## *Table of Content*

## *Introduction*

Elliott ActiveX components are designed to follow Microsoft's Windows Distributed Network Architecture (DNA). In Windows DNA, an application is broken down into three tier layers:

**Presentation Layer**
**Business Logic Layer**
**Database Layer**

Currently, Elliott Business Application software is considered between 1 to 2 layers. The Btrieve database that Elliott use can be considered as a database Layer. However, it is a very thin database layer for it does not provide as much functionality as the traditional SQL database. Even though Elliott Business Software currently supports both Btrieve and Micro Focus ISAM database file, Elliott ActiveX components will only support Btrieve database.

For the presentation and business logic, currently Elliott have both of them in the same application code. It is our intention to move Elliott Business Software toward a three tiers or even an n-tiers structure (see Windows DNA from Microsoft publication for more details). Our first step in supporting Windows DNA is to separate Presentation and Business Logic into two different layers.

The Business Logic layer, according to Windows DNA, is implemented as ActiveX components. By hiding the complicated business logic into the COM object (another name to say ActiveX), it make user interface layer a lot easier to implement. It also make coding much more manageable by re-use the same COM object. NETcellent utilize Merant's NetExpress 3.1 to create Elliott ActiveX component by using COBOL. You can rest easy that the ActiveX function we provide will behave the same way as in Elliott Business Software because they are essentially the same COBOL code that strip away the user interface portion.

In this documentation, we will mention words like ActiveX component, COM object and OLE Automation. Do not be confused because they are all interchangeable and refer to the same technology. OLE (Object Linking and Embedding) Automation is the earliest name that Microsoft adopts when this technology was first developed. It is later on called COM (Component Object Model). ActiveX is the trademark name Microsoft has for COM technology.

The Presentation layer can be Windows user interface, or the increasingly popular browser user interface. The Windows user interface can be developed with, for example, popular programming language like Visual Basic, as well many other programming language and tools available. The browser interface will utilize language like HTML, JavaScript, Perl, Java Applet/Servlet, or Active Server Page (ASP). However since our implementation with Business Logic layer is through ActiveX component, we will have to eliminate Perl, Java Applet/Servlet from our Presentation layer, unless the implementation does not require Elliott ActiveX components. For a web user interface, the most likely linking to Business Logic COM object layer is through the use of ASP page which is essentially VB script.

For Elliott Business Software, the separation of Presentation and Business Logic process will take place over time. Our first priority will be developing the most needed ActiveX components to support Elliott Business Software to interact with the World Wide Web. Even though these

ActiveX components are developed originally to support the World Wide Web, it is certainly not limited to the use of the World Wide Web only. Any programming language that support ActiveX under a supported operation system by Btrieve should work with Elliott's ActiveX components.

## *License Information*

The licensing requirement depends on whether you have a client or server application. If you have a server application (like web server) that utilize Elliott ActiveX Components, then the licensing of Elliott ActiveX components is per server. This means if you have two web servers at your location that utilize Elliott AcitveX components, then you need to have them licensed individually.

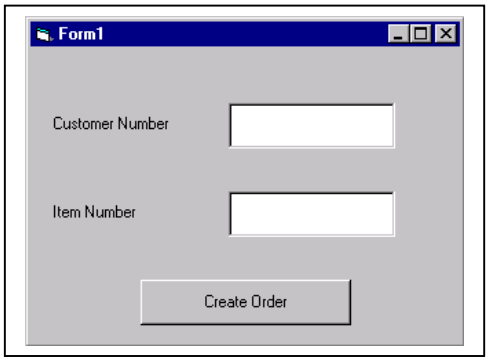
If you have a client application (like Windows desktop application) that utilize Elliott ActiveX Component, then the licensing of Elliott ActiveX component is per site.

If you have both Server and Client application that utilize Elliott ActiveX components, then the licensing is per server.

## *Example of Using Elliott ActiveX Component*

The following is an example by using Visual Basic to create Sales Order in Elliott. We will use the ActiveX component Elliott Create Sales Order (ELICRORD.DLL) to do so. For your information, all Elliott COM object has its own DLL file. The naming convention for all Elliott COM object is each object always start with "ELI" and it is always less or equal than 8 digits long. We never combine two Elliott ActiveX object into one DLL file.

To show you how ELICRORD COM component works, we will first start up Visual Basic and create a standard EXE project. Then, we will create two text boxes on the form with its id as TEXT1 and TEXT2 respectively. TEXT1 represent the text box entry for Customer Number and TEXT2 represent Item Number.




Then we create a command button 'Command1' to represent "Create Order" and enter the following codes for event Command1_Click:

```
Private Sub Command1_Click()
  Set X = CreateObject("ELICRORD")
  Dim OrderNo As Long
  Dim Status As Long
  X.CustomerNo = Text1.Text
  Status = X.CreateHeader("M:\MACOLA\DATA_20", OrderNo)
  If Status = 0 Then
    X.ItemNo = Text2.Text
    Status = X.CreateLineItem("M:\MACOLA\DATA_20", OrderNo)
    If Status = 0 Then
       Status = X.FinishOrder("M:\MACOLA\DATA_20", OrderNo)
      If Status = 0 Then
        MsgBox "Order# " & OrderNo & " Created Succuessfully"
      Else
        MsgBox "Error# " & Status & " When Complete Order"
      End If
    Else
      MsgBox "Error# " & Status & " When Create Line Item"
    End If
  Else
    MsgBox "Error# " & Status & " When Create Header"
  End If
  Set X = Nothing
End Sub
```

As you can see in this example, to create a sales order header, all you have to do is to supply a valid customer number and use the method **CreateHeader**. To create a line item, all you have to do is to supply a valid line item and the order number, then use the method **CreateLineItem**. At the end, we will need to use **FinishOrder** method to make the order complete. You may assign value to many additional properties in this component and system will then honor the value you assigned and use it. However, it is not required. For example, if you specify the Ship Via code to be "U", then ELICRORD component will use it when create the sales order header. If you do not specify it, then it will look up the customer file for default ship via code.

In all method calls of Elliott ActiveX components, if the return status code = 0, that means the method has been executed successfully. If not, then there's an error. For more information of what the error code stand for, please refer to the detail documentation of each COM object.

In each one of the method calls, the first parameter is always the data file path where the Elliott Btrieve data reside. There may be additional parameters for each method call. In this case, the CreateHeader method use the 2nd parameter to return the Order Number that system assigned sequentially. The CreateLineItem method use the 2nd parameter to specify the order number to add this line item. The same is true for FinishOrder method.

For a developer, the real value of using Elliott ActiveX components is to develop application that interacts with Elliott has become a lot easier. In Elliott Business Software, just the COP Line Item Screen portion, the COBOL source code has exceeded 12,000 lines. By encapsulating the complicated business logic of creating an Elliott sales order in the ELICRORD component, it relieve developer's burden to understand how sales order creation works and greatly speed up the developing process.

ELICRORD (Elliott Create Sales Order) COM component can be used on a web site with e-Business or e-Commerce ability to create Elliott Sales Order directly. It can also be used to create a Windows user interface front end to create Elliott Sales Order. For any reason that you need to insert sales order into Elliott, you should consider using ELICRORD COM component.

## *Special Considerations with Elliott ActiveX Components*

### Do Not Use Dim with Elliott ActiveX Components

In programming language like Visual Basic or VB script, always use **CreateObject** to create Elliott COM object. Even though for most ActiveX components, you should be able to use the following Visual Basic DIM statement to create the object:

```
Dim X as New ELICRORD
      Or
Dim X as ELICRORD
SET X = New ELICRORD
```

Unfortunately, this method does not work with Elliott's ActiveX components. Instead, you need to use the following method to create object:

```
Set X = CreateObject("ELICRORD")
```

With Active Server Page (ASP), you need to use the following method to create object:

```
Set X = Server.CreateObject("ELICRORD")
```

For ASP, this is working fine and Microsoft Visual InterDev tool will recognize Elliott's ActiveX object at developing time and displayed the helpful intelisense information.

However, for Visual Basic, if you use CreateObject to instantiate Elliott object, even though it works fine when the code is executed, the intelisense does not display at the developing time. To make intelisense work in Visual Basic, you need to use the Dim statement like the following:

```
Dim X as ELICRORD
'Set X = CreateObject("ELICRORD")
```

After done with the coding, before you execute the VB code, you will need to make sure to comment out the Dim statement like following:

```
'Dim X as ELICRORD
Set X = CreateObject("ELICRORD")
```

### Use Universal Naming Convention (UNC) for File Path

In the example of this document, we use the following method to create an Elliott Sales Order Header:

```
Status = X.CreateHeader("M:\MACOLA\DATA_20", OrderNo)
```

In this case, the path we supplies "M:\MACOLA\DATA_20", is a network path where M: drive is a network mapped drive. You should be aware of that all network mapped drive are logical assigned and can be different depend on the logon user.

If the application you developed is intended to work as a client application where it interact with the logon user, then path like "M:\MACOLA\DATA_20" is OK. However, if the application you developed is intended to work as a server application where it may interact other than the current logon desktop user (i.e. ASP), then you should use the Universal Naming Convention (UNC) to specify your file path:

\\NSI\SYS\MACOLA\DATA_20

Where:

NSI: This is your server name
SYS: This is either a volume name (Netware) or a Share name (NT)
\MACOLA\DATA_20: This is the physical path under that volume or share name

Since UNC path always works, we recommend you always use UNC path. However, you should also be aware that Elliott ActiveX component only allows for 32 bytes data file path at this moment. In some situations where you have a long server name or a long share name, you might exceed the 32bytes limitation. Please make sure your UNC path length is less or equal than this limitation.

## Trustee/Security Issue

When you develop a Web application with Active Server Page (ASP) that need to access Elliott's database, the most common mistake that you can make is the Trustee/Security problem. Under most situations, when an Active Server Page is executed, it is login as an anonymous user like:

IUSR_*SERVERNAME*

Where *SERVERNAME* is your web server machine name. IUSR_*SERVERNAME* account is created automatically for the Internet Information Server (IIS). Since most of the web access is anonymous access (means you don't need to type in a user name and password in order to access a web site), therefore, most of the time, your ASP pages are operated under the user account IUSR_*SERVERNAME*. When IUSR_*SERVERNAME* account is first created, it might only have limited right to access the current web server. It almost never has right to access data on a different server. Therefore, if your Elliott Btrieve Data resides on a different server from the web server, you must make sure the anonymous user has the privilege to access Elliott Data. You can accomplish this by doing either one of the following: (1) Create IUSR_*SERVERNAME* on the server where Elliott data reside and make sure this user has enough privilege to access Elliott data. Make sure to synchronize the password on both servers. (2) Assign anonymous user to an existing user that has privilege to access both web server and server where Elliott data reside.

## File Does Not Exist Will Stop ActiveX Component

In Elliott V7, when open a file I-O and if that file does not exist, Elliott V7 will automatically initialize it. However, in Elliott ActiveX component, this situation will cause the ActiveX component stop running instead of returning a status code "1". If you are using Elliott ActiveX component in the ASP page, you will either receive an internal server error 500, or a detail message showing the ASP page line with error code "800706BE" (Remote Procedure Call Failed). The cause for this is we currently do not support file creation inside of Elliott ActiveX.

Therefore, you should make sure that all the files has been properly initialized in Elliott V7 before you attempt to use an Elliott V7 ActiveX component. The way you can do this is to perform initialization of all the files module by module. The initialization routine in V7 will not initialize files that already exist. Therefore, you won't lose any data by initializing files.

## *Installation*

### Installation Components

There are two type of files needed for Elliott ActiveX Components:

(1) Elliott Run Time Files
(2) Elliott ActiveX Components

If you receive a CD package for Elliott ActiveX Components, insert the CD into your CD-ROM, Elliott Installation Utility should start automatically. If it does not start, click **setup.exe** on the CD-ROM drive to start the installation.

If you receive a single file installation package, the file name should be ELIAX.EXE. Simply execute this file either in DOS prompt, or from window explore. This will start the installation utility.

By default, Elliott ActiveX components will be installed into C:\ELLIAX directory if you use the installation utility. All Elliott ActiveX components will be registered correctly if you use installation utility and use the default C:\ELIAX as the target directory.

If you install Elliott ActiveX components through the installation utility and use the default path, then after you done with the installation, proceed to **Btrieve Components Installation**. However, if you choose a different path, or if you install the components manually, then you will need to perform the registration manually. Please refer to the next section for more details.

### Manual Installation and Registration

If you have successfully installed Elliott ActiveX components on one machine, it is possible for you to copy this directory to another machine. You may also install Elliott ActiveX components to a network path so all users on the network can share these files. However, you need to make sure that this directory is in search path and Elliott Components are registered manually.

You can add this directory into your search path by modifying C:\AUTOEXEC.BAT. You may also modify the login script to accomplish this. If you do not add this directory into your search path, system will complain certain DLL Files are missing when you execute the Elliott ActiveX components.

You can manually register Elliott COM object by using the following command at DOS prompt:

```
REGSVR32 ELICRORD
```

You only need to register the Elliott COM object that you intend to use. If system complains REGSVR32 is not a recognized command, please make sure C:\WINNT\SYSTEM32 (for WINNT/2000), or C:\WINDOWS\SYSTEM (for WIN95/98) is in search path.

Since NETcellent does not modify existing component's Class-Id, you do not need to register the components again if you install a newer version of the component manually. However, you may have difficulty install them if they are in used already (loaded in the memory). In that case, you may have to reboot your machine in order to install a newer version of the component. If you use MTS to manage your Elliott components, you can simply choose to stop the application in MTS

which will unload the DLL from memory and allow you to manually install the newer version of components without rebooting your machine.

## Btrieve Components Installation

All Elliott ActiveX components need Btrieve Client installed on the same machine. All Elliott ActiveX components are 32-bit and need 32-bit Btrieve client to operate. Even though it is possible to make it work with Btrieve 6.15.450, currently, we will only support PSQL 7 or higher version.

## Component Services (MTS)

Component Service is a standard feature in Windows 2000. In Windows NT V4.0, it comes with Windows NT optional pack free of charge, and it is called Microsoft Transaction Server (MTS). Component Services is an important feature in Windows DNA. It allows the Business Logic layer to be scalable and manageable. Essentially Component Service, or Microsoft Transaction Server (MTS) is a server of COM object. It manage COM object to achieve the following:

(1) Traditionally, a DLL utilized on a web server runs in the same memory space as the web server (in process which is more efficient than out process like EXE). Therefore, when the DLL crashes, it can stop the web server as well. By using MTS to manage COM object, it isolates COM objects in their own memory space. Therefore, if the COM object crashes, it only crashes its own memory space without affecting the main service like the web server. Also, when it crashes, the MTS is intelligently enough to know that the COM object should be started automatically from scratch the next time the COM object is being accessed.

(2) A COM object managed by MTS can physically locate on a different server than the client application who uses it. For example, a web server that utilize Elliott COM object. It is possible to place Elliott COM object on a different application server and improve the scalability. This capability is called DCOM (Distributed COM). To use DCOM, it is recommended that the ActiveX component should provide long method call where it pass all parameters in a single call. Currently, Elliott ActiveX components do not provide long method. When your application assigns value to each property, it will result in round trip traffic to the COM object server if they are located on a different machine. Therefore, we do not suggest you to use Elliott COM object on a different server than the client application. Eventually, we will develop long method, and keep the short method call for Elliott COM object.

(3) MTS can improve performance of the COM object by providing object pooling. COM object that's destroyed in the application still may be cached in MTS for the next application to access. Since instantiate an object may be time consuming, object pooling can potentially significantly improve system performance.

(4) Provide transaction support to allow multiple processes (ActiveX component) either all go through, or none of them go through to improve process integrity.

(5) Allow you to stop an application and unload DLL from memory. This will allow you to install newer version of Elliott ActiveX components without the need to reboot the machine.

You should be aware that Elliott COM object currently does not support transaction with MTS. However, all Elliott COM objects are designed to be stateless and therefore are fully supported by MTS.

It is recommended that you setup an MTS package to include all Elliott ActiveX COM components. This is especially important if you are developing a web application where you want to ensure that your web server is not going to stop if Elliott ActiveX COM components crash. To setup Elliott ActiveX COM components in an MTS package in Windows 2000, go to **start** -> **program** -> **Administrative Tools** -> **Component Services**.

In Component Services, on your left pane, expand **Component Services**, **Computers**, **My Computers** and you should see **COM+ Applications**. Highlight **COM+ Application** and right click. Choose New Application. When the Wizard comes up, choose to create an empty application and name it "Elliott ActiveX". You should then see "Elliott ActiveX" show up as a package on your right pane.

Now put all Elliott ActiveX Components into "Elliott ActiveX" application: Expand **COM+ Applications**, Expand "**Elliott ActiveX**". Highlight **Components** and right click. Choose New Component. When the Wizard comes up, you can choose either "Install New Components" or "Import Components that are already registered". Choose each one of Elliott ActiveX components until you are done.

Each application in Component services will be managed by a dllhost.exe. The additional memory burden of doing so is about 12-16 Meg for Elliott ActiveX. You may verify this by looking at the processes tab of task manager.

## *ActiveX Components*

In the following document, "**Set Property**" means the property that is to be set by the application. When a property is flagged as **Optional**, it means that you do not have to set the property before you make the method call. In most of the cases, when you do not set the optional property, system will lookup a default value. When you set a specific value for optional property, system will honor that value and use it. "**Get Property**" means the property is to be returned back from the ActiveX components.

PIC clause is to identify the data type and length of the property. Following are some examples:

```
      PIC 9(6):         Numeric 6 digits
      PIC X(6):         Alpha Numeric 6 digits
      PIC 9(3)V99:      3 digit Numeric left and 2 digit right of decimal
      PIC S9(3)V99:     Same as previous, however, negative value allowed
```

Generally speaking, you can pass a value to a property that's bigger than what Elliott ActiveX Component allowed, and no error will be raised. However, the value will be truncated.

## Calculate Freight (ELIGETFR.DLL)

Elliott Business Software has a Freight Calculation function. Once this function is setup, it can provide useful information for calculating freight. With this ActiveX component, you can calculate the freight for an order. You can also calculate freight if you know the customer and item. Or you can calculate freight if you know the customer and weight. It is assumed that you are going to ship the merchandize in one box.

Elliott Create Sales Order (ELICRORD) will also calculate freight for you if the feature is turned on in Elliott. Therefore, this component is mostly used to quote the freight amount to your customer before the sales order is created.

**Set Property**

```
OrderNo             pic 9(6).
CustomerNo          pic x(6).
ShipTo              pic x(4).             Optional
ShipViaCode         pic x(2).             Optional
ItemNo              pic x(15).
Quantity            PIC S9(9)V999.        Optional
TotalWeight         PIC S9(7)V99.
City                PIC X(15).            Optional
State               PIC X(2).             Optional
ZipCode             PIC X(10).            Optional
Country             PIC X(20).            Optional
TermsCode           PIC X(2).             Optional
```

You can provide the following information to get freight amount

```
(1) OrderNo
(2) CustomerNo + (ShipTo) + (ShipViaCode) + Item + (TotalWeight) + (Quantity)
(3) CustomerNo + (ShipTo) + (ShipViaCode) + TotalWeight + (Quantity)
```

The property in parenthesis means they are optional.  System will lookup and use a default if they are not provided.

**Get Property**

```
ShipViaDesc         PIC X(15).
FreightAmount       PIC S9(5)V99.
Weight              PIC S9(7)V99.
ShipViaNo           PIC X(2).
```

**Method:**

```
R = A.FreightCalculation(P1)
```

**Parameters:**

```
A  : Object
P1 : PASSING-PATH           PIC X(32).          Required
R  : PASSING-RETURN-CODE    pic 9(3).           Return Code (Variable)
```

**Return Code:**

```
  0 = OK
  1 = File Error
  2 = Data Missing (Customer No Missing)
  3 = Order Not On File
  4 = Data Missing (Both Item & Weight Missing)
  5 = Ship Via Not On File
  6 = Order Line Item Not On File
  7 = Customer Not On File
  8 = Ship To Not On File
  9 = Item Not On File
 10 = No Carrier Exist In This Ship Via
 11 = Carrier Code Not On File
 12 = Carrier Mode Not On File
 13 = Freight Zone Not On File
 14 = Freight Rate Not On File
 15 = No Service Available
```

**Code Example: (VB)**

```
'DIM A As ELIGETFR
Set A = CreateObject("ELIGETFR")
A.CustomerNo = "000100"
A.Item = "CLOCK"
R = A.FreightCalculation("F:\MACOLA\DATA_96\")
If R = 0 Then
  S = A.ShipViaDesc
  T = A.ShipViaNo
  U = A.Weight
  V = A.FreightAmount
  MsgBox "Freight Amount is " & V
End If
Set A = Nothing
```

The above example code show how to calculate freight amount if we try to ship Item "CLOCK" to CustomerNo "000100".  In addition to calculate Freight Amount,

this component can also be used to tell you the default ship via and weight information.

## Create Sales Order (ELICRORD.DLL)

This object creates sales order in Elliott. It can also be used to delete sales order. To create an order header, you only need to assign the property of CustomerNo. To create a line item, you only need to assign the property of ItemNo. For the property that you do not assign a value, system will automatically figure out a default value. However, if you do assign a value, then system will honor the value you assign. The property CheckNo, CheckDate, CheckAmount and PaymentType is used to apply prepaid amount or outstanding credit to the order. The can be used in conjunction with Credit Card Cash Receipt ActiveX component. HoldTrxID is used to assign the reserved inventory to the sales order to be created – to be used in conjunction with Hold Trx ActiveX components.

System assigns order number sequentially when the order header is created. However, system maintain a different sequence of order number that created by ActiveX component. To set the starting order number of ELICRORD, go to Global Setup -> COP Func -> Order Header -> Next Order Number of Web Order.

**Set Property**

| | | |
|---|---|---|
| OrdeNo | PIC 9(6). | Required (Except CreateHeader) |
| CustomerNo | PIC X(6). | Required |
| BillToName | PIC X(30). | Optional |
| BillToAddress1 | PIC X(30). | Optional |
| BillToAddress2 | PIC X(30). | Optional |
| BillToCity | PIC X(15). | Optional |
| BillToState | PIC X(2). | Optional |
| BillToZipCode | PIC X(10). | Optional |
| BillToCountry | PIC X(20). | Optional |
| ShipToNo | PIC X(4). | Optional |
| ShipToName | PIC X(30). | Optional |
| ShipToAddress1 | PIC X(30). | Optional |
| ShipToAddress2 | PIC X(30). | Optional |
| ShipToCity | PIC X(15). | Optional |
| ShipToState | PIC X(2). | Optional |
| ShipToZipCode | PIC X(10). | Optional |
| ShipToCountry | PIC X(20). | Optional |
| ShipInstr1 | PIC X(10). | Optional |
| ShipInstr2 | PIC X(40). | Optional |
| Comment1 | PIC X(40). | Optional |
| Comment2 | PIC X(35). | Optional |
| Comment3 | PIC X(35). | Optional |
| Location | PIC X(35). | Optional |
| ShipVia | PIC X(2). | Optional |
| FreightAmount | PIC X(2). | Optional |
| ItemNo | PIC S9(5)V99. | Required |
| Quantity | PIC X(15). | Required/Optional |
| UnitPrice | PIC S9(9)V999. | Optional |
| DiscountPercent | PIC S9(6)V9999. | Optional |
| RequestDate | PIC 9(3)V99. | Optional |
| CustomerPONo | PIC 9(8). | Optional |
| TermsCode | PIC X(2). | Optional |

```
OrderType                   PIC X(1).           Required/Optional
HoldTrxID                   PIC 9(9)            Optional
CheckNo                     PIC 9(6).           Optional
CheckDate                   PIC 9(8).           Optional
CheckAmount                 PIC 9(7)V99.        Optional
PaymentType                 PIC X(1).           Optional
MiscChargeAmount            PIC S9(5)V99        Optional
TaxCode1                    PIC X(3)            Optional
TaxCode2                    PIC X(3)            Optional
TaxCode3                    PIC X(3)            Optional
TaxPercent1                 PIC 9(3)V99         Optional
TaxPercent2                 PIC 9(3)V99         Optional
TaxPercent3                 PIC 9(3)V99         Optional
```

**Get Property**

```
SalesTaxAmount1             PIC S9(7)V99        Optional (Get)
SalesTaxAmount2             PIC S9(7)V99        Optional (Get)
SalesTaxAmount3             PIC S9(7)V99        Optional (Get)
```

**Method:**

```
R = A.CreateHeader(P1,P2)
R = A.CreateLineItem(P1,P2)
R = A.DeleteOrder(P1,P2)
R = A.FinishOrder(P1,P2)
```

FinishOrder method is used to assign the order status to complete or selected (I or C type of order), and indicate to other Elliott process that this order is now available for inquiring/processing. It also calculates the total sales, freight, tax and commission amount.

**Parameters:**

```
A  : Object
P1 : PASSING-PATH           PIC X(32).          Required
P2 : PASSING-ORDER-NO       PIC X(4) COMP-5     Required
R  : PASSING-RETURN-CODE    PIC 9(3).           Return Code (Variable)
```

**Return Code:**

```
  0 = OK
  1 = File Error
  2 = Data Missing (Customer No Missing) (Create Header)
  3 = Data Missing (Order No or Item Missing) (Create Line Item)
  4 = Data Missing (Order No Missing) (Finish & Delete Order)
  5 = Customer Not On File (Create Header)
  6 = Ship To Not On File (Create Header)
  7 = Order Not On File (Create Line Item & Delete Order)
  8 = Item Not On File (Create Line Item)
  9 = Inv Location Not On File (Create Line Item)
 10 = NSI Control File Not Found
 11 = Customer PO No Duplicate
 12 = Terms Code Not On File
 13 = Ship Via Code Not On File
 14 = Order Type Must Be "I" or "O"
 15 = No Serial Number Assign, Line Item Not Created
```

```
16 = Line Item Qty To Ship Must Be 1 when Hold Trx ID is provide
17 = Hold Trx Not Found
18 = Hold Trx Item No Mismatch
19 = Fail To Apply A/R Open Item

1025 = A1
1026 = A2
1027 = A1 & A2
1028 = A3
1029 = A1 & A3
1030 = A2 & A3
1031 = A1 & A2 & A3
1032 = A4
1033 = A1 & A4
1034 = A2 & A4
1035 = A1 & A2 & A4
.............................. (1036 - 1150)
1151 = A1 & A2 & A3 & A4 & A5 & A6 & A7

Formula = 1024 + A1 + A2 + A3 + A4 + A5 + A6 + A7
Yes =             1    2    4    8   16   32   64
No  =             0    0    0    0    0    0    0

Hold Status

A1 = This Order is On Hold Based on the AR Customer File
A2 = This Order is On Hold Based on the AR Terms Code
A3 = This Order is On Hold For Exceeding Credit Limit
A4 = This Order is On Hold For Account Is Past Due
A5 = This Customer is On Hold For Terms Code Customer On Hold
A6 = This Order & Customer is On Hold For FFL Expire
A7 = This Order is On Hold For Handgun & Ammo Coexisted
```

Please be aware, in this ELICRORD component, when you receive a return status code that's greater than 1024, it is actually not an error. Instead, it is a status code that indicates the order is put on hold (if this feature has turned on in Elliott).

**Code Example: (VB)**

```
'DIM A AS ELICRORD
Set A = CreateObject("ELICRORD")
A.CustomerNo = "000174"
A.ShipTo = "0001"
A.CustomerPONo = "WEB"
A.Location = "LA"
A.ShipInstr1 = "WEB ORDER"
A.CheckNo = 100
A.CheckDate = Date()
A.CheckAmount = 100
A.PaymentType = "P"
R = A.CreateHeader("F:\MACOLARD\DATA_96\",B)
IF R = 0 THEN
   A.ItemNo = "1005-1416-0932"
   A.Quantity = 10
   A.RequestDate = Date()
```

```
   R = A.CreateLineItem("F:\MACOLARD\DATA_96",B)
   IF R = 0 THEN
      R = A.FinishOrder("F:\MACOLARD\DATA_96",B)
      If R = 0 Then
            MsgBox "Successfully Creating Order# " & B
      Else
            R = A.DeleteOrder("F:\MACOLARD\DATA_96",B)
            MsgBox "Error #" & R & " Completing Order"
      End If
   ELSE
      R = A.DeleteOrder("F:\MACOLARD\DATA_96",B)
      MsgBox "Error# " & R & " Creating Line Item"
   END IF
ELSE
   MsgBox "Error# " & R & " Creating Order Header"
END IF
Set A = Nothing
```

## Order Object (ELIORDER.DLL)

This object handles Elliott Sales Order and is a superset of ELICRORD (Create Sales Order) object. It is our intention to eventually phase out support for ELICRORD object. ELIORDER object contains method to create, change and delete Elliott sales order. You can also use it for Order Inquiry purpose by giving specific order number, or customer# and get all the order belong to that customer. Special support of getting the serial number for "I" type of order (or "O" type of order that's selected) is also provided. In addition, user can get Starship Tracking Number information if they are available.

To create an order header, you only need to assign the property of OrderCustomerNo. To create a line item, you only need to assign the property of LineItemItemNo. For the property that you do not assign a value, system will automatically figure out a default value. However, if you do assign a value, then system will honor the value you assign. The property OrderCheckNo, OrderCheckDate, OrderPayment and OrderPaymentType is used to apply prepaid amount or outstanding credit to the order. The can be used in conjunction with Credit Card Cash Receipt ActiveX component. HoldTrxID is used to assign the reserved inventory to the sales order to be created – to be used in conjunction with Hold Trx ActiveX components.

System assigns order number sequentially when the order header is created. However, system maintain a different sequence of order number that created by ActiveX component. To set the starting order number of ELIORDER, go to Global Setup -> COP Func -> Order Header -> Next Order Number of Web Order.

**Set Property**

```
HoldTrxID                    PIC 9(9).      Optional
LineItemDiscPct              PIC 9(3)V99.   Optional
LineItemItemNo               PIC X(15).     Optional
LineItemPickSeq              PIC X(8).      Optional
LineItemQtyOrdered           PIC S9(9)V999  Optional
LineItemReqDate              PIC 9(8).      Optional
LineItemSeqNo                PIC 9(3).      Optional
LineItemUnitPrice            PIC S9(6)V9999 Optional
LineLsSerialNo               PIC X(15).     Optional
```

```
OrderBillToAddr1                PIC X(30).     Optional
OrderBillToAddr2                PIC X(30).     Optional
OrderBillToCity                 PIC X(15).     Optional
OrderBillToCountry              PIC X(20).     Optional
OrderBillToName                 PIC X(30).     Optional
OrderBillToNo                   PIC X(6).      Optional
OrderBillToState                PIC X(2).      Optional
OrderBillToZipCode              PIC X(10).     Optional
OrderCheckDate                  PIC 9(8).      Optional
OrderCheckNo                    PIC 9(6).      Optional
OrderComment1                   PIC X(35).     Optional
OrderComment2                   PIC X(35).     Optional
OrderComment3                   PIC X(35).     Optional
OrderCustomerNo                 PIC X(6).      Optional
OrderFreight                    PIC S9(5)V99   Optional
OrderFrtPayCode                 PIC X(1).      Optional
OrderLocation                   PIC X(2).      Optional
OrderNo                         PIC 9(6).      Required (Except CreateHeader)
OrderPayment                    PIC 9(7)V99.   Optional
OrderPaymentType                PIC X(1).      Optional
OrderPONo                       PIC X(25).     Optional
OrderShipInstr1                 PIC X(40).     Optional
OrderShipInstr2                 PIC X(40).     Optional
OrderShippingDate               PIC 9(8).      Optional
OrderShipToAddr1                PIC X(30).     Optional
OrderShipToAddr2                PIC X(30).     Optional
OrderShipToCity                 PIC X(15).     Optional
OrderShipToCountry              PIC X(20).     Optional
OrderShipToName                 PIC X(30).     Optional
OrderShipToNo                   PIC X(4).      Optional
OrderShipToState                PIC X(2).      Optional
OrderShipToZipCode              PIC X(10).     Optional
OrderShipVia                    PIC X(2).      Optional
OrderTermsCode                  PIC X(2).      Optional
OrderType                       PIC X(1).      Optional
LineItemQtyBackOrd              PIC S9(9)V999  Optional
LineItemQtyToShip               PIC S9(9)V999  Optional
OrderTaxCode1                   PIC X(3).      Optional
OrderTaxCode2                   PIC X(3).      Optional
OrderTaxCode3                   PIC X(3).      Optional
OrderTaxPct1                    PIC 9(2)V9999. Optional
OrderTaxPct2                    PIC 9(2)V9999. Optional
OrderTaxPct3                    PIC 9(2)V9999. Optional
OrderMiscCharge                 PIC S9(5)V99   Optional
OrderEDIFlag                    PIC X(1)       Optional
```

**Get Property**

```
LineItemBackorderable           PIC X(1).      Optional
LineItemCommCalcType            PIC X(1).      Optional
LineItemCommPctAmt              PIC S9(5)V99.  Optional
LineItemCopyToOrderNo           PIC 9(6).      Optional
LineItemCtlFlag                 PIC X(1).      Optional
LineItemDesc1                   PIC X(30).     Optional
LineItemDesc2                   PIC X(30).     Optional
LineItemDiscPct                 PIC 9(3)V99.   Optional
LineItemItemNo                  PIC X(15).     Required
```

```
LineItemLotExpDate              PIC 9(8).      Optional
LineItemLotNo                   PIC X(15).     Optional
LineItemNoOfPackage             PIC 9(4).      Optional
LineItemOrderNo                 PIC 9(6).      Optional
LineItemOrgItemNo               PIC X(15).     Optional
LineItemPickSeq                 PIC X(8).      Optional
LineIteMPOXrefSeqNo             PIC 9(3).      Optional
LineItemPrcLvlNo                PIC 9(2).      Optional
LineItemPriceFixedFlag          PIC X(1).      Optional
LineItemPriceOrg                PIC S9(6)V9999 Optional
LineItemPrmDate                 PIC 9(8).      Optional
LineItemProdCate                PIC X(3).      Optional
LineItemQtyBackOrd              PIC S9(9)V999  Optional
LineItemQtyOrdered              PIC S9(9)V999  Required/Optional
LineItemQtyReturnToStock        PIC S9(9)V999  Optional
LineItemQtyToShip               PIC S9(9)V999  Optional
LineItemReasonCode              PIC X(6).      Optional
LineItemReqDate                 PIC 9(8).      Optional
LineItemSelectCode              PIC X(1).      Optional
LineItemSeqNo                   PIC 9(3).      Optional
LineItemStockedFlag             PIC X(1).      Optional
LineItemTaxableFlag             PIC X(1).      Optional
LineItemTaxableFlag1            PIC X(1).      Optional
LineItemTaxableFlag2            PIC X(1).      Optional
LineItemTaxableFlag3            PIC X(1).      Optional
LineItemTotalQtyOrdered         PIC S9(9)V999  Optional
LineItemTotalQtyShipped         PIC S9(9)V999  Optional
LineItemUnitCost                PIC S9(6)V9999 Optional
LineItemUnitPrice               PIC S9(6)V9999 Optional
LineItemUnitWeight              PIC S9(5)V999  Optional
LineItemUOM                     PIC X(2).      Optional
LineItemVendorNo                PIC X(6).      Optional
LineLsBin                       PIC X(8).      Optional
LineLsItemNo                    PIC X(15).     Optional
LineLsLineNo                    PIC 9(3).      Optional
LineLsOrderNo                   PIC 9(6).      Optional
LineLsQty                       PIC S9(6)V999  Optional
LineLsSerialNo                  PIC X(15).     Optional
Order856ExportedFlag            PIC X(1).      Optional
OrderAccumFreight               PIC S9(5)V99   Optional
OrderAccumMiscCharge            PIC S9(5)V99   Optional
OrderAccumSalesTax              PIC S9(7)V99   Optional
OrderAccumTotalSaleAmt          PIC S9(7)V99   Optional
OrderAccumTotalTaxableAmt       PIC S9(7)V99   Optional
OrderApplyToNo                  PIC 9(6).      Optional
OrderARRef                      PIC X(30).     Optional
OrderBillToAddr1                PIC X(30).     Optional
OrderBillToAddr2                PIC X(30).     Optional
OrderBillToAddr3                PIC X(30).     Optional
OrderBillToCity                 PIC X(15).     Optional
OrderBillToCountry              PIC X(20).     Optional
OrderBillToFreeForm             PIC X(1).      Optional
OrderBillToName                 PIC X(30).     Optional
OrderBillToNo                   PIC X(6).      Optional
OrderBillToState                PIC X(2).      Optional
OrderBillToZipCode              PIC X(10).     Optional
OrderBOLPrinted                 PIC X(1).      Optional
```

```
OrderCashAcct                   PIC X(24).     Optional
OrderCheckDate                  PIC 9(8).      Optional
OrderCheckNo                    PIC 9(6).      Optional
OrderCommAmt                    PIC S9(6)V99   Optional
OrderComment1                   PIC X(35).     Optional
OrderComment2                   PIC X(35).     Optional
OrderComment3                   PIC X(35).     Optional
OrderCommPct                    PIC S9(2)V99   Optional
OrderCustBalMethod              PIC X(1).      Optional
OrderCustomerNo                 PIC X(6).      Required
OrderDate                       PIC 9(8).      Optional
OrderDateBilled                 PIC 9(8).      Optional
OrderDateEntered                PIC 9(8).      Optional
OrderDatePicked                 PIC 9(8).      Optional
OrderDepartment                 PIC X(8).      Optional
OrderDeptNo                     PIC X(6).      Optional
OrderDiscPct                    PIC 9(3)V99.   Optional
OrderEDIExportedFlag            PIC X(1).      Optional
OrderEDIFlag                    PIC X(1).      Optional
OrderEDIShipmentNo              PIC 9(10).     Optional
OrderEDIShipToFlag              PIC X(1).      Optional
OrderFreight                    PIC S9(5)V99   Optional
OrderFreightAcct                PIC X(24).     Optional
OrderFrtPayCode                 PIC X(1).      Optional
OrderInvoiceDate                PIC 9(8).      Optional
OrderInvoiceNo                  PIC 9(6).      Optional
OrderJobNo                      PIC X(6).      Optional
OrderLocation                   PIC X(2).      Optional
OrderMiscCharge                 PIC S9(5)V99   Optional
OrderMiscChargeAcct             PIC X(24).     Optional
OrderNo                         PIC 9(6).      Optional
OrderOrdExportedFlag            PIC X(1).      Optional
OrderPartPostFlag               PIC X(1).      Optional
OrderPayment                    PIC 9(7)V99.   Optional
OrderPaymentDisc                PIC 9(6)V99.   Optional
OrderPaymentType                PIC X(1).      Optional
OrderPhantomInv                 PIC X(1).      Optional
OrderPickTickCutoffDate         PIC 9(8).      Optional
OrderPONo                       PIC X(25).     Optional
OrderPOReqGen                   PIC X(1).      Optional
OrderPostedDate                 PIC 9(8).      Optional
OrderProfitCenter               PIC X(8).      Optional
OrderRMAStatus                  PIC X(1).      Optional
OrderSalesman1                  PIC X(3).      Optional
OrderSalesman2                  PIC X(3).      Optional
OrderSalesman3                  PIC X(3).      Optional
OrderSalesmanCommAmt1           PIC S9(7)V99   Optional
OrderSalesmanCommAmt2           PIC S9(7)V99   Optional
OrderSalesmanCommAmt3           PIC S9(7)V99   Optional
OrderSalesmanCommPct1           PIC 9(3)V99.   Optional
OrderSalesmanCommPct2           PIC 9(3)V99.   Optional
OrderSalesmanCommPct3           PIC 9(3)V99.   Optional
OrderSalesTax1                  PIC S9(6)V99   Optional
OrderSalesTax2                  PIC S9(6)V99   Optional
OrderSalesTax3                  PIC S9(6)V99   Optional
OrderSelectionCode              PIC X(1).      Optional
OrderShipInstr1                 PIC X(40).     Optional
```

```
OrderShipInstr2                 PIC X(40).     Optional
OrderShippingDate               PIC 9(8).      Optional
OrderShipToAddr1                PIC X(30).     Optional
OrderShipToAddr2                PIC X(30).     Optional
OrderShipToAddr3                PIC X(30).     Optional
OrderShipToCity                 PIC X(15).     Optional
OrderShipToCountry              PIC X(20).     Optional
OrderShipToFreeForm             PIC X(1).      Optional
OrderShipToName                 PIC X(30).     Optional
OrderShipToNo                   PIC X(4).      Optional
OrderShipToState                PIC X(2).      Optional
OrderShipToXrefNo               PIC X(17).     Optional
OrderShipToZipCode              PIC X(10).     Optional
OrderShipVia                    PIC X(2).      Optional
OrderStoreNo                    PIC X(6).      Optional
OrderTaxCode1                   PIC X(3).      Optional
OrderTaxCode2                   PIC X(3).      Optional
OrderTaxCode3                   PIC X(3).      Optional
OrderTaxPct1                    PIC 9(2)V9999. Optional
OrderTaxPct2                    PIC 9(2)V9999. Optional
OrderTaxPct3                    PIC 9(2)V9999. Optional
OrderTermsCode                  PIC X(2).      Optional
OrderTotalCost                  PIC S9(7)V99   Optional
OrderTotalSales                 PIC S9(7)V99   Optional
OrderTotalTaxableSales          PIC S9(7)V99   Optional
OrderTotalWeight                PIC S9(7)V999  Optional
OrderType                       PIC X(1).      Optional
ShipManifestDate                PIC X(160).    Optional
ShipTrackNo                     PIC X(500).    Optional
ShipVeriBoxNo                   PIC 9(3).      Optional
```

**Method:**

```
R = A.MoveHeaderCurrent(P1,P3,P4)
R = A.MoveHeaderFirst(P1,P3,P4)
R = A.MoveHeaderLast(P1,P3,P4)
R = A.MoveHeaderNext(P1,P3,P4)
R = A.MoveHeaderPrev(P1,P3,P4)
R = A.MoveLineItemCurrent(P1,P3,P4)
R = A.MoveLineItemFirst(P1,P3,P4)
R = A.MoveLineItemLast(P1,P3,P4)
R = A.MoveLineItemNext(P1,P3,P4)
R = A.MoveLineItemPrev(P1,P3,P4)
R = A.MoveLineLsCurrent(P1,P3,P4)
R = A.MoveLineLsFirst(P1,P3,P4)
R = A.MoveLineLsLast(P1,P3,P4)
R = A.MoveLineLsNext(P1,P3,P4)
R = A.MoveLineLsPrev(P1,P3,P4)
R = A.CreateHeader(P1,P2)
R = A.CreateLineItem(P1,P2)
R = A.DeleteOrder(P1,P2)
R = A.FinishOrder(P1,P2)
R = A.ChangeOrder(P1,P2)
R = A.ReapplyCredit(P1,P2)
```

**Parameters:**

```
A  : Object
P1 : PASSING-PATH           PIC X(32).            Required
P2 : PASSING-ORDER-NO       PIC X(4) COMP-5       Required
P3 : PASSING-KEY-USED       PIC X(4) COMP-5       Optional
P4 : PASSING-FILE-STATUS    PIC X(1).             Optional
R  : PASSING-RETURN-CODE    pic X(4) COMP-5       Return Code (Variable)
```

The FinishOrder method is used to assign the order status to complete or selected (I or C type of order), and indicate to other Elliott processes that this order is now available for inquiring/processing. It also calculates the total sales, freight, tax and commission amount.

The ChangeOrder method is used to change the line item qty ordered, qty to ship & qty BO information. The Order ActiveX Component only supports these three properties (LineItemQtyOrdered, LineItemQtyToShip, LineItemQtyBackOrd) now. We will support more properties in the future. In ChangeOrder method, system will automatically update item file, ATP file, line item audit trail file, and auto assign/unassign the serial number if it is a serialized item. Note: This method required a formula for checking the quantity: QtyOrdered = QtyToShip + QtyBO.

After a user specifies Change Order, the user needs to use FinishOrder to update the Order header record information.

MoveHeaderCurrent, MoveHeaderFirst, MoveHeaderLast, MoveHeaderNext, MoveHeaderPrev can help a user locate an order header record for order inquiry purposes.

MoveLineItemCurrent, MoveLineItemFirst, MoveLineItemLast, MoveLineItemNext, MoveLineItemPrev can help user locate order line item record for desired order.

MoveLineLsCurrent, MoveLineLsFirst, MoveLineLsLast, MoveLineLsNext, MoveLineLsPrev can help user locate order line serial records for a specific order line item.

Reapply Credit Method will update A/R Open Item file as well as Order Header Record. User should provide check number, check Date, payment amount, and payment type (“P” is default value).

Key Used parameter is used for MOVE method which the user needs to provide to the ActiveX Component. If Key Used = 0, the system will use the primary key to locate the record, otherwise the system will use the Alternate key to find the record.

File Status parameter is returned by ActiveX Component to indicate the current status of file. The following are the status codes:

“Y” = Record Found
“N” = Record Not Found
“E” = EOF encountered
“B” = BOF encountered

In Inquiry mode, user can get information from manifest system. Those information is store in ShipManifestDate property & ShipTrackNo property. Each of those property consist 20 elements. For ShipManifestDate, each element is a date field, and ShipTrackNo is alpha field of 25 characters long.

**Return Code:**

```
 0 = OK
 1 = File Error
 2 = Data Missing (Customer No Missing) (Create Header)
 3 = Data Missing (Order No or Item Missing) (Create Line Item)
 4 = Data Missing (Order No Missing) (Finish & Delete Order) (Reapply)
 5 = Customer Not On File (Create Header)
 6 = Ship To Not On File (Create Header)
 7 = Order Not On File (Create Line Item & Delete Order)
 8 = Item Not On File (Create Line Item)
 9 = Inv Location Not On File (Create Line Item)
10 = NSI Control File Not Found
11 = Customer PO No Duplicate
12 = Terms Code Not On File
13 = Ship Via Code Not On File
14 = Order Type Must Be "I" or "O"
15 = No Serial Number Assign, Line Item Not Created
16 = Line Item Qty To Ship Must Be 1 when Hold Trx ID is provide
17 = Hold Trx Not Found
18 = Hold Trx Item No Mismatch
19 = Fail To Apply A/R Open Item
20 = Customer No Not Provided (Order Inquiry & Customer Key)
21 = Order No Not Provided (Order Inquiry)
22 = Order Seq No Not Provided (Order Inquiry)
23 = Item No Not Provided (Order Inquiry)
24 = Picking Seq Not Provided (Order Inquiry)
25 = Serial No Not Provided (Order Inquiry)
26 = Data Missing (Order No & Seq No Missing) (Change Line Item)
27 = Order Line Item Not On File (Change Line Item)
28 = Quantity Formula Error (QtyOrdered <> QtyToShip + QtyBO)
29 = No Serial Number Assign, Change Not Effect
30 = Check No, Date, Amount, Payment Type Missing (Reapply Credit)
31 = Check No, Date, Amount, Payment Type Already Exist in the Order (Reapply)
32 = Tax Code Not Found
33 = Order Bill To Customer Not On File (Create Header)
34 = Bill To No Not Allowed in Create Header (Check Global Control Setup)

1025 = A1
1026 = A2
1027 = A1 & A2
1028 = A3
1029 = A1 & A3
1030 = A2 & A3
1031 = A1 & A2 & A3
1032 = A4
1033 = A1 & A4
1034 = A2 & A4
1035 = A1 & A2 & A4
.............................. (1036 - 1150)
1151 = A1 & A2 & A3 & A4 & A5 & A6 & A7

Formula = 1024 + A1 + A2 + A3 + A4 + A5 + A6 + A7
Yes =             1    2    4    8   16   32   64
No  =             0    0    0    0    0    0    0

Hold Status
```

```
 A1 = This Order is On Hold Based on the AR Customer File
 A2 = This Order is On Hold Based on the AR Terms Code
 A3 = This Order is On Hold For Exceeding Credit Limit
 A4 = This Order is On Hold For Account Is Past Due
 A5 = This Customer is On Hold For Terms Code Customer On Hold
 A6 = This Order & Customer is On Hold For FFL Expire
 A7 = This Order is On Hold For Handgun & Ammo Coexisted
```

Please be aware, in this ELICRORD component, when you receive a return status code that's greater than 1024, it is actually not an error. Instead, it is a status code that indicates the order is put on hold (if this feature has turned on in Elliott).

**Code Example: (VB)**

```
Create Order

Dim ReturnCode As Integer
Dim B As Double

OBJS.OrderCustomerNo = "871299"
'ObjS.OrderShipToNo = "8131"
OBJS.OrderPONo = "WEB"
OBJS.OrderShipInstr1 = "WEB Order"
OBJS.OrderShipInstr2 = "WEB Site Implementation"
OBJS.OrderComment1 = "12345678901234567890123456789012"
OBJS.OrderComment2 = "Comment 2: Davidson--------------000999999"
OBJS.OrderComment3 = "Comment 3: ActiveX"
OBJS.OrderLocation = "P"
OBJS.OrderShipVia = "FE"
OBJS.OrderType = "I"
OBJS.OrderFreight = 100
OBJS.OrderPayment = 76.25
OBJS.OrderCheckDate = Date
OBJS.OrderCheckNo = 238
OBJS.OrderPaymentType = "P"
OBJS.OrderFrtPayCode = "H"
ReturnCode = OBJS.CreateHeader("N:\MACOLARD\DATA_21\", B)
If ReturnCode = 0 Then
   'ObjS.HoldTrxID = 630
   OBJS.LineItemItemNo = "GX5040"
   OBJS.LineItemQtyOrdered = 1
   OBJS.LineItemReqDate = Date
   ReturnCode = OBJS.CreateLineItem("N:\MACOLARD\DATA_21\", B)
   If ReturnCode = 0 Then
      ReturnCode = OBJS.FinishOrder("N:\MACOLARD\DATA_21", B)
      If ReturnCode = 0 Then
         RstAttribute.Text = "Successful"
         RstFlag.Text = B
      Else
         RstAttribute.Text = "Fail - Cannot Finish Order"
         RstFlag.Text = ReturnCode
      End If
   Else
      RstAttribute.Text = "Fail - Line Item"
      RstFlag.Text = ReturnCode
```

```
   End If
Else
   RstAttribute.Text = "Fail - Header"
   RstFlag.Text = ReturnCode
End If

Change Order

Dim ReturnCode As Integer
Dim B As Double

OBJS.OrderCustomerNo = "906202"
OBJS.LineItemSeqNo = 2
OBJS.LineItemQtyOrdered = 6
'ObjS.OrderShipToNo = "8131"
ReturnCode = OBJS.ChangeOrder("N:\MACOLARD\DATA_21\", 906202)
If ReturnCode = 0 Then
   'ObjS.HoldTrxID = 630
   RstAttribute.Text = "Successful"
   RstFlag.Text = OBJS.LineItemQtyToShip
Else
   RstAttribute.Text = "Fail - Change Order"
   RstFlag.Text = ReturnCode
End If

Order Inquiry

Dim ReturnCode As Integer
Dim B As Double
Dim FILESTATUS As String

'Dim OBJS As ELIORDER
OBJS.OrderNo = "906202"
OBJS.LineItemSeqNo = 1
'ObjS.OrderShipToNo = "8131"
ReturnCode = OBJS.MoveLineItemNext("N:\MACOLARD\DATA_21\", 0, FILESTATUS)
If ReturnCode = 0 Then
   'ObjS.HoldTrxID = 630
   'Dim OBJS As ELIORDER
   If FILESTATUS = "Y" Then
      'RstAttribute.Text = Mid$(OBJS.ShipManifestDate, 1, 20)
      RstAttribute.Text = OBJS.LineItemItemNo
      RstFlag.Text = OBJS.OrderNo
   Else
      If FILESTATUS = "E" Then
         RstAttribute.Text = "EOF Encounter"
         RstFlag.Text = ReturnCode
      Else
         If FILESTATUS = "B" Then
            RstAttribute.Text = "BOF Encounter"
            RstFlag.Text = ReturnCode
         Else
            RstAttribute.Text = "File Not Found"
            RstFlag.Text = ReturnCode
         End If
      End If
   End If
```

```
End If
```

## Invoice Object (ELIINVOC.DLL)

This object handles Elliott Invoice History. ELIINVOC object contains method inquiry invoice history. In addition, user can get Starship Tracking Number information if it is available.

**Set Property**

| | | |
|---|---|---|
| InvItemInvoiceDate | PIC 9(8). | Optional |
| InvItemItemNo | PIC X(15). | Optional |
| InvItemSequenceNo | PIC 9(3). | Required |
| InvoiceCustomerNo | PIC X(6). | Optional |
| InvoiceNo | PIC 9(6). | Required |

**Get Property**

| | | |
|---|---|---|
| InvItemItemNo | PIC X(15). | Optional |
| InvItemMultFtrSerFlag | PIC X(1). | Optional |
| InvItemInvoiceDate | PIC 9(8). | Optional |
| InvItemExplodeKit | PIC X(1). | Optional |
| InvItemInvNo | PIC 9(6). | Optional |
| InvItemPOXrfSeqNo | PIC 9(3). | Optional |
| InvItemPrcLvlNo | PIC 9(2). | Optional |
| InvItemOrgBlanketSeqNo | PIC 9(3). | Optional |
| InvItemNoOfPackage | PIC 9(4). | Optional |
| InvItemOrgBlanketOrdNo | PIC 9(6). | Optional |
| InvItemCommPctOrAmt | PIC 9(5)V99. | Optional |
| InvItemControlledFlag | PIC X(1). | Optional |
| InvItemCommCalcType | PIC X(1). | Optional |
| InvItemBackorderableFlag | PIC X(1). | Optional |
| InvItemBMOrderNo | PIC 9(6). | Optional |
| InvItemDescription2 | PIC X(30). | Optional |
| InvItemDiscountPct | PIC 9(3)V99. | Optional |
| InvItemDescription1 | PIC X(30). | Optional |
| InvItemCopyToBMFlag | PIC X(1). | Optional |
| InvItemCustomerNo | PIC X(6). | Optional |
| InvItemPriceFixedFlag | PIC X(1). | Optional |
| InvItemPriceOrg | PIC S9(6)V9999 | Optional |
| InvItemProdCate | PIC X(3). | Optional |
| InvItemPromiseDate | PIC 9(8). | Optional |
| InvItemQtyBackOrdered | PIC S9(9)V999 | Optional |
| InvItemQtyOrdered | PIC S9(9)V999 | Optional |
| InvItemQtyReturnToStock | PIC S9(9)V999 | Optional |
| InvItemQtyToShip | PIC S9(9)V999 | Optional |
| InvItemReasonCode | PIC X(6). | Optional |
| InvItemRequestDate | PIC 9(8). | Optional |
| InvItemSelectCode | PIC X(1). | Optional |
| InvItemSequenceNo | PIC 9(3). | Optional |
| InvItemSerEffDate | PIC 9(8). | Optional |
| InvItemSerialLotNo | PIC X(15). | Optional |
| InvItemSerLotExpDate | PIC 9(8). | Optional |
| InvItemStockedFlag | PIC X(1). | Optional |
| InvItemStyleTempFlag | PIC X(1). | Optional |
| InvItemTaxableFlag | PIC X(1). | Optional |

```
InvItemTaxableFlag1             PIC X(1).           Optional
InvItemTaxableFlag2             PIC X(1).           Optional
InvItemTaxableFlag3             PIC X(1).           Optional
InvItemTotalQtyOrdered          PIC S9(9)V999       Optional
InvItemTotalQtyShipped          PIC S9(9)V999       Optional
InvItemUnitCost                 PIC S9(6)V9999      Optional
InvItemUnitOfMeasure            PIC X(2).           Optional
InvItemUnitPrice                PIC S9(6)V9999      Optional
InvItemUnitWeight               PIC S9(5)V999       Optional
InvItemVendorNo                 PIC X(6).           Optional
InvoiceAccumFreight             PIC S9(5)V99        Optional
InvoiceAccumMiscCharges         PIC S9(5)V99        Optional
InvoiceAccumSalesTax            PIC S9(7)V99        Optional
InvoiceAccumTotalSalesAmount    PIC S9(7)V99        Optional
InvoiceAccumTotalTaxableAmount  PIC S9(7)V99        Optional
InvoiceApplyToNo                PIC 9(6).           Optional
InvoiceARReference              PIC X(30).          Optional
InvoiceBillToAddress1           PIC X(30).          Optional
InvoiceBillToAddress2           PIC X(30).          Optional
InvoiceBillToAddress3           PIC X(30).          Optional
InvoiceBillToCity               PIC X(15).          Optional
InvoiceBillToCountry            PIC X(20).          Optional
InvoiceBillToFreeFormFlag       PIC X(1).           Optional
InvoiceBillToName               PIC X(30).          Optional
InvoiceBillToNo                 PIC X(6).           Optional
InvoiceBillToState              PIC X(2).           Optional
InvoiceBillToZipCode            PIC X(10).          Optional
InvoiceBOLPrinted               PIC X(1).           Optional
InvoiceCashAccountNo            PIC X(24).          Optional
InvoiceCheckDate                PIC 9(8).           Optional
InvoiceCheckNo                  PIC 9(6).           Optional
InvoiceCloseFlag                PIC X(1).           Optional
InvoiceComment1                 PIC X(35).          Optional
InvoiceComment2                 PIC X(35).          Optional
InvoiceComment3                 PIC X(35).          Optional
InvoiceCommissionAmount         PIC S9(6)V99        Optional
InvoiceCommissionPercent        PIC S9(2)V99        Optional
InvoiceCopyToBMFlag             PIC X(1).           Optional
InvoiceCustomerBalMethod        PIC X(1).           Optional
InvoiceCustomerNo               PIC X(6).           Optional
InvoiceDate                     PIC 9(8).           Optional
InvoiceDateBilled               PIC 9(8).           Optional
InvoiceDateEntered              PIC 9(8).           Optional
InvoiceDatePicked               PIC 9(8).           Optional
InvoiceDepartment               PIC X(8).           Optional
InvoiceDeptNo                   PIC X(6).           Optional
InvoiceDiscountPercent          PIC 9(3)V99.        Optional
InvoiceEDIExportedFlag          PIC X(1).           Optional
InvoiceEDIFlag                  PIC X(1).           Optional
InvoiceEDIShipment              PIC 9(10).          Optional
InvoiceEDIShipToFlag            PIC X(1).           Optional
InvoiceFreightAcctNo            PIC X(24).          Optional
InvoiceFreightAmount            PIC S9(5)V99        Optional
InvoiceFreightPayCode           PIC X(1).           Optional
InvoiceFullPayDate              PIC 9(8).           Optional
InvoiceJobNo                    PIC X(6).           Optional
InvoiceMfgingLocation           PIC X(2).           Optional
```

```
InvoiceMiscChargesAcctNo        PIC X(24).          Optional
InvoiceMiscChargesAmount        PIC S9(5)V99        Optional
InvoiceNo                       PIC 9(6).           Optional
InvoiceOrderDate                PIC 9(8).           Optional
InvoiceOrderNo                  PIC 9(6).           Optional
InvoiceOrdExportedFlag          PIC X(1).           Optional
InvoiceOriginalInvoiceNo        PIC X(6).           Optional
InvoicePartPostedFlag           PIC X(1).           Optional
InvoicePaymentAmount            PIC 9(7)V99.        Optional
InvoicePaymentDiscAmount        PIC 9(6)V99.        Optional
InvoicePaymentType              PIC X(1).           Optional
InvoicePhantomInvFlag           PIC X(1).           Optional
InvoicePickTickCutoffDate       PIC 9(8).           Optional
InvoicePONo                     PIC X(25).          Optional
InvoicePOReqGenFlag             PIC X(1).           Optional
InvoicePostedDate               PIC 9(8).           Optional
InvoiceProfitCenter             PIC X(8).           Optional
InvoiceRMAStatus                PIC X(1).           Optional
InvoiceSalesman1                PIC X(3).           Optional
InvoiceSalesman2                PIC X(3).           Optional
InvoiceSalesman3                PIC X(3).           Optional
InvoiceSalesmanCommAmt1         PIC S9(7)V99        Optional
InvoiceSalesmanCommAmt2         PIC S9(7)V99        Optional
InvoiceSalesmanCommAmt3         PIC S9(7)V99        Optional
InvoiceSalesmanPctComm1         PIC 9(3)V99.        Optional
InvoiceSalesmanPctComm2         PIC 9(3)V99.        Optional
InvoiceSalesmanPctComm3         PIC 9(3)V99.        Optional
InvoiceSalesTaxAmount1          PIC S9(6)V99        Optional
InvoiceSalesTaxAmount2          PIC S9(6)V99        Optional
InvoiceSalesTaxAmount3          PIC S9(6)V99        Optional
InvoiceSelectionCode            PIC X(1).           Optional
InvoiceShipInstr1               PIC X(40).          Optional
InvoiceShipInstr2               PIC X(40).          Optional
InvoiceShippingDate             PIC 9(8).           Optional
InvoiceShipToAddress1           PIC X(30).          Optional
InvoiceShipToAddress2           PIC X(30).          Optional
InvoiceShipToAddress3           PIC X(30).          Optional
InvoiceShipToCity               PIC X(15).          Optional
InvoiceShipToCountry            PIC X(20).          Optional
InvoiceShipToFreeFormFlag       PIC X(1).           Optional
InvoiceShipToName               PIC X(30).          Optional
InvoiceShipToNo                 PIC X(4).           Optional
InvoiceShipToState              PIC X(2).           Optional
InvoiceShipToXREFNo             PIC X(17).          Optional
InvoiceShipToZipCode            PIC X(10).          Optional
InvoiceShipViaCode              PIC X(2).           Optional
InvoiceStoreNo                  PIC X(6).           Optional
InvoiceTaxCode1                 PIC X(3).           Optional
InvoiceTaxCode2                 PIC X(3).           Optional
InvoiceTaxCode3                 PIC X(3).           Optional
InvoiceTaxPercent1              PIC 9(2)V9999.      Optional
InvoiceTaxPercent2              PIC 9(2)V9999.      Optional
InvoiceTaxPercent3              PIC 9(2)V9999.      Optional
InvoiceTermsCode                PIC X(2).           Optional
InvoiceTotalCost                PIC S9(7)V99        Optional
InvoiceTotalSaleAmount          PIC S9(7)V99        Optional
InvoiceTotalTaxableAmount       PIC S9(7)V99        Optional
```

```
InvoiceTotalWeight              PIC S9(7)V99          Optional
InvoiceType                     PIC X(1).             Optional
ShipManifestDate                PIC X(160).           Optional
ShipTrackNo                     PIC X(500).           Optional
ShipVeriBoxNo                   PIC 9(3).             Optional
```

In Inquiry mode, user can get information from manifest system. That information is stored in ShipManifestDate property & ShipTrackNo property. Each of the properties consists of 20 elements. For ShipManifestDate, each element is a date field, and ShipTrackNo is alpha field of 25 characters long.

**Method:**

```
R = A.ReadHeader(P1,P2)
R = A.ReadInvItem(P1,P2)
```

**Parameters:**

```
A  : Object
P1 : PASSING-PATH           PIC X(32).            Required
P2 : PASSING-FILE-STATUS    PIC X(1).             Optional
R  : PASSING-RETURN-CODE    pic X(4) COMP-5       Return Code (Variable)
```

ReadHeader method can help a user locate an invoice header record for invoice inquiry purposes.

ReadInvItem method can help user locate invoice line item record for desired invoice.
File Status parameter is returned by ActiveX Component to indicate the current status of file. The following are the status codes:

"Y" = Record Found
"N" = Record Not Found
"E" = EOF encountered
"B" = BOF encountered

**Return Code:**

```
  0 = OK
  1 = File Error
  2 = Order No Not Provided (Invoice Inquiry)
  3 = Customer No Not Provided (Invoice Inquiry & Customer Key)
  4 = Order Seq No Not Provided (Invoice Line Item Inquiry)
  5 = Item No Not Provided (Invoice Inquiry & Item Key)
  6 = Customer No or Item No Not Provided (Invoice Inquiry)
```

**Code Example: (VB)**

```
Dim ReturnCode As Integer
Dim B As Double
Dim FILESTATUS As String

'Dim OBJV As ELIINVOC
OBJV.InvoiceNo = 16011
ReturnCode = OBJV.ReadHeader("M:\MACOLA\DATA_90\", FILESTATUS)
If ReturnCode = 0 Then
   If FILESTATUS = "Y" Then
```

```
      RstAttribute.Text = CDate(OBJV.InvoiceDateBilled)
      RstFlag.Text = OBJV.InvoicePONo
   Else
      If FILESTATUS = "E" Then
         RstAttribute.Text = "EOF Encounter"
         RstFlag.Text = ReturnCode
      Else
         If FILESTATUS = "B" Then
            RstAttribute.Text = "BOF Encounter"
            RstFlag.Text = ReturnCode
         Else
            RstAttribute.Text = "File Not Found"
            RstFlag.Text = FILESTATUS
         End If
      End If
   End If
Else
   RstFlag.Text = ReturnCode
   RstAttribute.Text = "Failure"
End If

'Dim OBJV As ELIINVOC
OBJV.InvoiceNo = 16011
OBJV.InvItemSequenceNo = 2
ReturnCode = OBJV.ReadInvItem("M:\MACOLA\DATA_90\", FILESTATUS)
If ReturnCode = 0 Then
   If FILESTATUS = "Y" Then
      RstAttribute.Text = OBJV.InvItemItemNo
      RstFlag.Text = OBJV.InvItemQtyOrdered
   Else
      If FILESTATUS = "E" Then
         RstAttribute.Text = "EOF Encounter"
         RstFlag.Text = ReturnCode
      Else
         If FILESTATUS = "B" Then
            RstAttribute.Text = "BOF Encounter"
            RstFlag.Text = ReturnCode
         Else
            RstAttribute.Text = "File Not Found"
            RstFlag.Text = FILESTATUS
         End If
      End If
   End If
Else
   RstFlag.Text = ReturnCode
   RstAttribute.Text = "Failure"
End If
```

## eContact Object (ELIECONT.DLL)

This object handles Elliott eContact object.  ELIECONT object contains method to add, change and delete Elliott eContact. You can also use to add a contact relationship, or delete contact relationship.  You can also use it for view contact information and its relationship & role. Before adding a new contact, it is a good idea to check whether this contact exist or not by using the view contact method (viewcontact).  If contact does exist, you should not add the

contact again, otherwise, you will get an error code. If contact does exist, the method will also return the relationship (up to 20) back for this contact. If both contact and relationship exist, you should not add this contact. If contact exist, but that the relationship, then you should add the relationship only (addrelationship method).

**Add Contact or Add Relationship**

Add Contact (+Relationship) or add relationship only. When you use AddContact method to add a contact, it will add both the contact and relationship for you in one step. Therefore, not only you should specify the contact information, you should also specify the relationship information (DetailFile, ReferenceID, Role).

You will use AddRelationship method if the contact exists already. You can find this out by using the ViewContact method first. When add relationship, the role information is optional. Role information will be used to determine the security of this contact.

Whether a user has the security clearance to add or delete of contact or relationship will not be enforced by this ActiveX component, it is up to the application program to perform the security control.

If users need to add Contact, then proper information should be provided, like email address (EmailAddress property), (Contact ID will be automatically assigned by ActiveX and therefore do not needed when added a new contact), name (ContactName property) , …etc. System allows users to use primary relationship's address, phone and fax (LinkAddressFlag, LinkPhoneFlag, LinkFaxFlag properties) and as a result, users do not need to pass the address, phone and fax information. Same principle applied to credit card address (CreditCartdSameAddr property). Most of the other properties are not necessary when adding a contact and system will assume a default value. However, if you do set the property, system will honor the value you set.

**Change Contact**

You can't change a relationship. Relationship can only be added or deleted. However, you can change a contact by read the contact record first (ViewContact method) and all property for a contact shall return. From there, you can change certain property of that contact, then use the ChangeContact method. You can, for example, change the address, phone, fax, password or credit card of the contact.

Certain change will not make sense. For example, if users try to change address without changing the LinkAddressFlag to "N", then this change obviously does not make sense. A status code will return back to user.

When LinkAddressFlag is set to "Y", it means the address is linked to the primary relationship of this contact. As the primary relationship record is change, the contact address will be updated automatically.

**Delete Contact or Delete Relationship**

When you need to use DeleteContact method, make sure to set the property for EmailAddrss or ContactID, system will then delete the contact and all relationship for this contact. This, however, should not be the common way to use delete. Use this method with extreme caution.

When you try to delete a contact, most of the time, you really meant to delete the relationship. Therefore, you should look at the DeleteRelationship method. To use this method, please set the property for DetailFile (e.g. ARCUSFIL), reference ID (e.g. Customer number 000100) and contact ID (or e-mail address, ActiveX will be able to delete the relationship easier if you provide the concta id since it does not have to go through the process of finding the contact id from e-mail address.)

Using DeleteRelationship method, system will only delete the relationship record. If there's no other relationship for this contact, then the contact record will be deleted as well. If the deleted relationship are primary relationship, and secondary relationship still exist, one of the secondary relationship will be promoted as primary relationship.

**View Contact**

When use ViewContact method, please set the property of EmailAddress, or ContactID. This method will determine whether a contact record exists or not. If it exists, all Contact properties are returned. If not, an error code is returned. Also, ActiveX will return up to 20 relationship in the relationship structure (ContactRelationship Property). Each relationship structure is 63 bytes and contains the relationship information, role and security information. The first relationship is always the primary relationship.

View Contact will also return the description of each relationship if user set IncludeDescFlag to "Y". The description of relationship will store in RelationshipDesc property (array of 20, each element contains 30 bytes).

The structure of ContactRelationship will an array of 20. Each element contains the following information:

```
FileName          PIC X(8).
ReferenceID       PIC X(30).
Role              PIC X(15).
PrimaryRecINQ     PIX X(1).
      This flag determine whether this contact has the security clearance to
      inquire the primary record data.  Primary record can be, for example,
      ARCUSFIL or APVENFIL…etc.
PrimaryRecCHG     PIC X(1).
      This flag determine whether this contact has the security clearance to
      change the primary record data.
SurordINQ         PIC X(1).
      Security for Inquiry Subordinate Record.  Subordinate record depend on the
      primary record.  For example, If ARCUSFIL,000100 is the primary record,
      then subordinate records refer to Ship-To for Customer 000100 (i.e. Can
      this contact see the ship-to record of the customer that he/she has
      relationship to).  If ARSLMFIL,100 is the primary Record, then subordinate
      Record refer to Customer belong to Salesman 100 and the associated Ship-To
      records.
SurordADD         PIC X(1).
      Security for Add Subordinate Record.
SurordCHG         PIC X(1).
      Security for Change Subordinate Record.
SurordDEL         PIC X(1).
      Security for Delete Subordinate Record.
TrxINQ            PIC X(1).
```

```
      Security for Inquiry Trx Record.  Trx Record depends on the Primary
      Record.  If Primary Record is ARCUSFIL,000100, then this security will
      allow the contact to look at the sales order for Customer 0001000.
TrxADD            PIC X(1).
      Security for Add Trx Record (i.e. add sales order).
TrxCHG            PIC X(1).
      Security for Change Trx Record.
TrxDEL            PIC X(1).
      Security for Delete Trx Record.
```

The primary relationship will store in the first element.

If there are more than 20 relationships, a return code will returned back to indicate that there's an overflow to the relationship array.

During the return of relationship, we provide an option to exclude/include transaction file's relationship (sales order and purchase order). For example, users can specify return relationship for "ARCUSFIL" only. Another example, if we know this should be a salesperson function, we can ask to exclude relationship in "CPORDHDR". System will let user specify (1) Exclude/Include (ExcludeInclude property) (2) An array of 80 bytes for 10 files to include or exclude based on (1) (FileView property). If this information is not specified, then all relationships are returned.

**View Relationship**

Even though VeiwContact method will return both the contact and relationship information, it can only return up to 20 relationships for a contact. Sometimes, you may have situations that a contact has more than 20 relationships.  In that case, ViewContact will return back an error code.  If you need a way to consistently check whether a relationship exist or not, you should use the **ViewRelationship** method.

To use ViewRelationship method, you will need to set DetailFile, ReferenceId, ContactId (or EmailAddress).  System will return back the Contact Information. In the ContactRelationship structure, only the specific relationship will be returned at the first element.

**View All Contact**

**View All Contact** is a method that allow you to find all contacts within a particular contact relationship. To use ViewAllContact method, you will need to set DetailFile, ReferenceId. System will return back a contact table (Up to 20 elements). User can retrieve the contact information by using property **AllContact**. The first element of the table will be the primary contact record.

The structure of AllContact will an array of 20. Each element contains the following information:

```
ID          PIC 9(9).
Name        PIC X(30).
Position    PIC X(15).
E-mail      PIX X(60).
```

**Set Property**

```
AcceptHTMLFlag                      PIC X(1).       Optional
```

```
BlockAccess                         PIC X(1).       Optional
BlockAccessDate                     PIC 9(8).       Optional
BlockAccessReason                   PIC X(30).      Optional
BlockEmail                          PIC X(1).       Optional
BlockEmailDate                      PIC 9(8).       Optional
ContactAddress1                     PIC X(30).      Optional
ContactAddress2                     PIC X(30).      Optional
ContactCity                         PIC X(15).      Optional
ContactCountry                      PIC X(20).      Optional
ContactFax                          PIC X(12).      Optional
ContactID                           PIC 9(09).      Required
ContactName                         PIC X(30).      Required
ContactPassword                     PIC X(12).      Optional
ContactPasswordHint                 PIC X(20).      Optional
ContactPasswordHintValue            PIC X(20).      Optional
ContactPhone                        PIC X(12).      Optional
ContactPhoneExt                     PIC X(5).       Optional
ContactState                        PIC X(2).       Optional
ContactZip                          PIC X(10).      Optional
CreateDate                          PIC 9(8).       Optional
CreateTime                          PIC 9(6).       Optional
CreditCardAddress1                  PIC X(30).      Optional
CreditCardAddress2                  PIC X(30).      Optional
CreditCardCity                      PIC X(15).      Optional
CreditCardCountry                   PIC X(20).      Optional
CreditCardExpMo                     PIC 9(2).       Optional
CreditCardExpYr                     PIC 9(2).       Optional
CreditCardName                      PIC X(30).      Optional
CreditCardNumber                    PIC X(20).      Optional
CreditCardSameAddr                  PIC X(1).       Optional
CreditCardState                     PIC X(2).       Optional
CreditCardType                      PIC X(4).       Optional
CreditCardZip                       PIC X(10).      Optional
DetailFile                          PIC X(8).       Required
EmailAddress                        PIC X(60)       Required
ExcludeInclude                      PIC X(1).       Optional
FileView                            PIC X(80).      Optional
HomePhone                           PIC X(12).      Optional
IncludeDescFlag                     PIC X(1).       Optional
LinkAddressFlag                     PIC X(1).       Optional
LinkFaxFlag                         PIC X(1).       Optional
LinkPhoneFlag                       PIC X(1).       Optional
MobilePhone                         PIC X(12).      Optional
NoOfBouncebacks                     PIC 9(4).       Optional
PrimaryRelFlag                      PIC X(1).       Optional
ReferenceID                         PIC X(30).      Required
Role                                PIC X(15).      Optional
Position                            PIC X(15).      Optional
PrimaryContactFlag                  PIC X(1).       Optional
```

**Get Property**

```
AcceptHTMLFlag                      PIC X(1).
BlockAccess                         PIC X(1).
BlockAccessDate                     PIC 9(8).
BlockAccessReason                   PIC X(30).
BlockEmail                          PIC X(1).
```

```
BlockEmailDate                      PIC 9(8).
ContactAddress1                     PIC X(30).
ContactAddress2                     PIC X(30).
ContactCity                         PIC X(15).
ContactCountry                      PIC X(20).
ContactFax                          PIC X(12).
ContactID                           PIC 9(09).
ContactName                         PIC X(30).
ContactPassword                     PIC X(12).
ContactPasswordHint                 PIC X(20).
ContactPasswordHintValue            PIC X(20)>
ContactPhone                        PIC X(12).
ContactPhoneExt                     PIC X(5).
ContactRelationship                 PIC X(1260).
ContactState                        PIC X(2).
ContactZip                          PIC X(10).
CreateDate                          PIC 9(8).
CreateTime                          PIC 9(6).
CreditCardAddress1                  PIC X(30).
CreditCardAddress2                  PIC X(30).
CreditCardCity                      PIC X(15).
CreditCardCountry                   PIC X(20).
CreditCardExpMo                     PIC 9(2).
CreditCardExpYr                     PIC 9(2).
CreditCardName                      PIC X(30).
CreditCardNumber                    PIC X(20).
CreditCardSameAddr                  PIC X(1).
CreditCardState                     PIC X(2).
CreditCardType                      PIC X(4).
CreditCardZip                       PIC X(10).
EmailAddress                        PIC X(60)
HomePhone                           PIC X(12).
LinkAddressFlag                     PIC X(1).
LinkFaxFlag                         PIC X(1).
LinkPhoneFlag                       PIC X(1).
MobilePhone                         PIC X(12).
NoOfBouncebacks                     PIC 9(4).
RelationshipDesc                    PIC X(600).
AllContact                          PIC X(2280).
```

**Method:**

```
R = A.AddContact(P1)
      EmailAddress, ContactName, DetailFile and ReferenceID are required.  The
      other properties are optional and when not set, system will assume a
      default value.  When provided, system will honor the value you set in
      other properties.
R = A.AddRelationship(P1)
      DetailFile, ReferenceID, ContactID(or EmailAddress) are required.  Role is
      optional.
R = A.ChangeContact(P1)
      ContactId (or EmailAddress) is required.
R = A.DeleteContact(P1)
      ContactId (or EmailAddress) is required.
R = A.DeleteRelationship(P1)
      DetailFile, ReferenceId, ContactID(or EmailAddress) are required.
R = ViewContact(P1)
```

```
      ContactId or EmailAddress are required.
R = ViewRelationship(P1)
      DetailFile, ReferenceId, ContactID (or EmailAddress) are required.
R = ViewAllContact(P1)
      DetailFile, ReferenceId are required.
```

**Parameters:**

```
A  : Object
P1 : PASSING-PATH           PIC X(32).            Required
R  : PASSING-RETURN-CODE    pic X(4) COMP-5       Return Code (Variable)
```

**Return Code:**

```
  0 = OK
  1 = File Error
  2 = E-mail Address Missing (Add Contact)
  3 = Detail File & Ref No Missing (Add Contact/Relationship)
      (Delete Relationship)
  4 = E-mail Address or Contact ID Missing (Add Relationship)
      (Delete Contact/Relationship) (Delete Relationship)
      (View Contact)
  5 = Contact Record Not Found (Add Relationship/Change Contact)
      (Delete Contact/Relationship) (Delete Relationship)
      (View Contact)
  6 = Contact Record Already Existed (Add Contact)
  7 = Contact Role Not On File (Add Contact & Relationship)
  8 = NSI Control File Not Found (Add Contact)
  9 = Contact Name is Required (Add/Change Contact)
 10 = Link Address Flag Invalid (Add/Change Contact)
 11 = Link Phone Flag Invalid (Add/Change Contact)
 12 = Link Fax Flag Invalid (Add/Change Contact)
 13 = Block Access Flag Invalid (Add/Change Contact)
 14 = Block E-mail Flag Invalid (Add/Change Contact)
 15 = Credit Card Exp Year/Month Required (Add/Change Contact)
 16 = Credit Card No/Exp Invalid (Add/Change Contact)
 17 = Credit Card Name Required (Add/Change Contact)
 18 = Credit Card Same Address Flag Invalid (Add/Change Contact)
 19 = Accept HTML Flag Invalid (Add/Change Contact)
 20 = Contact ID Not Provided (Change Contact)
 21 = Use Primary Address Flag Turn On, But Address Info. Change
      (Change Contact)
 22 = Use Primary Phone Flag Turn On, But Phone Change.
      (Change Contact)
 23 = Use Primary Fax Flag Turn On, But Fax Change.
      (Change Contact)
 24 = Use same credit address, but address info changed (Change)
      (Change Contact)
 25 = No relationship found (View Contact)
 26 = More than 20 relationships found (View Contact)
 27 = Invalid Exclude/Include Value (Spaces,"E","I")
      (View Contact)
```

Please be aware, you will get a run-time error if the following files are missing:

SYCONREL.BTR

```
SYCONROL.BTR
SYCONTCT.BTR
```

**Code Example: (VB)**

```
Add Contact

OBJT.EmailAddress = "vic@netcellent.com"
OBJT.ContactName = "Roger Sueng"
OBJT.LinkAddressFlag = "Y"
OBJT.LinkPhoneFlag = "Y"
OBJT.AcceptHTMLFlag = "Y"
OBJT.DetailFile = "ARCUSFIL"
OBJT.ReferenceID = "000100"
ReturnCode = OBJT.AddContact("N:\MACOLAO\DATA_96\")
If ReturnCode = 0 Then
   RstAttribute.Text = "Successful"
   RstFlag.Text = OBJT.ContactAddress1
Else
   RstAttribute.Text = "Fail - Contact"
   RstFlag.Text = ReturnCode
End If

View Contact

Dim ReturnCode As Integer
Dim B As Double

OBJT.ContactID = 2
ReturnCode = OBJT.ViewContact("N:\MACOLAO\DATA_96\")
If ReturnCode = 0 Then
   RstAttribute.Text = "Successful"
   RstFlag.Text = OBJT.ContactAddress1
Else
   RstAttribute.Text = "Fail - Contact"
   RstFlag.Text = ReturnCode
End If
```

## Event Handling (ELIEVPRC.DLL)

This ActiveX component is used for the new feature "Event Handling" in Elliott V7.0. Event Handling allows the system to trigger E-mail, or other action when specific events occur. For example, a customer may wish to know when an e-Business site receives the out of stock merchandize. When requesting an Event Notification for a item, when the system receives the merchandize, a E-mail is sent to the customer automatically to notify the merchandize has arrived.

You can use this ActiveX component to subscribe to an event, cancel an event subscription, or extend an event subscription. The template is a very helpful when using events to help create an event without too much effort.

**Set Property**

```
ActionID                PIC 9(9).           Required (Cancel/Extend/Detail)
DetailFile              PIC X(8).           Required
ReferenceID             PIC X(30).          Required
Program                 PIC X(8).           Required
```

```
Type                    PIC X(8).          Required
Owner                   PIC X(10).         Optional
CreateDate              PIC 9(8).          Optional
CreateTime              PIC 9(6).          Optional
Source                  PIC X(10).         Optional (Recommend)
ActionType              PIC X(1).          Optional (Recommend)
RecurringType           PIC X(1).          Optional (Recommend)
NotifySubject           PIC X(80).         Optional
TriggerUser             PIC X(1).          Optional
NotifyFrom              PIC X(60).         Optional
ExpirationDate          PIC 9(8).          Optional
ExpirationTime          PIC 9(6).          Optional
CobolProgram            PIC X(8).          Optional
AddDaysWhenExt          PIC 9(4).          Required
DayToAlert              PIC 9(4).          Required
NextAlertDate           PIC 9(8).          Optional
TklNoteType             PIC X(06).         Optional
TklFolder               PIC X(10).         Optional
DetailType              PIC X(2).          Required
DetailSeqNo             PIC 9(4).          Optional
DetailLength            PIC 9(02).         Optional
DetailText              PIC X(80).         Optional
TemplateID              PIC 9(9).          Optional
AlertTemplateID         PIC 9(9).          Optional
ExpireTemplateID        PIC 9(9).          Optional
CobolParameters         PIC X(80).         Optional
ConsolidateFlag         PIC X(1).          Optional
FilterNo                PIC 9(2).          Required
FilterOper              PIC X(2).          Required
FilterRule              PIC X(80).         Required
RplName                 PIC X(14).         Required
RplValue                PIC X(60).         Required
```

**Get Property**

```
DetailFileX             PIC X(8).
ReferenceIDX            PIC X(30).
EmailAddress            PIC X(80).
ExpDate                 PIC 9(8).
```

**Method:**

```
R = A.CreateEvent(P1,P2)
R = A.CancelEvent(P1)
R = A.CreateEventDetail(P1)
R = A.ExtendEvent(P1)
R = A.ViewEvent(P1)
R = A.ObjectRef(A)
R = A.NewFilter(P1)
R = A.TemplateVar(P1)
```

**Parameters:**

```
A  : Object
P1 : PASSING-PATH                  PIC X(32).     Required
P2 : PASSING-ACTION-ID             PIC 9(9).      Required (C/E/D)
R  : PASSING-RETURN-CODE           PIC 9(3).      Return Code (Variable)
```

**Return Code:**

```
 0 = OK (Create/Create Datail/Cancel/Extend)
 1 = File Error (Create/Create Detail/Cancel/Extend/View)
 2 = No Action ID (Create Detail/Cancel/Extend/View)
 3 = Data Missing (File, Reference ID, Program, Type) (Create)
 4 = Invalid Action Type (Create)
 5 = Invalid Recurring Type (Create)
 6 = Invalid Trigger User (Create)
 7 = Add Days To Extended Missing (Create)
 8 = Days To Alert Missing (Create)
 9 = Expiration Date Mismatch (Create)
10 = Invalid COBOL Program (Create)
11 = Next Alert Date Mismatch (Create)
12 = No Need To Provide Tickler Information
13 = Detail Type Missing (Create Detail)
14 = No Event Record Exist (Cancel/Extend/View)
15 = Only E-mail have template ID
16 = Only E-mail & Day To Alert > 0 have Alert template ID
17 = Invalid Filter No Range (1-10)
18 = Invalid Filter Operator (EQ, NE, GT, GE, LT, LE ,BT)
19 = Only E-mail & Notify & Purge Have Expire template ID
```

**Code Example: (VB)**

```
'DIM A AS ELIEVPRC
Set A = CreateObject("ELIEVPRC")
R1 = A.ObjectRef(A)
A.DetailFile = "IMITMFIL"
A.ReferenceID = "CLOCK"
A.Program = "PO2001"
A.Type = ""
A.Owner = "SUPERVISOR"
A.Source = ""
A.ActionType = "E"
A.RecurringType = "O"
A.NotifySubject = "Notify"
A.AddDaysWhenExt = 30
A.DayToAlert = 10
A.TemplateID = 2
R = A.CreateEvent("F:\MACOLARD\DATA_96\",B)
A.ActionID = B
A.DetailType = "TX"
A.DetailSeqNo = 0
A.DetailLength = 80
A.DetailText = "This is a testing"
R = A.CreateEventDetail("F:\MACOLARD\DATA_96\")
A.ActionID = B
R = A.ExtendEvent("F:\MACOLARD\DATA_96\")
R = A.CancelEvent("F:\MACOLARD\DATA_96\")

R = A.ViewEvent("F:\MACOLARD\DATA_96\")
Z = A.DetailFileX
X = A.ReferenceIDX
Z = A.EmailAddress
E = A.ExpDate
```

```
A.FilterNo = 1
A.FilterOper = "NE"
A.FilterRule = "David"
R = A.NewFilter("F:\MACOLARD\DATA_21")

A.RplName = "Clayton"
A.RplValue = "Chienwu"
R = A.TemplateVar("F:\MACOLARD\DATA_21")
```

## Credit Card Cash Receipt (ELIAUTDP.DLL)

This ActiveX component is used for a new feature in Elliott V7.0 - Credit Card Cash Receipt process. It allow the received credit card payment to update the following Elliott area: (1) Customer File account balance, last cash receipt; (2) A/R Open Item File; (3) A/R Distribution to G/L File; (4) Credit Card Log file.

Source table should be setup in Elliott first to indicate the proper G/L cash account for Debiting. The A/R account always come from the customer file. Amount1 stand for the Credit Card Amount. Amount2 stand for any adjustment amount. For example, if you receive a $100 credit card amount and you charge 3% of using credit card. Then the Amount1 will be $100 and the Amount2 will be -3.

**Set Property**

```
CustomeNo               PIC X(6).               Required
TransactionDate         PIC 9(8).               Optional
TransactionTime         PIC 9(6).               Optional
ApplyTo                 PIC 9(6).               Optional
Amount1                 PIC S9(7)V99.           Required
Amount2                 PIC S9(7)V99.           Optional
CreditType              PIC X(4).               Optional
CerditNumber            PIC X(19).              Required
ExpYear                 PIC 9(4).               Required
ExpMonth                PIC 9(2).               Required
CardHolder              PIC X(30).              Optional
Address1                PIC X(30).              Optional
Address2                PIC X(30).              Optional
City                    PIC X(15).              Optional
State                   PIC X(2).               Optional
ZipCode                 PIC X(10).              Optional
Country                 PIC X(20).              Optional
ApprovalNo              PIC X(10).              Required
ReferenceNo             PIC X(10).              Optional
ReferenceText           PIC X(60).              Optional
Source                  PIC X(6).               Required
```

**Get Property**

```
CheckNo             pic 9(6).
```

**Method:**

```
R = A.AutoDeposit(P1)
```

**Parameters:**

```
P1 : PASSING-PATH                   PIC X(32).     Required
R  : PASSING-RETURN-CODE            PIC 9(3).      Return Code (Variable)
```

**Return Code:**

```
  0 = OK
  1 = File Error
  2 = Data Missing (No Customer No)
  3 = Data Missing (No Amount [AMOUNT-1])
  4 = Data Missing (No Credit Card Number)
  5 = Data Missing (No Expiration Year & Month)
  6 = Credit Card Type Mismatch
  7 = Data Missing (No Approval Number)
  8 = Credit Card Type invalid
  9 = Incorrect Credit Card Length
 10 = Expiration Year/Month invalid
 11 = This Credit Card Has Already Expired
 12 = Credit Card Number Invalid
 13 = Customer Record Locked, Please try again.
 14 = NSI Control File Not Found
 15 = Source Not On File
```

**Code Example: (VB)**

```
'DIM A AS ELIAUTDP
Set A = CreateObject("ELIAUTDP")
A.CustomeNo     = "000100"
A.Amount1       = 1000
A.CerditNumber  = "4223222222223333"
A.ExpYear       = 2001
A.ExpMonth      = 6
A.ApprovalNo    = "1123"
A.Source        = "WEB"
R = A.AutoDeposit("F:\MACOLARD\DATA_96\")
```

## Available To Promise Object (ELIATPOB.DLL)

This object handles Elliott ATP Object. ELIATPOB object contains method to inquiry Elliott ATP. The ATPInquiry method is very similar to Elliott ATP Inquiry Function. User will provide the item no (ItemNo property), location (Location Property, Optional), and Source (Source Property, Optional) for the inquiry. System will return all information (trx date, reference number, reference description, type, order number, location and quantity) of ATP in a table. The size of the table is controllable which user can specify in TalbeSize property. The maximum size of the table is 30.

The structure of the table is defined below:

```
ATP-TRX-DATE                PIC 99/99/99.
ATP-REF-NO                  PIC X(6).
ATP-REF-INFO                PIC X(30).
```

```
ATP-TYPE                    PIC X(1).
ATP-ORDER-NO                PIC X(9).
ATP-LOCATION                PIC X(2).
ATP-QTY                     PIC -999999999.999.
```

It is possible that all ATP trx for that item will exceed the size of the table. In that case, system will return the key for next trx in PrimaryKey property & AlternateKey property. User can provide this two information in next ATPInquiry method to retrieve next page of trx in the table.

It is also possible that all ATP trx for that item is less than the size of the table. User need to check TableCount Property to found the actual size of the table.

**Set Property**

```
ItemNo              PIC X(15).          Required
Location            PIC X(2).           Optional
       Location, Blank means all locations
Source              PIC X(1).           Optional
       Source, Blank means all source. Valid value "S" (Sales Orde), "B" (BOMP), "P" (Purchase
       Order) and "F" (Shop Floor Control)
PrimaryKey          PIC X(31).          Optional
AlternateKey        PIC X(23).          Optional
TableSize           PIC 9(2).           Optional
       User have the capability to manage the size of the table. It is up to 30 elements in the
       table.
```

**Get Property**

```
PrimaryKey          PIC X(31).          Optional
AlternateKey        PIC X(23).          Optional
ItemUnitPrice       PIC S9(6)V9999.     Optional
QuantityOH          PIC S9(9)V999.      Optional
QuantityAlloc       PIC S9(9)V999.      Optional
QuantityOnOrder     PIC S9(9)V999.      Optional
ATPTable            PIC X(2100).        Optional
TableCount          PIC 9(2).           Optional
```

**Method:**

```
R = A.ATPInquiry(P1)
```

**Parameters:**

```
A  : Object
P1 : PASSING-PATH           PIC X(32).          Required
R  : PASSING-RETURN-CODE    pic 9(3).           Return Code (Variable)
```

**Return Code:**

```
  0 = OK
  1 = File Error
```

```
  2 = Item No Missing
  3 = Table Size Must < 30
  4 = Item Not On File
  5 = Locatoin Not On File
  6 = Source (Space, S, B, P, F) Invalid
```

**Code Example: (VB)**

```
Set A = CreateObject("ELIATPOB")

A.ItemNo = "10/22-RB"
R = A.ATPInquiry("M:\MACOLA\DATA_21\")
If R = 0 Then
  IUP = A.ItemUnitPrice
  QOH = A.QuantityOH
  QAL = A.QuantityAlloc
  QOO = A.QuantityOnOrder
  For Idx = 1 to A.TableCount
     ATPTrxDate  = Cdate(MID$(A.ATPTable, (Idx - 1) * 70 +  1,  8))
     ATPRefNo    = MID$(A.ATPTable, (Idx - 1) * 70 +  9,  6)
     ATPRefDesc  = MID$(A.ATPTable, (Idx - 1) * 70 + 15, 30)
     ATPType     = MID$(A.ATPTable, (Idx - 1) * 70 + 45,  1)
     ATPOrderNo  = MID$(A.ATPTable, (Idx - 1) * 70 + 46,  9)
     ATPLocation = MID$(A.ATPTable, (Idx - 1) * 70 + 55,  2)
     ATPQty      = CDBL(MID$(A.ATPTable, (Idx - 1) * 70 + 57, 14))
  Next Idx
End If
Set A = Nothing
```

## Order Inquiry Object (ELIORDIQ.DLL)

This object handles Elliott Order Inquiry Object. ELIORDIQ object contains method to inquiry Elliott Order. It provides three ways to inquiry orders: Order Inquiry by Customer No, Order Inquiry By Item No & Order Inquiry by Order No. For Order Inquiry By CustomerNo method (OrderInquiryCustomer method), user need to provide customer no (CustomerNo property) to retrieve a list of order (OrderTable property). For Order Inquiry by ItemNo method (OrderInquiryItem method), user needs to provide item number to get a list of order (LineItemTable property). For Order Inquiry by OrderNo method (OrderInquiryOrderNo method), user need to give a specify order no to retrieve all line item (LineItemTable property) associated with that order.

The size of the table is controllable which user can specify in TalbeSize property. The maximum size of the table is 30.

You can also use ELIORDIQ object to retetrive Invoice History information. It is pretty simple by only set OrderInvoiceFlag property to "2" and you can inquiry invoice history instead of order information.

Sorting sequence is also controllable. You can inquiry order/invoice by ascending or decending sequence by setting SortingSeq property to either "A" or "D".

You can also exclude posted order by setting ExcludePostFlag to "Y". It is useful tool that will increase the performance of order inquiry.

The structure of the order table is defined below:

```
ORDER-DATE-ENTERED          PIC 99/99/99.     (001-008) (8)
ORDER-DATE                  PIC 99/99/99.     (009-016) (8)
ORDER-INVOICE-DATE          PIC 99/99/99.     (017-024) (8)
ORDER-SHIPPING-DATE         PIC 99/99/99.     (025-032) (8)
ORDER-TYPE                  PIC X(1).         (033-033) (1)
ORDER-STATUS                PIC X(1).         (034-034) (1)
ORDER-HOLD-STATUS           PIC X(1).         (035-035) (1)
ORDER-NO                    PIC 9(6).         (036-041) (6)
ORDER-INVOICE-NO            PIC 9(6).         (042-047) (6)
ORDER-PURCHASE-NO           PIC X(25).        (048-072) (25)
ORDER-SHIP-TO               PIC X(4).         (073-076) (4)
ORDER-SHIP-VIA              PIC X(2).         (077-078) (2)
ORDER-TERMS-CODE            PIC X(2).         (079-080) (2)
ORDER-SALESMAN-NO           PIC X(3).         (081-083) (3)
ORDER-LOCATION              PIC X(2).         (084-085) (2)
ORDER-JOB-NO                PIC X(6).         (086-091) (6)
ORDER-DATE-PICKED           PIC 99/99/99.     (092-099) (8)
ORDER-CUSTOMER-NO           PIC X(6).         (100-105) (6)
ORDER-BILL-TO-NAME          PIC X(30).        (106-135) (30)
ORDER-BILL-TO-CITY          PIC X(15).        (136-150) (15)
ORDER-BILL-TO-STATE         PIC X(2).         (151-152) (2)
ORDER-BILL-TO-ZIP-CODE      PIC X(10).        (153-162) (10)
ORDER-SHIP-TO-NAME          PIC X(30).        (163-192) (30)
ORDER-SHIP-TO-CITY          PIC X(15).        (193-207) (15)
ORDER-SHIP-TO-STATE         PIC X(2).         (208-209) (2)
ORDER-SHIP-TO-ZIP-CODE      PIC X(10).        (210-219) (10)
ORDER-TOTAL-SALE-AMOUNT     PIC -9999999.99.  (220-230) (11)
ORDER-TOTAL-WEIGHT          PIC -9999999.999. (231-242) (12)
ORDER-MISC-CHARGES-AMOUNT   PIC -99999.99.    (243-251) (9)
ORDER-FREIGHT-AMOUNT        PIC -99999.99.    (252-260) (9)
ORDER-SALES-TAX-AMOUNT      PIC -999999.99.   (261-270) (10)
ORDER-COMMISSION-PERCENT    PIC -99.99.       (271-276) (6)
ORDER-COMMISSION-AMOUNT     PIC -999999.99.   (277-286) (10)
ORDER-DATE-BILLED           PIC 99/99/99.     (287-294) (8)
ORDER-POSTED-DATE           PIC 99/99/99.     (295-302) (8)
ORDER-PART-POSTED-FLAG      PIC X(1).         (303-303) (1)
ORDER-EDI-FLAG              PIC X(1).         (304-304) (1)
ORDER-STORE-NO              PIC X(6).         (305-310) (6)
ORDER-BILL-TO-NO            PIC X(6).         (311-316) (6)
ORDER-RMA-STATUS            PIC X(1).         (317-317) (1)
ORDER-PHANTOM-INV-FLAG      PIC X(1).         (318-318) (1)
ORDER-DEPT-NO               PIC X(6).         (319-324) (6)
ORDER-BOL-PRINTED           PIC X(1).         (325-325) (1)
ORDER-EDI-EXPORTED-FLAG     PIC X(1).         (326-326) (1)
ORDER-SHIP-ACK-SENT         PIC 99/99/99.     (327-334) (8)
ORDER-PICK-CUTOFF-DATE      PIC 99/99/99.     (335-342) (8)
ORDER-ORD-EXPORTED-FLAG     PIC X(1).         (343-343) (1)
ORDER-SHIP-TO-XREF-NO       PIC X(17).        (344-360) (17)
```

```
ORDER-856-EXPORTED-FLAG     PIC X(1).          (361-361) (1)
ORDER-ORD-ACK-SENT          PIC X(1).          (362-362) (1)
ORDER-RESERVED              PIC X(138).        (363-500) (138)
```

It is possible that all order for that customer will exceed the size of the table. In that case, system will return the key for next trx in NextHeaderKey property & NextHeaderAltKey property. User can provide this two information in next OrderInquiryCustomer method to retrieve next page of orders in the table.

The structure of the line item table is defined below:

```
LINE-ITEM-ORDER-NO          PIC 9(6).            (001-006) (6)
LINE-ITEM-SEQ-NO            PIC 9(3).            (007-009) (3)
LINE-ITEM-CUSTOMER-NO       PIC X(6).            (010-015) (6)
LINE-ITEM-ITEM-NO           PIC X(15).           (016-030) (15)
LINE-ITEM-DESC-1            PIC X(30).           (031-060) (30)
LINE-ITEM-DESC-2            PIC X(30).           (061-090) (30)
LINE-ITEM-QTY-ORDERED       PIC -999999999.999.  (091-104) (14)
LINE-ITEM-QTY-TO-SHIP       PIC -999999999.999.  (105-118) (14)
LINE-ITEM-QTY-BO            PIC -999999999.999.  (119-132) (14)
LINE-ITEM-UNIT-PRICE        PIC -999999.9999.    (133-144) (12)
LINE-ITEM-DISC-PCT          PIC -999.99.         (145-151) (7)
LINE-ITEM-REQ-DATE          PIC 99/99/99.        (152-159) (8)
LINE-ITEM-PRM-DATE          PIC 99/99/99.        (160-167) (8)
LINE-ITEM-PICKING-SEQ       PIC X(8).            (168-175) (8)
LINE-ITEM-UOM               PIC X(2).            (176-177) (2)
LINE-ITEM-SERIAL-LOT-NO     PIC X(15).           (178-192) (15)
LINE-ITEM-QTY-RTN-TO-STOCK  PIC -999999999.999.  (193-206) (14)
LINE-ITEM-BACKORDERABLE     PIC X(1).            (207-207) (1)
LINE-ITEM-UNIT-WEIGHT       PIC -99999.999.      (208-217) (10)
LINE-ITEM-COMM-CALC-TYPE    PIC X(1).            (218-218) (1)
LINE-ITEM-COMM-PCT-OR-AMT   PIC -99999.99.       (219-227) (9)
LINE-ITEM-TAXABLE-FLAG      PIC X(1).            (228-228) (1)
LINE-ITEM-STOCKED-FLAG      PIC X(1).            (229-229) (1)
LINE-ITEM-CONTROLLED-FLAG   PIC X(1).            (230-230) (1)
LINE-ITEM-SELECT-CODE       PIC X(1).            (231-231) (1)
LINE-ITEM-TOTAL-QTY-SHIPPED PIC -999999999.999.  (232-245) (14)
LINE-ITEM-BM-ORDER-NO       PIC 9(6).            (246-251) (6)
LINE-ITEM-PO-XRF-SEQ-NO     PIC 9(3).            (252-254) (3)
LINE-ITEM-PROD-CATE         PIC X(3).            (255-257) (3)
LINE-ITEM-REASON-CODE       PIC X(6).            (258-263) (6)
LINE-ITEM-VENDOR-NO         PIC X(6).            (264-269) (6)
LINE-ITEM-ORG-ITEM-NO       PIC X(15).           (270-284) (15)
LINE-ITEM-SERIAL-FLAG       PIC X(1).            (285-285) (1)
LINE-ITEM-RESERVED          PIC X(115).          (286-400) (115)
```

It is possible that all line item for that customer/order will exceed the size of the table. In that case, system will return the key for next trx in NextLineKey property & NextLineAltKey (or NextInvAltKey, for invoice inquiry) property. User can provide this two information in next OrderInquiryItem/OrderInquiryOrderNo method to retrieve next page of orders in the table.

It is also possible that all order/line item for that customer/item/order is less than the size of the table. User need to check TableCount Property to found the actual size of the table.

**Set Property**

```
CustomerNo          PIC X(6).             Required (OrderInquiryCustomer)
ItemNo              PIC X(15).            Required (OrderInquiryItem)
OrderNo             PIC 9(6).             Required (OrderInquiryOrderNo)
OrderInvoiceFlag    PIC X(1).             Optional
      "I" for Invoice History. "O" for Order.
SortingSeq          PIC X(1).             Optional
      "A" : Ascending Sequence. "D": Descending Sequence.
ExcludePostFlag     PIC X(1).             Optional
      "Y" : Exclude all posted order/line item. (For Invoice History Inquiry,
      this flag is no use). "N" : All Order/Line Item.
NextHeaderKey       PIC X(6).             Optional
NextHeaderAltKey    PIC X(12).            Optional
NextLineKey         PIC X(9).             Optional
NextLineAltKey      PIC X(21).            Optional
NextInvAltKey       PIC X(23).            Optional
TableSize           PIC 9(2).             Optional
      Users have the capability to manage the size of the table. There can be up
      to 30 elements in the table.
```

**Get Property**

```
NextHeaderKey       PIC X(6).             Optional
NextHeaderAltKey    PIC X(12).            Optional
NextLineKey         PIC X(9).             Optional
NextLineAltKey      PIC X(21).            Optional
NextInvAltKey       PIC X(23).            Optional
OrderTable          PIC X(3000).          Optional
LineItemTable       PIC X(6000).          Optional
TableCount          PIC 9(2).             Optional
```

**Method:**

```
R = A.OrderInquiryCustomer(P1)
R = A.OrderInquiryItem(P1)
R = A.OrderInquiryOrderNo(P1)
```

**Parameters:**

```
A  : Object
P1 : PASSING-PATH           PIC X(32).          Required
R  : PASSING-RETURN-CODE    pic 9(3).           Return Code (Variable)
```

**Return Code:**

```
  0 = OK
  1 = File Error
  2 = Missing Customer No (Order Inquiry By Customer)
  3 = Missing Item No (Order Inquiry By Item)
  4 = Missing Order No (Order Inquiry By Order)
  5 = Table Size Must < 30 (All Method)
  6 = Sorting Sequence Must Be "A" (Ascending) or "D" (Decending)
```

**Code Example: (VB)**

```
Set A = CreateObject("ELIORDIQ")

A.CustomerNo = "871299"
R = A.OrderInquiryCustomer("M:\MACOLA\DATA_21\")
If R = 0 Then
  For Idx = 1 to A.TableCount
     OrderDateEntered  = Cdate(MID$(A.OrderTable, (Idx - 1) * 500 +  1,  8))
     OrderDate         = Cdate(MID$(A.OrderTable, (Idx - 1) * 500 +  9,  8))
     OrderInvoiceDate  = Cdate(MID$(A.OrderTable, (Idx - 1) * 500 + 17,  8))
     OrderShipDate     = Cdate(MID$(A.OrdreTable, (Idx - 1) * 500 + 25,  8))
     OrderType         = MID$(A.OrderTable, (Idx - 1) * 500 + 33,  1)
     OrderStatus       = MID$(A.OrderTable, (Idx - 1) * 500 + 34,  1)
     OrderHoldStatus   = MID$(A.OrderTable, (Idx - 1) * 500 + 35,  1)
     OrderNo           = Cint(MID$(A.OrderTable, (Idx - 1) * 500 +  36,  6))
     OrderInvoiceNo    = Cint(MID$(A.OrderTable, (Idx - 1) * 500 +  42,  6))
     OrderPONo         = MID$(A.OrderTable, (Idx - 1) * 500 + 48, 25)
     OrderShipTo       = MID$(A.OrdreTable, (Idx - 1) * 500 + 73,  4)
     OrderShipVia      = MID$(A.OrderTable, (Idx - 1) * 500 + 77,  2)
     OrderTermsCode    = MID$(A.OrderTable, (Idx - 1) * 500 + 79,  2)
     OrderSalesman     = MID$(A.OrderTable, (Idx - 1) * 500 + 81,  3)
     OrderLocation     = MID$(A.OrderTable, (Idx - 1) * 500 + 84,  2)
     OrderJobNo        = MID$(A.OrderTable, (Idx - 1) * 500 + 86,  6)
  Next Idx
End If
Set A = Nothing
```

## Item Inquiry Object (ELIITMIQ.DLL)

This object handles Elliott Item Inquiry Object. ELIITMIQ object contains method to inquiry Elliott Item Master File.  It will provides two way to inquiry the item: Read Item (List of item or list of stock of different item for single location) and Read Item all location (list of stock for single item for multiple location).

**ReadItem Method**

ReadItem method will list corresponding information of item in *ItemTable property* when user specify the item number information in *ItemNoTable property*. Property. If user specifies one particular location (*Location Property*) , the table will only contain the item information for that location. It the location information is missing, the ItemTable will only get the information from item file. User also need to provide the NoOfItem Property to tell the system how many item are store in the ItemNoTable.

**ReadItemAllLocation Method**

ReadItemAllLocation method will list all location information for one single item. The item information will store in the first element of ItemNoTable. Location property is irrelevant. System will Return NoOfItem and ItemTable to inform the user how many item in the table and the correspond item & inventory location record list in the table. The first element of ItemTalbe will be always from item file, and the rest of table will come from inventory location file.

The Following is the structure of ItemNoTable:

ItemNoTable will consisted of 100 elements with each element of 15 character long (Item No).

The Following is the structure of ItemTable (consisted of 100 element with each element of 1165 character long)

```
ERROR-STATUS                  PIC X(2).
ITEM-INDEX-ITEM-NO            PIC X(15).
ITEM-INDEX-ITEM-DESC          PIC X(30).
ITEM-DESCRIPTION-2            PIC X(30).
ITEM-PRODUCT-CATEGORY         PIC X(3).
ITEM-USER-DEFINED-CODE        PIC X(2).
ITEM-MFGING-LOCATION          PIC X(2).
ITEM-QUANTITY-ON-HAND         PIC -999999999.999.
ITEM-QUANTITY-ALLOCATED       PIC -999999999.999.
ITEM-QTY-BO                   PIC -999999999.999.
ITEM-QTY-ON-ORDER-NO-SIGN     PIC 999999999.999.
ITEM-ORDER-UP-TO-LEVEL        PIC 999999999.999.
ITEM-REORDER-LEVEL            PIC 999999999.999.
ITEM-AVERAGE-COST             PIC -999999.9999.
ITEM-PRICE                    PIC -999999.9999.
ITEM-PRICE-UOM                PIC X(2).
ITEM-PRICE-RATIO              PIC 9999.999.
ITEM-WEIGHT                   PIC 99999.999.
ITEM-UNIT-OF-MEASURE          PIC X(2).
ITEM-PICKING-SEQUENCE         PIC X(8).
ITEM-BACKORDERABLE-FLAG       PIC X(1).
ITEM-TAXABLE-FLAG             PIC X(1).
ITEM-END-ITEM-CODE            PIC X(1).
ITEM-STARTING-SALE-DATE       PIC 99/99/99.
ITEM-ENDING-SALE-DATE         PIC 99/99/99.
ITEM-SALE-PRICE               PIC -999999.9999.
ITEM-PRICES-APPLY-FLAG        PIC X(1).
ITEM-DISCS-APPLY-FLAG         PIC X(1).
ITEM-USAGE-PTD                PIC -999999999.999.
ITEM-USAGE-YTD                PIC -999999999.999.
ITEM-PRIOR-YEAR-USAGE         PIC -999999999.999.
ITEM-PRIOR-YEAR-SALES         PIC -999999999.999.
ITEM-QTY-RETURNED-YTD         PIC -999999999.999.
ITEM-QTY-SOLD-PTD             PIC -999999999.999.
ITEM-QTY-SOLD-YTD             PIC -999999999.999.
ITEM-SALES-PTD                PIC -999999999.99.
ITEM-SALES-YTD                PIC -999999999.99.
ITEM-COST-PTD                 PIC -999999999.99.
ITEM-COST-YTD                 PIC -999999999.99.
ITEM-RECOM-MINIMUM-ORDER      PIC 999999999.999.
ITEM-ECONOMIC-ORDER-QTY       PIC 999999999.999.
ITEM-AVERAGE-USAGE            PIC 999999999.999.
ITEM-USAGE-WGHT-FACTOR        PIC 9.99.
ITEM-SAFETY-STOCK             PIC 999999999.999.
ITEM-SAFETY-FACTOR            PIC 9.9.
ITEM-AVERAGE-FRCST-ERROR      PIC 999999999.999.
ITEM-SUM-OF-ERRORS            PIC -999999999.999.
ITEM-USAGE-FILTER             PIC 9.9.
ITEM-LEAD-TIME                PIC 9(3).
ITEM-MAT-TYPE                 PIC X(1).
ITEM-SUBSTITUTE               PIC X(15).
```

```
ITEM-LAST-COST                PIC -999999.9999.
ITEM-STD-COST                 PIC -999999.9999.
ITEM-PRIME-VENDOR-NO          PIC X(6).
ITEM-ORDER-MINIMUM            PIC 999999999.999.
ITEM-ORDER-MULTIPLE           PIC 9(4).
ITEM-TARGET-MARGIN            PIC 9(3).
ITEM-PUR-UNIT-OF-MEASURE      PIC X(2).
ITEM-FILLER                   PIC X(4).
ITEM-DATE-LAST-SOLD           PIC 99/99/99.
ITEM-QTY-LAST-SOLD            PIC -999999999.999.
ITEM-P-AND-IC-CD              PIC X(3).
ITEM-ACTIVITY-CODE            PIC X(1).
ITEM-STOCKED-FLAG             PIC X(1).
ITEM-CONTROLLED-FLAG          PIC X(1).
ITEM-PUR-OR-MFG-CODE          PIC X(1).
ITEM-MS-ITEM-FLAG             PIC X(1).
ITEM-INVENTORY-CLASS          PIC X(1).
ITEM-CYCLE-COUNT-CODE         PIC X(1).
ITEM-DATE-LAST-COUNTED        PIC 99/99/99.
ITEM-COMMODITY-CODE           PIC X(4).
ITEM-BUYER-OR-ANALYST         PIC X(2).
ITEM-DRAWING-RELEASE-NO       PIC X(6).
ITEM-DRAWING-REVISION-NO      PIC X(2).
ITEM-ROUTING-RELEASE-NO       PIC X(6).
ITEM-ROUTING-REVISION-NO      PIC X(2).
ITEM-ROUTING-NO               PIC X(5).
ITEM-ORDER-POLICY-CODE        PIC X(1).
ITEM-PLANNING-PERIOD          PIC 9(3).
ITEM-PLANNING-LEAD-TIME       PIC 9(3).
ITEM-PLANNING-ORDER-MULT      PIC 9(4).
ITEM-MRP-TIME-FENCE-SHOP-DAYS PIC 9(3).
ITEM-MRP-MULT-LOC-QTY-OH      PIC -999999999.999.
ITEM-STOCK-STATUS-CODE        PIC X(1).
ITEM-LOW-LEVEL-CODE           PIC 9(2).
ITEM-ACTIVE-ORDERS            PIC 9(5).
ITEM-CUTOFF-QTY               PIC -999999999.999.
ITEM-PCT-ERR-LAST-COUNT       PIC -99.99.
ITEM-COMMISSION-PCT-OR-AMT    PIC 99999.99.
ITEM-CALC-COMMISSION-TYPE     PIC X(1).
ITEM-SERIAL-LOT-FLAG          PIC X(1).
ITEM-SERIAL-WARRANTY-DAYS     PIC 9(3).
ITEM-PUR-TO-INV-RATIO         PIC 9999.999.
ITEM-DESC-SEARCH              PIC X(30).
ITEM-QUANTITY-ON-ORDER        PIC -999999999.999.
ITEM-NOTE-1                   PIC X(30).
ITEM-NOTE-2                   PIC X(30).
ITEM-NOTE-3                   PIC X(30).
ITEM-NOTE-4                   PIC X(30).
ITEM-NOTE-5                   PIC X(30).
ITEM-USER-DATE                PIC 99/99/99.
ITEM-USER-AMOUNT              PIC -9999999.99.
ITEM-LANDED-COST-FACTOR       PIC 99.999999.
OP10-FILLER                   PIC X(92).
ITEM-CAD-DRAWING-NAME         PIC X(8).
ITEM-FRZ-QTY-ON-HAND          PIC -999999999.999.
ITEM-FRZ-COST                 PIC -999999.9999.
ITEM-KIT-PRICE-ROLLUP         PIC X(1).
```

```
ITEM-FEATURE-PRICE-OPTION     PIC X(1).
ITEM-LAND-FIX-COST            PIC -99999.9999.
ITEM-VOLUME                   PIC 999999.9999.
ITEM-DUTY-PERCENT             PIC -9999.9999.
ITEM-AVG-COST-2               PIC -999999.9999.
ITEM-STD-COST-2               PIC -999999.9999.
ITEM-LAST-COST-2              PIC -999999.9999.
ITEM-USER-LAST-ACCESS         PIC X(10).
ITEM-DATE-LAST-ACCESS         PIC 99/99/99.
ITEM-STYLE-CODE               PIC X(15).
FILLER                        PIC X(5).
```

Note: ErrorCode in the table will indicate the error status of each item, the error code system provide will the following:

"00" : OK
"01" : Item File Not Found
"02" : Item File & Inventory Location Not Found

**Set Property**

```
ItemNoTabl          PIC X(1500)          Required
Location            PIC X(2)             Optional
NoOfItem            PIC 9(4)             Required (ReadItem)
```

**Get Property**

```
NoOfitem            PIC 9(4)             Required (ReadItemAllLocation)
ItemTable           PIC X(116500)        Required
```

**Method:**

```
R = A.ReadItem(P1)
R = A.ReadItemAllLocation(P1)
```

**Parameters:**

```
A  : Object
P1 : PASSING-PATH           PIC X(32).           Required
R  : PASSING-RETURN-CODE    pic 9(3).            Return Code (Variable)
```

**Return Code:**

```
  0 = OK
  1 = File Error
  2 = Missing Item No (Read Item with multiple location)
  3 = No of Item Must Provide (Read Item)
  4 = Item Not Found
```

**Code Example: (VB)**

```
Set A = CreateObject("ELIITMIQ")


Dim ReturnCode As Integer
```

```
Dim B As Double

OBJY.ItemNoTable = "1004-9959-0601 1004-9959-0602 1005-1416-0932"
OBJY.NoOfItem = 3
OBJY.Location = "LA"
ReturnCode = OBJY.ReadItem("M:\MACOLA\DATA\")
If ReturnCode = 0 Then
   If Mid$(OBJY.ItemTable, 1, 2) = "00" Then
      RstAttribute.Text = CDbl(Mid$(OBJY.ItemTable, 85, 14))
      RstFlag.Text = CDbl(Mid$(OBJY.ItemTable, 99, 14))
   Else
      RstFlag.Text = Mid$(OBJY.ItemTable, 1, 2)
      RstAttribute.Text = "Failure"
   End If
Else
   RstFlag.Text = ReturnCode
   RstAttribute.Text = "Failure"
End If

OBJY.ItemNoTable = "1005-1416-0932"
ReturnCode = OBJY.ReadItemAllLocation("M:\MACOLA\DATA\")
If ReturnCode = 0 Then
   If Mid$(OBJY.ItemTable, 1, 2) = "00" Then
      RstAttribute.Text = CDbl(Mid$(OBJY.ItemTable, 85, 14))
      RstFlag.Text = CDbl(Mid$(OBJY.ItemTable, 99, 14))
   Else
      RstFlag.Text = Mid$(OBJY.ItemTable, 1, 2)
      RstAttribute.Text = "Failure"
   End If
Else
   RstFlag.Text = ReturnCode
   RstAttribute.Text = "Failure"
End If
```

## Get Restricted Item (ELIRSTIM.DLL)

Restricted Item function is a new feature in Elliott V7 under the Attribute feature. The purpose of this feature is to determine whether an item can be sold to a local area; or whether an item can be sold to a customer based on their license or setup arrangement.

Once restricted item function is properly setup on Elliott side, you can use this ActiveX component to check whether there's a problem selling this item to a customer.

**Set Property**

```
ItemNo              pic x(15).          Required
UserDefinedCode     pic x(2).           Optional
ProductCategory     pic x(3).           Optional
CustomerNo          pic x(6).           Optional
CustomerType        pic x(5).           Optional
ShipToNo            pic x(4).           Optional
City                pic x(15).          Optional (Recommend)
State               pic x(2).           Optional (Recommend)
ZipCode             pic x(10).          Optional (Recommend)
```

```
Country             pic x(10).            Optional (Recommend)
```

**Method:**

```
R = A.FindConsumerRestrictedItem(P1,P2,P3)
R = A.FindDealerRestrictedItem(P1,P2,P3)
R = A.ObjectRef(A)
```

**Parameters:**

```
A  : Object
P1 : PASSING-PATH           PIC X(32).            Required
P2 : PASSING-RST-ATTRIB     PIC X(15).            Return Attribute (Variable)
P3 : PASSING-RST-FLAG       PIC X(1).             Return Attribute (Variable)
R  : PASSING-RETURN-CODE    pic 9(3).             Return Code (Variable)
```

**Return Code:**

```
 0 = OK
 1 = File Error
 2 = Item No Not Provided
```

**Code Example: (VB)**

```
'DIM A AS ELIRSTIM
Set A = CreateObject("ELIRSTIM")
R1 = A.ObjectRef(A)
A.Item = "CLOCK"
A.City = "WALNUT"
A.State = "CA"
A.ZipCode = "91766"
A.Country = "USA"
R = A.FindConsumerRestrictedItem("F:MACOLARD\DATA_96\",B,C)
```

## Customer Object (ELICUSTO.DLL)

This object is used to add or view a customer record in Elliott. Feature for Change or Delete will be provided in the future. When a customer is added, if a certain property are not set, it will get its default value from customer number "&&&&&&".

**Set Property**

```
CustomerNo              PIC X(6).             Require (View)
CustomerName            PIC X(30).            Require (Add)
CorrespondName          PIC X(30).            Optional
Address1                PIC X(30).            Optional
Address2                PIC X(30).            Optional
City                    PIC X(15).            Optional
State                   PIC X(2).             Optional
ZipCode                 PIC X(10).            Optional
Country                 PIC X(10).            Optional
Contact                 PIC X(20).            Optional
PhoneNumber             PIC X(12).            Optional
FaxNumber               PIC X(12).            Optional
SalesmanNo              PIC X(3).             Optional
```

```
CustomerType            PIC X(5).           Optional
BalanceMethod           PIC X(1).           Optional
StmntFrequency          PIC X(1).           Optional
OrderLocation           PIC X(2).           Optional
ShipViaCode             PIC X(2).           Optional
TermsCode               PIC X(2).           Optional
Comment1                PIC X(30).          Optional
Comment2                PIC X(30).          Optional
Note1                   PIC X(30).          Optional
Note2                   PIC X(30).          Optional
Note3                   PIC X(30).          Optional
Note4                   PIC X(30).          Optional
Note5                   PIC X(30).          Optional
UserDate                PIC 9(8).           Optional
UserAmount              PIC S9(7)V99.       Optional
CreditLimit             PIC 9(9).           Optional
CreditRating            PIC X(4).           Optional
CreditHoldFlag          PIC X(1).           Optional
AcctCollector           PIC X(3).           Optional
FinChgFlag              PIC X(1).           Optional
Territory               PIC X(2).           Optional
ARAccountNo             PIC X(24).          Optional
UPSZone                 PIC X(2).           Optional
DiscountPct             PIC 9(3)V99.        Optional
TaxableFlag             PIC X(1).           Optional
TaxCode1                PIC X(3).           Optional
TaxCode2                PIC X(3).           Optional
TaxCode3                PIC X(3).           Optional
ExemptNo                PIC X(15).          Optional
AllowSubItems           PIC X(1).           Optional
AllowBO                 PIC X(1).           Optional
AllowPartShip           PIC X(1).           Optional
PrintDunnFlag           PIC X(1).           Optional
SlsmnStartDate          PIC 9(8).           Optional
```

**Get Property**

```
CustomerName          PIC X(30).
CorrespondName        PIC X(30).
Address1              PIC X(30).
Address2              PIC X(30).
City                  PIC X(15).
State                 PIC X(2).
ZipCode               PIC X(10).
Country               PIC X(10).
Contact               PIC X(20).
PhoneNumber           PIC X(12).
FaxNumber             PIC X(12).
SalesmanNo            PIC X(3).
CustomerType          PIC X(5).
BalanceMethod         PIC X(1).
StmntFrequency        PIC X(1).
OrderLocation         PIC X(2).
ShipViaCode           PIC X(2).
TermsCode             PIC X(2).
CustStartDate         PIC 9(8).
Balance               PIC S9(9)V99.
```

```
LastSaleDate          PIC 9(8).
LastSaleAmt           PIC S9(9)V99.
CreditHoldFlag        PIC X(1).
```

**Method:**

```
R = A.AddCustomer(P1,P2)
R = A.ViewCustomer(P1)
```

**Parameters:**

```
A  : Object
P1 : PASSING-PATH                  PIC X(32).     Required
P2 : PASSING-CUSTOMER-NO           PIC X(6).      Required
R  : PASSING-RETURN-CODE           PIC 9(3).      Return Code (Variable)
```

**Return Code:**

```
  0 = OK
  1 = File Error
  2 = Data Missing (Customer No Missing) (Inquiry)
  3 = NSI Control File Not Found
  4 = Customer Not On File (Inquiry)
  5 = No Default Customer '&&&&&&'
  6 = Data Missing (Customer Name Missing) (Add)
  7 = Salesman Not On File
  8 = Customer Type Not On File
  9 = Balance Method Must Be "B" or "O"
 10 = Statement Frequency Must Be "N", "D", "W", "M", "S", "Q"
 11 = Ship Via Code Not On File
 12 = Terms Code Not On File
 13 = Credit Hold Flag Must Be "Y" or "N"
 14 = Fin Chg Flag Must Be "Y" or "N"
 15 = A/R Account No Not On File
 16 = Taxable Flag Must Be "Y" or "N"
 17 = Allow Substitute Item Flag Must Be "Y" or "N"
 18 = Backorderable Flag Must Be "Y" or "N"
 19 = Partial Shipment Flag Must Be "Y" or "N"
 20 = Print Dunning Letter Flag Must Be "Y" or "N"
 21 = Tax Code Not On File
 22 = Duplication For Tax Code 1 & 2 & 3
```

**Code Example: (VB)**

```
'DIM A AS ELICCUST
Set A = CreateObject("ELICCUST")
A.CustomerName = "Clayton Chen"
A.Address1 = "3096 Temple Avenue"
A.City = "Pomona"
A.State = "CA"
A.ZipCode = "91766"
A.PhoneNumber = "909-622-5009"
A.Contact = "Chienwu Chen"
R = A.AddCustomer("M:\MACOLARD\DATA_96\",B)

A.CustomerNo = "000100"
R = A.ViewCustomer("M:\MACOLARD\DATA_96\")
```

```
X = A.CustAddress1
```

## Calculate Price (ELIGETPC.DLL)

This component will calculate a price based on Elliott pricing table. The CalcPrice is before any customer discount percent while CalcNetPrice is after the customer discount percent applied. If an item is on sales and the system setup does not allow for additional discount, then the CalcNetPrice will reflect the correct final price. For the reason, it is usually a good choice to use CalcNetPrice as the price to quote.

**Set Property**

```
CustomerNo          PIC X(6).
ItemNo              PIC X(15).
ProductCategory     PIC X(3).
CustomerType        PIC X(5).
Quantity            PIC S9(9)V999.
PriceDate           PIC 9(8).
UOM                 PIC X(2).
OrderDisc           PIC 9(3)V99.
OnSaleFlag          PIC X(1).
```

**Get Property**

```
CalcPrice           PIC S9(6)V9999.
CustDisc            PIC S9(3)V99.
CalcNetPrice        PIC S9(6)V9999.
```

**Method:**

```
R = A.CalculatePrice(P1)
```

**Parameters:**

```
A  : Object
P1 : PASSING-PATH           PIC X(32).          Required
R  : PASSING-RETURN-CODE    pic 9(3).           Return Code (Variable)
```

**Return Code:**

```
 0 = OK
 1 = File Error
 2 = Price By Whole Flag Is Turn On
 3 = Missing Date (Customer No or Item No)
 4 = Item Not On File
 5 = Customer Not On File
```

**Code Example: (VB)**

```
'DIM A AS ELIGETPC
Set A = CreateObject("ELIGETPC")
A.CustomerNo = "000100"
A.Item = "CLOCK"
A.Quantity = 1
```

```
R = A.CalculatePrice("F:\MACOLA\DATA_96\")
S = A.CalcPrice
```

## Get Item Information (ELIGITEM.DLL)

This component will return some basic item data including quantity information. Depend on the location specify, the quantity information may come from item file or inventory location file.

**Set Property**

```
ItemNo          X(15).                      Required
Location        X(2).                       Required
```

**Get Property**

```
QuantityOH              PIC S9(9)V999.
QuantityAlloc           PIC S9(9)V999.
QuantityOnOrd           PIC S9(9)V999.
QuantityBO              PIC S9(9)V999.
Desc1                   PIC X(30).
Desc2                   PIC X(30).
StartingSaleDate        PIC 9(8).
EndingSaleDate          PIC 9(8).
Price                   PIC S9(6)V9999.
SalesPrice              PIC S9(6)V9999.
ProductCategory         PIC X(3).
UserDefinedCode         PIC X(2).
SalesYTD                PIC S9(9)V99.
```

**Method:**

```
R = A.ItemData(P1)
```

**Parameters:**

```
P1 : PASSING-PATH                   PIC X(32).     Required
R  : PASSING-RETURN-CODE            PIC 9(3).      Return Code (Variable)
```

**Return Code:**

```
 0 = OK
 1 = File Error
 2 = Data Missing (Item No or Loc Missing)
 3 = Item Not On File
 4 = Inv Location Not On File
```

**Code Example: (VB)**

```
'DIM A AS ELIGITEM
Set A = CreateObject("ELIGITEM")
A.ItemNo = "1005-1416-0932"
A.Location = "LA"
R = A.ItemData("F:\MACOLARD\DATA_96\")
Qty = A.QuantityOH
```

## Hold/Cancel Inventory (ELIHDTRX.DLL)

Hold Inventory is a new feature in Elliott V7.0 that allow inventory to be set aside for a particular customer/salesman without entering a sales order. With this component, you can reserve inventory in Elliott. You can also cancel your reservation. You may also extend the reservation period when it getting close to expire.

**Set Property**

```
ItemNo              pic x(15).          Required
Location            pic x(2).           Required
SerialNo            pic x(15).          Optional (Sometimes Required)
UserID              pic x(10).          Required
Quantity            pic 9(9)V999        Required
Description         pic x(30)           Optional
EMailAddress        pic x(60).          Optional
NoOfDays            pic 9(3).           Required
```

**Get Property**

```
ItemNo              pic x(15).
Location            pic x(2).
SerialNo            pic x(15).
UserID              pic x(10).
Quantity            pic 9(9)V999
Description         pic x(30)
EMailAddress        pic x(60).
```

**Method:**

```
R = A.HoldInventory(P1,P2)
R = A.CancelInventory(P1,P2)
R = A.ExtInventory(P1,P2,P10)
R = A.ViewInventory(P1,P2)
R = A.ObjectRef(A)
```

**Parameters:**

```
A  : Object
P1 : PASSING-PATH           PIC X(32).          Required
P2 : PASSING-TRX-ID         PIC 9(9).           Required
P10: PASSING-NO-OF-DAYS     PIC 9(3).           Required
R  : PASSING-RETURN-CODE    pic 9(3).           Return Code (Variable)
```

**Return Code:**

```
 0 = OK (Hold & Cancel & Extend)
 1 = File Error (Hold & Cancel & Extend)
 2 = Data Missing (Item No, Location, User ID) (Hold)
 3 = Item Not On File (Hold)
 4 = Inventory Location Not Found (Hold)
 5 = No Trx ID (Cancel & Extend)
 6 = No Hold Trx (Cancel)
```

```
 7 = Serialized Item Quantity Must Be 1
 8 = Non-Serialized Item Serial Number Must Be Blank
 9 = No Serial Number For Serialized Item
10 = No Extend No of Days (Extend)
```

**Code Example: (VB)**

```
'DIM A AS ELIHDTRX
Set A = CreateObject("ELIHDTRX")
R1 = A.ObjectRef(A)
A.Item = "CLOCK"
A.Location = "LA"
A.UserID = "1000"
A.Quantity = 10
A.NoOfDays = 15
R = A.HoldInventory("F:\MACOLARD\DATA_96\",B)
R = A.CancelInventory("F:\MACOLARD\DATA_96",B)
R = A.ExtInventory("F:\MACOLARD\DATA_96",B,10)
R = A.ViewInventory("F:\MACOLARD\DATA_96",B)
S = A.ItemNo
T = A.Location
U = A.UserID
V = A.Quantity
W = A.EmailAddress
```