

VPME 9.1 Data Handler Methods

Key: (1) If a method had an equivalent in the prior VPME, the name of the equivalent method is shown in parentheses following the new method name; (2) If a parameter existed in the prior equivalent method, the sequence order of the parameter in the prior method is shown in parentheses following the parameter name.

- **AddRecord** (oApp.AddRecord) – Adds a record to or copies the current record in the current cursor or the cursor with the alias specified in the parameter stcAlias.
 - Parameters:
 - stcAlias (1) – The alias of the cursor to which the record will be added.
 - stlCopy (6) – Specifies whether the current record should be copied (.T.) or a new record should be created (.F.).
 - stcRulesID (5) – Passed on to business rules methods so that code in those methods can run conditionally if necessary. If this method is run from default form class code the contents of the form's cRulesID property is passed to this method.
 - stlNoRecycle (3) – Specifies whether a deleted record is recalled (.F.) or a record is appended (.T.).
 - stlNoDefaultValues (7) – Specifies whether or not the DefaultValues method should be run to fill in default values into the fields of the new record.
 - stlBlankPKOnCopy (12) – Specifies whether or not the PK field(s) of the copied record are blanked-out.
 - stlPreSaveDefaults (13) – Specifies whether the add-time default values or pre-save default values are placed in the fields.
 - stcUpdateAliases (14) – A comma-delimited list of aliases that specifies the additional cursors into which a new or copied record should be created.
 - stnParentRow (8) – Used by VPM forms to fill in FK field value(s) in a child record with the PK field value(s) from the current parent record.
 - stcChildFKFlds (9) – Used by VPM forms to fill in FK field value(s) in a child record with the PK field value(s) from the current parent record.
 - stcParentPKFlds (10) – Used by VPM forms to fill in FK field value(s) in a child record with the PK field value(s) from the current parent record.
 - stcParentAlias (11) – Used by VPM forms to fill in FK field value(s) in a child record with the PK field value(s) from the current parent record.
 - Returns:
 - .T. – The record was created.
 - .F. – The record was not created.
- **CalculateCheckSum** (oApp.oSec.FieldCheckSum) – Calculates a four-character check sum value for the string that is passed to this method. By default it is assumed that the string contains a check sum as the first four characters. Therefore, the check sum calculation starts with the fifth character in the string. The stnStartingPosition parameter can be used to specify an alternate calculation starting position.
 - Parameters:
 - stcString (1) – The string on which the calculation is performed.
 - stnStartingPosition – Specifies the starting position of the calculation.
 - Required Parameters:
 - stcString
 - Returns:
 - A four-character check sum string.
- **CloseViewSourceTables** (oApp.CloseViewSourceTables) – When a view is opened, VFP opens the view's source table(s). When a view cursor is closed, the source table(s) are left open. This method is used in conjunction with the OpenView method to close the view's source table(s) when the view cursor is closed.
 - Parameters:
 - stlInitialize (1) – When .T., causes the recording of open view source tables to commence. When .F., causes those tables to be closed.

- Properties:
 - aCloseViewSourceTables[<alias>]
 - lCloseViewSourceTables
 - nCloseViewSourceTables
 - Returns:
 - .T.
- **CreateErrorRecord** (oApp.CreateErrorRecord) – Logs an error in the Errors table.
 - Parameters:
 - stnErrorNumber – The error number.
 - stcErrorMessage – The error message.
 - stcClassOrProgram – The name of the class or program that contains the code where the error occurred.
 - stcMethodOrProcedure – The name of the class or procedure that contains the code where the error occurred.
 - stnSourceLineNumber – The line number of the line of code that caused the error to occur.
 - stcCode – The line of code that caused the error to occur.
 - Required Parameters:
 - stnErrorNumber
 - stcErrorMessage
 - stcClassOrProgram
 - stcMethodOrProcedure
 - stnSourceLineNumber
 - stcCode
 - Returns:
 - .T. – The record was created.
 - .F. – The record was not created.
- **CursorAdapterIdentification** – Various processes in VPM require that data dictionary information be retrieved for cursor adapter cursors. This method saves/retrieves the information that identifies the table, view, or cursor adapter whose data dictionary information should be used for a particular cursor adapter cursor.
 - Parameters:
 - stcAlias – The alias of the cursor adapter cursor for which the information is saved/retrieved.
 - stcCAIdentification – The information that is saved is parsed from the string passed in this parameter. If this parameter is empty the information is retrieved. Otherwise the information in this parameter is saved. Format of string in this parameter: <DBC or Cursor Adapter Library name>!<Table, View, or Cursor Adapter name>
 - Required Parameters:
 - stcAlias
 - Properties:
 - cCAID_Alias
 - cCAID_DDD_Name
 - cCAID_DDTV_Name
 - Returns:
 - .T. – The information was saved, or the information was retrieved and loaded into the properties.
 - .F. – The information was not saved, or the information was not retrieved and the properties contain default (empty) values.
- **CursorIdentification** – Various processes in VPM require that data dictionary information be retrieved for cursors. This method identifies the information necessary to find data dictionary records for a cursor. For cursor adapter cursors this information cannot be determined directly from the cursor itself and is therefore determined through a subsequent call to the CursorAdapterIdentification method.
 - Parameters:

- stcAlias – The alias of the cursor for which the information is retrieved.
 - Required Parameters:
 - stcAlias
 - Properties:
 - cCursorID_DDD_Name
 - cCursorID_DDTV_Name
 - nCursorID_SourceType
 - Returns:
 - .T. – The properties were loaded and contain values.
 - .F. – The properties contain default (empty) values.
- **DataChanged** (oApp.DataChanged) – Determines if a change has been made to the current record of the cursor open in the current work area or the cursor specified in the parameter stcAlias. Also checks for changes made to any of the cursors specified in the parameter stcAdditionalAliases. For any of the cursors specified in stcAdditionalAliases that use table buffering, determines if there are any modified or deleted records in the cursor. Otherwise, just determines if the current record has been modified.
 - Parameters:
 - stcAlias (1) – The alias of the cursor for which the determination is made.
 - stcAdditionalAliases (3) – A comma-delimited list of aliases that specifies the additional cursors to be included in the determination.
 - Returns:
 - .T. – A change has been made.
 - .F. – A change has not been made.
- **DecryptString** (oApp.oSec.StringDecrypt) – Decrypts the string passed to this method.
 - Parameters:
 - stcString (1) – The string to be decrypted.
 - stcEncryptionCharacter (2) – Specifies an alternate character to be used in the decryption algorithm.
 - Required Parameters:
 - stcString
 - Returns:
 - A string of the same length as the string that was passed to this method.
- **DefaultValues** (oApp.DefaultValues) – Places default values into fields in the current record of the cursor open in the current work area or the cursor specified in the parameter stcAlias. Default values are determined by evaluating the Default expression stored for each field in the SDATADDF table. If this method is run for a copied record, the Default expression is evaluated only for primary key fields.
 - Parameters:
 - stcAlias (1) – The alias of the cursor that contains the record into which the default values are placed.
 - stlPreSaveDefaults (9) – Specifies whether the add-time default values or pre-save default values are placed in the fields.
 - stcRulesID – Passed on to business rules methods so that code in those methods can run conditionally if necessary. If this method is run from default form class code the contents of the form's cRulesID property is passed to this method.
 - stlCopy (5) – Specifies whether or not the default values are being placed in a new record or a copied record.
 - stlBlankPKOnCopy (8) – Specifies whether or not the PK field(s) of the copied record are blanked-out.
 - stnParentRow (3) – Used by VPM forms to fill in FK field value(s) in a child record with the PK field value(s) from the current parent record.
 - stcChildFKFlds (4) – Used by VPM forms to fill in FK field value(s) in a child record with the PK field value(s) from the current parent record.
 - stcParentPKFlds (6) – Used by VPM forms to fill in FK field value(s) in a child record with the PK field value(s) from the current parent record.

- stcParentAlias (7) – Used by VPM forms to fill in FK field value(s) in a child record with the PK field value(s) from the current parent record.
 - Returns:
 - .T. – Default values were placed in the fields.
 - .F. – Default values were not placed in the fields.
- **DeleteFor** (oApp.DeleteFor) – Deletes records in the cursor specified by the parameter stcAlias using the expression specified in the parameter stcExpression. If the parameter stcAlias is empty and a cursor is open in the current work area, the records in that cursor are deleted. The deletions are performed using the DELETE FOR command and the TABLEUPDATE function instead of using the DeleteRecord method, which means the records are deleted without the PreDelete/PostDelete code being run, without the data handler's RI routine being run, and without the Audit Trail functionality being run.
 - Parameters:
 - stcAlias (1) – The alias of the cursor that contains the records to be deleted.
 - stcExpression (3) – The expression used to select the records to be deleted.
 - Required Parameters:
 - stcExpression
 - Returns:
 - .T. – The records were deleted.
 - .F. – The records were not deleted.
- **DeleteRecord** (oApp.DeleteRecord) – Deletes the current record in the cursor specified in the parameter stcAlias. If the parameter stcAlias is empty and a cursor is open in the current work area, the current record in that cursor is deleted.
 - Parameters:
 - stcAlias (1) – The alias of the cursor that contains the record to be deleted.
 - stcRulesID (7) – Passed on to business rules methods so that code in those methods can run conditionally if necessary. If this method is run from default form class code the contents of the form's cRulesID property is passed to this method.
 - stnGoToRecNo (3) – The record number of the record where the record pointer is positioned after the deletion is completed. If this parameter is not used the record pointer is positioned on the record following the deleted record according to the master tag.
 - stlNoAuditTrail (5) – Specifies whether or not Audit Trail records are created for the deletion.
 - stlNoRICall – Specifies whether or not the RI routine is called. If not, both the RI check and the Audit Trail functionality are disabled.
 - stlNoRICheck (4) – Specifies whether or not RI check is performed.
 - Properties:
 - cRIMessage – Contains error information when 2, 5, or 7 value is returned.
 - Returns:
 - 0 – Deletion completed.
 - 1 – PreDelete method or code in mPreDeleteCode returned .F., deletion aborted.
 - 2 – Failed VPM RI check, deletion aborted.
 - 3 – Delete trigger failed, deletion aborted.
 - 4 – TABLEUPDATE failed, deletion aborted.
 - 5 – Error occurred, deletion aborted.
 - 6 – Transaction failed, deletion aborted.
 - 7 – TABLEUPDATE failed, deletion aborted (ODBC error occurred).
 - 8 – PostDelete method or code in mPostDeleteCode returned .F., deletion aborted.
 - 9 – PreDeletePreTransaction method or code in mPreDeletePreTransactionCode returned .F., deletion aborted.

- **EmptyRequiredFields** – Determines if any of the required fields in the current record of the current cursor or the cursor with the alias specified in the parameter stcAlias are empty or contain a null value. Required fields are those that have the Integrity Option set to “On”. This method returns the number of required fields that do not contain a value. Those required fields that do not contain a value are loaded into the array aEmptyRequiredFields.
 - Parameters:
 - stcAlias – The alias of the cursor that contains the record for which the check is performed.
 - stcNotBlankFields – If only specific fields should be checked, a comma-delimited list of fields can be passed in this parameter. If there are fields from more than one cursor in the list, the field names should contain the alias (<alias>.<field name>) so that only those fields in the current alias (stcAlias) will be checked.
 - stlCheckNoAlias – If one or more of the fields in the stcNotBlankFields list do not contain the alias, this parameter should be used to specify when those fields should be checked.
 - Properties:
 - aEmptyRequiredFields[<field name>, <field description>]
 - Returns:
 - The number of required fields that do not contain a value (corresponds to number of rows in aEmptyRequiredFields).

- **EncryptString** (oApp.oSec.StringEncrypt) – Encrypts the string passed to this method.
 - Parameters:
 - stcString (1) – The string to be encrypted.
 - stcEncryptionCharacter (2) – Specifies an alternate character to be used in the encryption algorithm.
 - Required Parameters:
 - stcString
 - Returns:
 - A string of the same length as the string that was passed to this method.

- **EvaluateExpression** (oApp.EvaluateExpression) – Selects the alias or work area specified by the parameter stAliasOrWorkArea and then evaluates the expression specified in the parameter stcExpression. If the parameter stAliasOrWorkArea is empty and a cursor is open in the current work area, the expression is evaluated with the current work area still selected.
 - Parameters:
 - stAliasOrWorkArea (1) – The alias or work area to be selected before the expression in stcExpression is evaluated.
 - stcExpression (3) – The expression that is evaluated.
 - Required Parameters:
 - stcExpression
 - Returns:
 - A value that is the result of the evaluation of the expression.

- **FieldValidation** – Validates the value in the stValue parameter against the validation rule defined in the data dictionary for a field that is specified in the stcDDDName, stcDDTVName, and stcDDFName parameters.
 - Parameters:
 - stcDDDName – Along with the following two parameters identifies the field for which the value in stValue is validated
 - stcDDTVName – See above for description.
 - stcDDFName – See above for description.
 - stValue – The value being validated.
 - stcRulesID – Passed on to business rules methods so that code in those methods can run conditionally if necessary. If this method is run from default control class code the contents of the control's cValidRulesID property is passed to this method.
 - stcRefFilterExpr – The filter expression to be applied to the table that is used to validate the value.
 - staRefFilterVariables – The filter variables and values to be used with the view that is used to validate the value.
 - stlRecordLevel – If .T., the field is being validated prior to a record being saved.
 - Required Parameters:
 - stcDDDName
 - stcDDTVName
 - stcDDFName
 - stValue
 - Properties:
 - cFieldValidationReturnMessage
 - Returns:
 - .T. – The value passed the validation check.
 - 0 (zero) – The value failed the validation check.
 - The value returned by the developer's code.
- **FieldValidation_RecordLevel** – Called from the SaveRecord method. Calls the FieldValidation method to determine if any of the fields in the current record of the current cursor or the cursor with the alias specified in the parameter stcAlias contain an invalid value. This method returns the number of fields that contain an invalid value. Information about the fields that contain an invalid value is loaded into the array aRecordLevelInvalidFields.
 - Parameters:
 - stcAlias – The alias of the cursor that contains the record for which the field values are validated.
 - stcRulesID – Passed on to business rules methods so that code in those methods can run conditionally if necessary. If this method is run from default form class code the contents of the form's cRulesID property is passed to this method.
 - Properties:
 - aRecordLevelInvalidFields [<field name>, <field description>, <validation message>]
 - Returns:
 - The number of fields that contain an invalid value (corresponds to number of rows in aRecordLevelInvalidFields).
- **GenerateGUID** – Generates a 16-character binary GUID (Globally Unique ID). Since this 16-character string can include illegal XML characters that cannot be transferred, if necessary, via XML, the 3-digit numeric ANSI value for each of the 16 characters is used to create a 48-character (digit) GUID of valid XML characters. It is this 48-character GUID that is returned by default.
 - Parameters:
 - stlShortGUID – If .T., causes the 16-character GUID to be returned instead of the 48-character GUID.
 - Returns:
 - A 16- or 48-character string.

- **GenerateKey** (S<Prefix>DGK1.PRG) – Generates a character or numeric value to be placed in a field. Normally, this method is run to generate a value to be placed in a key field of a new record. The last used key field values are stored in the table SDATADDGK and are accessed using the view SDATA_V!DDGK_V.
 - Parameters:
 - stcFieldPK (1) – The string used to find the field's record in the Generated Key table.
 - stlGenerateNumeric (2) – The last value generated is stored in a character field in the Generated Key table. However, the value returned by this method does not have to be stored in a character field. If this parameter is .T., the value returned by this method will be numeric, which can be stored in a numeric, integer, float, double, or currency field. Otherwise, the value returned by this method will be of character type, which can be stored in a character or memo field.
 - stcDDDName – Along with the following two parameters identifies the field for which the key is being generated. If this is the first time a key is being generated for the field, these parameter values are placed in the cDescription field of the SDATADDGK record that is created.
 - stcDDTVName – See above for description.
 - stcDDFName – See above for description.
 - Required Parameters:
 - stcFieldPK
 - Returns:
 - A value to be placed in the field.
 - .F. – A value for the field cannot be generated.

- **GenerateKey_Increment** (S<Prefix>DGK2.PRG) – Increments the character value passed to this method by the GenerateKey method and returns that incremented character value.
 - Parameters:
 - stcLastUsed (1) – The value that this method is to increment.
 - Returns:
 - A character string representing the incremented value.

- **GetAlias** – Returns the alias of the cursor open in the current work area.
 - Returns:
 - The alias of the cursor that is open in the current work area or the null string if a cursor is not open in the current work area.

- **GetDDCK** – Retrieves into a view cursor the SDATADDCK (Candidate Keys) records for the table, view, or cursor adapter specified in the parameters.
 - Parameters:
 - stcDDDName – Along with the following parameter identifies the table, view, or cursor adapter for which the records are retrieved.
 - stcDDTVName – See above for description.
 - stlFindLocal – If the stcDDDName and stcDDTVName properties identify a remote view and that remote view is not in the data dictionary, this parameter specifies whether or not the data dictionary information for the related local view should be used for the remote view.
 - Required Parameters:
 - stcDDDName
 - stcDDTVName
 - Returns:
 - .T. – The view cursor was created.
 - .F. – The view cursor was not created.

- **GetDDD** – Retrieves into an object the SDATADDD (Databases and Cursor Adapter Libraries) record for the database or cursor adapter library specified in the parameters.
 - Parameters:

- stcDDDName – Identifies the database or cursor adapter library for which the record is retrieved.
 - stcPKValue – The PK value of the record to be retrieved.
 - Required Parameters:
 - stcDDDName; or
 - stcPKValue
 - Properties:
 - oDDDRRecord
 - Returns:
 - .T. – The object was created.
 - .F. – The object was not created.
- **GetDDF** (oApp.GetDD2Record) – Retrieves into an object the SDATADDF (Fields) record for the field specified in the parameters.
 - Parameters:
 - stcDDDName – Along with the following two parameters identifies the field for which the record is retrieved.
 - stcDDTVName (1) – See above for description.
 - stcDDFName (2) – See above for description.
 - stlFindLocal – If the stcDDDName and stcDDTVName properties identify a remote view and that remote view is not in the data dictionary, this parameter specifies whether or not the data dictionary information for the related local view should be used for the remote view.
 - stcPKValue – The PK value of the record to be retrieved.
 - stcParentPKValue – Along with the stcDDFName parameter identifies the field for which the record is retrieved. This parameter (the PK value of an SDATADDTV record) identifies the table, view, or cursor adapter that contains the field identified in the stcDDFName parameter.
 - Required Parameters:
 - stcDDDName, stcDDTVName, and stcDDFName; or
 - stcPKValue; or
 - stcParentPKValue and stcDDFName
 - Properties:
 - oDDFRecord
 - Returns:
 - .T. – The object was created.
 - .F. – The object was not created.
- **GetDDF_All** – Retrieves into a view cursor the SDATADDF (Fields) records for the table, view, or cursor adapter specified in the parameters.
 - Parameters:
 - stcDDDName – Along with the following parameter identifies the table, view, or cursor adapter for which the records are retrieved.
 - stcDDTVName – See above for description.
 - stlFindLocal – If the stcDDDName and stcDDTVName properties identify a remote view and that remote view is not in the data dictionary, this parameter specifies whether or not the data dictionary information for the related local view should be used for the remote view.
 - stcParentPKValue – This parameter (the PK value of an SDATADDTV record) identifies the table, view, or cursor adapter for which the records are retrieved.
 - Required Parameters:
 - stcDDDName and stcDDTVName; or
 - stcParentPKValue
 - Returns:
 - .T. – The view cursor was created and contains at least one record.
 - .F. – The view cursor was not created or does not contain any records.

- **GetDDFH** – Retrieves into an object the SDATADDFH (Field Help) record for the field specified in the parameters.
 - Parameters:
 - stcDDDName – Along with the following two parameters identifies the field for which the record is retrieved.
 - stcDDTVName – See above for description.
 - stcDDFName – See above for description.
 - Required Parameters:
 - stcDDDName
 - stcDDTVName
 - stcDDFName
 - Properties:
 - oDDFHRecord
 - Returns:
 - .T. – The object was created.
 - .F. – The object was not created.

- **GetDDFP** – Retrieves into an object the SVPMDDFP (Field Picklists) record for the field specified in the parameters.
 - Parameters:
 - stcDDDName – Along with the following two parameters identifies the field for which the record is retrieved.
 - stcDDTVName – See above for description.
 - stcDDFName – See above for description.
 - stcID – Identifies the specific Picklist definition whose record is to be retrieved for the field. If empty, the default Picklist definition record is retrieved for the field. The parameter value corresponds to the value in the cID field in the SVPMDDFP record to be retrieved. If a Picklist is being brought up for control, the parameter value also corresponds to the value in the cPicklistID property of the control for which the Picklist is being brought up.
 - Required Parameters:
 - stcDDDName
 - stcDDTVName
 - stcDDFName
 - Properties:
 - oDDFPRecord
 - Returns:
 - .T. – The object was created.
 - .F. – The object was not created.

- **GetDDGK** – Retrieves into an object the SDATADDGK (Generated Keys) record for the field specified in the parameters.
 - Parameters:
 - stcDDDName – Along with the following two parameters identifies the field for which the record is retrieved.
 - stcDDTVName – See above for description.
 - stcDDFName – See above for description.
 - Required Parameters:
 - stcDDDName
 - stcDDTVName
 - stcDDFName
 - Properties:
 - oDDGKRecord
 - Returns:
 - .T. – The object was created.
 - .F. – The object was not created.

- **GetDDIT** (oApp.GetDIndRecord) – Retrieves into an object the SDATADDIT (Index Tags) record for the tag specified in the parameters.
 - Parameters:
 - stcDDDName – Along with the following two parameters identifies the tag for which the record is retrieved.
 - stcDDTVName (1) – See above for description.
 - stcTag (2) – See above for description.
 - stlFindLocal – If the stcDDDName and stcDDTVName properties identify a remote view and that remote view is not in the data dictionary, this parameter specifies whether or not the data dictionary information for the related local view should be used for the remote view.
 - stcPKValue – The PK value of the record to be retrieved.
 - stcParentPKValue – Along with the stcTag parameter identifies the tag for which the record is retrieved. This parameter (the PK value of an SDATADDTV record) identifies the table, view, or cursor adapter whose tag is identified in the stcTag parameter.
 - Required Parameters:
 - stcDDDName, stcDDTVName, and stcTag; or
 - stcPKValue; or
 - stcParentPKValue and stcTag
 - Properties:
 - oDDITRecord
 - Returns:
 - .T. – The object was created.
 - .F. – The object was not created.
- **GetDDIT_All** – Retrieves into a view cursor the SDATADDIT (Index Tags) records for the table, view, or cursor adapter specified in the parameters.
 - Parameters:
 - stcDDDName – Along with the following parameter identifies the table, view, or cursor adapter for which the records are retrieved.
 - stcDDTVName – See above for description.
 - stlFindLocal – If the stcDDDName and stcDDTVName properties identify a remote view and that remote view is not in the data dictionary, this parameter specifies whether or not the data dictionary information for the related local view should be used for the remote view.
 - stcParentPKValue – This parameter (the PK value of an SDATADDTV record) identifies the table, view, or cursor adapter for which the records are retrieved.
 - Required Parameters:
 - stcDDDName and stcDDTVName; or
 - stcParentPKValue
 - Returns:
 - .T. – The view cursor was created and contains at least one record.
 - .F. – The view cursor was not created or does not contain any records
- **GetDDIT_PK** – Retrieves into an object the SDATADDIT (Index Tags) record for the PK tag of the table, view, or cursor adapter specified in the parameters.
 - Parameters:
 - stcDDDName – Along with the following parameter identifies the table, view, or cursor adapter for which the PK tag record is retrieved.
 - stcDDTVName – See above for description.
 - stlFindLocal – If the stcDDDName and stcDDTVName properties identify a remote view and that remote view is not in the data dictionary, this parameter specifies whether or not the data dictionary information for the related local view should be used for the remote view.
 - stcParentPKValue – This parameter (the PK value of an SDATADDTV record) identifies the table, view, or cursor adapter for which the PK tag record is retrieved.
 - Required Parameters:

- stcDDDName and stcDDTVName; or
 - stcParentPKValue
 - Properties:
 - oDDITRecord
 - Returns:
 - .T. – The object was created.
 - .F. – The object was not created.
- **GetDDIT_TagType** – Retrieves into an object the SDATADDIT (Index Tags) record for the type of tag specified in the parameter stnTagType for the table specified in the other parameters.
 - Parameters:
 - stcDDDName – Along with the following parameter identifies the table for which the record is retrieved.
 - stcDDTVName – See above for description.
 - stcParentPKValue – This parameter (the PK value of an SDATADDTV record) identifies the table for which the record is retrieved.
 - stnTagType – 1 (Candidate), 2 (VFP Primary), 3 (Unique)
 - Required Parameters:
 - stcDDDName, stcDDTVName, and stnTagType; or
 - stcParentPKValue and stnTagType
 - Properties:
 - oDDITRecord
 - Returns:
 - .T. – The object was created.
 - .F. – The object was not created.
- **GetDDSI** – Retrieves into a view cursor the SDATADDSI (Set Integrity) records for the field specified in the parameters.
 - Parameters:
 - stcDDDName – Along with the following two parameters identifies the field for which the records are retrieved.
 - stcDDTVName – See above for description.
 - stcDDFName – See above for description.
 - stlFindLocal – If the stcDDDName and stcDDTVName properties identify a remote view and that remote view is not in the data dictionary, this parameter specifies whether or not the data dictionary information for the related local view should be used for the remote view.
 - stcParentPKValue – This parameter (the PK value of an SDATADDF record) identifies the field for which the records are retrieved.
 - Required Parameters:
 - stcDDDName, stcDDTVName, and stcDDFName; or
 - stcParentPKValue
 - Returns:
 - .T. – The view cursor was created and contains at least one record.
 - .F. – The view cursor was not created or does not contain any records.

- **GetDDTV** (oApp.GetDD1Record) – Retrieves into an object the SDATADDTV (Tables, Views, and Cursor Adapters) record for the table, view, or cursor adapter specified in the parameters.
 - Parameters:
 - stcDDDName – Along with the following parameter identifies the table, view, or cursor adapter for which the record is retrieved.
 - stcDDTVName (1) – See above for description.
 - stlFindLocal – If the stcDDDName and stcDDTVName properties identify a remote view and that remote view is not in the data dictionary, this parameter specifies whether or not the data dictionary information for the related local view should be used for the remote view.
 - stcPKValue – The PK value of the record to be retrieved.
 - Required Parameters:
 - stcDDDName and stcDDTVName; or
 - stcPKValue
 - Properties:
 - oDDTVRecord
 - Returns:
 - .T. – The object was created.
 - .F. – The object was not created.

- **GetFilter** (oApp.GetFilter) – Returns the filter expression that is in use with the current cursor or the cursor with the alias specified in the parameter stcAlias.
 - Parameters:
 - stcAlias (1) – The alias of the cursor for which the current filter expression is returned.
 - Returns:
 - The current filter expression.

- **GetOrder** – Returns the name of the master (controlling) index tag for the current cursor or the cursor with the alias specified in the parameter stcAlias.
 - Parameters:
 - stcAlias – The alias of the cursor for which the current master tag name is returned.
 - Returns:
 - The name of the master (controlling) tag if a tag is in use.
 - The null string if no tag is in use.

- **GetRecNo** (oApp.GetRecordNumber) – Returns the record pointer position (record number) of the current cursor or the cursor with the alias specified in the parameter stcAlias.
 - Parameters:
 - stcAlias (1) – The alias of the cursor for which the current record pointer position is returned.
 - Returns:
 - The current record number if the record pointer is not at end-of-file.
 - The number 0 (zero) if the record pointer is at end-of-file.

- **GoEOF** (oApp.GoEOF) – Moves the record pointer to end-of-file in the current cursor or the cursor with the alias specified in the parameter stcAlias.
 - Parameters:
 - stcAlias (1) – The alias of the cursor in which the record pointer is moved.
 - Returns:
 - .T.

- **GoFirst** (oApp.First) – Moves the record pointer to the first record in the current cursor or the cursor with the alias specified in the parameter stcAlias. The pointer is moved according to the master tag.
 - Parameters:
 - stcAlias (1) – The alias of the cursor in which the record pointer is moved.
 - Returns:

- The record number of the selected (first) record.
 - The number 0 (zero) if no records are in the cursor or the alias specified in the parameter stcAlias not in use.
- **GoLast** (oApp.Last) – Moves the record pointer to the last record in the current cursor or the cursor with the alias specified in the parameter stcAlias. The pointer is moved according to the master tag.
 - Parameters:
 - stcAlias (1) – The alias of the cursor in which the record pointer is moved.
 - Returns:
 - The record number of the selected (last) record.
 - The number 0 (zero) if no records are in the cursor or the alias specified in the parameter stcAlias not in use.
- **GoNext** (oApp.Next) – Moves the record pointer in the current cursor or the cursor specified in the parameter stcAlias. The pointer is moved according to the master tag.
 - Parameters:
 - stcAlias (2) – The alias of the cursor in which the record pointer is moved.
 - stnRecs (1) – The record pointer is moved the number of records specified in this parameter. If a positive number, the record pointer is moved forward. If a negative number, the record pointer is moved backward. If empty, the record pointer is positioned one record forward.
 - Returns:
 - 0 – Not EOF() or BOF()
 - 1 – EOF()
 - 2 – BOF()
 - 3 – EOF() and BOF()
- **GoRecNo** – Moves the record pointer to the record specified in the parameter stnRecNo in the current cursor or the cursor with the alias specified in the parameter stcAlias.
 - Parameters:
 - stcAlias – The alias of the cursor in which the record pointer is moved.
 - stnRecNo – The record number where the record pointer should be positioned. If 0 (zero), the record pointer is moved to end-of-file.
 - Required Parameters:
 - stnRecNo
 - Returns:
 - .T. – The record pointer was moved to the desired record.
 - .F. – The record pointer was not moved to the desired record
- **IsBOF** (oApp.IsBOF) – Determines whether or not the record pointer has been attempted to be moved above the first record in the current cursor or the cursor with the alias specified in the parameter stcAlias. When this happens, the record pointer remains on the first record and this method returns .T..
 - Parameters:
 - stcAlias (1) – The alias of the cursor for which the determination is made.
 - Returns:
 - .T. – BOF() returned .T.
 - .F. – BOF() returned .F.
- **IsEOF** (oApp.IsEOF) – Determines whether or not the record pointer is at end-of-file for the current cursor or the cursor with the alias specified in the parameter stcAlias.
 - Parameters:
 - stcAlias (1) – The alias of the cursor for which the determination is made.
 - Returns:
 - .T. – The record pointer is at end-of-file.
 - .F. – The record pointer is not at end-of-file.

- **IsNew** (oApp.IsNew) – Determines whether the current record in the current cursor or the cursor with the alias specified in the parameter stcAlias is new (has been created but not yet saved).
 - Parameters:
 - stcAlias (2) – The alias of the cursor for which the determination is made.
 - stlAndEmpty (1) – If .T., the determination is made as to whether the record is new and no changes have been made to the new record.
 - Properties:
 - lIsNew_Recycled – Set to .T. if the record is new and recycled.
 - Returns:
 - .T. – The record is new but changed (stlAndEmpty = .F.) or is new and unchanged (stlAndEmpty = .T. or .F.).
 - .F. – The record is new but changed (stlAndEmpty = .T.) or is not new (stlAndEmpty = .T. or .F.).

- **IsUsed** – Determines whether or not a cursor is open in the current work area, or whether or not the cursor with the alias specified in the parameter stcAlias is open.
 - Parameters:
 - stcAlias – The alias of the cursor for which the determination is made.
 - Returns:
 - .T. – A cursor is open in the current work area (if stcAlias is empty) or a specific cursor is open in any work area (if alias is specified in stcAlias).
 - .F. – A cursor is not open in the current work area (if stcAlias is empty) or a specific cursor is not open in any work area (if alias is specified in stcAlias).

- **LocateFor** (oApp.Locate_For) – Performs a LOCATE with a FOR expression to find a selected record or records.
 - Parameters:
 - stcAlias (4) – The alias of the cursor in which the LOCATE is performed.
 - stcLocateExpr (1) – The expression used to find the record(s).
 - stlUseCurrentTag (2) – If .F. (the default), and the master tag does not have a FOR expression, performs the LOCATE without a tag in use to speed up the LOCATE.
 - stlContinue (3) – If .T., performs a CONTINUE after the initial LOCATE. If the CONTINUE is successful (a record is found), the record number of that record is placed in the property nContinueRec and the record pointer is moved back to the initially located record. This parameter can be used to determine if more than one record matches the search criteria.
 - Properties:
 - nContinueRec – Contains a record number if CONTINUE was successful, 0 (zero) otherwise.
 - Returns:
 - .T. – A record was found.
 - .F. – A record was not found.

- **OpenCursorAdapter** – Opens a cursor adapter cursor. The cursor adapter object that is created must be kept in memory (not released) until the cursor is no longer needed. Therefore, the object cannot be released within this method, and will be stored in the array aCursorAdapterObjects.
 - Parameters:
 - stcClassName – The CursorAdapter class name from which the cursor adapter object is instantiated.
 - stcModule – The name of the VCX file or program that contains the class specified in stcClassName (can include path).
 - stcInApplication – The name of the VFP application (EXE or APP) containing the VCX file specified in stcModule (extension must be included).
 - stcAlias – Used to override the Alias property value of the cursor adapter object.
 - stlNoData – Passed to the cursor adapter object's CursorFill method.
 - stcCAIdentification – Identifies the table, view, or cursor adapter whose information in the data dictionary should be used for the cursor adapter cursor being created. The table, view, or cursor adapter is identified as a string formatted as: <DBC or Cursor Adapter Library name>!<Table, View, or Cursor Adapter name>. If this parameter is not passed, the cursor adapter itself is assumed to be in the data dictionary and is identified as stcModule+'!'+stcClassName. See the CursorAdapterIdentification method for more information.
 - stlNoCursorSchema – If .T., .F. is passed to the cursor adapter object's CursorFill method.
 - stOptions – Passed to the cursor adapter object's CursorFill method.
 - stSource – Passed to the cursor adapter object's CursorFill method.
 - stParameter1 through stParameter10 – The parameters are passed to the Init event of the cursor adapter object when the object is instantiated.
 - Required Parameters:
 - stcClassName
 - Properties:
 - aCursorAdapterObjects[<alias>, <object reference>]
 - Returns:
 - .T. – The cursor was created.
 - .F. – The cursor was not created.
- **OpenDatabase** (oApp.OpenDBC) – Opens the database specified in the parameter stcDatabase. If stcDatabase is empty, all of the databases listed in the SDATADDD table are opened.
 - Parameters:
 - stcDatabase (1) – The name of the database to be opened.
 - Returns:
 - .T. – The database was opened.
 - .F. – The database was not opened.
- **OpenTable** (oApp.OpenTable) – Opens the table specified in the parameter stcTable.
 - Parameters:
 - stcTable (1) – The name of the table to open. The path and/or extension can be included. If empty (null, blank, .F., or no parameter), a SELECT 0 is issued (an unused work area is selected).
 - stcAlias (2) – The alias to be assigned to the cursor when the table is opened. If empty (null, blank, .F., or no parameter), the alias will be the same as the table name.
 - stlAgain (3) – Determines whether or not the AGAIN clause is included in the USE command that is used to open the table. If empty (null, blank, .F., or no parameter), the AGAIN clause is not included. Otherwise, the AGAIN clause is included.
 - stlExclusive (4) – Determines whether or not the EXCLUSIVE clause is included in the USE command that is used to open the table. If empty (null, blank, .F., or no parameter), the EXCLUSIVE clause is not included. Otherwise, the EXCLUSIVE clause is included.
 - stlNoUpdate (5) – Determines whether or not the NOUPDATE clause is included in the USE command that is used to open the table. If empty (null, blank, .F., or no parameter), the NOUPDATE clause is not included. Otherwise, the NOUPDATE clause is included.

- Returns:
 - .T. – The table was opened.
 - .F. – The table was not opened.
- **OpenView** (oApp.OpenView) – Opens a view after first defining the variables that are used in the WHERE clause (filter) of the view's SELECT statement. Filter variables that are not used in the current situation (not defined in the parameter staFilterVariables) will be set to values that will cause those variables to not filter out any records. This allows a single view to be used in any number of situations that only require the use of a subset of the filter variables.
 - Parameters:
 - stcView (1) – The name of the view to open.
 - stcDBC (2) – The name of the database that contains the view to open. The path and/or extension can be included.
 - staFilterVariables – The reference to an array that contains the definition of view filter variables that will be set before the view cursor is created, causing records to be excluded from the cursor. If empty (null, blank, zero, .F., or no parameter), no filter variables are defined. Otherwise, a reference to a two-dimensional array that defines the variables used in the WHERE clause of the view's SELECT statement. Prior to the view being opened the variables listed in the array are defined using the name from column 1 and the value from column 2.
 - stcAlias (3) – The alias to be assigned to the cursor when the view is opened. If empty (null, blank, .F., or no parameter), the alias will be the same as the view name.
 - stlAdmin (4) – Determines whether or not the ADMIN clause is included in the USE command that is used to open the view. If empty (null, blank, .F., or no parameter), the ADMIN clause is not included. Otherwise, the ADMIN clause is included.
 - stlExclusive (5) – Determines whether or not the EXCLUSIVE clause is included in the USE command that is used to open the view. If empty (null, blank, .F., or no parameter), the EXCLUSIVE clause is not included. Otherwise, the EXCLUSIVE clause is included.
 - stlNoData (6) – Determines whether or not the NODATA clause is included in the USE command that is used to open the view. If empty (null, blank, .F., or no parameter), the NODATA clause is not included. Otherwise, the NODATA clause is included.
 - stlNoRequery (7) – Determines whether or not the NOREQUERY clause is included in the USE command that is used to open the view. If empty (null, blank, .F., or no parameter), the NOREQUERY clause is not included. Otherwise, the NOREQUERY clause is included.
 - stlNoUpdate (8) – Determines whether or not the NOUPDATE clause is included in the USE command that is used to open the view. If empty (null, blank, .F., or no parameter), the NOUPDATE clause is not included. Otherwise, the NOUPDATE clause is included.
 - stlOnline (9) – Determines whether or not the ONLINE clause is included in the USE command that is used to open the view. If empty (null, blank, .F., or no parameter), the ONLINE clause is not included. Otherwise, the ONLINE clause is included.
 - Required Parameters:
 - stcView
 - stcDBC
 - Returns:
 - .T. – The cursor was created.
 - .F. – The cursor was not created.
- **ReleaseCursorAdapterObject** – Releases the cursor adapter object for the cursor with the alias specified in the parameter stcAlias. The object that is released is stored in the array aCursorAdapterObjects by the OpenCursorAdapter method.
 - Parameters:
 - stcAlias – The alias of the cursor for which the object is to be released.
 - Required Parameters:
 - stcAlias
 - Returns:
 - .T.

- **RequeryView** – Performs a REQUERY() on a view cursor after first defining the variables that are used in the WHERE clause (filter) of the view's SELECT statement. Filter variables that are not used in the current situation (not defined in the parameter staFilterVariables) will be set to values that will cause those variables to not filter out any records. This allows a single view to be used in any number of situations that only require the use of a subset of the filter variables.
 - Parameters:
 - stcAlias – The alias of the view cursor for which a REQUERY() will be performed.
 - staFilterVariables – The reference to an array that contains the definition of view filter variables that will be set before the REQUERY() is performed, causing records to be excluded from the cursor. If empty (null, blank, zero, .F., or no parameter), no filter variables are defined. Otherwise, a reference to a two-dimensional array that defines the variables used in the WHERE clause of the view's SELECT statement. Prior to the REQUERY() being performed the variables listed in the array are defined using the name from column 1 and the value from column 2.
 - Returns:
 - .T. – Data was retrieved successfully.
 - .F. – Data was not retrieved.
- **RestoreRecord** (oApp.Restore) – Cancels the changes to the current record in the cursor specified in the parameter stcAlias. If the parameter stcAlias is empty and a cursor is open in the current work area, the changes to the current record in that cursor are cancelled.
 - Parameters:
 - stcAlias (1) – The alias of the cursor in which the changes will be cancelled.
 - stlRestoreAll – If .T., the changes to all modified records in the cursor are cancelled.
 - stcAdditionalAliases (3) – Changes will also be cancelled in the cursors that are specified in this comma-delimited list of aliases. For any of the cursors specified in this list that use table buffering, changes will be cancelled for each modified record. Otherwise, just the changes in the current record are cancelled.
 - stcRulesID (6) – Passed on to business rules methods so that code in those methods can run conditionally if necessary. If this method is run from default form class code the contents of the form's cRulesID property is passed to this method.
 - Returns:
 - .T.
- **SaveRecord** (oApp.Save) – Saves the changes to the current record in the cursor specified in the parameter stcAlias. If the parameter stcAlias is empty and a cursor is open in the current work area, the changes to the current record in that cursor are saved.
 - Parameters:
 - stcAlias (1) – The alias of the cursor in which the changes will be saved.
 - stlSaveAll (8) – If .T., the changes to all modified records in the cursor are saved. Table buffering must be in use on the cursor for this parameter to be used.
 - stcAdditionalAliases (3) – Changes will also be saved in the cursors that are specified in this comma-delimited list of aliases. For any of the cursors specified in this list that use table buffering, changes will be saved for each modified record. Otherwise, just the changes in the current record are saved.
 - stcRulesID (7) – Passed on to business rules methods so that code in those methods can run conditionally if necessary. If this method is run from default form class code the contents of the form's cRulesID property is passed to this method.
 - stlNoNewAuditTrail (4) – Specifies whether or not Audit Trail records are created when a new record is saved.
 - stlNoChangeAuditTrail (5) – Specifies whether or not Audit Trail records are created when a modified record is saved.
 - stlNoRICall (9) – Specifies whether or not the RI routine is called. If not, the RI check, Audit Trail functionality, duplicate PK check, and blank PK check are all disabled.

- stcNotBlankFields – This parameter is passed on to the EmptyRequiredFields method when the save is performed. See the description of that method for a definition of this parameter.
 - stlNoRecordLevelValidation – Specifies whether or not the record-level field validation will be performed.
 - Properties:
 - cRIMessage – Contains error information when 2, 4, 7, 9, 15, 16, 18, or 22 is returned.
 - Returns:
 - 0 – Save completed.
 - 1 – Failed duplicate primary key check, save aborted.
 - 2 – Failed VPM RI check, save aborted.
 - 3 – Primary key field has null or blank value, save aborted.
 - 4 – TABLEUPDATE failed, save aborted (unknown error).
 - 5 – DefaultValue method returned .F., save aborted.
 - 6 – PreSave method or code in mPreSaveCode returned .F., save aborted.
 - 7 – EmptyRequiredFields method found at least one required field that does not contain a value, save aborted.
 - 8 – Transaction failed, save aborted.
 - 9 – Error occurred, save aborted.
 - 10 – TABLEUPDATE failed, save aborted (record is in use by another user).
 - 11 – TABLEUPDATE failed, save aborted (insert trigger failed).
 - 12 – TABLEUPDATE failed, save aborted (update trigger failed).
 - 13 – TABLEUPDATE failed, save aborted (delete trigger failed).
 - 14 – TABLEUPDATE failed, save aborted (field doesn't accept null values).
 - 15 – TABLEUPDATE failed, save aborted (field rule was violated).
 - 16 – TABLEUPDATE failed, save aborted (row rule was violated).
 - 17 – TABLEUPDATE failed, save aborted (uniqueness of an index was violated).
 - 18 – TABLEUPDATE failed, save aborted (ODBC error occurred).
 - 19 – PostSave method or code in mPostSaveCode returned .F., save aborted.
 - 20 – Failed duplicate candidate key check, save aborted.
 - 21 – PreSavePreTransaction method or code in mPreSavePreTransactionCode returned .F., save aborted.
 - 22 – FieldValidation_RecordLevel method found at least one field that contains an invalid value, save aborted.
- **SelectAlias** (oApp.Select_Alias) – Selects the alias specified in the parameter stcAlias. If stcAlias is empty and a cursor is open in the current work area, that work area remains selected.
 - Parameters:
 - stcAlias (1) – The alias of the cursor to be selected.
 - Returns:
 - .T. – The specified alias (work area) was selected.
 - .F. – The specified alias (work area) was not selected.
- **SelectWorkArea** – Selects the work area specified in the parameter stnWorkArea. If stnWorkArea is zero or non-numeric an unused work area is selected.
 - Parameters:
 - stnWorkArea – The number of the work area to be selected.
 - Returns:
 - .T.
- **SetFilter** (oApp.SetFilter) – Sets or removes a filter on the current cursor or the cursor with the alias specified in the parameter stcAlias. A filter is removed by passing the null string or .F. in the parameter stcFilterExpression.
 - Parameters:
 - stcAlias (1) – The alias of the cursor to which the filter is set.
 - stcFilterExpression (3) – The filter expression to be set on the cursor.

- stcDDDName
 - stcDDTVName
 - stcDDFName
 - Returns:
 - .T. – The record was updated.
 - .F. – The record was not updated.
- **UpdateFor** – Updates records in the cursor specified by the parameter stcAlias using the UPDATE command. If the parameter stcAlias is empty and a cursor is open in the current work area, the records in that cursor are updated. The updates are performed using the UPDATE command and the TABLEUPDATE function instead of using REPLACES and the SaveRecord method, which means the records are updated without the PreSave/PostSave code being run, without the data handler's RI routine being run, and without the Audit Trail functionality being run.
 - Parameters:
 - stcAlias – The alias of the cursor that contains the records to be updated.
 - staSet – The reference to a three-dimensional array that contains the information used to build the SET clause of the UPDATE command, which defines how the records are updated. The columns in the array should contain the following: Column 1 – field name, Column 2 – the value to be placed in the field or the expression to be evaluated and result placed in the field, Column 3 – "V" (if column 2 contains a value) or "E" (if column 2 contains an expression).
 - stcWhereExpression – The expression used in the WHERE clause of the UPDATE command.
 - Required Parameters:
 - staSet
 - Returns:
 - .T. – The records were updated.
 - .F. – The records were not updated.
- **UserValidation** – Validates the User ID and optional password that are passed to this method.
 - Parameters:
 - stcUserID – Specifies the User ID to be validated.
 - stcPassword – Specifies the password of the user to be validated.
 - stlSaveToProperty – Specifies whether or not the User ID should be saved to the cUserID property. The value in the cUserID property is placed in error records and Audit Trail records when created.
 - stlPasswordRequired – Specifies whether or not the password must be validated. If not, just the User ID will be validated.
 - Required Parameters:
 - stcUserID
 - Returns:
 - 0 – Validation check was successful.
 - 1 – Empty User ID and password, validation failed.
 - 2 – Empty User ID, validation failed.
 - 3 – Empty password, validation failed.
 - 4 – UserLogin_V view could not be opened, validation failed.
 - 5 – User ID not found, validation failed.
 - 6 – Invalid data in User record, validation failed.
 - 7 – Invalid password, validation failed.