Functions for interfacing other applications with WinMENTOR

DocImpServer is a module intended for programmers who want to interface various applications with the **WinMENTOR** program.

As a software component, **DocImpServer** is a **DCOM** server containing functions both for querying data from the **WinMENTOR** database and for updating certain catalogs or issuing documents.

Because the structure of the **WinMENTOR** database is quite complex, DocImpServer allows programmers to develop applications that execute data queries and updates without needing to know the **WinMENTOR** data structures and table relationships. Data transfer between the client application and **DocImpServer** is done through *Olevariant* data blocks, which essentially are arrays of strings containing information concatenated with ";".

Depending on the specific development desired, programmers receive support in the form of a project containing examples of connecting to **DocImpServer** and launching basic functions. They are also provided with information on the functions they can use to achieve the desired results, as well as the structure they need to send or receive the data.

With over 100 implemented functions, DocImpServer allows:

- Reading the Articles catalog
- Reading the Partners catalog
- Reading the Warehouses catalog
- Reading the Personnel catalog
- Checking stocks
- Checking partner balances
- Checking price category catalogs and prices per item/client
- Updating the Partners catalog
- Updating the Articles catalog
- Generating orders from clients
- Generating invoices and delivery notes to clients
- Generating invoices from internal or external suppliers (DVI documents)
- Generating warehouse delivery notes
- Generating consumption vouchers
- Generating transfers between warehouses
- Generating cash transactions with sales in the store.

I. DESCRIPTION OF FUNCTIONS

To understand the explanations below, you should already be familiar with the operation of the **WinMENTOR** application.

It is preferable that the Server be copied to the directory where Mentor.exe is located so that the BDE session can use the existing Private subdirectory there. The COM server implements the following functions:

1. GetListaFirme: OleVariant;

In **WinMENTOR**, the user can manage data for an indefinite number of companies; upon entering the session, they will select the working company.

This function returns a list with the abbreviated names of the companies found in the **WinMENTOR** database directory.

2. GetListaLuni(const NumeFirma: WideString): OleVariant;

Data in **WinMENTOR** is managed at the level of working months, so the user must specify the working month when entering the session.

This function returns a list with the existing months on disk for the company with the abbreviated name specified in the NumeFirma parameter. The names of the working months are in the format "yyyy_mm" (as they are on disk, the names of the subdirectories containing the information of the working months).

3. SetNumeFirma(const NumeFirma: WideString): Integer;

Sets the name of the working company (the **NumeFirma** parameter); Returns **1** if the function succeeded or **0** if not. To read any error message, the *GetListaErori* function described below must be called.

Observation: In **WinMENTOR**, the database is structured in directories, each company having allocated a directory whose name is given by the abbreviated name of the company.

For each working accounting month, a subdirectory is created with the name AAAA_MM;

Example: the data of the EXEMPLU company is located in the directory

C:\winment\data\example.

The data for each accounting month is in subdirectories like

C:\winment\data\example\2022_04

4. SetLunaLucru(An, Luna: Integer): Integer;

Sets the working month (parameters **An**, **luna**);

Returns **1** if the function succeeded or **0** if not. The error message is retrieved with the GetListaErori function.

5. SetDocsData(DocData: OleVariant);

Sends in a list of strings (the **DocData** parameter) the content of a data packet containing documents to be imported into **WinMENTOR**.

Here's an example written in Delphi about using this function:

```
VA := VarArrayCreate([0, Lista.Items.Count -1 ], varOleStr);
for i := 0 to Lista.Items.Count - 1 do VA[i] :=  Lista.Items[i];
WinMDocImp.SetDocsData(VA);
```

Lista is a string concatenation in which each line of a text file to be transmitted is loaded one by one.

WinMDocImp is an instance of **DocImpServer**.

6. DateValide: Integer;

The function validates the content of a package of outgoing invoices sent through the *SetDocsData* function.

Returns, if the data is valid, or **0** if not. Error messages are retrieved with the *GetListaErori* function;

7. GetListaErori: OleVariant;

Transmits a list of strings containing error messages returned by the server after executing certain functions.

8. ImportaFacturi: Integer;

Import the invoices from the data transmitted with *SetDocsData*. Return the total number of imported invoices.

Calling the *GetErrorList* function is useful because, even if invoices have been imported, it's possible that some of them could not be modified or deleted because they were being edited by other users. The list returned by *GetErrorList* may contain such information.

Here's an example of using the functions described above (Consider *DocImp* an instance of *IDocImpObject* that you will create when launching the client program):

```
if (DocImp.SetNumeFirma('TEST') == 0)  then
 {
   ListaErori= DocImp.GetListaErori;
   Print error list;
   return;
 }
```

```
if (DocImp.SetLunaLucru(2003, 4) == 0)
then
  {
    ListaErori= DocImp.GetListaErori;
    Print error list;
    return;
  }

if (DocImp.Datevalide == 0) then
  {
    ListaErori= DocImp.GetListaErori;
    Print error list;
  }
else
  { nr=DocImp.ImportaFacturi;     // nr will contain the number of invoices imported
    ListaErori= DocImp.GetListaErori;     // read any errors or warning messages
  }
```

9. GetListaParteneri(out Error: Integer): OleVariant;

Returns a list of strings with information about Partners. Each line contains information about a partner in the following format:

1. *Partner ID;*
2. *Name;*
3. *Fiscal Code;*
4. *Locality;*
5. *Address;*
6. *Phone;*
7. *Contact person;*
8. *Classification Symbol; from the NCLASEP.DB table*
9. *Classification; // from the NCLASEP.DB table*
10. *Price category symbol;* // from the NCATPRET.DB table
11. *Price category name; // from the NCATPRET.DB table*
12. *Agent Brand;* // MARCA column from the NPERS.DB table
13. *Agent Name;* // NUME column from the NPERS.DB table
14. *Agent Surname; //* PRENUME *column from the NPERS.DB table*
15. *Due Date;*
16. *Discount;*
17. *Associated Discount Criterion Name;*

18. Partner office names; // separated by "~" if there are multiple;

19. External Code; // CODEXTERN from the NPART.DB table

20.Partner blocked; // Flag can be YES or NO

21. Credit on sale;

22. Fiscal code;

23. Bank account;

24. Office localities; // separated by "~" if there are multiple, corresponding to the order from field 18

25. Country;

26.Assigned agent brands at offices; // separated by "~" if there are multiple, corresponding to the order from field 18

27. Observations;

28. Flag (D) indicating whether the listed offices have a headquarters status // separated by "~",

29. Office postal codes; // separated by "~", corresponding to the order from field 18

30. Office email addresses // separated by "~", corresponding to the order from field 18

31. Contact person phone numbers;

32. Flag indicating whether the partner is a natural person or legal entity (PJ or PF)

33. Partner's default currency;

34. Date added to the database;

35. Assigned routes to the partner;

36. Puncte acumulate;

37. Office fiscal codes // separated by "~", corresponding to the order from field 18

38. Office type information // separated by "~", corresponding to the order from field 18.

If, for various reasons, the list could not be transmitted, the *Error* parameter takes the value *1*, and error messages are retrieved with the *GetListaErori* function

10. GetStocArticole(out Error: Integer): OleVariant;

Returns a list of strings with information about existing items in stock. Each line contains information about an item in the format:

„CodExtern;Denumire;UM;PretVanzare;Stoc;SimbolClasa;DenClasa;IDProducator;Den.Producator;IDFurnizor;DenFurnizor;SimbolGestiune;DenGestiune;CotaTVA;Flag;PretCuTVA;StocRezervat de comezi; ”

("ExternalCode;Name;UnitOfMeasure;SellingPrice;Stock;ClassSymbol;ClassName;ManufacturerID;ManufacturerName;SupplierID;SupplierName;WarehouseSymbol;WarehouseName;VATRate;Flag;PriceWithVAT;ReservedStockByOrders;")

Flag // specifies whether VAT is included in the selling price (Y/N);

If, for various reasons, the list could not be transmitted, the *Error* parameter takes the value 1, and error messages are retrieved with the *GetListaErori* function.

11. GetStocArticol(const ArticolID: WideString;out Error: Integer): OleVariant;

Returns a string with information about the stock of an item in the format

„CodExtern;Denumire;UM;PretVanzare;Stoc".

("ExternalCode;Name;UnitOfMeasure;SellingPrice;Stock".)

As above, the Error parameter takes the value 1 if an error has occurred.

12. GetSoldPart(const PartID: WideString; out Error: Integer): OleVariant;

Returns a string with information about the balance of a partner in the format:

„CodExtern;Denumire;Sold".

("ExternalCode;Name;Balance").

The *Error* parameter has the same significance as above.

13. GetListaPersonal

Returns:

„Nume;Prenume;Marca;CNP;flag EsteACtiv (Da/Nu);flag EsteAgent (Da/Nu);Serie Buletin;Numar buletin;CodPostal;"

("LastName;FirstName;EmployeeID;SSN;IsActiveFlag (Yes/No);IsAgentFlag (Yes/No);IDCardSeries;IDCardNumber;PostalCode;")

14. GetSoldDetaliat(PartID: WideString; out Error: Integer)

Details the balance on unpaid invoices and advances.

Lines regarding each invoice have the structure:

„Factura;NrFactura;DataFactura;Rest_de_Incasat";

("Invoice;InvoiceNumber;InvoiceDate;RemainingAmountToBeCollected";)

Advances (if any) are returned after unpaid invoices in the structure:

„Avans;DocumentAvans;DataDocument;SumaAvans;"

("Advance;AdvanceDocument;DocumentDate;AdvanceAmount;".)

In both types of lines, "Invoice" and "Advance" are texts defining the line type; for advances, AdvanceAmount will be negative.

15. GetListaGestiuni(out Error: Integer): OleVariant

Generates the list of warehouses in the format *Simbol;Denumire (Symbol; Name.)*

If *Error* <> 0, the operation failed (read error list).

16. GetClaseParteneri(out Error: Integer): OleVariant

Generates the list of warehouses in the format: *Simbol;Denumire*; (Symbol; Name).

If *Error* <> 0, the operation failed (read error list).

17. ExistaFactura(Numar: Integer): Integer;

Tests the existence of the invoice with the number "NUMBER".

The function can return the values:

-1: function execution error (reads error list);

0: invoice does not exist;

1: invoice exists and is processed;

2: invoice exists and is unprocessed.

18. GenCodArticole: Integer;

Generates external codes in the item nomenclature.

Returns -1 if there were errors (reads error list) or the number of updated items;

To read the items from **WinMENTOR**, you need to call the function:

19. GetProducts(const LastSyncDate: WideString; out Error: Integer): OleVariant;

As an input parameter, send the date and time of the last synchronization in the structure: 'dd.mm.yyyy hh:mm:ss';

The function returns data in the structure:

IDArticol; // IDArticle;

Denumire; // Name;

Den_UM; // UnitOfMeasureName;

IDProducator; // ManufacturerID*;*

Denumire Producator; // ManufacturerName;

TipSerie; // SeriesType;

DataAdaugarii; // DateAdded;

DataUltimeiModificari; // LastUpdateDate;

Tip unitate de masura; // Measurement unit type;

Cod Intern WinMentor; // Internal Code WinMentor;

Simbol Clasa // Class Symbol

20. GetStergeriProduse(const LastSyncDate: WideString; out Error: Integer): OleVariant;

Returns the list of items deleted after the date and time sent in the LastSyncDate parameter.

The returned structure will be:

„Cod Intern WinMentor;Data si ora stergerii;”

("Internal Code WinMentor;Deletion Date and Time;")

To add new partners, use the function:

21. AdaugaPartener(const InfoPart: WideString): Integer;

The structure of a *InfoPart* line is:

1. *ID Partener; // Partner ID;*
2. *Denumire partener; // Partner name;*
3. *Cod Fiscal; // Fiscal Code;*
4. *Sediul in localitatea; // Headquarters in the locality;*
5. *Adresa sediu; // Headquarters address;*
6. *Telefon sediu; // Headquarters phone;*
7. *Persoane de contact; //* separate prin „~” // Contact persons; // separated by "~"
8. *Simbol Clasa; // Class Symbol;*
9. *Simbol categorie de pret; // Price category symbol;*
10. *ID Agent implicit; // Default Agent ID;*
11. *Nr. Registrul comertului; // Trade Registry Number;*
12. *Observatii; // Remarks;*
13. *Simbol banca; // Bank symbol; // separated by "~" if there are more;*
14. *Nume banca; // Bank name; // separated by "~" if there are more;*
15. *Localitate banca; // Bank locality; // separated by "~" if there are more;*
16. *Cont banca; // Bank account; // separated by "~" if there are more;*
17. *Zi implicita plata; // Default payment day;*
18. *Nume sediu secundar // Secondary headquarters name; // separated by "~" if there are more;*
19. *Adresa sediului secundar; // Secondary headquarters address; // separated by "~" if there are more;*
20. *Telefonul sediului secundar; // Secondary headquarters phone;// separated by "~" if there are more;*
21. *Localitatea sediului secundar; // Secondary headquarters locality;// separated by "~" if*

there are more;

22. ID Agent pentru sediului secundar; // Agent ID for secondary headquarters; // separated by "~" if there are more;

23. CodExtern // External Code

24. Simboluri auto judete sedii de livrare // Auto symbols for delivery county headquarters

25. Simbol auto judet sediu principal // Auto symbol for main headquarters county

26. Flag care indica daca partenerul este persoana fizica (PF) // Flag indicating if the partner is a natural person (NP)

27. Scadenta implicita la vanzare // Default sales deadline

28. Simbol tip contabil implicit // Default accounting account symbol

29. Flag (P) care indica daca partenerul este producator // Flag (P) indicating if the partner is a producer

30. adresa eMail sediu social // eMail address for social headquarters

*31. adrese eMail sedii de livrare (*concatenate cu „~”) // eMail addresses for delivery headquarters (concatenated with "~")

32. TVAIncasare // VAT Collection (if D then the partner has VAT on collection)

33. SerAI // Invoice Series

34. NrAI // Invoice Number

35. Simbol Auto tara Sediu principal // Auto symbol for main headquarters country

If there are columns that you do not manage in your application or do not intend to reach **WinMENTOR**, you will send an empty string (i.e., ;;). However, the number of separators ";" must be respected.

21. GetNextPartID(out Error: Integer): WideString

The function is used to obtain the next available partner ID for partners. (This mechanism assumes that partner IDs are numeric).

22. ModificaPartener(const InfoPart: WideString): Integer;

InfoPart has the same structure as *AdaugaPartener* (AddPartner).

23. GetListaBanci(out Error: Integer): OleVariant;

Returns the list of national banks from **WinMENTOR** in the structure:
„Simbol;Denumire”
("Symbol;Name").

24. ComenziValide: Integer;

Validates the package of orders sent through SetDocsData. Returns **1** if the package is valid and **0** if not (in which case the error list must be read).

25. MonetareValide : Integer;

Validates the package of monetary transactions sent through the *SetDocsData* function. Returns **0** if the package contains errors and **1** if the package is valid.

26. ImportaMonetare: Integer;

Imports the package of monetary transactions and returns the number of transactions imported.

27. GetOferte(out Error : integer) : OleVariant.
Output data structure:
"PartID;ArtID;OfferStartDate;OfferEndDate;Price;Quantity"
(„PartID;ArtID;DataInceputOferta;DataSfarsitOferta;Pret;Cantitate”)

28. GetClaseArticole(out Error: Integer): OleVariant;
Returns:
„SimbolClasa;NumeClasa;”

("ClassSymbol;ClassName;")
To read the item nomenclature from **WinMENTOR**, use:

29. GetNomenclatorArticole(out Error: Integer): OleVariant;

Returns the item nomenclature in the structure:
CodExtern(Intern); // ExternalCode*(*Internal*);*
Denumire; // Name*;*
DenUM; // UnitOfMeasureName*;*
PretVanzare; // SellingPrice*;*
SimbolClasa; // ClassSymbol*;*
DenClasa; // ClassName*;*
CodExtern(Intern)Producator; // ManufacturerExternalCode*(*Internal*);*
Den.Producator; // ManufacturerName*;*
GestImplicita; // DefaultWarehouse*;*
CodExtern // ExternalCode (useful to know when the unique identifier is InternalCode);
CotaTVA; // VATRate*;*
DenUMSecundaraImplicita; // DefaultSecondaryUnitOfMeasure*;*
ParitateUMSecundaraImplicita; // SecondaryUnitOfMeasureParity*;*
Masa; // Weight;
Serviciu; // Service;
CodVamal; // CustomsCode;
PretMinim; // MinimumPrice;
CantImplicita; // DefaultQuantity;
PretValuta; // CurrencyPrice;
DataAdaug; // DateAdded;
Masa; // Weight;
PretVCuTVA; // PriceWithVAT;
Locatie; // Location;
PretReferinta; // ReferencePrice;

30. FactIntrareValida : Integer;

Validates the package of entry invoices transmitted through SetDocsData. Returns **0** if the package contains errors and **1** if the package is valid.
If the package is invalid, the errors that occurred are read with the *GetListaErori* (GetErrorList function).

The actual document import is done with the function:

31. ImportaFactIntrare : Integer;

Returns the number of imported invoices.
Here's an example of using these 2 functions above:

```
Erori.Clear;
  if WinMDocImp.FactIntrareValida = 0 then begin // I have validation errors
    ListaErori := WinMDocImp.GetListaErori;
    if VarIsArray(ListaErori) then
for i := VarArrayLowBound(ListaErori, 1) to VarArrayHighBound(ListaErori, 1) do
Erori.Items.Add(ListaErori[i]);
  end
  else ShowMessage('Au fost importate ' + Inttostr(WinMDocImp.ImportaFactIntrare));
```

32. GetVanzariExt(out Error: Integer): OleVariant;

Returns the list of sales for the working month in the structure:
„PartID; Zi; PrefixDoc; NrDoc; ArtID; Cant; DenUM; Pret; DenGest; CodInternArt; Locatie client; Adresa; Comision; CodFisca; MarcaAgent; ValAchizitie; CodPostal; ClasaArticol;”

("PartID; Day; DocPrefix; DocNumber; ArtID; Quantity; UnitOfMeasureName; Price; WarehouseName; InternalArticleCode; ClientLocation; Address; Commission; FiscalCode; AgentBrand; PurchaseValue; PostalCode; ArticleClass;").

33. GetIncasariClienti(An1, Luna1, An2, Luna2: Integer; const PartID: WideString; out Error: Integer): OleVariant

Returns only the total value of receipts within a specified interval.

34. GetVanzariLuna

Returns the list of invoices issued in a working month.
The returned structure is:

```
IDPartener; //  PartnerID;
Zi; //  Day;
NrFactura; // InvoiceNumber;
ID Articol; //  ArticleID;
NumarComanda; //  OrderNumber;
Cant; //  Quantity;
DenUM; //  UnitOfMeasureName;
Pret; //  Price;
MarcaAgent; //  AgentBrand;
```

Valoare Factura. // InvoiceValue.

35. GetSolduriExt - returns the breakdown of customer balances, which are composed of invoices and advances.

For a line describing a balance derived from an invoice, the structure is:

ID Partener; // PartnerID;
Factura; // Invoice; (a text indicating the source of the balance)
NrFactura; // InvoiceNumber;
DataFactura; // InvoiceDate;
Rest de plata; // RemainingPayment;
Termen de plata; // PaymentTerm;
Partner location for which the invoice was issued;
Marca Agent // Agent Brand;
Valoare factura // Invoice value;
Observatii factura; // Invoice remarks;

For an advance line, the structure is:
ID Partener; // PartnerID;
Avans; // Advance; (a text indicating the source of the balance);
Tip Document plata avans (exemplu Chit, CEC, OP etc); *// Payment Document Type (e.g., Receipt, Check, Bank Transfer, etc.);*
Nr Document plata avans; // Payment Document Number;
Data Document plata avans; // Payment Document Date;
Rest avans; // Remaining Advance;
Marca Agent incasator; // Receiving Agent Brand;
Valoare initiala avans; // Initial Advance Value;

36. GetComenziNefacturate
Returnează lista articolelor comandate, dar încă nefacturate pe structura:
IDArticol; // ArticleID;
Numar Comanda; // Order Number;
Cant; // Quantity;
Den Articol; // Article Name;

37. IncasariValideExt - validates the package with receipts sent through *SetDocsData*
38. ImportaIncasariExt – imports the package with receipts.

For setting the partner identification field in the database, use:

39. Procedure SetIDPartField(const FieldName: WideString);
FieldName can take the values: *CodExtern, CodFiscal sau CodIntern*
(ExternalCode, FiscalCode, or InternalCode;)

For setting the article identification field in the database, use:

40. Procedure SetIDArtField(const FieldName: WideString);
FieldName can take the values: *CodExtern sau CodIntern.*
(ExternalCode or *InternalCode*.)

41. GenCodParteneri: Integer

The function returns the next internal code, external code, or fiscal code, depending on the value of the constant *„Cod pentru identificare partener”* ("Code for partner identification") from *Constante generale > Import date din alte aplicații*. (General Constants > Import data from other applications).

42. GetPretVanzare(const ArtID, PartID: WideString; out Error: Integer): OleVariant;

The function returns the selling price or a list of selling prices if there are multiple prices for an article for a given partner.

43. GetListaCatPret(out Error: Integer): OleVariant;

Returns a list of price categories for the current company.

44. GetListaArtCatPret(out Error: Integer): OleVariant;

Returns a list of price categories specified for each article.

45. GetListaLocalitati(out Error: Integer): OleVariant;

Returns a list of locations for each partner of the current company.

46. GetSolduri(out Error: Integer): OleVariant;

Returns a list of strings with information about agent receipts. Each line contains information in the format:

„NrDoc;DataDoc;Rest;Termen;Agent;Valoare”
("DocNumber;DocDate;Balance;Term;Agent;Value")

Advances on agents are returned in the format:

„Doc;NrDoc;DataDoc;Suma;Agent;Valoare”
("Doc;DocNumber;DocDate;Amount;Agent;Value")

47. GetListaCarnete(out Error: Integer): OleVariant;

The function returns a list of booklets of documents used for internal exits to internal clients and transfers between warehouses.

48. GetNumarFactura(const SimbolCarnet: WideString; out Error: Integer): Integer;

The function checks if an invoice number is used in a given document booklet.

49. GetIncasariClienti(An1, Luna1, An2, Luna2: Integer; const PartID: WideString; out Error: Integer): OleVariant;

The function returns a list of customer receipts within a given interval *Year1.Month1 - Year2.Month2*.

50. ImportaComenzi: Integer;

Imports internal orders from data transmitted with *SetDocsData*. Returns the total number of imported orders.

51. ComenziValide: Integer;

The function validates if the structure of the internal order to be imported is valid.

52. SetIndexNart(const aIndexName: WideString);

A method used before implementing the *SetIDArtField* function. Currently, it is no longer necessary, but it has been kept in case someone still uses it.

53. ComenziValideExt: Integer;

The function validates if the structure of the order to be imported is valid.

54. ImportaComenziExt: Integer;

Imports orders from the data transmitted with *SetDocsData*. Returns the total number of imported orders.

55. GetVersiuni(out VerMentor, VerServer: Double): Integer;

56. GetInfoPers(IDPers: Integer; out Error: Integer): OleVariant;

Returns a list with the first name and last name of each employee.

57. GetInfoPart(const PartID: WideString; out Error: Integer): OleVariant;

Returns the name of the partner for the provided PartID.

58. GetListaClienti(AnInceput, LunaInceput: Integer;out Error: Integer): OleVariant;

Returns a list of strings with information about customers, starting from a specific month of a year, in the format:

„CodIntern;CodExtern;Denumire;CodFiscal;Localitate;Judet;Adr;Tel;MarcaAgent;DataFact; SediiPart;Simbol_Clasa;Denumire_Clasa;LocalitSedii”

("InternalCode;ExternalCode;Name;FiscalCode;City;County;Address;Phone;AgentBrand;DateFact;Branches;Symbol_Class;Name_Class;BranchLocations")

59. GetStringConstanta(Id: Integer; const Simbol: WideString): WideString;

60. GetArticoleVandute(const PartID: WideString; MarcaAgent, AnInceput, LunaInceput: Integer;out Error: Integer): OleVariant;

The function returns a string with the articles sold to a specific client by an agent starting from the provided month.year.

61. GetUltimeleVanzari(const ArtID, PartID: WideString; MarcaAgent, Cate: Integer; out Error: Integer): OleVariant;

GetDocFromFile;
GetVanzariExt

GetIntrari
GetListaSubunit
SetClasaArt
GetInfoBon
GetInfoBonExt
GetInfoIesiri
GetInfoIesiriExt
GetStocArtDetaliat
GetListaArtCatPretExt
BonuriConsumValide
ImportaBonuriConsum
GetListaDelegati
SetCmdImplicitAcceptat
GetOferteClienti
GetCritDiscPeArticole
GetCritDiscPeClase
GetCritDiscPart
GetStocuriPeGestiuni
GetReceptii
GetTransferuri
CheckDocument
GetTranzactiiInCurs
TransferuriValide
ImportaTransferuri
GetSuppliersOrders
AddProduct
SetReceivingList
GetReceivingStatus
GetInventoryOrders
SetInventoryOrders
GetDispozitiiDeLivrare
SetPickedList
ExistaFacturaExt
GetStocArticoleExt
ExistaFacturaIntrare
PotIntroduceDoc
GetDeliveryOrders
SetDeliveryList
GetListacarneteExt
GetInfoCmdNefacturate
GetNomenclatorLocalitati
GetMonede
NCValide
ImportaNoteContabile
GetStergeriProduse
GetInfoCmdSubNefacturate
AdaugaGestiune
ActualizeazaAcceptat
GetReceptiiIntrSubunit
GetInfoCmdBon

GetInfoCmdBonuri
GetIncasariFactura
PlatiValideExt
ImportaPlatiExt
GetNirAtribute
SetDescarcareAutomata
ModiProduct
AddClasaArt
ModifPretValide
ImportaModifPret
GetStocInitialAmbalaj
GetMiscariAmbalaje
GetStocArticoleExt2
ComenziFurnValide
ImportaComenziFurn
GetSoldFactNeop
GetIncasariLuna