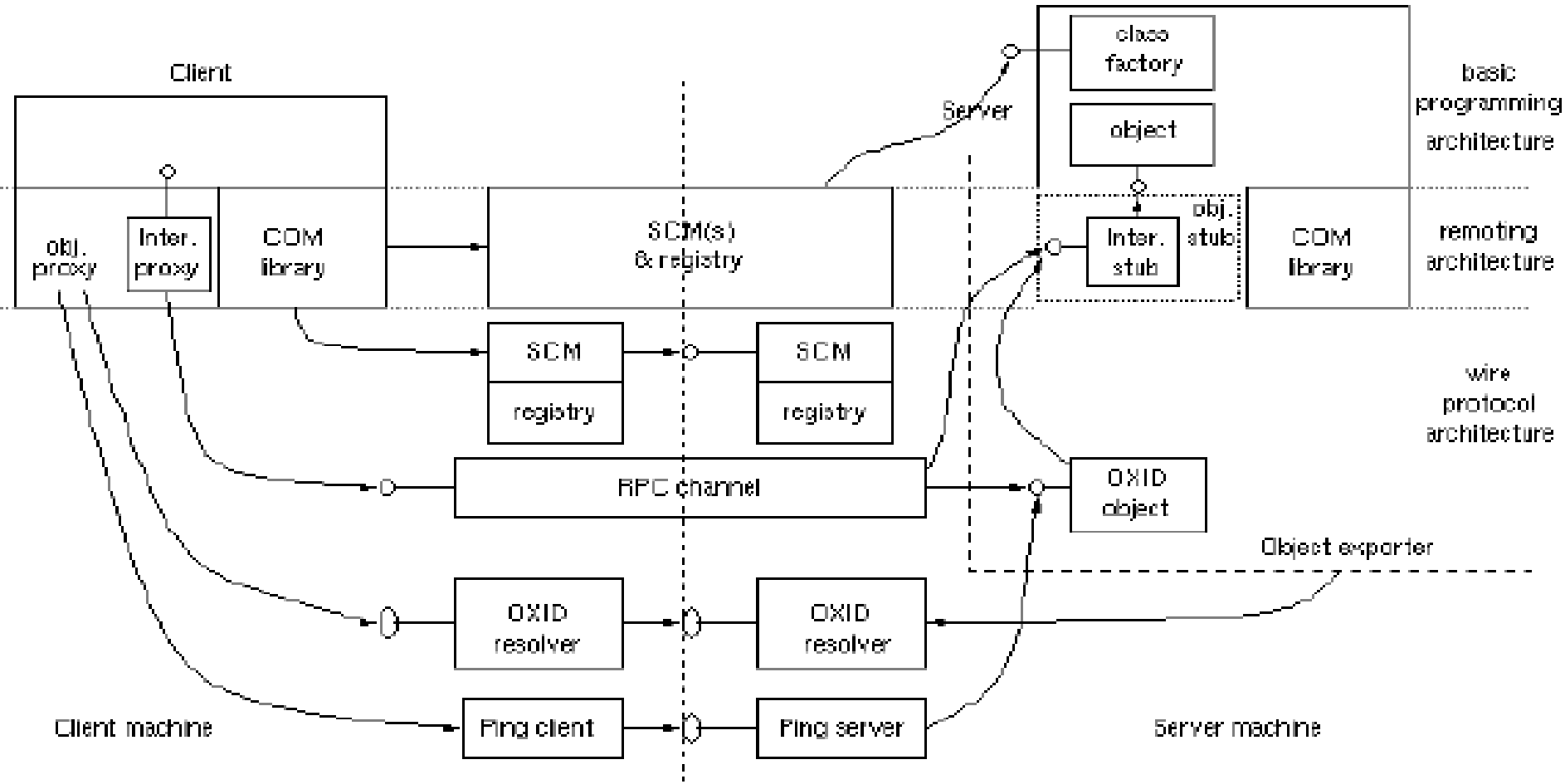
DCOM I CORBA – KARAKTERISTIKE



DCOM I CORBA – KARAKTERISTIKE

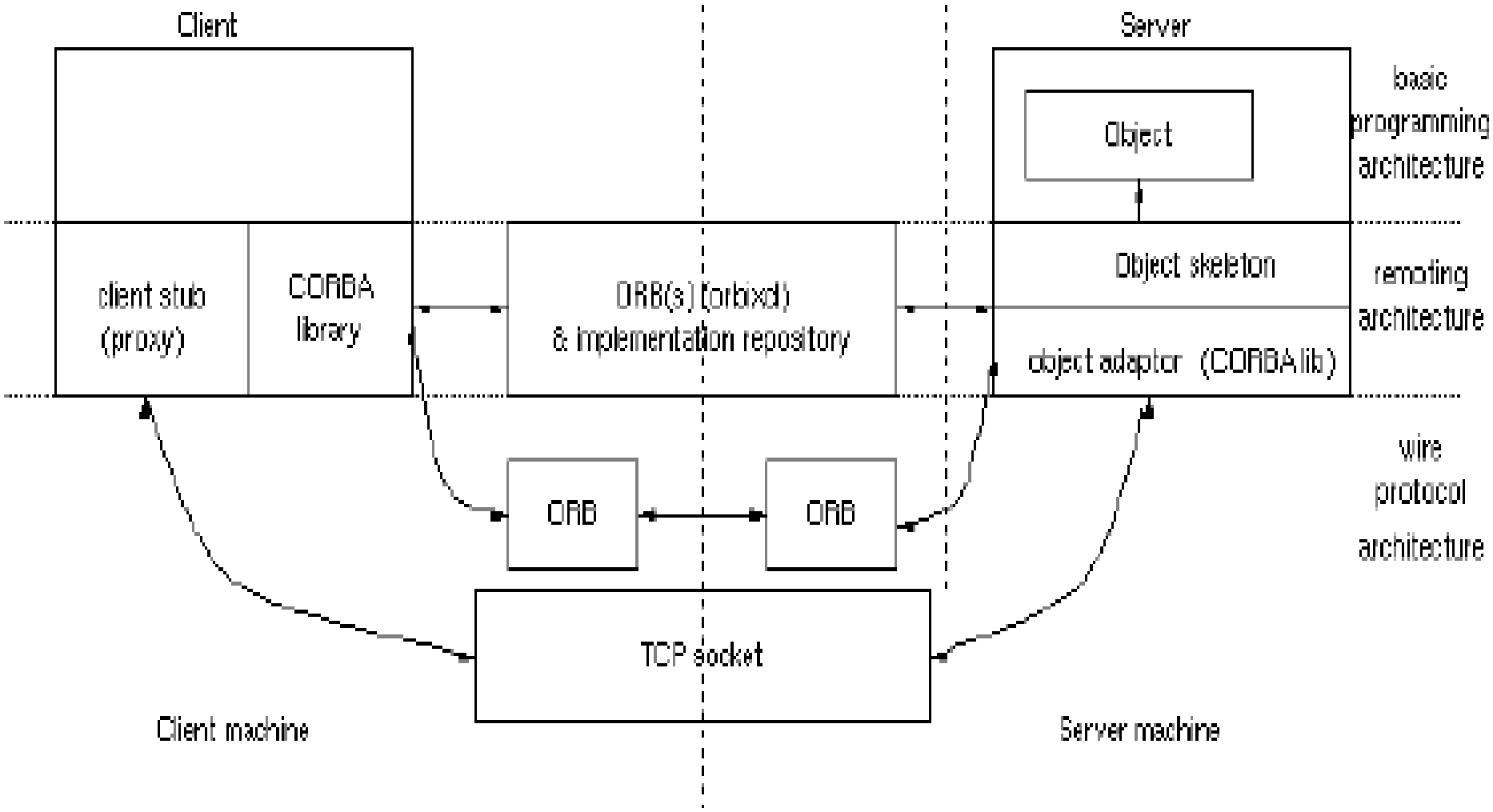
- Na serverskoj strani, wire protokol isporučuje poruku ka serverskom stabu, koji zatim raspakiva poruku zahtjeva i poziva onu funkciju na objektu koja je adekvatna pozivu.
-
- Kod **DCOM**-a klijentski stab se naziva **proxy** a serverski stab se naziva **stub**.
- Kod CORBA se klijentski stab naziva **stub** , a serverski stab se naziva **skeleton**.
- Naredna dva slajda pokazuju ukupnu arhitekturu DCOM i CORBA

DCOM UKUPNA ARHITEKTURA



SCM – SERVICE CONTROL MANAGER ,
OXID- OBJECT EXPORTER IDENTIFIER

CORBA UKUPNA ARHITEKTURA



ORB – OBJECT REQUEST BROKER ,

DCOM I CORBA – ARHITEKTURA

- U okviru arhitekture uočavamo tri sloja:

- Najgornji sloj je **bazna programska arhitektura** koja je vidljiva onima koji razvijaju programe za klijente i objekte servera.

- Srednji sloj je **udaljena arhitektura** (**remoting architecture**) koja na transparentan način čini razumljivim pointere interfejsa ili reference objekata, kroz različite procese.

- Najdonji sloj je **wire protokol arhitektura (wire protocol architecture),** koja dalje proširuje udaljenu arhitekturu da bi mogla da radi kroz različite mašine**.**

GORNJI SLOJ - BAZNA PROGRAMSKA ARHITEKTURA

- Na najgornjem sloju pokazaćemo kako klijent zahtjeva objekat i poziva njegove metode, i kako server kreira instancu objekta i čini je raspoloživom za klijenta.

- Programi klijenta i servera interaktiraju kao da su rezidentni u istom adresnom prostoru na istoj mašini.

- Glavna razlika izmedju DCOM i CORBA na ovom sloju je kako klijent specificira interfejs kao i COM-ov **class factories i Iunknown** interfejs metod**e.**

- U narednoj tabeli ćemo pokazati korak po korak opis i ilustrirati ga na slikama koje slijede poslije tabele, za DCOM a onda za CORBA arhitekturu.

GORNJI SLOJ - BAZNA PROGRAMSKA ARHITEKTURA

DCOM	CORBA
Aktivacija objekta	
1. Klijent poziva iz COM biblioteke CoCreateInstance() , sa CLSID_Grid i IID_IGrid1	1. Klijent poziva klijentski stab: grid::_bind(), koji je statička funkcija u stabu
2. COM infrastruktura starta objektni server za CLSID_Grid	2. ORB starta server koji sadrži objekat koji podržava intefejsni grid
3. Server kreira class factories za sve podržavane CLSID-jeve, i poziva CoRegisterClassObject() da bi registrirao svaku factory. Server blokira dok ćeka, dogadjaj koji bi setovao signal da server nije više potreban. Daljnji dolazeći zahtjevi od klijenta će biti opsluživani od drugih threadova.	3. Server instantinizira sve podržavane objekte. Server poziva : CORBA::BOA::impl_is_ready() da kaže ORB da je spreman da prihvati zahtjev klijenta. BOA-baze object adapter

GORNJI SLOJ - BAZNA PROGRAMSKA ARHITEKTURA

DCOM	CORBA
Aktivacija objekta	
4. COM dobija IClassFactory pointer ka CLSID_Grid factory, i poziva CreateInstance() na njemu.	4. ORB vraća objektnu referencu za grid kao gridVar ka klijentu.
5. U CreateInstance() , server kreira instancu objekta i pravi QueryInterface() poziv da bi dobio interfejs pointer na IID_IGrid1 interfejs	
6. COM vraća interfejs pointer kao pIGrid1 ka klijentu.	

IGrid1 i IGrid2 su dva interfejsa u DCOM za dvije grupe metoda.

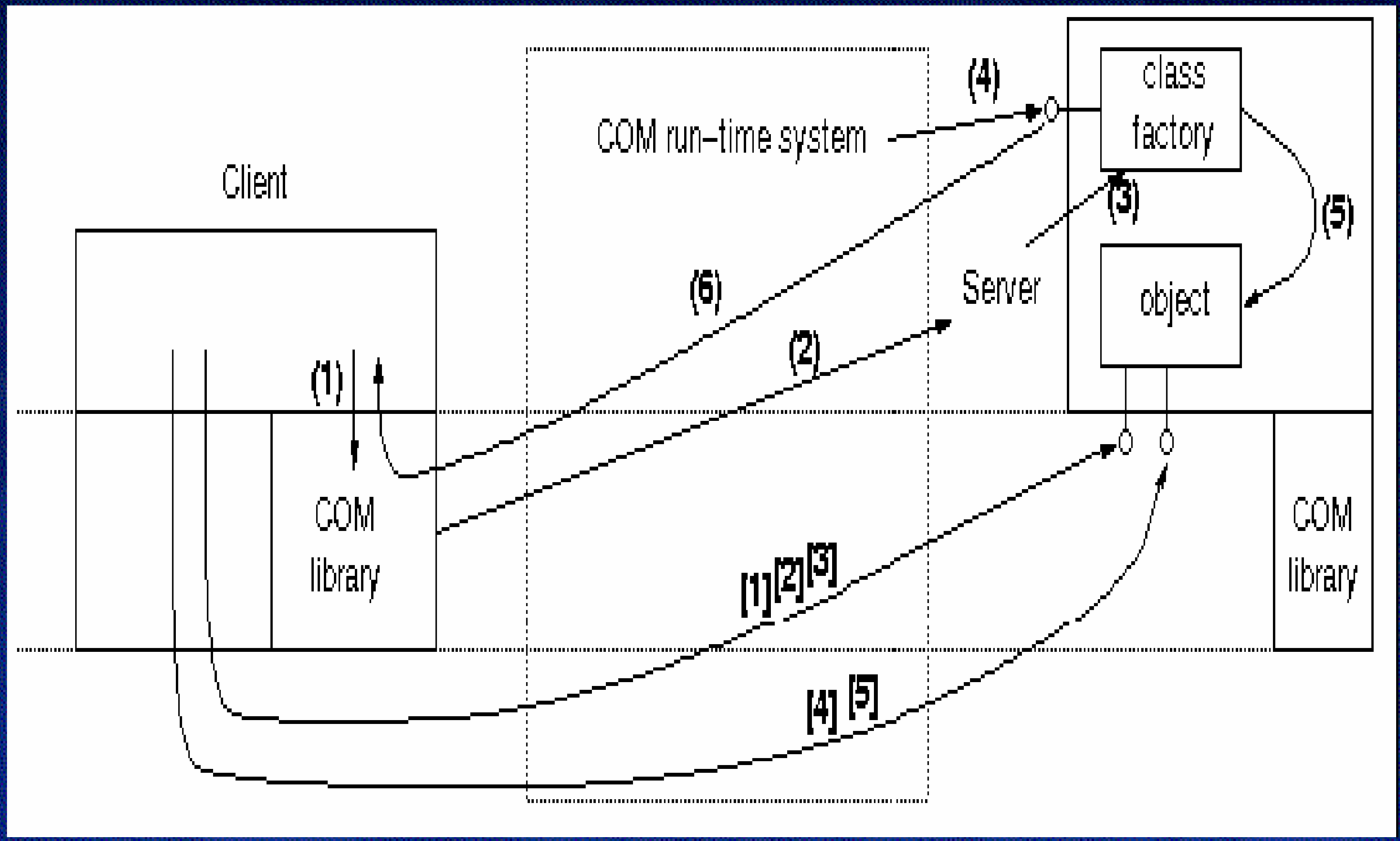
GORNJI SLOJ - BAZNA PROGRAMSKA ARHITEKTURA

DCOM	CORBA
Metod Pozivanja (invocation)	
1. Klijent poziva pIGrid1->get() koji će eventualno pozvati Cgrid::get() kod servera.	1. ORB vraća objektnu referencu za grid kao gridVar ka klijentu.
2. Da se dobije pointer na drugi interfejs IID_IGrid2 od iste instance objekta, klijent poziva pIGrid1->QueryInterface(), koji poziva sa svoje strane Cgrid::QueryInterface.	2. Klijent poziva gridVar-> reset() koji poziva grid_i::reset()
3. Kada završi sa korištenjem pIGrid1, klijent poziva pIGrid1->Release().	

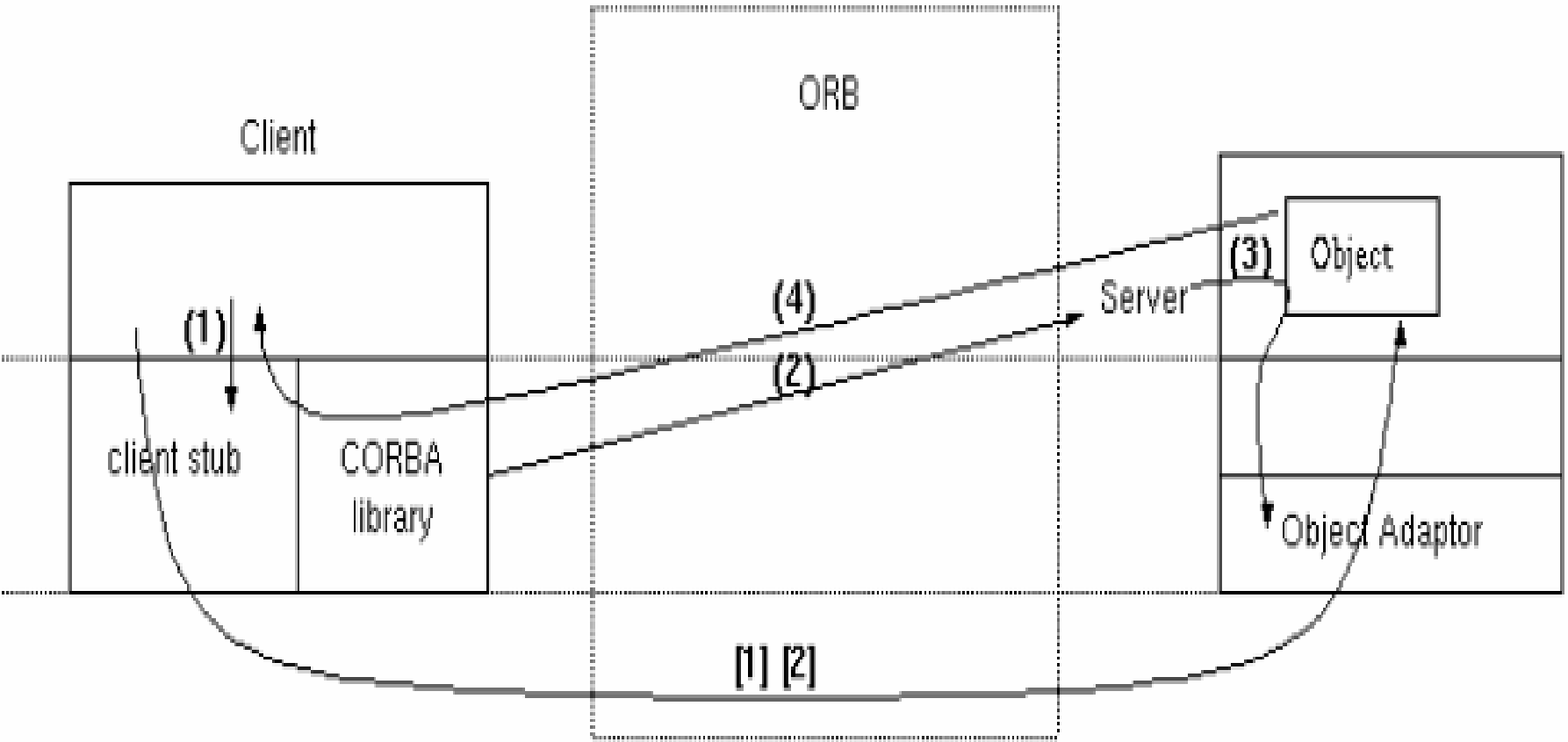
GORNJI SLOJ - BAZNA PROGRAMSKA ARHITEKTURA

DCOM	CORBA
Metod	Pozivanja (invocation)
4. Klijent poziva pIGrid2->reset() koji poziva CGrid::reset.	
5. Klijent poziva pIGrid2->Release() Koji poziva Cgrid::Release()	

NAJGORNJI SLOJ - DCOM KORACI



NAJGORNJI SLOJ – CORBA KORACI



SREDNJI SLOJ – UDALJENA ARHITEKTURA

- Srednji sloj se sastoji od infrastrukture koja je neophodna da obezbjedi klijenta i servera da su oni u istom adresnom prostoru. Odgovarajuće ilustracije blokova ovog sloja su pokazane za DCOM i CORBA arhitekturu na narednim slajdovima.

- Glavna razlika izmedju DCOM i CORBA na ovom sloju je kako su serverski objekti registrirani i kada su instance proxy/stub/skeleton kreirane.

- Da bi se poslali podaci kroz različite adresne prostore zahtjeva se proces koji se naziva **marshalling** i **unmarshalling**.

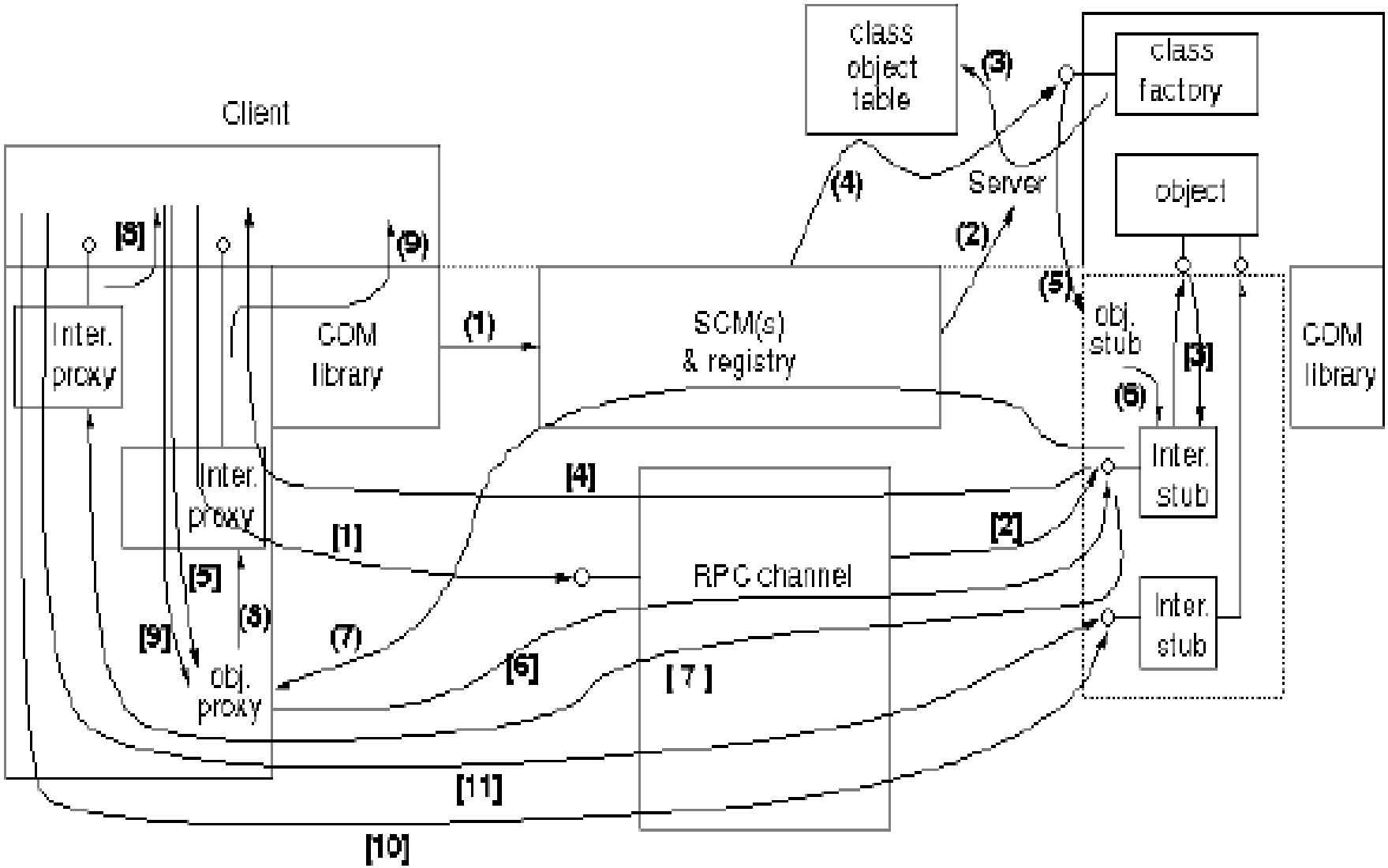
SREDNJI SLOJ – UDALJENA ARHITEKTURA

- Marshalling pakuje pozivne parametre metoda (na strani klijenta) ili povratne vrijednosti (u prostoru servera) u standardni format za transmisiju.

- Unmarshalling je obrnuta operacija, koja raspakiva standardni format u odgovarajući način predstavljanja podataka u adresnom prostoru procesa koji je primio poruku.

- Implementirajući **Imarshall** interfejs, serverski objekat deklariše da on hoće da kontroliše kako i koji se podaci maršaliraju i demaršaliraju, i kako klijent treba da komunicira sa serverom. (kod custom marshaling mehanizma).

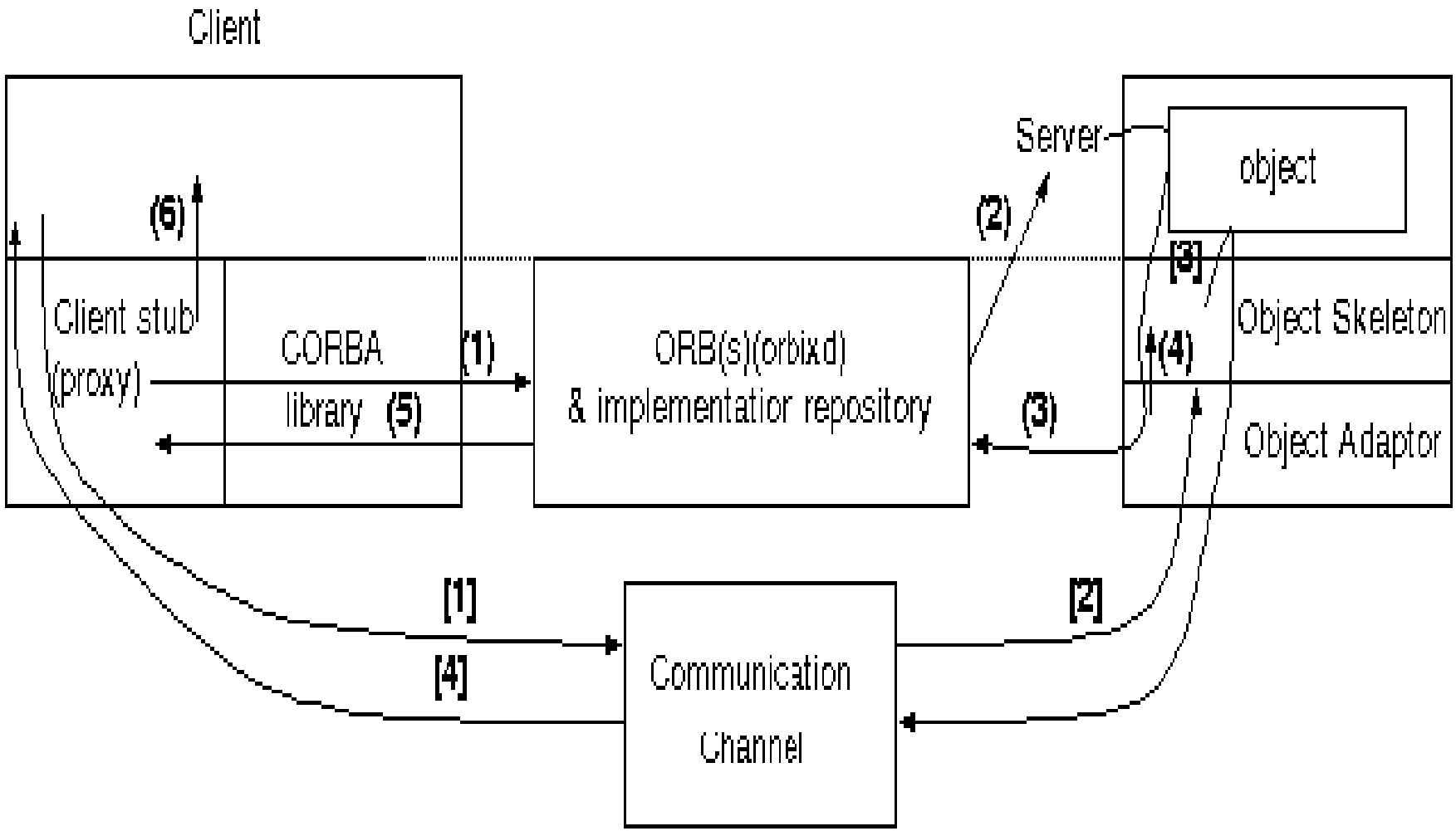
SREDNJI SLOJ – DCOM KORACI



SREDNJI SLOJ – UDALJENA ARHITEKTURA

- Kod CORBA , ORB djeluje kao objektni bas. Objektni adapter (OA), sjedi na vrhu ORB, i odgovoran je za konektiranje objektne implementacije na ORB.
- Objektni adapteri obezbjedjuju servise kao što su: generiranje i interpretacija objektnih referenci, metode pozivanja, aktivacija i deaktivacija objekta, mapiranje referenci objekta na implementacije.
- Bazni objektni adapter (BOA) definira objektni adapter koji može biti korišten za najkonvecionalnije implementacije objekta.
- U novije vrijeme POA (Portable object adapter) je uveden kao zamjena za BOA.

SREDNJI SLOJ – CORBA KORACI



DONJI SLOJ – WIRE PROTOKOL ARHITEKTURA

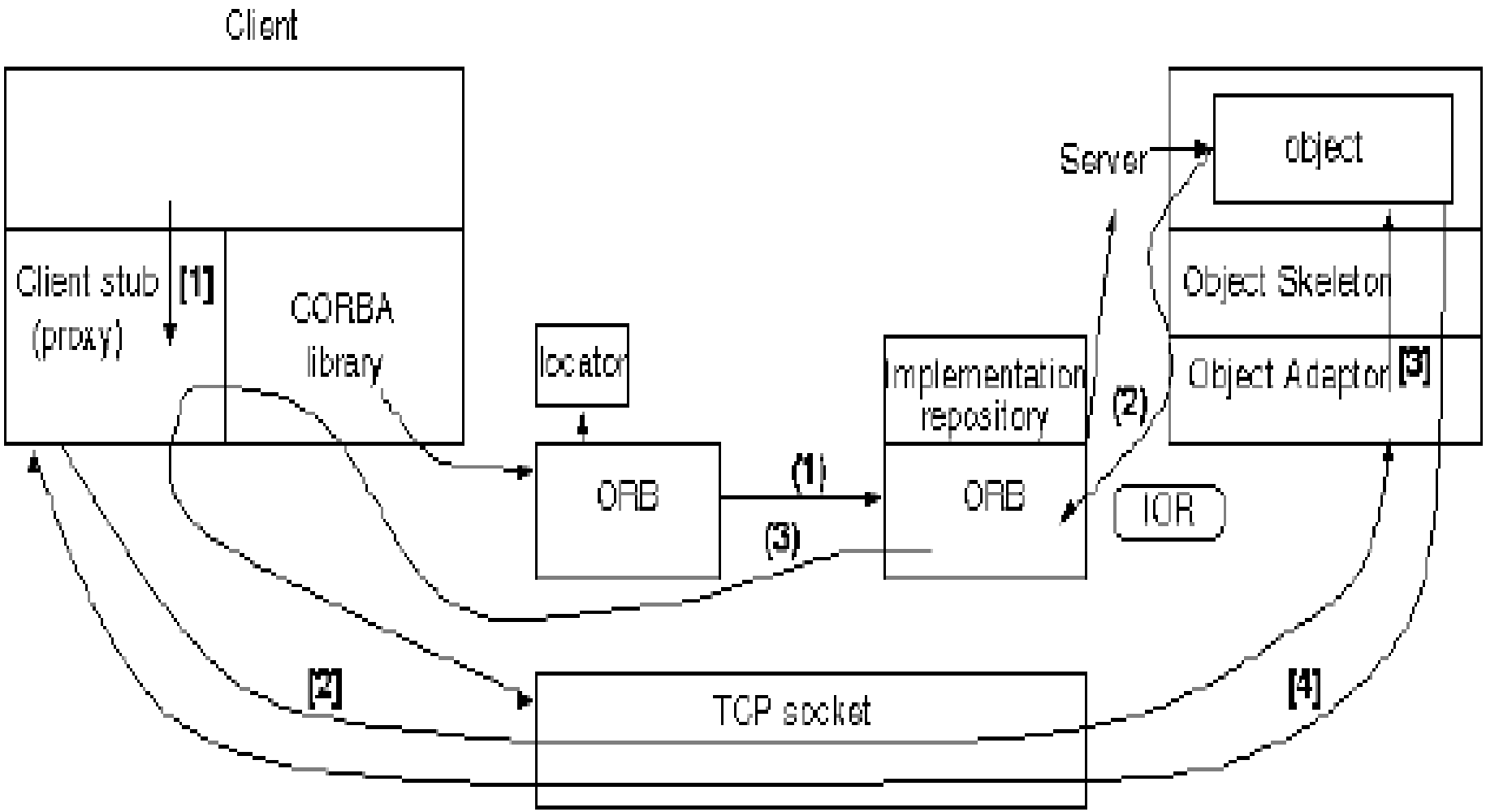
- Najdonji sloj specificira wire protokol koji podržava klijenta i server koji su na različitim mašinama. Naredni slajdovi pokazuju korake kroz koje prolaze DCOM i CORBA kroz ovaj sloj.

- Glavna razlika izmedju DCOM i CORBA na ovom sloju uključuje kako daljinski pointeri interfejsa ili reference objekta su predstavljeni da prenesu serversku informaciju ka klijentu, kao i standardni format u kojem su podaci maršalirani za transmisiju u heterogenom okruženju.

DONJI SLOJ – WIRE PROTOKOL ARHITEKTURA

- Primjetimo da CORBA ne specificira protokol za komunikaciju izmedju klijenta i objektnog servera koji se izvršavaju na ORB koji je obezbjedio isti proizvodjač (Vendor).
- Protokol za inter-ORB komunikaciju izmedju ORB-jeva istog Vendora , zavisi od Vendora.
- Medjutim da bi se podržavala interoperabilnost izmedju različitih ORB proizvoda, specificiran je opšti Inter-ORB protokol (General Inter-ORB Protocol-GIOP).
- Specifično mapiranje GIOP na TCP/IP konekciju je definirano, i poznato je kao Internet Inter-ORB protokol (IIOP).

DONJI SLOJ – CORBA KORACI



DONJI SLOJ – WIRE PROTOKOL ARHITEKTURA

- DCOM wire protokol je uglavnom baziran na OSF DCE RPC specifikaciji. Ova specifikacija uključuje predstavljanje objektne reference udaljenog objekta, kao IRemUnknown interfejs za optimiziranje performanse udaljenih Iunknown metoda poziva, kao i pinging protokol.
- Pinging dozvoljava serverskom objektu da prikuplja odbaćene reference udaljenog objekta (garbage-collect) kada udaljeni klijent abnormalno završi sa radom.
- Kada klijent dobije interfejs pointer na udaljeni objekat po prvi put, ping klijentski kod na klijentskoj mašini , dodaje objekat u ping set i periodično šalje ping na serversku mašinu da bi rekao serveru da je klijent još živ.

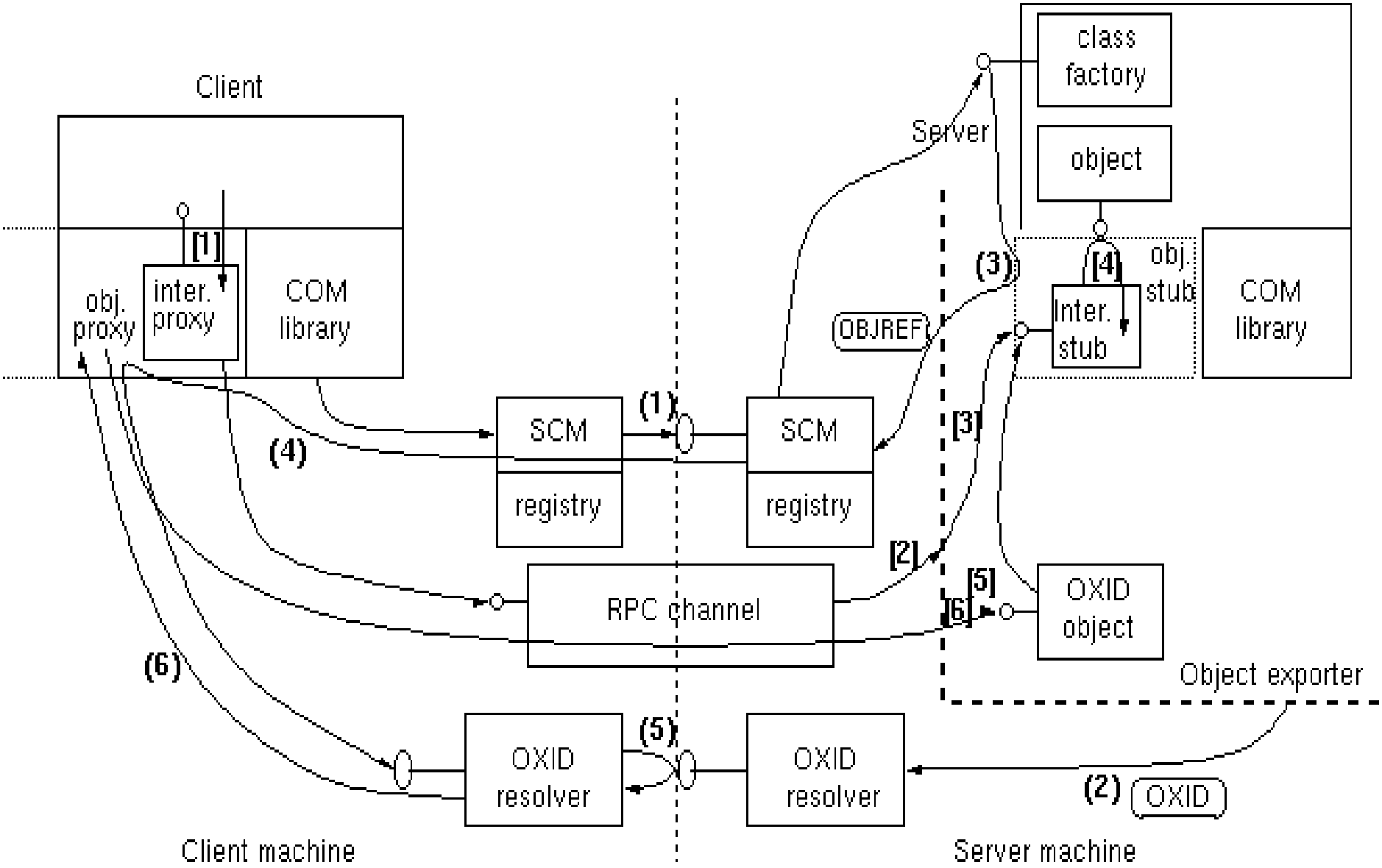
DONJI SLOJ – WIRE PROTOKOL ARHITEKTURA

- Nedolaženje prethodno odredjenog broja konsekutivnih pingova indicira serveru da je klijent nenormalno završio sa radom, te interfejs pointer koji je on držao može biti oslobodjen.

- Da bi se optimizirala performansa, pingovi se šalju po mašini i na inkrementalni način. Oni takodjer mogu biti piggy-backed na normalne poruke.

- Kada je god to neophodno, funkcionalnost pinga se takodjer može isključiti da bi se smanjio saobračaj kroz mrežu.

DONJI SLOJ – DCOM KORACI



DCOM I CORBA – SUMARNO

- Opis troslojne arhitekture DCOM i CORBA je pokazao da su one vrlo slićne.
- Obadvije obezbjedjuju distribuiranu objektnu infrastrukturu za transparentno aktviranje i pristupanje udaljenim objektima.
- Naredna tabela daje sumarni pregled termina i entiteta za dvije arhitekture.
- DCOM podržava objekte sa višestrukim interfejsima i obezbjedjuje standardni QueryInterface() metod da navigava izmedju interfejsa.
- Ovo takodjer uvodi pojam objektnog proxy/stub-a koji dinamički puni višestruke interfejsne proxy/stub-ove u daljinski sloj.
- Ovakav koncept ne postoji kod CORBA arhitekture.

DCOM I CORBA – SUMARNO

- Svaki CORBA interfejs naslijedjuje od **CORBA::Object,** konstruktor koji implicitno izvršava takve zajedničke taskove kao što je registracija objekta, generiranje, reference objekta, instantinizacija skeletona, itd.

- Kod DCOM ovakvi taskovi se izvršavaju eksplicitno od strane serverskog programa, ili se dinamički izvršavaju od strane DCOM-a u run-time-u.

- DCOM wire protokol je strogo vezan za RPC, dok kod CORBA to nije slučaj.

DCOM I CORBA – SUMARNO

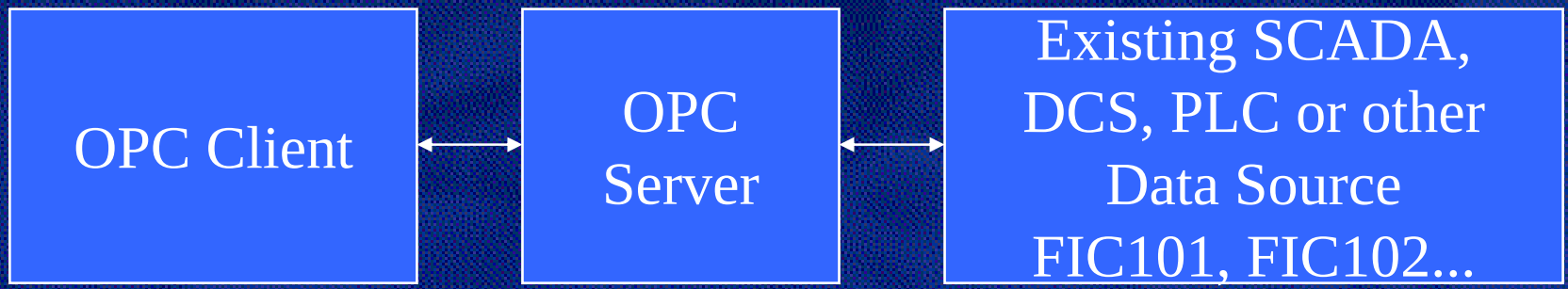
Karakteristike	DCOM	CORBA
Gornji sloj : Bazna programska arhitektura		
Common base class	Iunknown	CORBA::Object
Object class Identifier	CLSID	Ime interfejsa
Interface identifier	IID	Ime interfejsa
Client-side object activation	CoCreateInstance()	Poziv metoda **bind**
Object handle	Pointer interfejsa	Referenca objekta
Srednji sloj: Udaljena arhitektura		
Mapiranje imena u implementaciju	Registar	Implementacioni repositorij
Tip informacije za metode	Tip biblioteka	Interfejs repozitorij
Lokacija implementacije	SCM	ORB

DCOM I CORBA – SUMARNO

Karakteristike	DCOM	CORBA
Srednji sloj: Udaljena arhitektura		
Implementacija aktivacije	SCM	OA
Client-side stub	proxy	Stub/proxy
Server side stub	stub	skeleton
Donji sloj: Wire protokol arhitektura		
Server end point resolver	OXID resolver	ORB
Server endpoint	Objekt eksporter	OA
Referenca objekta	OBJREF	IOR
Generiranje reference objekta	Objekt eksporter	OA
Marshalling formata podataka	NDR(network data representation)	CDR
Identifikator instance interfejsa	IPID (Interface pointer identifier)	Object_key

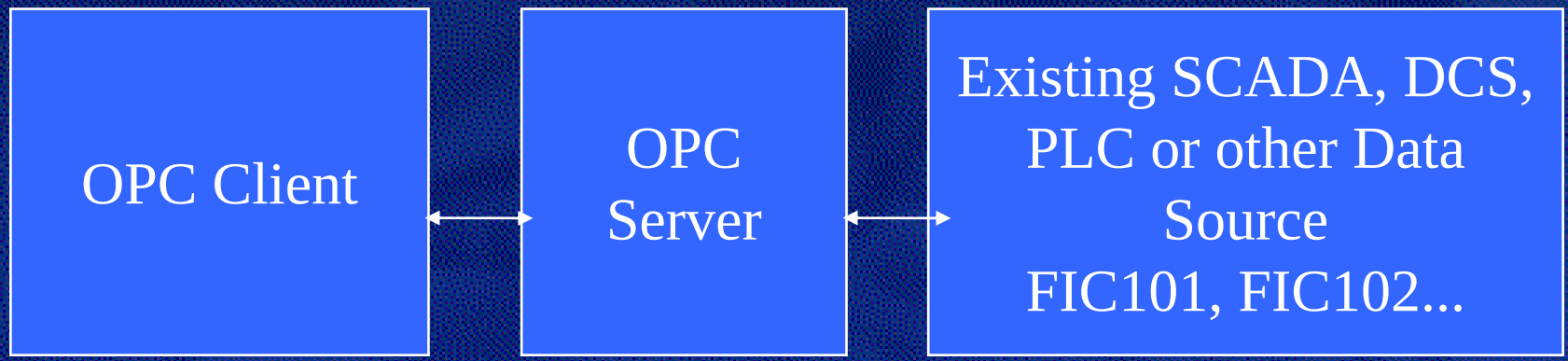
Predpostavke o OPC arhitekturi

- Svaki OPC server kao što je DA ili A&E je poseban objekat.
- DA Server obezbjedjuje prozor u Postojeće podatke: on nije poseban konfiguracioni sistem.
- Podacima se pristupa preko imena (string), koje je u opštem slućaju specifično ili za vendora ili hadver.
- Podaci koji su na listama itema se mogu čitati eksplicitno (polling), ili se mogu kreirati pretplate (subscription).



Predpostavke o OPC arhitekturi

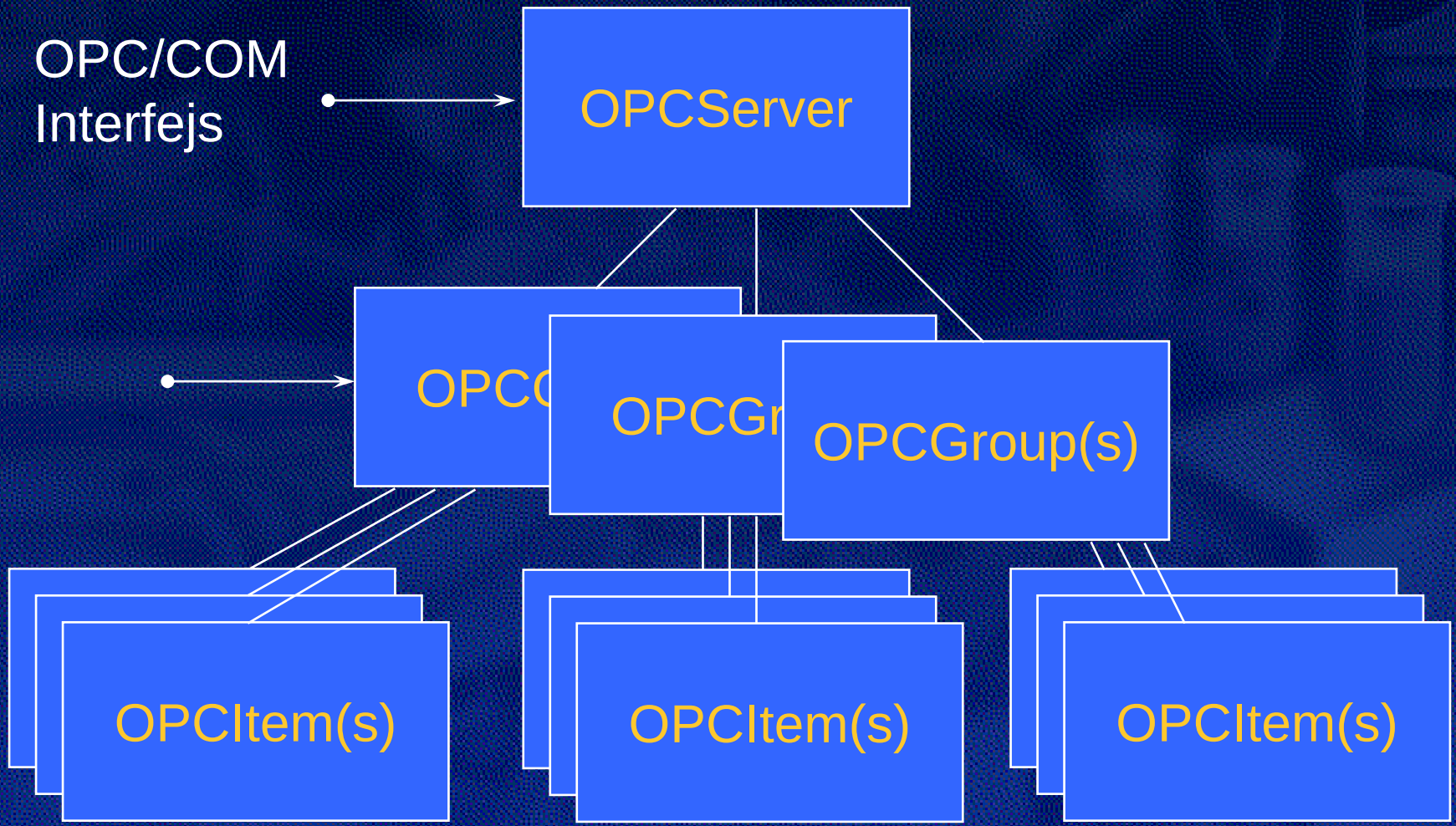
- Osnovna predpostavka o OPC Serveru je da postoji neka vrsta kontrolne ili nadzorne mašine koja se izvršava nezavisno od bilo čega što OPC Server radi, i OPC server je ustvari samo prozor u ovu data engine mašinu.
- Ova predpostavka vodi logično do modela koji je prikazan na slijedećoj slici:



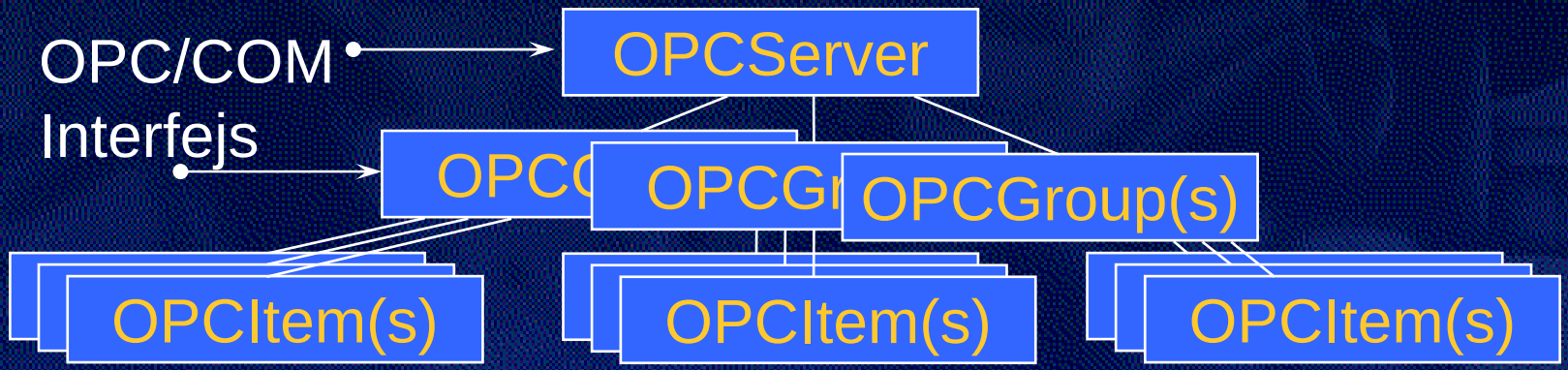
Predpostavke o klijentskim aplikacijama

- Klijentske aplikacije su zainteresovane za podskup podataka itema (Tagovi), koji su raspoloživi kod OPC servera u cashu iz I/O podsistema na koji je OPC server nadogradjen
- Aplikacije su zainteresirane za mnoge različite podskupove itema podataka u različitim trenutcima vremena i mogu imati promjenjljive zahtjeve na vrijeme osvježavanja i rezoluciju.
- Aplikacije žele da budu nezavisne od strukture podataka (ili objekata) koje koriste podsistemi na kojima je OPC server nadogradjen (tj. one žele imati simbolički pristup podacima).

Logički Objektni Model



Logički Objektni Model



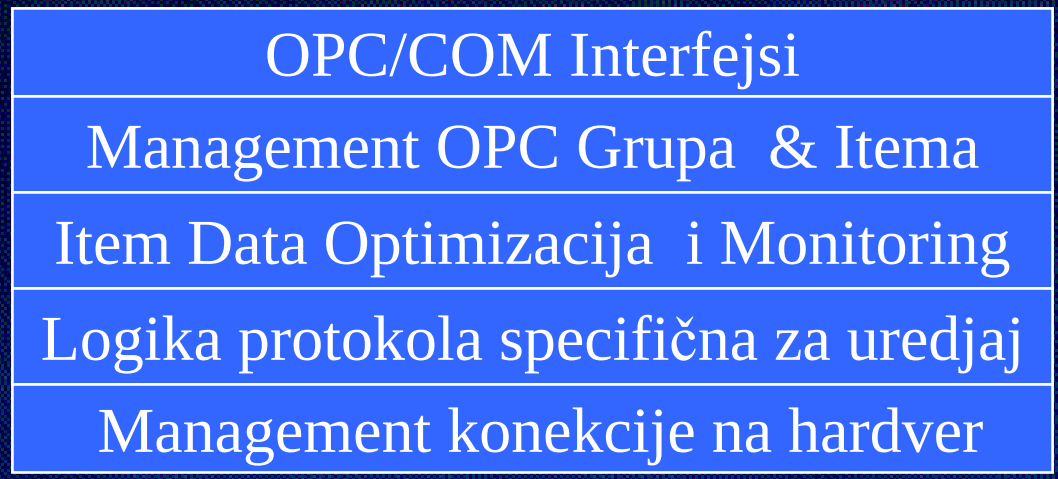
Vidimo da je OPC Server COM objekat na koji se konektira aplikacija. OPC grupe se dinamički kreiraju, od strane aplikacija (klijenata), i sadrže liste tagova i njihove atribute (u OPC terminologiji se zovu detalji- Items). Jedna klijentska HMI aplikacija može kreirati grupu za svaki svoj ekran. Izvještaji u okviru HMI mogu takodjer kreirati svoju grupu tagova koji oni koriste za svoje izvještavanje. Sadržaj Grupe i skupa itema može dinamički varirati u vremenu, zavisno od potreba aplikacije. (ovo nema uticaja na onu kontrolnu data mašinu koja se nalazi u pozadini ovoga).

Logički Objektni Model

Ovo je donekle analogno sa **RowSets** kako se koristi kod OLEDB. Postoji mnogo podataka u bazi, mi želimo da gledamo podskup njih u našoj aplikaciji, i taj podskup može varirati u vremenu zavisno od potreba aplikacije. Postoje dva bazna pristupa kojima se ovo moglo riješiti: pomenuti ‘’rowset’’ pristup koji je primjenjen , ili ‘’čistiji’’ objektno orijentisani pristup kod kojeg mi ‘’proxiramo’’ procesne objekte i izlažemo njihove osobine i metode prema aplikacijama. Softverski dizajneri u OPC Fondaciji i implementatori OPC Server/Klijent tehnologije su osjetili da bi ovaj drugi čisto objektno orijentisani pristup bio pretvrd u smislu efikasnosti komunikacije i razmjene podataka, uz probleme kao što su da se svi Vendori (softverski proizvodjači) slože o strukturi objekata i njihovim osobinama i metodama. Nadalje, ovakav pristup ne bi dozvolio da se eksponiraju na ovaj način podaci iz neke OODB (object oriented data base).

Tipični dizajn Servera

Tipični dizajn Servera



SERVER obezbjedjuje menadjment OPC grupe kao i kontrole, posreduje i optimizira pristup fizičkim uredjajima od strane višestrukih klijenata. Server je u suštini I/O drajver koji ‘’razumije’’ kako da razgovara sa nekim specifičnim provajderom podataka (hardver i softver), i nakon toga eksponira podatke od ovog data provajdera preko standardnog OPC interfejsa

Serverski Interfejsi

- Server je COM objekat koji obezbjedjuje interfejse:
- **IOPCServer**
- **IOPCBrowseServerAddressSpace (optional)**
- **IOPCCommon (2.0)**
- **IOPCItemProperties(2.0)**
- **IConnectionPointContainer(2.0)**

Ovi interfejsi, preko pridruženih metoda, dozvoljavaju aplikaciji da putem OPC klijenta može da:

- kreira i briše grupe
- da pretražuje raspoložive tagove (iteme)
- Da odredi atribute ili polja pridružena sa tagom
- Da dobije informaciju o statusu servera
- Da bude obavještena kada server ode u shutdown

Serverski Interfejsi

- Dakle , Server je COM objekat koji obezbjedjuje slijedeće interfejse:

- **IOPCServer**
- **IOPCBrowseServerAddressSpace (optional)**
- **IOPCCommon (2.0)**
- **IOPCItemProperties(2.0)**
- **IConnectionPointContainer(2.0)**

Serverski Interfejsi

- **IOPCServer** – je glavni interfejs za OPC server. OPC server se registruje kod operativnog sistema

- **IConnectionPointContainer** – obezbjedjuje pristup ka konekcionoj tačci za IOPCShutdown

  (pojedinačno za svaki od klijenata)
- **IOPCCommon** – obezbjedjuje uspostavljanje upita za LocaleID, koji će se koristiti za datu klijent server sesiju.
- **IOPCBrowseServerAddressSpace** (optional) – ovaj interfejs obezbjedjuje način kako klijenti browsuju raspoložive data iteme kod servera, dajući mu listu definicija za ITEM ID.
- **IOPCItemProperties**(2.0) – ovaj interfejs koriste klijenti da browsuju za raspoložive osobine (atribute ili parametre), udružene sa ITEMID, kao i da čitaju tekuće vrijednosti ovih osobina.

Interfejsi za objekat grupe

- Objekat grupe je COM objekat koji obezbjedjuje:
- **IOPCGroupStateMgt**
- **IOPCAsyncIO2 (2.0 - replaces IOPCAsyncIO)**
- **IOPCItemMgt**
- **IOPCSyncIO**
- **IConnectionPointContainer (2.0 replaces IDataObject)**

Interfejsi za objekat grupe

Interfejsi na nivou grupe omogućavaju aplikaciji da putem OPC klijenta može:

- da dodaje ili sklanja iteme iz sastava grupe
- da uravlja sa brzinom osvježavanja (update rate) podataka u grupi
- da čita ili upisuje na vrijednosti itema u grupi
- da se pretplati na podatke u grupi na bazi izuzeća (exception)

Primjetimo da item nije definisan kao COM objekat , i ne eksponira nikakve interfejse u kastom modelu. Postoji niz razloga za ovo a uglavnom su vezani za efiasnost i kompleksnost Servera.

Interfejsi za serversku grupu

- **IOPCGroupStateMgt** – omogućava klijentu da upravlja sa stanjem grupe. Primarno mu omogućava da mjenja brzinu ažuriranja, kao i aktivno stanje grupe.

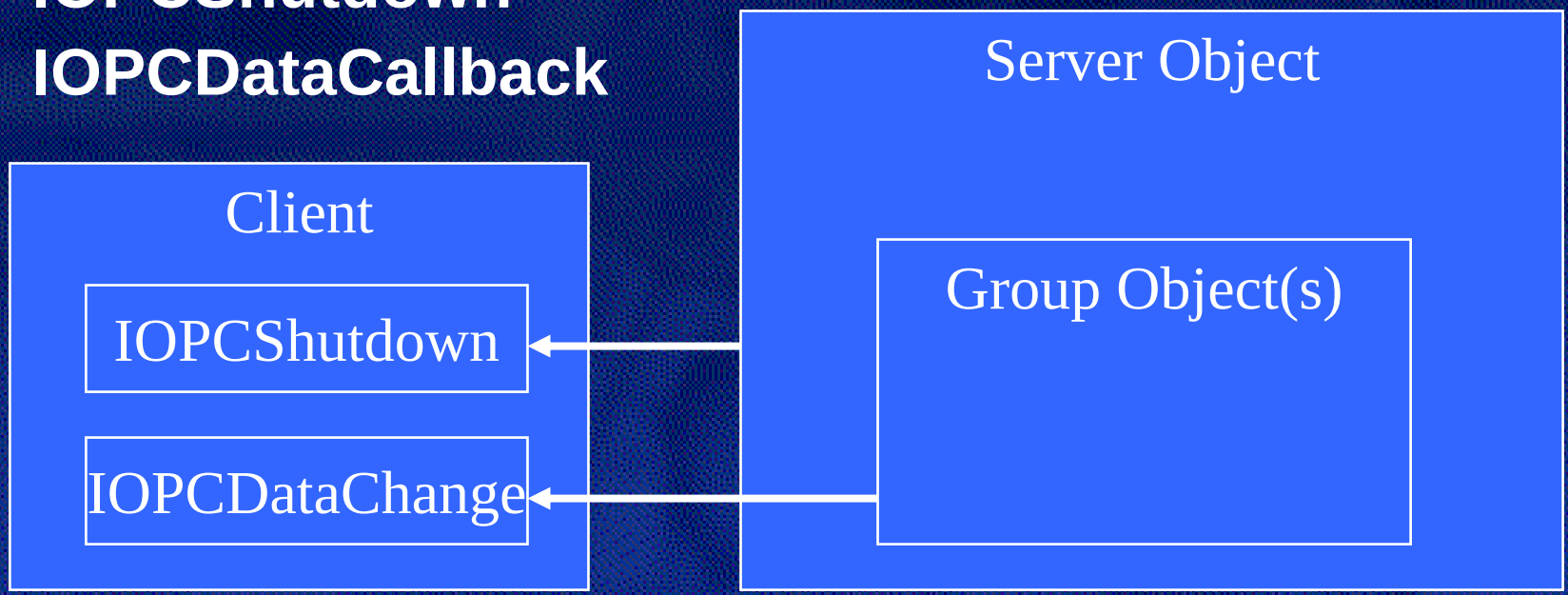
- **IOPCAsyncIO2** (2.0 – zamjenio je IOPCAsyncIO)- ovaj interfejs se koristi da kontrolira konekciju uspostavljenu sa IconnectionPoint. Dozvoljava klijentu da izvršava asinhrone read i write operacije na serveru.

- **IOPCItemMgt** - omogućava klijentu da dodaje, otklanja, i kontrolira ponašanje itema u grupi.
- **IOPCSyncIO** – omogućava klijentu da izvršava sinhrone read i write operacije na serveru.
- **IConnectionPointContainer** – ovaj interfejs obezbjedjuje funkcionalnost Callback konekcije izmedju klijenta i grupe za efikasan prenos podataka (mnogo itema po transakciji).

Interfejsi na klijentskoj strani

- Klijent obezbjedjuje 2 COM interfejsa koje server može pozivati:

- **IOPCShutdown**
- **IOPCDataCallback**



Čitanje i upisivanje podataka

- Postoje tri načina da klijent dobije podatke

- **** IOPCSync::Read*** (iz cache memorije ili iz registara fizičkog uredjaja)
- * ***IOPCAsyncIO2::Read*** (sa uredjaja)
- * ***IOPCCallback::OndataChange()*** , bazirano na izuzeću , koje može takodjer biti trigerovano sa *IOPCAsyncIO2::Refresh*
- Nadalje postoje dva načina da se izbace podatci preko OPC servera na fizički izlaz:

- • ***IOPCSyncIO::Write***
- • ***IOPCAsyncIO2::AsyincWrite***

Općenito o grupama

- Svaka grupa ima ime. Za privatne grupe ime mora biti jedinstveno za sve privatne grupe koje pripadaju tom klijentu.
- Za javne grupe ime mora biti jedinstveno medju svim javnim grupama. Dok klijent može promjeniti ime privatne grupe, ime javne grupe ne može biti promjenjeno.
- Privatna grupa i javna grupa mogu imati isto ime samo ako klijent nije povezan sa javnom grupom istog imena.
- Imena grupa su *case sensitive*. To znači da Grupa1 je različita od grupa1.

Općenito o grupama

- Grupe i detalji (items) unutar grupa imaju flagove aktivnosti (active flag).
- Aktivno stanje grupe se održava nezavisno od aktivnog stanja detalja .
- Promjena stanja aktiviteta grupe ne mjenja stanje detalja.
- Klijenti setuju i resetuju flagove aktiviteta za grupe i detalje kao efikasan način i alternativa dodavanju i uklanjanju grupa i detalja u njima.
- Ako je na HMI displeju prikaz minimiziran , onda detalji koji dolaze sa servera na taj prikaz se neće prikazivati i klijent može resetovati njihove flagove tako da mu ih server ne šalje i time rastereti komunikaciju.
- Drugi način minimizacije ovog saobračaja je poziv ***OnDataChange*** unutar adresnog prostora klijenta.

Javne gupe

- **IOPCServerPublicGroups** (opcioni interfejs)
- Ovaj opcioni interfejs dozvoljava management javnih grupa
- Moguće je dizajnirati aplikaciju tako da iste grupe podataka se koriste od strane više klijenata. U ovakvim slućajevima opciona mogućnost servera sa javnim grupama obezbjedjuje pogodan mehanizam za klijente i servere da dijele ove grupe.
- Javne grupe mogu biti kreirane od strane servera ili može ih kreirati i klijent.
- Kada ih kreira klijent, one se prvo kreiraju kao privatne grupe a onda se konvertuju u javne grupe sa pozivom **MoveToPublic**.

Javne gupe

- Klijent može izbrojati (enumerate) raspoložive javne grupe po imenima koristeći :
- **IOPCServer::CreateGroupEnumerator**. Može se spojiti ('*connect*') na javnu grupu pozivajući **GetPublicGroupByName**.
- Može ispitati sadržaj grupe putem : **IEnumOPCItemAttributes**.
- Može doznačiti klijentske handles i tipove podataka koji su poželjni za specifičnog klijenta koristeći razne **IOPCItemMgt** funkcije.
- Kada se klijent spoji na javnu grupu, on se ponaša vrlo slićno kao i kod spajanja na privatnu grupu. On može aktivirati ili deaktivirati grupu ili detalje u grupi. On može postaviti klijent handles za grupu i detalje unutar grupe. On može postaviti zahtjevani tip podataka za detalje u grupi sa kojim mu server treba slati te podatke.

Osnovne predpostavke

- Svaki OPC Server kao što je i ovaj za Alarme i dogadjaje je poseban objekat koji je komplementaran sa ostalim OPC objektima.
- Alarmi i dogadjaji su bazirani na pretplatama (subscription)
- Tipovi dogadjaja su:
  - Jednostavni (simple)
  - Praćenje (tracking)
  - Uslov (alarm)
- Uslovi (Conditions) mogu biti:
  - sa jednim stanjem (single state)
  - sa više stanja (multi state)

Osnovne predpostavke

Interfejsi za alarme i dogadjaje su na posebnim COM objektima koji su različiti od OPC DA servera. Ovo dozvoljava da se mogu modularno implementirati. Interfejsi su, za razliku od OPC DA, bazirani na pretplatama (subscription) i nisu polirani. Dogadjaji koji stižu od OPC A&E Servera ka klijentu putem pretplate se, kako smo već pokazali na prethodnom slajdu, mogu svrstati u tri klase:

- Jednostavni (simple), što znači da se radi o informacionom tekstu koji nema neko konkretno stanje vezano za njega.
- Praćenje (tracking), što je u suštini tekst koji indicira da je neki aktor izvršio neku akciju nad ciljem
- Uslov (alarm), koji su najčešće oni za koje smo zainteresirani u praćenju i nadzoru procesa i poslovnih dogadjaja u realnom vremenu.

Osnovne predpostavke

Uslovi su obićno definisani kao varijable sa tri stanja :

- omogućen
- aktivan
- potvrdjen

Uslovi mogu biti udruženi sa oblastima. Ovo će se odraziti na dvije osobine interfejsa : filtriranje i pregledanje oblasti.

Uslovi na koje se klijent pretplaćuje imaju imena koja će biti slata klijentima. Ova imena predstavljaju ujedno i mogućnost da se koriste kod filtriranja.

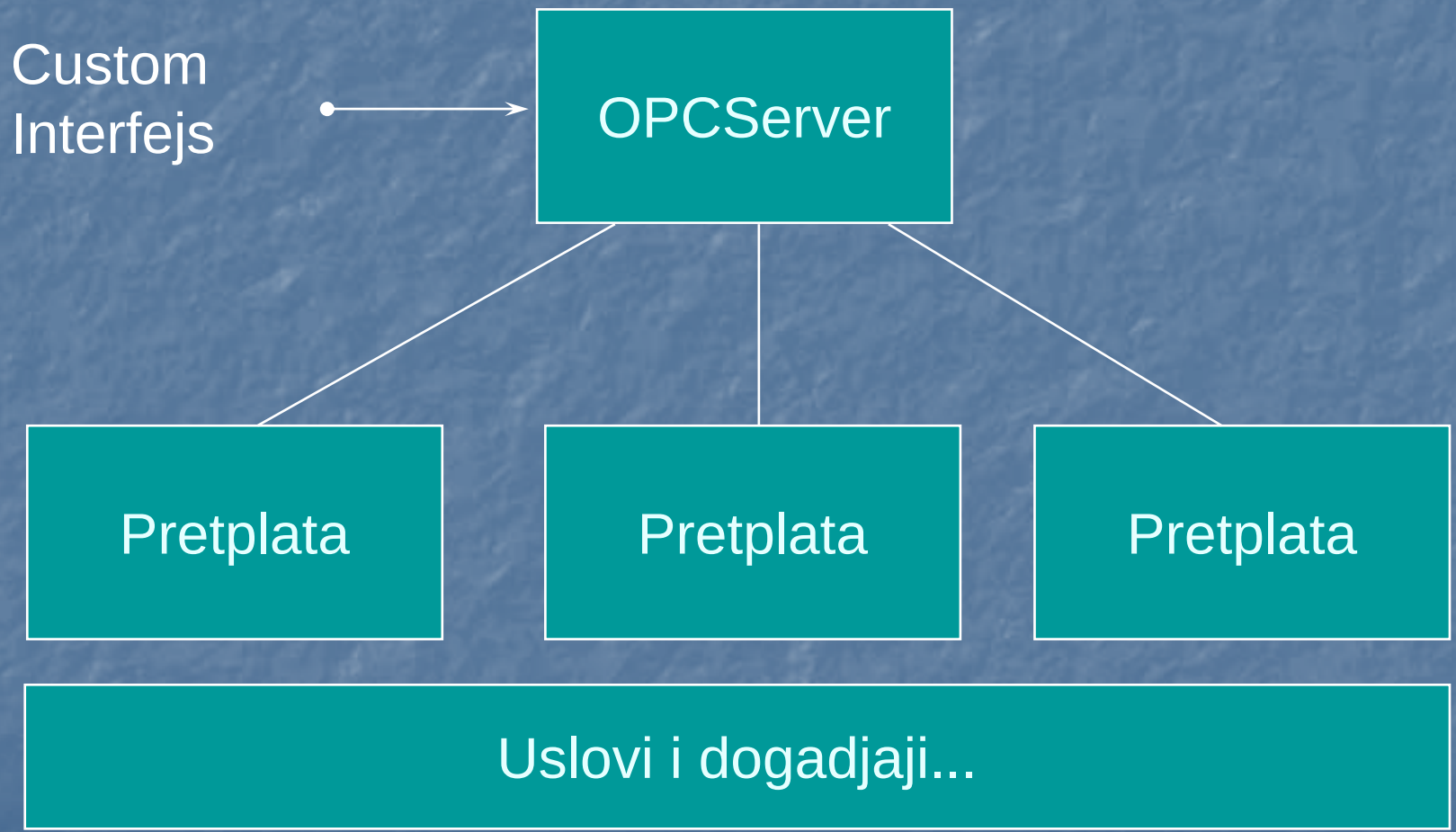
Server može da udruži ove uslove sa kategorijama kao što su napr. : ‘’ alarm nivoa’’ ili ‘’alarm temperature’’.

Klijent se pretplačuje ne Server na bazi filtera kojeg specificira, a ne na eksplicitnoj listi uslova i stanja tih uslova.

Uključenje filtera u dizajn OPC A&E Servera ima za cilj da smanji promet kroz računarske mreže za 80-90%.

Ostala fltriranja koja nisu podržana ovim dizajnom sa serverske strane mogu biti realizovana na klijentskoj strani.

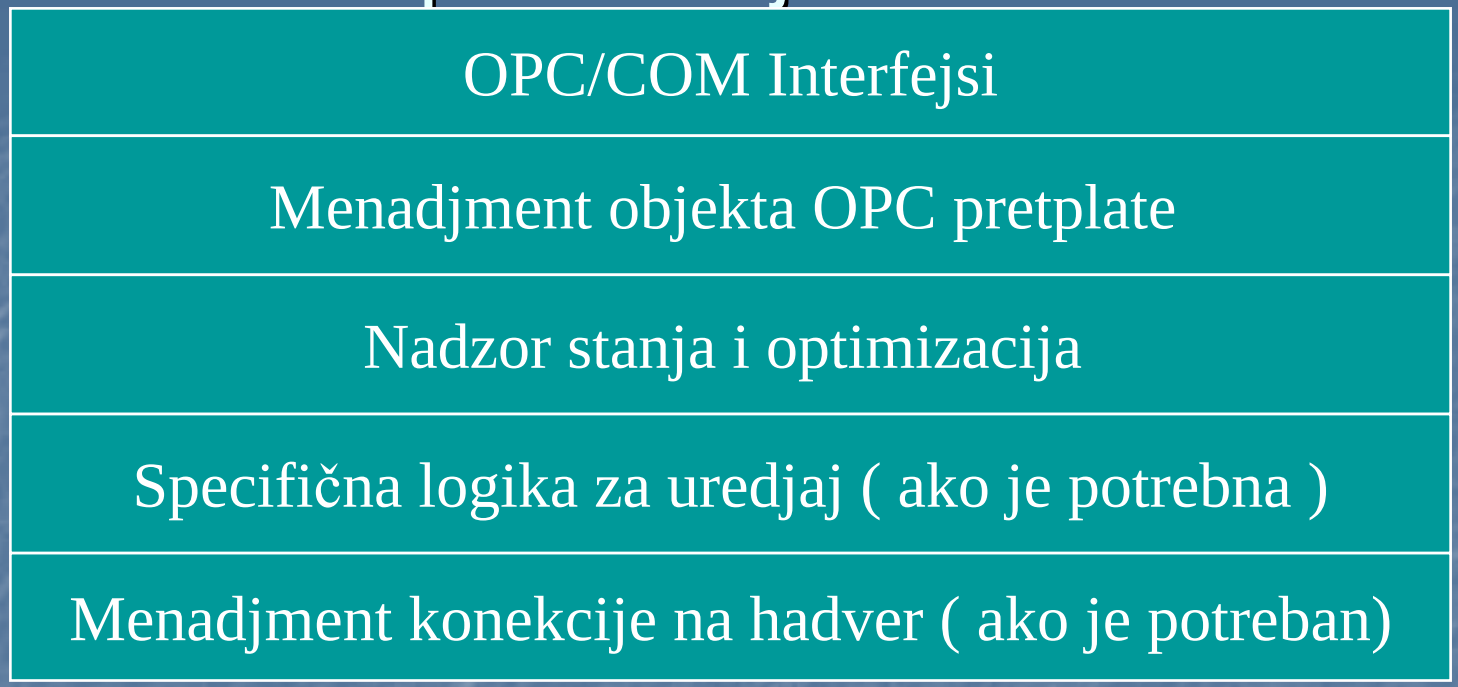
Logički Objektni Model



Logički Objektni Model

Vidimo jedan OPC Serversi objekat na koji se konektiraju aplikacije putem OPC klijenata. Primjetimo da će svaki OPC klijent vidjeti logički nezavisni COM objekat. Ispod njega je kolekcija objekta pretplata. Ovi objekti se dinamički kreiraju putem aplikacije klijenta da dobije pretplatu koja može biti filtrirana na različite načine. Najčešći sučaj je da većina aplikacija koristi jednu pretplatnu konekciju sa jednim filterom.

Tipični dizajn servera



Server obezbjedjuje menadjment OPC pretplata kao i kontrolira, posreduje i optimizira pristup logici i hardveru za višestruke klijente.
Server je i ovdje ustvari vrsta I/O drajvera, koji ‘’zna’’ kako da razgovara sa hardverom i od njega dobija podatke o specifičnim alarmima i dogadjajima.
Server o ovim dogadjajima obavještava klijente koji su izvršili pretplatu, putem OPC interfejsa.

Interfejsi servera

Server je COM objekat koji obezbjedjuje:

- **IOPCCommon**
- **IOPCEventServer**
- **IConnectionPointContainer**

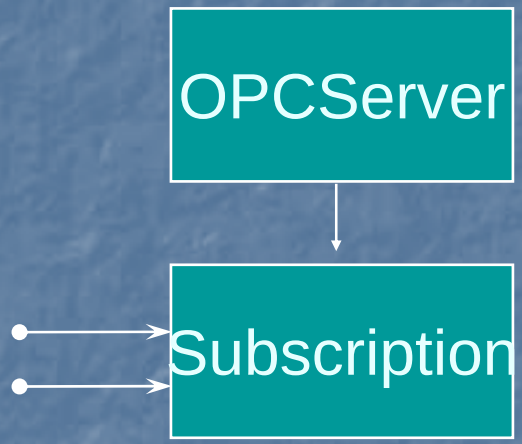
Metodi svakog od ovih pobrojanih interfejsa, obezbjedjuju slijedeće funkcionalnosti, aplikaciji koja se konektira preko OPC klijenta:

- da prevede kodove grešaka u razumljivi tekst
- da dobije statusnu informaciju o serveru
- da kreira i upravlja pretplatama
- da odredi raspoložive kriterije filtera za Server
- da pretražuje rasoložive alarmne oblasti i uslove
- da omogući, onemogući i potvrdi uslove.

Interfesi objekta pretplate

Pretplata je COM objekat koji obezbjedjuje: :

- **IOPCEventSubscriptionMgt**
- **IConnectionPointContainer**

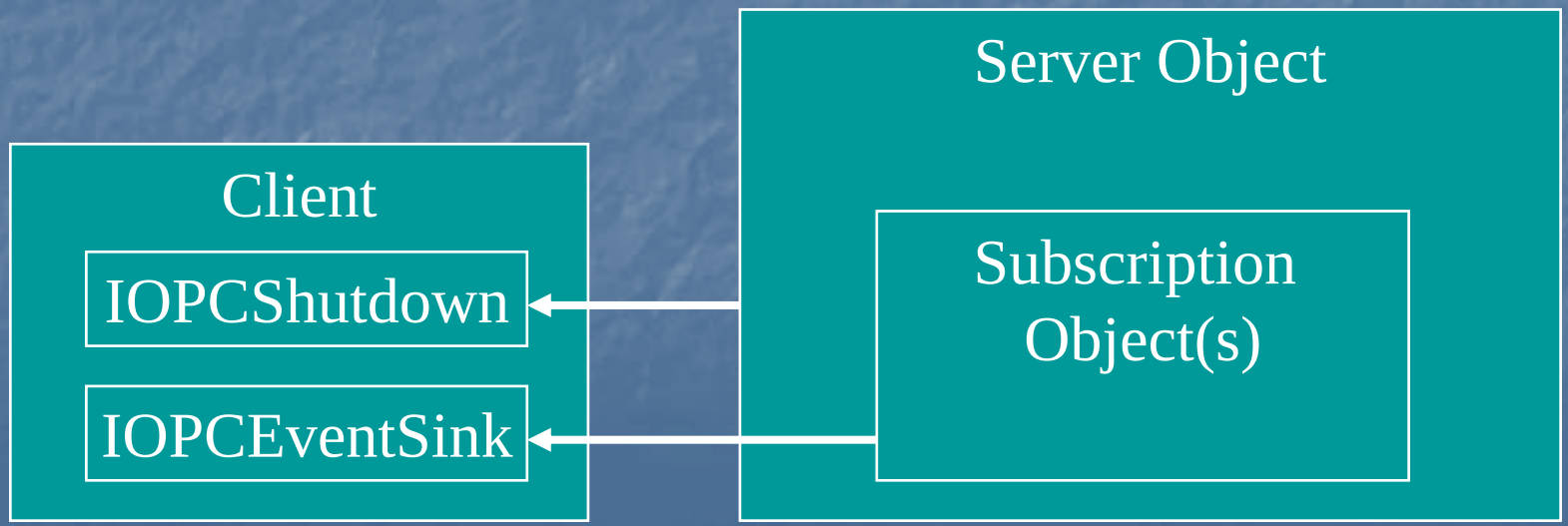


-Ovi interfejsi omogućavaju klijentu da:
-uspostavi callback konekciju
-da ‘’pauzira’’ i da ponovno uspostavi (resume) konekciju
- da setuje i modificira kriterije filtera.
-da indicira Serveru da treba da pošalje i dodatne parametre zajedno sa osnovnom informacijom, kada uslov promjeni stanje.
-da dobije update na stanje svih nadziranih uslova.

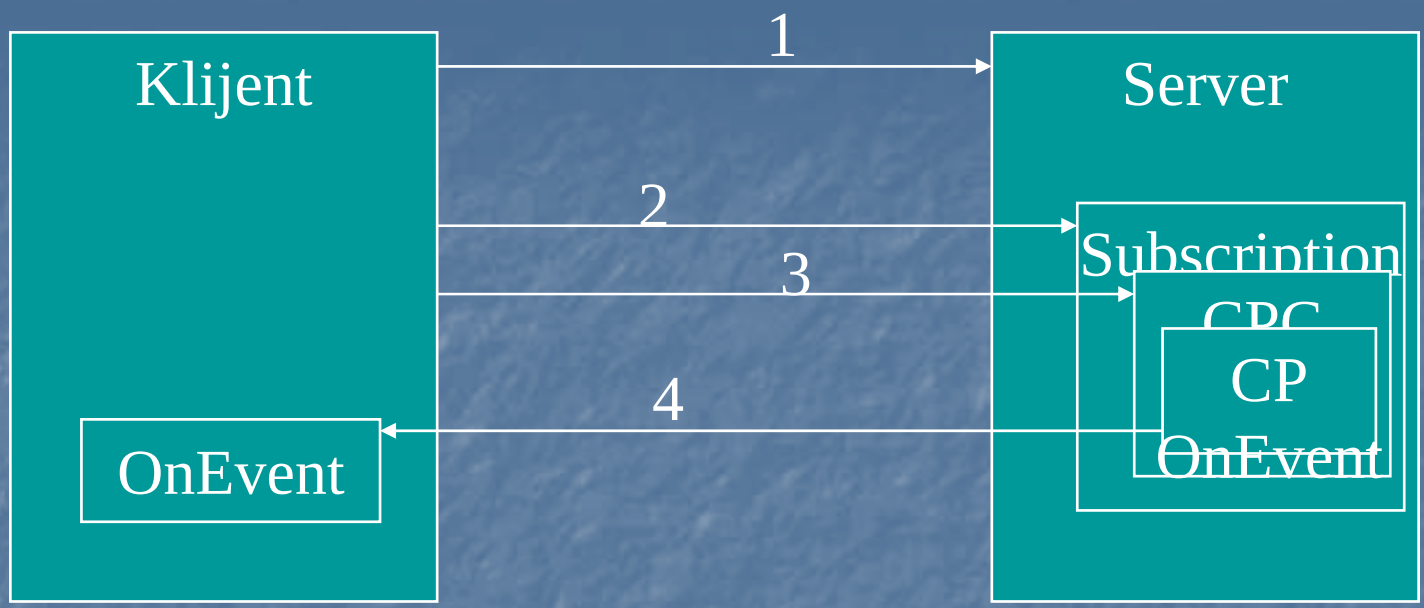
Interfejsi na strani klijenta

Sa strane klijenta, on obezbjedjuje dva COM interfejsa koja omogućuju Serveru da može da poziva (callback):

- **IOPCShutdown:** se poziva od strane Servera kada on traži od klijenta da oslobodi sve interfejse i da se diskonektira od Servera.
- **IOPCEventSink :** je gdje Server šalje obavjesti na koje se je klijent pretplatio



Konekcije izmedju klijenta i servera



Ovaj dijagram sumira kako klijent i Server rade zajedno:

1. Klijent kreira serverski objekat.
2. Klijent kreira pretplatu unutar servera i postavlja filtere
3.a. Klijent locira ConnectionPointContainer unutar objekta pretplate
3.b Klijent locira specifični ConnectionPoint objekat za IOPCOnEvent unutar CPC i konektira svoj callback na ovaj objekat.
4. Server šalje alarme i dogadjaje klijentu putem ovog callbacka.

Performanca i fleksibilnost

- OPC Server alarma i dogadjaja (A&E) je kompletan, snažan i fleksibilan mehanizam pošto je on kombinacija najboljih softverskih riješenja u razmjeni podataka izmedju aplikacija i objekata unutar njih.
- OPC Server za alarme i dogadjaje je brz mehanizam razmjene podataka jer je razvijan od početka sa distribuiranom mrežnom arhitekturom kao okruženjem za distribuirane aplikacije.

Sumarne karakteristike

OPC za alarme i dogadjaje pružaju:

- Podršku za ažuriranja podataka izmedju servera i klijenata na bazi izuzeća (exception) i preko filtriranih pretplata
- Projektovani su i kodirani da budu nezavisni od Vendora i platforme
- Dozvoljavaju konekcije i na jednostavne kao i na kompleksne uredjaje.
- Su optimizirani za korištenje u mrežama i distribuiranoj arhitekturi