KEYREF

This complex data type identifies a key in a table and the fields in this key. This gives you access to the key and the fields it contains. The keyref object can refer to any key in any table in the database.

ACTIVE
Ok := KeyRef.ACTIVE
Use this function to find out if the key is enabled or not.

FIELDCOUNT
No := KeyRef.FIELDCOUNT
Use this function to return the number of fields that have been defined in a key. These functions returns an error if no key is selected.

FIELDINDEX
FieldRef := KeyRef.FIELDINDEX(Index)
Use this function to return the fieldref of the field that has this index in the key referred to by the keyref variable.

RECORD
RecordRef := KeyRef.RECORD
Use this function to return the recordref of a key. This function returns an error if no key is selected.

NUMERIC

ABS
NewNumber := ABS(Number)
Use this function to calculate the absolute value of a number. ABS always returns a positive numeric value or zero.

POWER
NewNumber := POWER(Number, Power)
Use this function to raise a number to a power.

RANDOM
Number := RANDOM(MaxNumber)
Use this function to return a pseudo-random number.

RANDOMIZE
RANDOMIZE([Seed])
Use this function to generate a set of random numbers, from which RANDOM will select a random number.

ROUND
NewNumber := ROUND(Number [, Precision] [, Direction])
Use this function to round the value of a number variable. The optional parameter tells the system how to round Number. The default rounding method is '='. There are three different options for rounding: '=', '>', '<'

RECORD

This complex data type corresponds to a row in a table. Each record consist of fields (which form the columns of the table). A record is typically used to hold information about a fixed number of properties.

ASCENDING
[IsAscending] := Record.ASCENDING([SetAscending])
Use this function to change or check the order in which the system will search through a C/SIDE table.

CALCFIELDS
[Ok :=] Record.CALCFIELDS(Field1, [Field2],...)
Use this function to update the FlowFields in a record.

CALCSUMS
[Ok :=] Record.CALCSUMS (Field1, [Field2],...)
Use this function to calculate the total of a column of SumIndexFields in a C/SIDE table.

CHANGECOMPANY
[Ok :=] Record.CHANGECOMPANY([CompanyName])
Use this function to redirect references to table data from one company to another.

CLEARMARKS
Record.CLEARMARKS
Use this function to remove all marks on a record.

CONSISTENT
Record.CONSISTENT(Consistent)
Use this function to mark a C/SIDE table as being consistent or inconsistent from an administrative point of view, which you define.

COPY
Record.COPY(FromRecord)
Use this function to copy a record from a C/SIDE table. All filters, marks, and keys are included in the copy.

COPYFILTER
Record.COPYFILTER(FromField, ToRecord.ToField)
Use this function to copy the filter set for one field and apply it to another field.

COPYFILTERS
Record.COPYFILTERS(FromRecord)
Use this function to copy all filters set by SETFILTER or SETRANGE from one record to another.

COUNT
Number := Record.COUNT
Use this function to count the number of records in a C/SIDE table.

COUNTAPPROX
Number := Record.COUNTAPPROX
Use this function to obtain an approximate count of the number of records in the table, for example, for updating progress bars or displaying informational messages. The count is approximate because it uses statistical information maintained by SQL Server, which is not kept precisely in synchronization with modifications to the table and is not under transaction control.

CURRENTKEY
CurrentKey := Record.CURRENTKEY
Use this function to return the current key of a database table.

DELETE
[Ok :=] Record.DELETE([RunTrigger])
Use this function to delete a record in a C/SIDE table.

DELETEALL
Record.DELETEALL([RunTrigger])
Use this function to delete all records in a C/SIDE table that fall within a specified range.

FIELDACTIVE
Ok := Record.FIELDACTIVE(Field)
Use this function to check whether a field is enabled or not.

FIELDERROR
Record.FIELDERROR(Field, [Text])
Use this function to stop the execution of the code (cause a run-time error, in fact) and create an error message for a field.

FIELDNAME
Name := Record.FIELDNAME(Field)
Use this function to return the name of a field as a string.

FIELDNO
Number := Record.FIELDNO(Field)
Use this function to return the number assigned to a field in the table description.

FILTERGROUP
[CurrGroup] := Record.FILTERGROUP([NewGroup])
Use this function to select a filtergroup and to find the number of the current filtergroup. A filtergroup can contain a filter for a Record that has been set earlier with SETFILTER or SETRANGE. The total filter applied is the combination of all the filters set in all the filtergroups. C/SIDE uses 7 FILTERGROUPS internally: 0 Std, 1 Global, 2 Form, 3 Exec, 4 Link, 5 Temp, 6 Security.

FIND
Ok := Record.FIND([Which])
Use this function to find a record in a C/SIDE table based on the values stored in keys. Which the system how to perform the search. If SearchStr contains '=', '>' or '<', you must assign values to all fields of the current and primary keys before you call FIND.

GET
[Ok :=] Record.GET([Value] ,...)
Use this function to find a record based on values stored in primary key fields. This function always uses the primary key for the table and ignores any filters. The system does not change the current key and filters after you call this function.

GETFILTER
String := Record.GETFILTER(Field)
Use this function to return a list of the filters within the current filter group that are applied to a field.

GETFILTERS
String := Record.GETFILTERS
Use this function to return a string which contains a list of the filters within the current filter group for all fields in a record. In addition, this function also returns the state of MARKEDONLY.

GETPOSITION
String := Record.GETPOSITION([UseNames])
Use this function to return a string that contains the primary key of the current record.

GETRANGEMAX
Value := Record.GETRANGEMAX(Field)
Use this function to return the maximum value in a range for a field.

GETRANGEMIN
Value := Record.GETRANGEMIN(Field)
Use this function to return the minimum value in a range for a field.

GETVIEW
String := Record.GETVIEW([UseNames])
Use this function to return a string that describes the current sort order, key and filters on a table.

HASFILTER
Ok := Record.HASFILTER
Use this function to determine if the system has attached a filter to a record within the current filter group.

INIT
Record.INIT
Use this function to initialize a record in a C/SIDE table. The system does not initialize primary key or timestamp fields.

ISEMPTY
Empty := Record.ISEMPTY
Use this function to find out whether a C/SIDE table or a filtered set of records is empty. When you are using SQL Server, this function is faster than using the Record.COUNT function and then testing the result for zero.

INSERT
[Ok :=] Record.INSERT([RunTrigger])
Use this function to insert a record into a C/SIDE table.

LOCKTABLE
Record.LOCKTABLE([Wait] [, VersionCheck])
Use this function to lock a C/SIDE table to protect it from write transactions that conflict with each other. The SQL Server Option for Navision only supports the default values for the parameters of the LOCKTABLE function – LOCKTABLE(TRUE,FALSE).

MARK
[IsMarked] := Record.MARK([SetMarked])
Use this function to mark a record. You can also use this function to find out if a record is marked.

MARKEDONLY
[IsMarkedOnly] := Record.MARKEDONLY([SetMarkedOnly])
Use this function to tell the system to activate a special filter. After you use this function, your view of the table only includes records marked by this function.

MODIFY
[Ok :=] Record.MODIFY([RunTrigger])
Use this function to modify a record in a C/SIDE table.

MODIFYALL
Record.MODIFYALL(Field, NewValue [, RunTrigger])
Use this function to modify a field in all records within a range you specify.

NEXT
Steps := Record.NEXT([Steps])
Use this function to step through a specified number of records and retrieve a record. Steps is used to define the direction of the search and how many records to step over. > 0: Search Steps records forwards in the table. < 0: Search Steps records backwards in the table. = 0 No effect. If you do not specify Steps, the system finds the next record.

READCONSISTENCY
Ok := Record.READCONSISTENCY
Use these functions to determine whether the table supports read consistency.

READPERMISSION
Ok := Record.READPERMISSION
Use this function to find out if you can read from a table. This function can test for both full read permission and a partial read permission that has been granted with a security filter.

RECORDLEVELLOCKING
Ok := Record.RECORDLEVELLOCKING
Use these functions to find out whether the table supports record level locking. When you are using SQL Server, you can use record level locking. When you are using the Navision Database Server, you cannot use record level locking.

RUN — Form.RUN
Use this function to create and launch a form you specify. You can use CLEAR to remove the form.

RUNMODAL — [Action] := Form.RUNMODAL
Use this function to create, launch, and close the form you specify. The optional return code tells you what action the user took. The possible return values are: OK, Cancel, LookupOK, LookupCancel, Yes, No, Close, Helpform, RunObject, RunSystem.

SAVERECORD — CurrForm.SAVERECORD
Use this function to save the current record shown on the form.

SETRECORD — Form.SETRECORD(Record)
Use this function to select the current record shown on the form.

SETSELECTIONFILTER — CurrForm.SETSELECTIONFILTER(Record)
Use this function to have the system note the records the user has selected on the form, mark those records in the table specified, and set the filter to "marked only".

SETTABLEVIEW — SETTABLEVIEW(Record)
Use this function to apply the Table View on the current record as the table view for the form, report or dataport.

UPDATE — CurrForm.UPDATE[(SaveRecord)]
Use this function to save the current record and then update the controls in the form. If you set the SaveRecord parameter to FALSE, this function will not save the record before the system updates the form.

UPDATECONTROLS — CurrForm.UPDATECONTROLS
Use this function to reload the captions of all controls on the current form. This is necessary when the user changes the caption class of a control after the form has been loaded.

UPDATEEDITABLE — UPDATEEDITABLE(Editable)
Use this function to dynamically change the setting of the Editable property of a field, form or control.

URL — String:=Form.URL([UseNames])
This function returns a string that contains the full URL to a form.

VISIBLE — [IsVisible] := Form.VISIBLE([SetVisible])
Use this function to return the current setting of the Visible property of a form or control, and to change the setting of the property.

WIDTH — [CurrWidth] := Form.WIDTH([NewWidth])
Use this function to return the current setting of the Width property of a form or control, and to set this property to a new value.

XPOS — [CurrXPos] := Form.XPOS([NewXPos])
Use this function to return the current setting of the XPos property of a form or control, and to set this property to a new value.

YPOS — [CurrYPos] := Form.YPOS([NewYPos])
Use this function to return the current setting of the YPos property of a form or control, and to set this property to a new value.

GUID

Use this data type to give a unique identifying number to any database object. The Globally Unique Identifier (GUID) data type is a 16 byte binary data type. This data type is used for the global identification of objects, programs, records and so on. The important property of a GUID is that each value is globally unique. The value is generated by an algorithm, developed by Microsoft, which assures this uniqueness. The standard textual representation is {12345678-1234-1234-1234-1234567890AB}.

CREATEGUID — Guid :=CREATEGUID()
Use this function to create a new unique GUID. The value can then be assigned to a GUID data type or a text data type. Use the text data type if you want to compare the GUID to another text string.

ISNULLGUID — Ok := ISNULLGUID(Guid)
Use this function to check whether or not a value has been assigned to a GUID. A null GUID that consists only of zeros is valid but must never be used for reference purposes.

INSTREAM & OUTSTREAM

The InStream (input stream) and OutStream (output stream) data types are generic stream objects that you can use to read from or write to files and BLOBs. In addition, the InStream and OutStream data types enable data to be read from and sent to objects of the types Automation and OCX. The Microsoft XML DOM can read from an InStream object and write to an OutStream object.

InStream.EOS — IsEOS:= InStream.EOS()
Use this function to find out whether or not an input stream has reached End of Stream (EOS).

InStream.READ — [{Read}:=] InStream.Read(Variable, [Length])
Use this function to read a specified number of bytes from an InStream object. Data is read in binary format.

InStream.READTEXT — [{Read}:=] InStream.ReadText(Text, [Length])
Use this function to read text from an InStream object. READTEXT reads the specified number of bytes, the maximum length of the string or until the end of the line. Data is read in text format.

OutStream.WRITE — [{Written}:=] OutStream.Write(Variable, [Length])
Use this function to write a specified number of bytes to the stream. Data is written in binary format.

OutStream.WRITETEXT — [{Written} :=] OutStream.WriteText([Text, [Length]])
Use this function to write text to an OutStream object. Data is written in text format.

RELATION — TableNumber := Record.RELATION(Field)
Use this function to find out the table relationship of a given field.

RENAME — [Ok]:= Record.RENAME(Value1, [Value2],...)
Use this function to change a primary key in a C/SIDE table.

RESET — Record.RESET
Use this function to remove all filters, including any special filters set by MARKEDONLY, and change the current key to the primary key. The system also removes any marks on the record and clears any C/AL variables on the record.

SETCURRENTKEY — [Ok :=] Record.SETCURRENTKEY(Field1, [Field2],...)
Use this function to select a key for a table.

SETFILTER — Record.SETFILTER(Field, String, [Value],...)
Use this function to assign a filter to a field you specify.

SETPERMISSIONFILTER — Record.SETPERMISSIONFILTER
Use this function to apply the user's security filter to a Record variable. The security filter is combined with any other filters that are placed on the Record variable with SETFILTER or SETRANGE. This C/AL function only applies to the SQL Server Option for Navision.

SETPOSITION — Record.SETPOSITION(String)
Use this function to set the fields in a primary key on a record to the values specified in the supplied string. The remaining fields are left untouched.

SETRANGE — Record.SETRANGE(Field [,FromValue] [.ToValue])
Use this function to set a simple filter, such as a single range or a single value, on a field.

SETRECFILTER — Record.SETRECFILTER
Use this function to set the values in the primary key fields of the current record as a record filter.

SETVIEW — Record.SETVIEW(String)
Use this function to set the current sort order, key and filters on a table.

TABLECAPTION — Caption := Record.TABLECAPTION
Use this function to return the current caption of a table as a string.

TABLENAME — Name := Record.TABLENAME
Use this function to return the name of a C/SIDE table.

TESTFIELD — Record.TESTFIELD(Field, [Value])
Use this function to test to see if the contents of a field match a given value. If the contents differ from the given value, the system displays an error message.

TRANSFERFIELDS — Record.TRANSFERFIELDS(FromRecord [, InitPrimaryKeyFields])
Use this function to copy all matching fields in one record to another record.

VALIDATE — Record.VALIDATE(Field [, NewValue])
Use this function to call the triggers for the field you specify.

WRITEPERMISSION — Ok := Record.WRITEPERMISSION
Use this function to find out if you can write to a table. This function can test for both full write permission and a partial write permission that has been granted with a security filter. A write permission consists of Insert, Delete and Modify permissions.

RECORDID

This data type contains the table number and the primary key of a table. You can store a RecordID in the database but you cannot set filters on a RecordID.

GETRECORD — RecordRef := RecordID.GETRECORD
Use this function to return a recordref that refers to the record identified by recordID.

TABLENO — No := RecordID.TABLENO
Use this function to return the table number of the table identified by recordid. This function returns an error if the record is blank.

RECORDREF

This complex data type identifies a row in a table. Each record consist of fields (which form the columns of the table). A record is typically used to hold information about a fixed number of properties. The RecordRef object can refer to any table in the database. Use the RecordRef.OPEN function to select the table you want to access. When you use the RecordRef.OPEN function a new object is created. This object contains references to the open table, filters and the record itself and all the fields it contains.

ASCENDING — [IsAscending :=] RecordRef.ASCENDING([SetAscending])
Use this function to change or check the order in which the system will search through the table referred to by the recordref.

CAPTION — Caption := RecordRef.CAPTION
Use this function to return the caption of the table that is currently selected. This function returns an error if no table is selected.

CLOSE — RecordRef.CLOSE
Use this function to close the current form or table.

COUNT — Number := RecordRef.COUNT
Use this function to count the number of records that are within the filters that are currently applied to the table referred to by the recordref.

COUNTAPPROX — Number := RecordRef.COUNTAPPROX
Use this function to obtain an approximate count of the number of records in the table, for example, for updating progress bars or displaying informational messages.

QUERYREPLACE [IsQueryreplace :=] File.QUERYREPLACE([SetQueryreplace])
This function is used to determine whether the system should query the user before overwriting a file if it already exists.

READ [Read] := File.READ(Variable)
Use this function to read from an ASCII or binary file. If TEXTMODE is set to TRUE, the system reads a line of text from the file, evaluates it and sets the variable equal to the result. If TEXTMODE is set to FALSE, the system determines the number of bytes to read based on the size of the variable.

RENAME [Ok:=] File.RENAME(OldName, NewName)
Use this function to rename an ASCII or binary file.

SEEK File.SEEK(Position)
Use this function to set a file pointer to a new position in an ASCII or binary file.

SETSTAMP [Ok] := File.SETSTAMP(Name, Date [, Time])
Use this function to set a time stamp for a file.

TEXTMODE [IsTextmode] := File.TEXTMODE([SetTextmode])
This function is used to set whether a file should be opened as an ASCII file or a binary file.

TRUNC File.TRUNC
Use this function to truncate an ASCII or binary file to the current position of the file pointer.

WRITE File.WRITE(Value)
Use this function to write to an ASCII or binary file. If TEXTMODE is set to TRUE and Value is an integer, the system formats the integer into text and writes the result, followed by a new line character. If Value is a record, the system separates each field with a tab character. If TEXTMODE is FALSE and Value is an integer, the system writes the integer as an integer which is four bytes long.

WRITEMODE [IsWritemode :=] File.WRITEMODE([SetWritemode])
Use this function before you use OPEN to set or test whether you can write to a file in later calls.

FORM

Variables of this complex data type store forms. Forms contain simpler elements called controls. Controls are used to display information to the user or to receive information from the user.

ACTIVATE [Ok :=] Form.ACTIVATE
Use this function to make a form or control active.

ACTIVE IsActive := Form.ACTIVE
Use this function to find out if the current form is active or inactive.

CAPTION [CurrCaption] := Form.CAPTION([NewCaption])
Use this function to return the current caption of an object as a string, and to set a new caption for the object.

CLOSE Form.CLOSE
Use this function to close the current form or table.

EDITABLE [IsEditable] := From.EDITABLE([SetEditable])
Use this function to return the current setting of the Editable property, and to change the setting of the property.

FORM Subform := Form.FORM
Use this function to access a form that is a subform of the current form - that is, the form that is defined as the SubFormID of a subform control.

FORM.RUN FORM.RUN(Number [, Record] [, Field])
Use this function to create and launch a form object, which you specify.

FORM.RUNMODAL [Action] := Form.RUNMODAL(Number [, Record] [, Field])
Use this function to create, run, and close a form object, which you specify. The system runs the form modally.

GETRECORD Form.GETRECORD(Record)
Use this function to retrieve the current record shown on the form.

HEIGHT [CurrHeight] := Form.HEIGHT([NewHeight])
Use this function to return the current setting of the Height property of a form or control, and to set this property to a new value.

LOGHEIGHT [CurrLogHeight] := Form.LOGHEIGHT([NewLogHeight])
Use this function to return the current setting of the LogHeight property of a form, and to set this property to a new value.

LOGWIDTH [CurrLogWidth] := Form.LOGWIDTH([NewLogWidth])
Use this function to return the current setting of the LogWidth property of a form, and to set this property to a new value.

LOOKUPMODE [CurrLookupMode] := Form.LOOKUPMODE([NewLookupMode])
Use this function to return the current setting of the LookupMode property of a form, and to set this property to a new value.

MAXIMIZEDONOPEN [CurrMaximized] := Form.MAXIMIZEDONOPEN([NewMaximized])
Use this function to return the current setting of the MaximizedOnOpen property of a form, and to set this property to a new value.

MINIMIZEDONOPEN [CurrMinimized] := From.MINIMIZEDONOPEN([NewMinimized])
Use this function to return the current setting of the MinimizedOnOpen property of a form, and to set this property to a new value.

OBJECTID String := Form.OBJECTID([UseNames])
This function returns a string in the "form xxx" format, where xxx is the name or number of the application object.

CURRENTKEY CurrentKey := RecordRef.CURRENTKEY
Use this function to return the current key of the table referred to by the recordref. The current key is returned as a string.

CURRENTKEYINDEX [CurrKeyIndex :=] RecordRef.CURRENTKEYINDEX([NewKeyIndex])
Use this function to return or set the current key of the table referred to by the recordref. The current key is set or returned as a number.

DELETE [Ok :=] RecordRef.DELETE([RunTrigger])
Use this function to delete a record in a C/SIDE table.

DELETEALL RecordRef.DELETEALL([RunTrigger])
Use this function to delete all records in a C/SIDE table that fall within a specified range.

DUPLICATE RecordRef := RecordRef.DUPLICATE
Use this function to duplicate the table that contains the recordref.

FIELD Field := RecordRef.FIELD(FieldNo)
Use this function to return the recordref of the field that has the number fieldno in the table that is currently selected. If no field has this number, the function returns an error.

FIELDCOUNT Count := RecordRef.FIELDCOUNT
Use this function to return the number of fields in the table that is currently selected or to return the number of fields that have been defined in a key. These functions returns an error if no table or no key is selected

FIELDEXIST Exist := RecordRef.FIELDEXIST(FieldNo)
Use this function to find out if the field that has the number fieldno exists in the table that is referred to by the recordref. The function returns an error if no table is currently selected.

FIELDINDEX Field := RecordRef.FIELDINDEX(Index)
Use this function to return the fieldref of the field that has this index in the table referred to by the recordref.

FILTERGROUP [CurrGroup :=] RecordRef.FILTERGROUP([NewGroup])
Use this function to change the filter group that is being applied to the table. A filtergroup can contain a filter for a RecordRef that has been set earlier with SETFILTER or SETRANGE. The total filter applied is the combination of all the filters set in all the filtergroups.

FIND [Ok :]= RecordRef.FIND([Which])
Use this function to find a record in a table based on the values stored in the key fields.

GET [Ok:=]RecordRef.GET(RecordID)
Use this function to find a record based on the ID of the record.

GETFILTERS String := RecordRef.GETFILTERS
Use this function to find out which filters have been applied to the table referred to by the recordref.

GETPOSITION String := RecordRef.GETPOSITION([UseNames])
Use this functions to return a string that contains the primary key of the current record.

GETTABLE RecordRef.GETTABLE(rec)
Use this function to make a recordref variable use the same table instance as a record variable.

GETVIEW String := RecordRef.GETVIEW([UseNames])
Use this function to return a string that describes the current sort order, key and filters on a table.

HASFILTER Ok := RecordRef.HASFILTER
Use this function to find out whether or not a filter has been applied to the table referred to by a recordref.

INIT RecordRef.INIT
Use this function to initialize a record in a table.

INSERT [Ok :=] RecordRef.INSERT([RunTrigger])
Use this function to insert a record into a table.

ISEMPTY Empty := RecordRef.ISEMPTY
Use this function to find out whether any records exist within a filtered set of records in a table.

KEYCOUNT Count := RecordRef.KEYCOUNT
Use this function to return the number of keys that exist in the table that is referred to by the recordref. This function returns an error if no table is selected.

KEYINDEX Key := RecordRef.KEYINDEX(Index)
Use this function to return the keyref of the key that has this index in the table that is currently selected.

LOCKTABLE RecordRef.LOCKTABLE([Wait] [, VersionCheck])
Use this function to lock a table to protect it from write transactions that conflict with each other.

MODIFY [Ok :=] RecordRef.MODIFY([RunTrigger])
Use this function to modify a record in a C/SIDE table.

NAME Name := RecordRef.NAME
Use this function to return the name of the table that is currently selected. This function returns an error if no table is selected.

NEXT [Steps :=] RecordRef.NEXT([Steps])
Use this function to step through a specified number of records and retrieve a record.

NUMBER No := RecordRef.NUMBER
Use this function to return the table ID (number) of the table that contains the record referred to by the recordref.

OPEN RecordRef.OPEN(No[, Temp][, CompanyName])
Use this function to make a RecordRef variable refer to a table which is identified by its number in a particular company.

READCONSISTENCY Ok := RecordRef.READCONSISTENCY
Use this function to know whether or not read consistency is supported.

READPERMISSION	Ok := RecordRef.READPERMISSION Use this function to find out if you can read from a table. This function can test for both full read permission and a partial read permission that has been granted with a security filter.
RECORDID	RecordID := RecordRef.RECORDID Use this function to return the RecordID of the record that is currently selected in the table. If no table is selected, an error is generated.
RECORDLEVELLOCKING	Ok := RecordRef.RECORDLEVELLOCKING Use this function to find out whether the table supports record level locking.
RESET	RecordRef.RESET Use this function to remove all filters, including any special filters set by MARKEDONLY and change the current key to the primary key. The system also removes any marks on the record and clears any C/AL variables on the record.
SETPERMISSIONFILTER	RecordRef.SETPERMISSIONFILTER Use this function to apply the user's security filter to a RecordRef variable. The security filter is combined with any other filters that are placed on the RecordRef variable with SETFILTER or SETRANGE.
SETPOSITION	RecordRef.SETPOSITION(String) Use this function to set the fields in a primary key on a record to the values specified in the supplied string. The remaining fields are left untouched.
SETRECFILTER	RecordRef.SETRECFILTER Use this function to set a filter on a record that is referred to by a recordref.
SETTABLE	RecordRef.SETTABLE(rec) Use this function to make a record variable use the same table instance as a recordref variable.
SETVIEW	RecordRef.SETVIEW(String) Use this function to set the current sort order, key and filters on a table.
WRITEPERMISSION	Ok := RecordRef.WRITEPERMISSION Use this function to find out if you can write to a table. This function can test for both full write permission and a partial write permission that has been granted with a security filter. A write permission consists of Insert, Delete and Modify permissions.

REPORT

Use this complex data type to store reports. Reports contain a number of simpler elements called controls. Controls are used to display information to the user or receive information from the user.

BREAK	BREAK Use this function to exit from a loop or a trigger in a data item trigger of a report or dataport.
CREATETOTALS	CREATETOTALS(Var1 [, Var2] ,...) Use this function to maintain totals for a variable in the same way as totals are maintained for fields by using the TotalFields property.
NEWPAGE	NEWPAGE Use this function to force a page break when printing a report.
NEWPAGEPERRECORD	[IsNewPagePerRecord] := NEWPAGEPERRECORD([SetNewPagePerRecord]) Use this function to return the current setting of the NewPagePerRecord property, and to set this property to a new value.
OBJECTID	String:=Report.OBJECTID([UseNames]) This function returns a string in the "report xxx" format, where xxx is the name or number of the application object.
PAGENO	[CurrPageNo] := PAGENO([NewPageNo]) Use this function to return the current page number of a report, and to set a new page number.
PAPERSOURCE	CurrReport.PAPERSOURCE(PaperBinNo [, PhysicalPage]) Use this function to return the paper source used for the current page or a specified page, and to set a new paper source.
PREVIEW	IsPreview := PREVIEW Use this function to determine whether a report is being printed in preview mode or not.
PRINTONLYIFDETAIL	[IsPrintOnlyIfDetail] := PRINTONLYIFDETAIL([SetPrintOnlyIfDetail]) Use this function to return the current setting of the PrintOnlyIfDetail property, and to set this property to a new value.
QUIT	QUIT Use this function to abort the processing of a report or dataport.
REPORT.RUN	REPORT.RUN(Number [, ReqWindow] [, SystemPrinter] [, Record]) Use this function to load and execute the report you specify.
REPORT.RUNMODAL	REPORT.RUNMODAL(Number [, ReqWindow] [, SystemPrinter] [, Record]) Use this function to load and execute the report you specify.
RUN	Report.RUN Use this function to load and execute the report you specify.
RUNMODAL	Report.RUNMODAL Use this function to load and execute the report you specify.
SAVEASHTML	[Ok :=] Report.SAVEASHTML(Number, FileName [,SystemPrinter] [, Rec]) [Ok :=] Report.SAVEASHTML(FileName) Use this function to save a report as an HTML file. A browser that supports HTML version 3.0 or later is recommended for viewing the file.

FIELDERROR	FieldRef.FIELDERROR([Text]) Use this function to stop the execution of the code (cause a run-time error, in fact) and create an error message for a field.
GETFILTER	String := FieldRef.GETFILTER Use this function to return the filter within the current filter group that are applied to a field.
GETRANGEMAX	Value := FieldRef.GETRANGEMAX Use this function to return the maximum value in a range for a field.
GETRANGEMIN	Value := FieldRef.GETRANGEMIN Use this function to return the minimum value in a range for a field.
LENGTH	Length := FieldRef.LENGTH Use this function to return the maximum size of the field (the size specified in the DataLength property of the field).
NAME	Name := FieldRef.NAME Use this function to return the name of a field as a string.
NUMBER	No := FieldRef.NUMBER Use this functions to return the number of the table or of the field.
OPTIONCAPTION	OptionCaption := FieldRef.OPTIONCAPTION Use this function to return the option caption of the field that is currently selected.
OPTIONSTRING	OptionString := FieldRef.OPTIONSTRING Use this function to return the list of options that are available in the field that is currently selected.
RECORD	RecordRef := FieldRef.RECORD Use this function to return the recordref of the field that is currently selected.
RELATION	TableNumber := FieldRef.RELATION Use this function to find out the table relationship of a given field.
SETFILTER	FieldRef.SETFILTER(String [, Value],...) Use this function to assign a filter to a field you specify.
SETRANGE	FieldRef.SETRANGE([FromValue] [, ToValue]) Use this function to set a simple filter, such as a single range or a single value, on a field.
TESTFIELD	FieldRef.TESTFIELD([Value]) Use this function to see if the contents of a field match a given value.
TYPE	Type := FieldRef.TYPE Use this function to return the data type of the field that is currently selected.
VALIDATE	FieldRef.VALIDATE([NewValue]) Use this function to enter a new value into a field and have the new value validated by the properties and code that have been defined for that field.
VALUE	[CurrValue :=] FieldRef.VALUE([NewValue]) Use this function to set or get the value of the field that is currently selected.

FILE

Variables of this data type give you access to files. Files can be opened in text or binary mode.

CLOSE	File.CLOSE Use this function to close a file which has been opened by OPEN.
COPY	[Ok :=] File.COPY(FromName, ToName) Use this function to copy a file.
CREATE	[Ok :=] File.CREATE(Name) Use this function to create and open an ASCII or binary file. If the file exists, the system will truncate it and then open it.
CREATEINSTREAM	File.CreateInStream(Stream) Use this function to create an InStream object for a file. This enables you to stream data into the file.
CREATEOUTSTREAM	File.CreateOutStream(Stream) Use this function to create an OutStream object for a file. This enables you to stream data out of the file.
CREATETEMPFILE	File.CreateTempFile Use this function to create a temporary file. This enables you to save data of any format to a temporary file. This file has a unique name and will be stored in the temporary files folder.
ERASE	[Ok] := File.ERASE(Name) Use this function to erase a file.
EXISTS	[Ok :=] File.EXISTS(Name) Use this function to determine if a file exists.
GETSTAMP	[Ok]:= File.GETSTAMP(Name, Date [, Time]) Use this function to find out the time at which a file was last written to (return a time stamp).
LEN	Length := File.LEN Use this function to return the length of an ASCII or binary file.
NAME	Name := File.NAME Use this function to return the name of an ASCII or binary file.
OPEN	[Ok]:= File.OPEN Use this function to open an existing ASCII or binary file. As compared to CREATE, this function does not create the file if it does not exist.
POS	Position := File.POS Use this function to return the current position of the file pointer in an ASCII or binary file.

CREATEDATETIME
DateTime := CREATEDATETIME(Date, Time)
Use this function to create a datetime from a date and a time.

CURRENTDATETIME
Datetime := CURRENTDATETIME
Use this function to return the current datetime.

DATE2DMY
Number := DATE2DMY(Date, What)
Returns the day, month, or year based on a date.

DATE2DWY
Number := DATE2DWY(Date, What)
Returns the day of the week, week number, and year based on the input Date.

DATI2VARIANT
Variant := DATI2VARIANT(Date, Time)
Use this system date function to create a variant that contains a VT_DATE.

DMY2DATE
Date := DMY2DATE(Day [, Month] [, Year])
Use this function to return a Date based on a day, month, and year.

DT2DATE
Date := DT2DATE(Datetime)
Use this function to return the date part of a datetime.

DT2TIME
DT2TIME
Use this function to return the time part of a datetime.

DWY2DATE
Date := DWY2DATE(WeekDay [, Week] [, Year]))
Use this function to return a Date based on a weekday, a week, and a year.

NORMALDATE
NormalDate := NORMALDATE(Date)
Use this function to return the normal date (as opposed to the closing date) for the argument Date.

ROUNDDATETIME
NewDateTime := ROUNDDATETIME(Datetime [, Precision][, Direction])
Use this function to round a datetime.

TIME
Time := TIME
Use this function to retrieve the current time from the operating system.

TODAY
Date := TODAY
Use this function to return the current date set in the operating system.

VARIANT2DATE
Date := VARIANT2DATE(Variant)
Use this system date function to return a date from a VT_DATE variant.

VARIANT2TIME
Time := VARIANT2TIME(Variant)
Use this system date function to return a time from a VT_DATE variant.

WORKDATE
[WorkDate]:= WORKDATE([NewDate])
Use this function to return the current work date or to set a new work date.

DIALOG

Variables of this complex data type store dialog windows. These variables also give you access to a number of dialog functions, such as OPEN, CLOSE, and so on.

BEEP
BEEP(Frequency, Duration)
Use this function to sound a tone through the computer's speaker.

CLOSE
Dialog.CLOSE
Use this function to close a dialog window which has been opened by OPEN.

CONFIRM
Ok := Dialog.CONFIRM(String [, Default] [, Value1] ,...)
Use this function to create a dialog box which prompts the user for a yes or no answer.

ERROR
ERROR(String [, Value1, ...])
Use this function to display an error message and end the execution of C/AL code.

INPUT
NewControlID := Dialog.INPUT([ControlID] [,Variable])
Use this function to read what a user enters into a field in a window.

MESSAGE
MESSAGE(String [, Value1, ...])
Use this function to display a text string in a message window.

OPEN
Dialog.OPEN(String [, Variable1], ...)
Use this function to open a dialog window.

STRMENU
OptionNumber := Dialog.STRMENU(OptionString [, DefaultNumber])
Use this function to create a menu window that displays a series of options.

UPDATE
Dialog.UPDATE([Number] [, Value])
Use this function to update the value of a '#'-or '@' field in the current window.

YIELD
YIELD
Use this function to pass control to the operating system, specifically DOS/Windows 3.x, so it can process events. Once the operating system finishes, you regain control.

FIELDREF

This complex data type identifies a field in a table and gives you access to this field. The fieldref object can refer to any field in any table in the database.

ACTIVE
Ok := FieldRef.ACTIVE
Use this function to check whether the field that is currently selected is enabled or not.

CALCFIELD
[Ok :=] FieldRef.CALCFIELD
Use this function to update a FlowField in a record.

CALCSUM
[Ok:=] FieldRef.CALCSUM
Use this function to calculate the total of a SumIndexFields in a table.

CAPTION
Caption := FieldRef.CAPTION
Use this function to return the current caption of a field referred to by a fieldref as a string.

CLASS
Class := FieldRef.CLASS
Use this function to return the fieldclass of the field that is currently selected.

SAVEASXML
[Ok :=] Report.SAVEASXML(Number, FileName [,SystemPrinter] [, Rec])
[Ok :=] Report.SAVEASXML(FileName)
Use this function to save a report as an XML file. The report can then be exported to User Portal.

SETTABLEVIEW
SETTABLEVIEW(Record)
Use this function to apply the Table View on the current record as the table view for the form, report or dataport.

SHOWOUTPUT
[IsShow] := SHOWOUTPUT ([SetShow])
Use this function to return the current setting of whether a section should be outputted or not, and to change this setting.

SKIP
SKIP
Use this function to skip the current iteration of the current data item.

TOTALSCAUSEDBY
FieldNo := TOTALSCAUSEDBY
Use this function to determine which field caused a group total to be calculated - meaning determining which field changed contents and thereby concluded a group.

URL
String:=Report.URL([UseNames])
This function returns a string with the full URL to a report.

USEREQUESTFORM
[IsUseRequestForm] := USEREQUESTFORM([SetUseRequestForm])
Use this function to return the current setting of the UseReqForm property, and to set this property to a new value. This function should be used before the request form is run - that is, in the OnInitReport trigger. Although it will not cause an error if it is used elsewhere, it will have no effect.

STRINGS

CONVERTSTR
NewString := CONVERTSTR(String, FromCharacters, ToCharacters)
Use this function to convert the characters in a string based on the characters in the strings FromCharacters and ToCharacters, which serve as conversion tables.

COPYSTR
NewString := COPYSTR(String, Position [, Length])
Use this function to copy a substring of any length from a specific position in a string (text or code) to a new string. If you omit Length, the resulting string includes all characters from Position to the end of the string.

DELCHR
NewString := DELCHR(String [, Where] [, Which])
Use this function to delete one or more characters in a string.

DELSTR
NewString := DELSTR(String, Position [, Length])
Use this function to delete a substring inside a string (text or code).

FORMAT
String := FORMAT(Value [, Length] [, FormatNumber | FormatString])
Use this function to format a value into a string.

INCSTR
NewString := INCSTR(String)
Use this function to increase a positive number or decrease a negative number inside a string by one (1).

INSSTR
NewString := INSSTR(String, SubString, Position)
Use this function to insert a substring into a string.

LOWERCASE
NewString := LOWERCASE(String)
Use this function to convert all letters in a string to lowercase.

MAXSTRLEN
MaxLength := MAXSTRLEN(String)
Use this function to return the maximum defined length of a string variable.

PADSTR
NewString := PADSTR(String, Length [, FillCharacter])
Use this function to change the length of a string to a length you define. The system does this by either truncating the string or adding filler characters at the end of the string.

SELECTSTR
NewString := SELECTSTR(Number, CommaString)
Use this function to retrieve a substring from a comma-separated string.

STRCHECKSUM
CheckNumber :=STRCHECKSUM(String [, WeightString] [, Modulus])
Use this function to calculate a checksum for a string containing a number.

STRLEN
Length := STRLEN(String)
Use this function to return the length of a string you define.

STRPOS
Position := STRPOS(String, SubString)
Use this function to search for a substring inside a string.

STRSUBSTNO
NewString := STRSUBSTNO(String [,Value1, ...])
Use this function to replace %1, %2, %3... and #1, #2, #3... fields in a string with the values you provide as optional parameters.

UPPERCASE
NewString := UPPERCASE(String)
Use this function to convert the letters in a string to uppercase.

SYSTEM

ARRAY

ARRAYLEN
Length := ARRAYLEN(Array [, Dimension])
Use this function to return the total number of elements in an array or the number of elements in a specific dimension.

COMPRESSARRAY
[Count =:] COMPRESSARRAY(StringArray)
Use this function to move all non-empty strings (text) in an array to the beginning of the array. The resulting StringArray has the same number of elements as the input array, but empty entries and entries that contain only blanks appear at the end of the array.

COPYARRAY — COPYARRAY(NewArray, Array, Position [, Length])
Use this function to copy one or more elements in an array to a new array. You can only copy from one-dimensional arrays. Repeat the COPYARRAY function to copy two- and three-dimensional arrays.

CODECOVERAGE

CODECOVERAGELOG — [IsActive]:= CODECOVERAGELOG([NewIsActive])
Use this function to start and stop the logging of code. You can also use it to retrieve the current logging status. You must only start the Code Coverage tool from the command prompt when you want to get a total overview of the code used when running Navision or when you are testing the application.

LANGUAGE

GLOBALLANGUAGE — [LanguageID]:= GLOBALLANGUAGE([NewLanguageID])
Use this function to set and retrieve the current C/SIDE global language setting The LanguageID is a standard Windows language ID. The Windows Language virtual table contains a list of these IDs and the corresponding names and short names..

LANGUAGE — [CurrLanguage]:= LANGUAGE([NewLanguage])
Use this function to set and retrieve the current language setting for an object (form, report or datapor t).

WINDOWSLANGUAGE — LanguageID:= WINDOWSLANGUAGE
Use this function to retrieve the current Windows language setting.

OPERATING SYSTEM

COMMANDLINE — String := COMMANDLINE
Use this function to return a list of the parameters used to start Navision.

CONTEXTURL — String:=CONTEXTURL
Use this function to return a context string that defines the current position of the running objects. Here are two examples of a context string:
navision://client/run?database=filename&company=companyname
navision://client/r un?server=servername&company=companyname&servertype=MSSQL

ENVIRON — String := ENVIRON(Name)
Use this function to return a string associated with an environment variable. If the environment variable does not exist, the string that is returned may contain garbage.

GUIALLOWED — [Ok:=] GUIALLOWED()
Use this operating system function to check whether the C/AL code is allowed to show any information on the screen. When you run Navision Application Server, GUIALLOWED always returns FALSE and any call to CONFIRM or d.OPEN, or any attempt to use a form or dataport will generate an error.

HYPERLINK — HYPERLINK(URL)
This function starts up Microsoft Internet Explorer, passing a URL as an argument to that program.

OSVERSION — String := OSVERSION
Use this function to retu rn a string which contains the name and version of the operating system or operating environment. This string tells you the type and version of the operating system or operating environment. Here are some typical examples of what the system returns: Windows 98 -> Windows_95_4.10; Windows NT -> Windows_NT_4.0; Windows 2000 -> Windows_NT_5.0; Windows XP -> Windows_NT_5.1.

SHELL — [ReturnCode]:= SHELL(Name [, Param, ...])
Use this function to execute external programs and operating system commands from C/AL programs. You can run this function modally or non-modally, depending on whether or not you include the return value from the external program in your code. To pass multiple parameters to the command, enter the parameters, either as individual arguments or as a string with the arguments separated by spaces. The total length of the strings cannot exceed 128 characters.

SLEEP — SLEEP(Duration)
Use this function to return control to the operating system for a specifiable amount of time (milliseconds).

VARIABLE

CLEAR — CLEAR(Variable)
Use this function to clear the value of a single variable. CLEAR also clears all filters that were set if the variable is a record and resets the key to the primary key. Use the CLEARALL function to clear all internal variables, keys, and filters in the object and in any associated objects such as reports, forms, codeunits, and so on that contain C/AL code. Note, however, that CLEARALL does not affect or change values for variables in single instance codeunits.

CLEARALL — CLEARALL
Use this function to clear all internal variables (except REC variables), keys, and filters in the object and in any associated objects, such as reports, forms, codeunits, and so on that contain C/AL code. CLEARALL works by calling CLEAR repeatedly on each variable. However, this is not the case with codeunits, where the CLEARALL function works by calling CLEARALL inside the codeunit. It deletes the contents of the codeunit, whereas CLEAR only deletes the reference to the codeunit.

COPYSTREAM — [Ok :=] COPYSTREAM(OutStream, InStream)
Use this variable function to copy the information contained in an InStream to an OutStream.

VISIBLE — [IsVisible] := Form.VISIBLE([SetVisible])
Use this function to return the current setting of the Visible property of a form or control, and to change the setting of the property.

WIDTH — [CurrWidth] := Form.WIDTH([NewWidth])
Use this function to return the current setting of the Width property of a form or control, and to set this property to a new value.

XPOS — [CurrXPos] := Form.XPOS([NewXPos])
Use this function to return the current setting of the XPos property of a form or control, and to set this property to a new value.

YPOS — [CurrYPos] := Form.YPOS([NewYPos])
Use this function to return the current setting of the YPos property of a form or control, and to set this property to a new value.

DATABASE

CHECKLICENSEFILE — CHECKLICENSEFILE(KeyNumber)
Use this function to check a key in the license file of the system.

COMMIT — COMMIT
Use this function to end the current write transaction.

COMPANYNAME — Name := COMPANYNAME
Use this function to return the current company name.

CURRENTTRANSACTIONTYPE — [TransactionType :=] CURRENTTRANSACTIONTYPE([TransactionType])
This function can be used both to return the current transaction type and set a new type to be assigned. The following basic transaction types are available: Browse, Snapshot, UpdateNoLocks, Update, Report. [SQL]

SELECTLATESTVERSION — SELECTLATESTVERSION
This function forces the latest version of the database to be used.

SERIALNUMBER — String := SERIALNUMBER
Use this function to return a string which contains the serial number of the license file for your Navision system.

USERID — ID := USERID
Use this function to have the system return the ID of the current user.

DATAPORT

Dataports are objects that are used for importing data from and exporting data to external text files.

BREAK — BREAK
Use this function to exit from a loop or a trigger in a data item trigger of a report or dataport.

DATAPORT.RUN — DATAPORT.RUN(Number [, ReqWindow] [, Record])
Use this function to load and execute the dataport you specify.

DATAPORT.RUNMODAL — DATAPORT.RUNMODAL(Number [, ReqWindow] [, Record])
Use this function to load and execute the dataport you specify.

FILENAME — [CurrFileName] := FILENAME([NewFileName])
Use this function to return the current setting of the FileName property of a dataport, and to set this property to a new value.

IMPORT — [IsImport] := IMPORT([SetImport])
Use this function to return the current setting of the Import property, and to change the setting of the property.

QUIT — QUIT
Use this function to abort the processing of a report or dataport.

RUN — Dataport.RUN
Use this function to load and execute the dataport you specify.

RUNMODAL — Dataport.RUNMODAL
Use this function to load and execute the dataport you specify.

SETTABLEVIEW — SETTABLEVIEW(Record)
Use this function to apply the Table View on the current record as the table view for the form, report or dataport.

SKIP — SKIP
Use this function to skip the current iteration of the current data item.

DATES & TIMES

Use this simple data type DATE to denote dates ranging from January 1, 0 (the year zero) to December 31, 9999. The system defines an undefined date as 0D. Use the data type DATETIME to denote the date and time of day. The datetime is stored in the database as Coordinated Universal Time (UTC). Use the data type DURATION to represent the difference between two datetimes, in milliseconds. This value can be negative. It is a 64 bit integer. Use the simple data type TIME to denote a time. The system defines an undefined time as 0T. Any time between 00:00:00 to 23:59:59 is valid.

CALCDATE — NewDate := CALCDATE(DateExpression [, Date])
Calculates a new date based on a date expression and a reference date.

CLOSINGDATE — ClosingDate := CLOSINGDATE(Date)
Use this function to return the closing date for a Date.

AUTOMATION

The Automation data type is used to reference an automation server. In order to use an automation server in C/SIDE, define a variable of type Automation and give it a name. C/SIDE can receive events from an Automation server.

CREATE — [Ok :=] CREATE(Automation [,NewServer])
Use this function to create an Automation object.

ISCLEAR — Ok := ISCLEAR(Automation)
Use this variable function to check whether an automation object has been created or not.

VARIABLEACTIVE — IsActive := VARIABLEACTIVE(Variable)
Use this function to determine if a variable, such as field or a control, is active or inactive.

BLOB

The maximum size of a BLOB is normally determined by your system's disk storage capacity. However, the maximum size in C/SIDE is 2GB.

CREATEINSTREAM — Blob.CreateInStream(Stream)
Use this function to create an InStream object for a BLOB (Binary Large Object).
This enables you to stream data into the BLOB.

CREATEOUTSTREAM — Blob.CreateOutStream(Stream)
Use this function to create an OutStream object for a BLOB (Binary Large Object).
This enables you to stream data out of the BLOB.

EXPORT — [ExportName :=] := Blob.EXPORT([Name [, CommonDialog]])
Use this function to export a BLOB (Binary Large Object).

HASVALUE — HasValue := Blob.HASVALUE
Use this function to determine if a BLOB (Binary Large Object) has a value.

IMPORT — [ImportName :=] := Blob.IMPORT([Name [, CommonDialog]])
Use this function to import a BLOB (Binary Large Object).

CODEUNIT

Use this complex data type to store units of C/AL code. Codeunits contain a number of user-defined functions.

Codeunit.RUN — [Ok] := Codeunit.RUN(Number [, Record])
Use this function to load and execute the unit of C/AL code you specify.

RUN — [Ok] := Codeunit.RUN(VAR Record)
Use this function to load and execute the unit of C/AL code you specify. To use this function, you can specify a C/SIDE table associated with the codeunit when you defined the codeunit properties. This lets you pass a variable with the function. The transaction that the codeunit contains is always committed due to the boolean return value.

CONTROLS

ACTIVATE — [Ok :=] Form.ACTIVATE
Use this function to make a form or control active.

DECIMALPLACESMAX — [CurrMaxDecimals] := DECIMALPLACESMAX([NewMaxDecimals])
Use this function to return the current setting of the maximum number of decimal places for a control (field or text box), and to set a new value.

DECIMALPLACESMIN — [CurrMinDecimals] := DECIMALPLACESMIN([NewMinDecimals])
Use this function to return the current setting of the minimum number of decimal places for a control (field or text box), and to set a new value.

EDITABLE — [IsEditable] := From.EDITABLE([SetEditable])
Use this function to return the current setting of the Editable property, and to change the setting of the property.

ENABLED — [IsEnabled] := ENABLED([SetEnabled])
Use this function to return the current setting of the Enabled property of a control, and to change the setting of the property.

HEIGHT — [CurrHeight] := Form.HEIGHT([NewHeight])
Use this function to return the current setting of the Height property of a form or control, and to set this property to a new value.

INLINEEDITING — [IsInLineEditing] := INLINEEDITING([SetInLineEditing])
Use this function to return the current setting of the InLineEditing property of a table box or a matrix box, and to change the setting of the property.

UPDATEEDITABLE — UPDATEEDITABLE(Editable)
Use this function to dynamically change the setting of the Editable property of a field, form or control.

UPDATESELECTED — UPDATESELECTED(Selected)
Use this function to mark a control as selected (which will normally be displayed in reverse video, but this depends upon the Windows color scheme that the end user has chosen.)

UPDATEFONTBOLD — UPDATEFONTBOLD(FontBold)
Use this function to dynamically change the setting of the FontBold property of a control.

UPDATEFORECOLOR — UPDATEFORECOLOR(ForeColor)
Use this function to dynamically change the setting of the ForeColor property of a control.

UPDATEINDENT — UPDATEINDENT(Indent)
Use this function set the Indent property of a text box.

EVALUATE — [Ok :=] EVALUATE(Variable, String)
Use this function to evaluate a string representation of a value into its normal representation. The system assigns the result to a variable.

VARIANT

The C/AL variant data type can contain any variants from OCX and Automation objects (VT_VARIANT). The variant data type can contain the following C/AL data types: record, file, action, codeunit, Automation, boolean, option, integer, decimal, char, text, code, date, time, binary, DateFormula, TransactionType, InStream and OutStream.

DATI2VARIANT — Variant := DATI2VARIANT(Date, Time)
Use this system date function to create a variant that contains a VT_DATE.

ISACTION — Ok := Variant.ISACTION
Use this function to find out whether a C/AL variant contains an action variable or not.

ISAUTOMATION — Ok := Variant.ISAUTOMATION
Use this function to find out whether a C/AL variant contains an automation variable or not.

ISBINARY — Ok := Variant. ISBINARY
Use this function to find out whether a C/AL variant contains a binary variable or not.

ISBOOLEAN — Ok := Variant. ISBOOLEAN
Use this function to find out whether a C/AL variant contains a boolean variable or not.

ISCHAR — Ok := Variant. ISCHAR
Use this function to find out whether a C/AL variant contains a char variable or not.

ISCODE — Ok := Variant. ISCODE
Use this function to find out whether a C/AL variant contains a code variable or not.

ISCODEUNIT — Ok := Variant.ISCODEUNIT
Use this function to find out whether a C/AL variant contains a codeunit variable or not.

ISDATE — Ok := Variant. ISDATE
Use this function to find out whether a C/AL variant contains a date variable or not.

ISDATEFORMULA — Ok := Variant.ISDATEFORMULA
Use this function to find out whether a C/AL variant contains a dateformula variable or not.

ISDECIMAL — Ok := Variant. ISDECIMAL
Use this function to find out whether a C/AL variant contains a decimal variable or not.

ISFILE — Ok := Variant.ISFILE
Use this function to find out whether a C/AL variant contains a file variable or not.

ISINSTREAM — Ok := Variant.ISINSTREAM
Use this function to find out whether a C/AL variant contains an InStream variable or not.

ISINTEGER — Ok := Variant. ISINTEGER
Use this function to find out whether a C/AL variant contains an integer or not.

ISOPTION — Ok := Variant. ISOPTION
Use this function to find out whether a C/AL variant contains an option variable or not.

ISOUTSTREAM — Ok := Variant.ISOUTSTREAM
Use this function to find out whether a C/AL variant contains an OutStream variable or not.

ISRECORD — Ok := Variant.ISRECORD
Use this function to find out whether a C/AL variant contains a record variable or not.

ISTEXT — Ok := Variant. ISTEXT
Use this function to find out whether a C/AL variant contains a text variable or not.

ISTIME — Ok := Variant. ISTIME
Use this function to find out whether a C/AL variant contains a time variable or not.

ISTRANSACTIONTYPE — Ok := Variant.ISTRANSACTIONTYPE
Use this function to find out whether a C/AL variant contains a transactiontype variable or not.

VARIANT2DATE — Date := VARIANT2DATE(Variant)
Use this system date function to return a date from a VT_DATE variant.

VARIANT2TIME — Time := VARIANT2TIME(Variant)
Use this system date function to return a time from a VT_DATE variant.

GENERAL INFO

RGB Color Model

The system uses the RGB color model for specifying colors. This model specifies the intensity of red, green, and blue on a scale of 0 to 255, with 0 (zero) indicating the minimum intensity. The settings of the three colors are converted to a single integer value by using this formula: RGB value := Red + (Green*256) + (Blue*256*256)

Examples:

Red	Green	Blue	RGB value	Color
255	0	0	255	Red
0	255	0	65280	Green
0	0	255	16711680	Blue
0	255	255	16776960	Cyan
255	0	255	16711935	Magenta
255	255	0	65535	Yellow
255	255	255	16777215	White
128	128	128	8421504	Gray
0	0	0	0	Black

Virtual Tables

2000000001	Object
2000000002	User
2000000003	Member Of
2000000004	User Role
2000000005	Permission
2000000006	Company
2000000007	Date
2000000009	Session
2000000010	Database File
2000000020	Drive
2000000022	File
2000000024	Monitor
2000000026	Integer
2000000028	Table Information
2000000029	System Object
2000000037	Performance
2000000038	AllObj
2000000039	Printer
2000000040	License Information
2000000041	Field
2000000042	OLE Control
2000000043	License Permission
2000000044	Permission Range
2000000045	Windows Language
2000000046	Automation Servers
2000000049	Code Coverage
2000000050	Windows Object
2000000051	Navision Database Server
2000000052	Windows Group Member
2000000053	Windows Access Control
2000000054	Windows Login
2000000055	SID - Account ID
2000000056	User SID
2000000058	AllObjWithCaption
2000000059	Breakpoints
2000000203	Database Key Groups

Caption Classes

When you have set the CaptionClass property on a field or control, users can configure the caption of the field or control without having to modify code. C/SIDE passes the value of the CaptionClass property to the trigger with ID 15 on Codeunit 1, which translates the caption class to a caption that users can see.
C/SIDE calls this trigger with a language and a caption class. The trigger must convert the caption class into the specific caption for that language and return it as a string. Language is specified as an integer. Caption class is a text.

Data Types

ACTION
AUTOMATION
BIGINTEGER
BINARY
BLOB
BOOLEAN
CHAR
CODE
CODEUNIT
DATAPORT
DATE
DATEFORMULA
DATETIME
DECIMAL
DIALOG
DURATION
FIELDREF
FILE
FORM
GUID
INSTREAM
INTEGER
KEYREF
OCX
OPTION
OUTSTREAM
RECORD
RECORDID
RECORDREF
REPORT
TABLEFILTER
TEXT
TIME
TRANSACTIONTYPE
VARIANT

Repetitive Statements

```
FOR <control variable> := <start value> TO <end value> DO
  <statement>

FOR <control variable> := <start value> DOWNTO <end value> DO
  <statement>

WHILE <boolean expression> DO
  <statement>

REPEAT <statement>
  { <statement> }
UNTIL <boolean expression>
```

Conditional Statements

```
IF <boolean expression> THEN
  <statement 1>
[ELSE <statement 2>]

CASE <expression> OF
  <value set 1> : <statement 1>;
  …
  <value set n> : <statement n>;
  [ELSE <statement n+1>]
END;
```

mibuso.com

Microsoft Business Solutions online community

Quick Reference

Microsoft Business Solutions Navision 3.70

Table of contents:

Author: Luc Van Dyck
Website: http://www.mibuso.com