Vložené řídicí systémy
Standardy OPC

Pavel Balda
ZČU v Plzni, FAV, KKY

Osnova přednášky

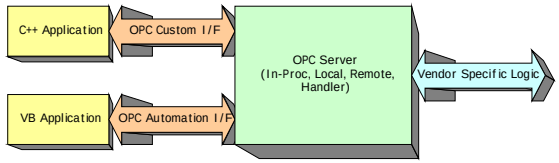
- OPC Common
- OPC Data Access

OPC – OLE for Process Control

- OPC Foundation (neziskové sdružení, **www.opcfoundation.org**) zavedlo několik standardizovaných protokolů na bázi OLE/COM
  - Snaha o zlepšení interoperability mezi aplikacemi v oblasti automatizace a řízení, řídicími systémy a kancelářskými aplikacemi v oblasti řízení procesů
  - Protokoly jsou založeny na standardech OLE/COM firmy Microsoft.
  - Definují standardní objekty, metody a vlastnosti pro servery poskytující informace v reálném čase, např. distrituované řídicí systémy, programovatelné automaty (PLC), inteligentní snímače, apod.
  - Informace se v reálném čase komunikují do zařízení podporujících OLE/COM (servery, aplikace, apod.)
- Komunikace prostřednictvím OPC má architekturu Klient-Server.
  - Server poskytuje data, komunikuje s fyzickými zařízeními
  - Klient komunikuje se serverem, který plní jeho požadavky

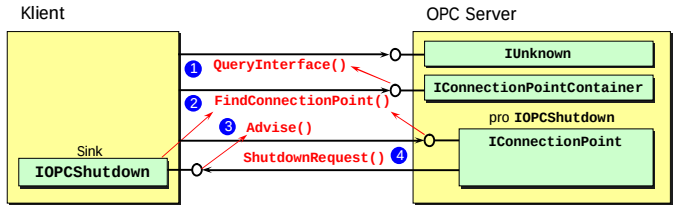
Základy OPC – Interfacy

- Popsány v dokumentu OPC Common Definitions and Interfaces
- Specifikace OPC obsahují vždy dvě sady interfaců:
  - Custom interfacy – základní rozhraní implementovaná OPC servery
  - Automation interfacy – nepovinná rozhraní odvozená od Idispatch
  - Custom interfacy poskytují maximální výkon, Automation interfacy jsou vhodné pro skriptovací jazyky, např. Visual Basic, Excel, apod.
- Povinná a volitelná rozhraní
  - OPC server musí mít implementovánu veškerou funkčnost z povinných interfaců. Klient komunikuje se serverem pomocí povinných interfaců
  - OPC server může mít implementovány nepovinné (volitelné) interfacy. Klient však musí být navržen tak, aby pracoval i když server volitelný interface nepodporuje. Pokud je daný volitelný interface implementován, musí být implementovány všechny jeho funkce



Základy OPC – Ukončení OPC serverů

- Pro všechny typy serverů (Data Access, Alarms & Events, ...) definuje OPC jednotný způsob oznámení klientům, že server bude ukončen (shutdown)
- Implementováno pomocí tzv. „Connection Points“
  - Způsob, jak volat z komponenty funkce z klienta (podobné delegátům z C#)
  - 1. Klient volá **`QueryInterface()`** pro **`IConnectionPointContainer`**
  - 2. Z **`IConnectionPointContainer`** se volá **`FindConnectionPoint()`** pro **`IID_IOPCShutDown`**, která vrací **`IConnectionPoint`**
  - 3. Z **`IConnectionPoint`** se volá **`Advise()`**s **`IOPCShutdown`**
  - 4. Chce-li server ukončit činnost volá pro všechny klienty funkci **`ShutdownRequest()`**



Základy OPC - IOPCCommon

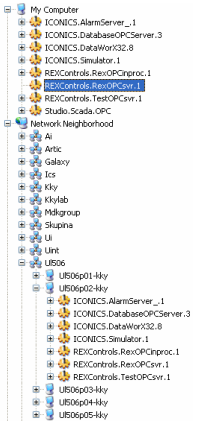
- Interface **`IOPCCommon`** využívají OPC servery všech typů
  - Obsahuje funkce pro nastavení a dotázání se na dané Locale (nastavení jazyka a země, určující formátování čísel, data a času, apod.)
  - Funkce pro ladicí účely
- Funkce:
  - **`SetLocaleID()`** – nastavuje identifikátor locale (language a sublanguage)
  - **`GetLocaleID()`** – vrací naposled nastavený identifikátor locale
  - **`QueryAvailableLocaleIDs()`** – vrací seznam dostupných locale pro komunikaci klient-server
  - **`GetErrorString()`** – pro daný kód chyby vrací její řetězcovou reprezentaci
  - **`SetClientName()`** – umožňuje klientovi aby nepovinně registroval své jméno na serveru (zejména pro ladicí účely)

Základy OPC – Instalace a registrace

- Kategorie komponent (Component Categories)
  - Umožňují rozdělit OPC servery do skupin podle implementované funkčnosti. Každá kategorie je identifikována GUIDem, nazývaným CATID
  - Pro každý server je registrován klíč **`Implemented Categories`**, ve kterém jsou ukládány všechny kategorie, do kterých server patří (server může implementovat několik kategorií, např. OPC Data Access 2.0, OPC Data Access 3.0, OPC Alarm&Event, ...)
- Vytvoření klíčů v Registry
  - Řídí se mechanismem automatické registrace (self-registration) definované v COMu
  - DLL servery exportují funkce **`DllRegisterServer`** (pro registraci) a **`DllUnregisterServer`** (pro zrušení registrace)
  - EXE servery se registrují parametrem příkazového řádku **`/RegServer`**, registrace se ruší zavoláním serveru s parametrem **`/UnregServer`**. Místo znaku **`/`** lze použít i znak **`-`**
  - Servery by neměly registrovat Proxy/Stub DLL knihovny. Ty se registrují pomocí exportovaných funkcí **`DllRegisterServer`** a **`DllUnregisterServer`**

Základy OPC – Vyhledávání serverů

- OPC Foundation dodává vyhledávač (server browser) **`OPCENUM.EXE`** a Proxy/Stub DLL **`OPCCOMN_PS.DLL`**
  - Může být instalován na jakémkoliv počítači
  - Přistupuje k lokální Registry do klíčů Component Categories
  - Implementuje interface **`IOPCServerList`**, pomocí kterého lze pracovat s daným cílovým počítačem
- Registrace
  - **`OPCENUM /RegServer`** – registruje browser jako EXE server
  - **`OPCENUM /Service`** – registruje browser jako službu Windows NT (jen Win NT4, 2000, XP, ?)
  - **`REGSVR32 OPCComn_ps.dll`** – registruje Proxy/Stub DLL
- Metody interfacu **`IOPCServerList`**
  - **`EnumClassesOfCategories()`**– vrací interface **`IEnumGUID`** pomocí kterého lze procházet všechny servery implementující dané kategorie
  - **`GetClassDetails()`** – pro dané **`CLSID`** serveru vrací **`ProgID`** a uživatelsky čitelné jméno serveru
  - **`CLSIDFromProgID()`** – pro dané **`ProgID`** vyhledá **`CLSID`** serveru (vhodné pro Automation)

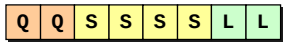


Typ VARIANT (zjednodušeno)

- Datová struktura pro ukládání hodnot libovolného typu

```
struct __tagVARIANT {
    VARTYPE vt;        // WORD, definuje VT_*
    WORD wReserved1;
    WORD wReserved2;
    WORD wReserved3;
    union {
       LONGLONG          llval;        // VT_I8.
       LONG              lVal;         // VT_I4.
       BYTE              bVal;         // VT_UI1.
       SHORT             iVal;         // VT_I2.
       FLOAT             fltVal;       // VT_R4.
       DOUBLE            dblVal;       // VT_R8.
       VARIANT_BOOL      boolVal;      // VT_BOOL.
       SCODE             scode;        // VT_ERROR.
       CY                cyVal;        // VT_CY.
       DATE              date;         // VT_DATE.
       BSTR              bstrVal;      // VT_BSTR.
       IUnknown          * punkVal;    // VT_UNKNOWN.
       IDispatch         * pdispVal;   // VT_DISPATCH.
       SAFEARRAY         * parray;     // VT_ARRAY|*.
       /* ... Mnoho dalších položek ... */
    };
};
```


Příznaky kvality signálu



- Ke každé hodnotě položky z OPC je přidruženo 8 bitů, tzv. příznaků kvality (podobné ale jednodušší než kvalita ve specifikaci Fieldbus)
- **QQ** – bity určující základní kvalitu signálu
  - **00** – Špatná (BAD) kvalita, hodnota je nepoužitelná z důvodu uvedených v bitech **SSSS**
  - **01** – Nejistá (UNCERTAIN) kvalita, důvod je opět uveden v bitech **SSSS**
  - **10** – Tato kombinace se v OPC nevyužívá
  - **11** – Dobrá (GOOD) kvalita, hodnotu lze používat bez problémů
- **SSSS** – substatus blíže specifikuje kvalitu (např. příčinu selhání) a má různý význam pro různé hodnoty bitů **QQ**
- **LL** – příznaky překročení mezí
  - **00** – neomezeno
  - **01** – hodnota podkročila spodní mez (saturace)
  - **10** – hodnota překročil horní mez (saturace)
  - **11** – hodnota je konstantní

Substatus příznaků kvality signálu

QQ = 00 … BAD

SSSS	Bitově	Popis
0	0000	Nespecifikovaná příčina
1	0001	Chyba konfigurace
2	0010	Nepřipojeno
3	0011	Selhání zařízení
4	0100	Selhání čidla
5*	0101	Poslední známá hodnota
6	0110	Selhání komunikace
7	0111	Mimo provoz
8	1000	Čekání na počáteční data
9-15		Rezervováno

QQ = 01 … UNCERTAIN

SSSS	Bitově	Popis
0	0000	Nespecifikovaná příčina
1*	0001	Poslední použitelná hodnota
4	0100	Nepřesné čidlo
5	0101	Překročení rozsahu jednotek
6**	0110	Sub-normální hodnota
Ostatní		Rezervováno

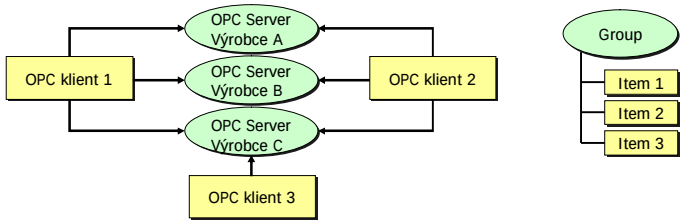
QQ = 11 … GOOD

SSSS	Bitově	Popis
0	0000	Nespecifikováno
6	0110	Místní hodnota (přepsání)
Ostatní		Rezervováno

*) Význam těchto chybových kódů se liší: pro BAD je chyba způsobena chybou komunikace po delší dobu, pro UNCERTAIN jde o chybu právě proběhlé komunikace
**) Hodnota vzniká z několika zdrojů; je dobrý jen menší počet těchto zdrojů než je požadováno

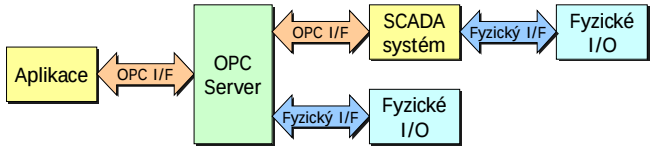
OPC Data Access 3.0

- OPC DA (Data Access) je nejznámější a nejčastěji využívaná OPC specifikace
- Specifikace popisuje OPC COM objekty a jejich interfacy implementované OPC servery
- OPC klient se může připojit k OPC serverům od jednoho i více výrobců (vendor)
- OPC server se skládá z několika typů objektů: server, group, item
  - OPC server udržuje svůj stav a obsahuje OPC grupy (container)
  - OPC grupa spravuje svůj stav a obsahuje položky OPC položky (items)
  - OPC položka obsahuje hodnotu (typ **VARIANT**), příznaky kvality a časovou značku



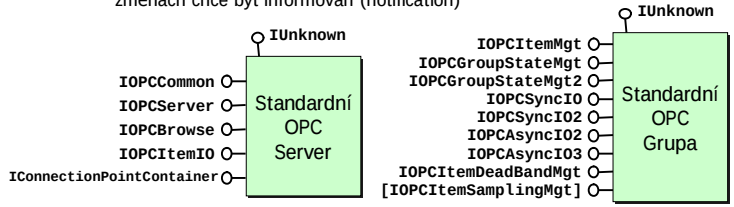
Kde je OPC vhodné?

- OPC je primárně navrženo pro přístup k datům ze síťového serveru
- OPC interfacy však mohou být použity na mnoha místech v aplikaci
  - Přenos dat z fyzických zařízení do SCADA (Supervisory Control And Data Acquisition) nebo DCS (Distributed Control Systems)
  - Přenos dat ze systémů SCADA nebo DCS do aplikace
- Architektura a návrh OPC dovoluje klientské aplikaci přistupovat k datům z mnoha OPC serverů, mnoha dodavatelů, běžících na různých počítačích v síti prostřednictvím jediného objektu serveru



Přehled objektů a interfaců

- OPC server poskytuje přístup (čtení, zápis, komunikaci) k množině datových zdrojů (sources)
- OPC klient se připojuje k OPC serveru a komunikuje s ním prostřednictvím interfaců
  - Server umožňuje klientům vytvářet grupy a manipulovat s nimi
  - Grupy jsou prostředkem klienta pro organizaci dat, ke kterým chce přistupovat. Grupy mohou být aktivovány/deaktivovány jako celek
  - Grupa též umožňuje, aby si klient „předplatil“ seznam položek, o jejichž změnách chce být informován (notification)



Interface IOPCServer

- **IOPCServer** je hlavním (povinným) interfacem OPC serveru
- Obsahuje funkce:
  - **AddGroup()** – server vytvoří grupu a vrátí interface (ukazatel) požadovaný klientem
  - **GetErrorString()** – stejná funkce jako v interfacu **IOPCCommon**
  - **GetGroupByName()** – vrátí požadovaný interface pro grupu s daným jménem dříve vytvořenou daným klientem
  - **GetStatus()** – vrací aktuální stav serveru. Může být klientem volána periodicky
  - **RemoveGroup()** – odstraňuje grupu z paměti
  - **CreateGroupEnumerator()** – vytváří různé objekty pro prohledávání (enumerátory) grup vytvořených serverem

IOPCServer – AddGroup()

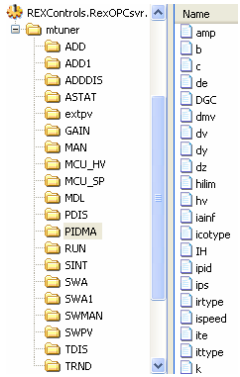
- Funkce **AddGroup()** přidává do serveru požadovanou grupu.
- Funkce má následující přesnou syntaxi v jazyku IDL

```
HRESULT AddGroup(
    [in, string] LPCWSTR szName,          // jméno grupy
    [in] BOOL bActive,                    // TRUE ... grupa bude hned aktivní
    [in] DWORD dwRequestedUpdateRate,     // nejkratší perioda aktualizace v [ms]
    [in] OPCHANDLE hClientGroup,          // handle klienta ke grupě
    [unique, in] LONG *pTimeBias,         // posun času v časovém pásmu v [min]
    [in] FLOAT * pPercentDeadband,        // procentuální změna v hodnotě
      // položky, která způsobí zaslání aktualizace klientovi (jen pro analogové hodnoty)
    [in] DWORD dwLCID,                    // jazyk pro vracení textových informací
    [out] OPCHANDLE * phServerGroup,      // ukazatel na jedinečný handle
      // grupy v serveru
    [out] DWORD *pRevisedUpdateRate,      // revidovaná hodnota
      // dwRequestedUpdateRate vrácená serverem
    [in] REFIID riid,                     // Požadovaný typ interfacu
    [out, iid_is(riid)] LPUNKNOWN * ppUnk // Vrácený ukazatel na interface
);
```

  - Ve specifikaci je dále popsáno chování této funkce a jsou uvedeny přesné návratové kódy **HRESULT**, které má funkce v konkrétních případech vracet
- Obdobně jsou popsány všechny funkce všech interfaců. Pro detaily viz specifikaci.

Interface IOPCBrowse

- Interface **IOPCBrowse** poskytuje vylepšené metody (oproti zrušenému interfacu **IOPCBrowseServerAddressSpace**) pro procházení adresního prostoru serveru a získávání vlastností položek (item properties)
- Obsahuje funkce:
  - **Browse()** – prochází jednu větev adresového prostoru a vrací nulu nebo víc struktur **OPCBROWSEELEMENT**. Adresní prostor serveru je hierarchický. Jeho stromová struktura je analogická hierarchii souborového systému – jednotlivé větve odpovídají adresářům, listy, tj. položky odpovídají souborům
  - **GetProperties()** – pro každou položku z pole položek vrací pole struktur **OPCITEMPROPERTIES**. Vlastnosti jednotlivých položek lze považovat za „podpoložky“. Navíc mohou být vlastnosti rovněž reprezentovány jako samostatné položky



Interface IOPCItemIO

- Interface **IOPCItemIO** pro velmi snadný přístup k datům z jednoduchých aplikací
  - Ve většině případů bude přístup přes interfacy OPC grup mnohem výkonnější!
  - Z hlediska výkonnosti se chová tento interface jako by byla vytvořena grupa, přidány do ní položky, vykonáno jedno čtení nebo zápis a grupa zrušena
- Druhým účelem je poskytnout metodu pro zápis časových značek a příznaků kvality do serverů, které tuto funkčnost podporují
- Obsahuje funkce:
  - **Read()** – čte jednu nebo několik hodnot, příznaků kvality a časových značek pro zadané položky. Funkce je podobná stejnojmenné funkci z **IOPCSyncIO**
  - **WriteVQT()**– zapisuje jednu nebo několik hodnot, příznaků kvality a časových značek pro zadané položky. Funkce je podobná stejnojmenné funkci z interfacu **IOPCSyncIO2**. Zapisované hodnoty jsou uloženy v poli struktur **OPCITEMVQT**, v každé z těchto struktur jsou příznaky, co má být zapsáno. Je-li typ hodnoty roven **VT_EMPTY**, hodnota se nezapisuje. Je-li hodnota příznaků pro kvalitu a/nebo časovou značku rovna **FALSE**, kvalita a/nebo časová značka se nenastavuje.

OPC Grupy

- Objekt **OPCGroup** obsluhuje pro server skupinu jednotlivých položek
- OPCGroup má některé obecné vlastnosti, ovlivňující fungování interfaců a metod
  - Name – jméno grupy. Musí být jednoznačné pro grupy daného klienta; rozlišuje malá a velká písmena; klient jej může měnit
  - Cached Data – metody grupy (dále) umožňují klientovi specifikovat zda mají být vracená data z vyrovnávací paměti (CACHE) nebo přímo ze zařízení (DEVICE)
  - Active – příznak „aktivity“ grup a jejich položek. Umožňuje optimalizovat zatížení komunikace a CPU. Položky a grupy, které ho nemají nastaven nemusí být aktualizovány v paměti CACHE
  - Update Rate – perioda aktualizace grupy zadaná klientem. Server by měl mít „dobrou vůli“ splnit požadavek klienta. Určuje, že volání funkcí klienta (callback) by nemělo být častější než s touto periodou a že CACHE by měla být aktualizována alespoň s touto periodou
  - Time Zone – určuje časový posun mezi daným zařízením a klientem, používá se (zřídka) pro výpočet lokálního času zařízení
  - Percent Deadband – používá se pro generování výjimky překročení meze (exception limit)
    - Generována pokud absolutní hodnota rozdílu mezi hodnotou v CACHE a aktuální hodnotou je větší než (EUHigh – EULow) * Percent_Deadband / 100; EUHigh a EULow jsou vlastnosti položky určující rozsah položky v inženýrských jednotkách (např. m, kg, °C,...)
  - Client Handle – posílán v každém volání callback, umožňuje identifikovat grupu, které daná data patří

Čtení a zápis dat

- Data lze do klienta dostat (číst) šesti způsoby (funkcemi)
  - **Read()** z **IOPCSyncIO** – z CACHE nebo DEVICE
  - **Read()** z **IOPCAsyncIO2** – z DEVICE
  - **OnDataChange()** z **IOPCDataCallback** – založena na výjimkách, může být iniciována funkcí **Refresh()** z **IOPCAsyncIO2**
  - **Read()** z **IOPCItemIO**
  - **ReadMaxAge()** z **IOPCSyncIO2**
  - **ReadMaxAge()** z **IOPCAsyncIO3**
  - Poslední tři způsoby jsou z CACHE nebo DEVICE, určeno „lhůtou prošlosti“ dat v CACHE
- Existuje 5 způsobů jak data zapsat do serveru
  - **Write()** z **IOPCSyncIO**
  - **Write()** z **IOPCAsyncIO2**
  - **WriteVQT()** z **IOPCItemIO**
  - **WriteVQT()** z **IOPCSyncIO2**
  - **WriteVQT()** z **IOPCAsyncIO3**

Interface IOPCItemMgt

- **IOPCItemMgt** umožňuje klientům přidávat a rušit položky grupy a řídit jejich chování
- Obsahuje funkce:
  - **AddItems()** – přidává jednu nebo více položek do grupy
  - **ValidateItems()** – zjišťuje, zda dané položky jsou platné, tj. zda mohou být přidány do grupy bez chyby; vrací informaci o typu položky
  - **RemoveItems()** – ruší položky z grupy, opak funkce AddItems()
  - **SetActiveState()** – aktivuje nebo deaktivuje jednu nebo více položek. Tím je určeno, zda data těchto položek v CACHE budou platná a zda budou položky obsaženy ve zpětném volání funkce **OnDataChange()**
  - **SetClientHandles()** – mění klientský handle (celočíselný identifikátor) jedné nebo několika položek v grupě
  - **SetDatatypes()** – mění požadovaný typ dat pro jednu nebo více položek grupy
  - **CreateEnumerator()** – vytváří interface pro procházení položek grupy

21

Interface IOPCGroupStateMgt

- **IOPCGroupStateMgt** umožňuje klientovi pracovat se stavem grupy jako celku, zejména měnit periodu aktualizace a aktivní stav
  - **GetState()** – vrací aktuální stav grupy
  - **SetState()** – nastavuje vlastnosti stavu grupy (update rate, příznak active, časový posun, percent deadband, locale, klientský handle)
  - **SetName()** – nastavuje jedinečné jméno grupy
  - **CloneGroup()** – vytváří kopii grupy a nastavuje jí jedinečné jméno. Duplikuje všechny vlastnosti grupy, položky a vlastnosti položek. Nová grupa je nezávislá na grupě, ze které byla vytvořena.
- **IOPCGroupStateMgt2** rozšiřuje předchozí interface o funkce řídicí zpětná volání v situaci, kdy se „nic neděje“ :
  - **SetKeepAlive()** – nastavuje čas, po kterém je zavolána aktualizace klienta v případě, že se žádná položka nemění. Server tím dává najevo, že „ještě žije“
  - **GetKeepAlive()** – vrací aktuálně nastavený čas „KeepAlive“

22

Interfacy IOPCSyncIO, IOPCSyncIO2

- **IOPCSyncIO** slouží pro synchronní čtení a zápis ze/do serveru. Čeká se na dokončení operace v daných funkcích
  - **Read()** – čte hodnotu, příznaky kvality a časovou značku jedné nebo několika položek v grupě. Data lze číst z CACHE (měla by být čtena v rámci **Update Rate** a **Percent Deadband**) i z DEVICE. Data čtená z CACHE jsou platná pouze pokud je aktivní jak grupa, tak i konkrétní položka
  - **Write()** – zapisuje hodnoty jedné nebo několika položek do DEVICE. Zápisy nejsou ovlivněny nastavením příznaku ACTIVE grupy ani položky
- **IOPCSyncIO2** doplňuje **IOPCSyncIO** o vylepšené funkce čtení a zápisu
  - **ReadMaxAge()** – podobná funkci **Read**. O čtení z CACHE nebo DEVICE rozhoduje sám server podle parametru **MaxAge**. Pokud jsou data v CACHE „mladší“ než **MaxAge**, čtou se z CACHE, jinak z DEVICE
  - **WriteVQT()** – podobná funkci **Write()**, navíc však umožňuje nastavit kvalitu a časovou značku

23

Interfacy IOPCAsyncIO2, IOPCAsyncIO3

- **IOPCAsyncIO2** umožňuje klientovi realizovat asynchronní čtení a zápis ze/do serveru. Operace je zařazana do fronty a daná funkce se hned vrací (klient nečeká na provedení čtení/zápisu).
  - Klient generuje **TransactionID**, které posílá do funkcí **Read()**, **Write()** a **Refresh()**. Tato hodnota je předávána serverem ve zpětném volání
  - **Read()** – čte jednu nebo několik položek z grupy. Výsledky se vracejí pomocí rozhraní **IOPCDataCallback** klienta, které je předáno serveru přes **IConnectionPointContainer**
  - **Write()** – zapisuje jednu nebo několik položek grupy. Využívá se rozhraní **IOPCDataCallback**
  - **Refresh2()** – vynutí zavolání funkce **OnDataChange()** z **IOPCDataCallback** pro všechny aktivní položky grupy (nezávisle na tom, zda se změnily či ne)
  - **Cancel2()** – požadavek na ukončení probíhající transakce
  - **SetEnable()** – hodnota **FALSE** blokuje volání **OnDataChange()** při změně
  - **GetEnable()** – vrací naposled nastavenou hodnotu funkcí **SetEnable()**
- **IOPCAsyncIO3** rozšiřuje **IOPCAsyncIO2** o funkce
  - **ReadMaxAge()** – „asynchronní“ verze stejnojmenné funkce z **IOPCSyncIO2**
  - **WriteVQT()** – podobná funkci **Write()**, navíc umožňuje nastavit kvalitu a časovou značku
  - **RefreshMaxAge()** – podobná funkci **Refresh2()**, parametr **MaxAge** určuje, odkud se budou hodnoty brát

24

Interface IOPCDeadbandMgt

- **IOPCDeadbandMgt** umožňuje pracovat s různým Percent Deadband pro každou položku v grupě
  - **SetItemDeadband()** – potlačí deadband grupy a nastaví zadanou hodnotu pro všechny specifikované položky
  - **GetItemDeadband()** – vrátí hodnoty „mrtvých zón“ pro všechny specifikované položky
  - **ClearItemDeadband()** – smaže individuální deadbandy položek a obnoví pro ně deadband grupy

Interface IOPCDataCallback

- **IOPCDataCallback** je interfacem klienta
  - Klient jej pomocí **IConnectionPointContainer** a **IConnectionPoint** zaregistruje pro zpětné volání (callback) ze serveru
  - Používá se pouze společně s „asynchronními“ interfacy
- Obsahuje funkce:
  - **OnDataChange()** – tuto metodu volá server z OPC grupy při vygenerování výjimky (překročení Percent Deadband z rozsahu a po volání funkce **Refresh2()**
  - **OnReadComplete()** – předává čtená data položek klientovi. Je volána serverem po ukončení funkcí pro asynchronní čtení
  - **OnWriteComplete()** – předává klientovi návratové kódy asynchronních zápisů z **IOPCAsyncIO2**
  - **OnCancelComplete()** – oznamuje klientovi ukončení metody **Cancel2()** z **IOPCAsyncIO2**