Vision Q.400

Image Processing

Version 7.1.0.0 ActiveX Controls

Q.VITEC GmbH

Hagenburger Str. 54, 31515 Wunstorf, Germany

Tel.:+49(0)5031-949 43 20 • Fax: +49(0)5031-949 43 29

Internet: www.qvitec.de

Date: 2014-07-22

Table of Contents

1 Functionality: ActiveX Control VisionQ400Control

This documentation is an addition to the reference manual for Vision Q.400. For the details about Vision Q.400, which are necessary to understand the functionality described here, please have a look into the reference manual of Vision Q.400.

Vision Q.400 can be used as OLE-Automation server. For easier access to this COM-interface, the ActiveX Control **VisionQ400Control** can be used. This ActiveX Control is automatically registered by the installation of Vision Q.400.

By default , COM places instances of the ActiveX Control **VisionQ400Control** in a so called "single-threaded apartment", so the object won't suffer concurrent thread accesses. That means that if your client uses more than one thread, and you use only one instance of the ActiveX Control **VisionQ400Control**, only the thread in which the instance is created can access it. All other threads must have their own instances of the control, or the have to post, not send(!), messages to the thread, which owns the control if they want to access it. (A description of the COM apartment model is found at
http://www.codeguru.com/Cpp/COM-Tech/activex/apts/article.php/c5529/.)

Vision Q.400 is licensed by a dongle. This doingle can be used to license the ActiveX client, too. To check the license the VisionQ400Control method getProperty() with the property name "CheckClientLicense" can be used.

To use Vision Q.400 as a server, the client, which uses the ActiveX Control, must be connected to Vision Q.400 by calling the method **connectToServer**. All other methods return an error until connectToServer is called.

If Vision Q.400 was not started when a client calls connectToServer, Vision Q.400 is implicitly started by the call.

To disconnect a client, the method **disconnectFromServer** has to be called. If Vision Q.400 is implicitly started by a client, and the client which calls disconnectFromServer, is the last connected client, Vision Q.400 is implicitly finished.

The method **exitServer** disconnects automatically from Vision Q.400 before Vision Q.400 is finished. If several clients are connected to Vision Q.400, and one client calls exitServer, all other clients are disconnected from Vision Q.400, too, and they receive the signal VISIONQ400_SIG_EXITING. The client which calls exitServer does not receive the signal VISIONQ400_SIG_EXITING.

If Vision Q.400 is finished, but not by a call from an OLE client, all OLE clients are disconnected from Vision Q.400, and they receive the signal VISIONQ400_SIG_EXITING.

The event **ErrorOccurred** is independent from the methods of VisionQ400Control and may occur at any time. Therefore we recommend implementing it. Otherwise it can happen that Vision Q.400 does not work anymore, but the client does not know this.

Please consider that your client has to be synchronised with the help of the signals Vision Q.400 sends. The event **SignalRecieved** informs you about all signals Vision Q.400 sends to the ActiveX Control VisionQ400Control. Therefore we recommend implementing it. Synchronisation may especially be necessary after starting or changing an application.

If you do not synchronise your client, Vision Q.400 may not be ready for the next call of a method, and the methods of VisionQ400Control may return an error code, e.g. the method startApplication may return VARIANT_FALSE. Synchronization matters are described in the Vision Q.400 reference manual in the chapter "Interfaces -> Introduction".

1.1 Naming Convention

If in the following the terms TRUE, or FALSE (respectively true, or false) are used, the refer always to the VARIANT_BOOL data type: TRUE (or true) mean always VARIANT_TRUE, FALSE (or false) mean always VARIANT_FALSE.

If the terms double, long, or short are used the data types DOUBLE, LONG, or SHORT are meant.

1.2 A Note on Visual Studio

Visual Studio, even Visual Studio 2012, is a 32 Bit executable.
(See http://stackoverflow.com/questions/13603854/visual-studio-2012-64-bit).
This means that if Vision Q.400 is installed on a 64 Bit system, Visual Studio will not find the ActiveX Control VisionQ400Control, because this is 64 bit. To solve this the file VisionQ400Control_32.ocx is found in the installation directory of Vision Q.400. If this file is registered, the 32 Bit version of the ActiveX Control VisionQ400Control is registered, too. Now Visual Studio will find it.

There is no problem to create 64 Bit executables with the 32 Bit version of the ActiveX Control VisionQ400Control. They will run.

2 Methods of the ActiveX Control VisionQ400Control

2.1 General Methods

2.1.1 connectToServer(sleepTime)

The method **connectToServer** connects a client to the Vision Q.400 server. Before all other methods can be used, the client must be connected to Vision Q.400 by a call of this method.

Return Value

VARIANT_BOOL

The return value is TRUE if the methods succeeds, otherwise FALSE. If the return value is FALSE, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument		**Type**	**Range**	**Description**
[IN]	sleepTime	LONG		Time in seconds to sleep before the connection to Vision Q.400 is performed.

Remarks

If Vision Q.400 is already connected, the method returns immediately.
If sleepTime is less or equal to zero, the connection is immediately performed without any sleep.

If Vision Q.400 is not running, it is implicitly started by a call to this method.

If Vision Q.400 is started by a client, and the Vision Q.400 property "Interfaces -> OLE -> Start Vision Q.400 Hidden by an OLE Client" is checked, Vision Q.400 is started without any visible window. To show Vision Q.400 later, the method showServer() with the parameter show set to TRUE may be called.

If Vision Q.400 is started by a client and the creation fails, e.g. by a missing license file, a timeout of two minutes occurs before connectToServer returns. (The length of the timeout is fixed by MICROSOFT and cannot be influenced.)

The sleep time is especially helpful if you wish to establish an automatic connection between Vision Q.400 and a client at system startup. **We found out that if we do not have a delay of 20 seconds between the automatic start of the client and its connection to Vision Q.400, synchronization problems between grabber and camera can be happen.** In this case we recommend connecting your client to Vision Q.400 with the call connectToServer(20).

Please refer to the function "setProperty()" which can be used to set connection parameters.

Visual Basic Example

```
Private Sub Command1_Click()

    Dim bReturn As Boolean
    Dim sErrorText As String

    bReturn = VisionQ400Control1.connectToServer(0)
    If bReturn = False Then
      sErrorText = VisionQ400Control1.getLastErrorText()
      MsgBox (sErrorText)
    End If

End Sub
```

2.1.2 disconnectFromServer(saveSettingsSetupMode)

The method **disconnectFromServer** disconnects a client from the Vision Q.400 server. A call to all other methods will fail until the client is connected again.

Return Value

VARIANT_BOOL

The return value is TRUE if the methods succeeds, otherwise FALSE. If the return value is FALSE, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument	Type	Range	Description
[IN] saveSettingsSetupMode	SHORT	-1..1	This parameter is only used if Vision Q.400 is in setup mode, and if it is finished when the last client is disconnected. (See the remarks section.) - 1: The currently open application is not saved, even if it was modified. 0: If the currently open application was modified, Vision Q.400 displays a dialog, and it is asked if the application should be saved. If “Cancel” is selected, this is ignored, that means Vision Q.400 is finished, but the currently open application is not saved, even if it was modified. If the Vision Q.400 user rights management is activated, and the currently logged in user does not have the permission to save an application, the currently open application is not saved. 1: The currently open application is saved if it was modified. In all cases, if Vision Q.400 is password protected, the password is not asked for.

Remarks

If Vision Q.400 was started by a call to connectToServer, it is finished when the last client is disconnected.

If Vision Q.400 is NOT started by a client, and is hidden, and the last client is disconnected, Vision Q.400 is NOT finished, but it can only be seen in the task manager. To avoid this, Vision Q.400 is shown if the last client is disconnected.

Visual Basic Example

```
Private Sub Command2_Click()
    Dim bReturn As Boolean
    Dim sErrorText As String

    bReturn = _
          VisionQ400Control1.disconnectFromServer(0)
    If bReturn = False Then
      sErrorText = VisionQ400Control1.getLastErrorText()
      MsgBox (sErrorText)
    End If

End Sub
```

2.1.3 exitServer()

The method **exitServer** is called to exit Vision Q.400. The additional behaviour beside exiting Vision Q.400 depends on the ”Shut Down On Exit” settings in the “Vision Q.400 Settings...” item of Vision Q.400.

Return Value

VARIANT_BOOL

The return value is TRUE if the methods succeeds, otherwise FALSE. If the return value is FALSE, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument

none

Remarks

If several clients are connected to Vision Q.400, and one client calls exitServer, all other clients are disconnected from Vision Q.400, too, and they receive the signal VISIONQ400_SIG_EXITING. The client, which does call this method, does not receive the signal VISIONQ400_SIG_EXITING.

If a password is set in Vision Q.400, the password is not asked for.

The method exitServer fails in setup mode.

Visual Basic Example

```
Private Sub Command3_Click()
    Dim bReturn As Boolean
    Dim sErrorText As String
    bReturn = VisionQ400Control1.exitServer()
    If bReturn = False Then
      sErrorText = VisionQ400Control1.getLastErrorText()
      MsgBox (sErrorText)
    End If
End Sub
```

2.1.4 getState()

The method **getState** returns the current state of Vision Q.400.

Return Value

LONG

The return value contains the current state of Vision Q.400. It is a combination of any of the following values:

Name	**Value**	**Meaning**
VISIONQ400_STATE_IN_SETUP_MODE	0x000000001	Vision Q.400 is in setup mode
VISIONQ400_STATE_IN_RUN_MODE	0x000000002	Vision Q.400 is in run mode
VISIONQ400_STATE_APPLICATION_LOAD	0x000000004	An application is loaded or Vision Q.400 is loading an application
VISIONQ400_STATE_READY	0x000000008	Vision Q.400 is ready
VISIONQ400_STATE_START_WILL_FAIL	0x000000100	This is not a state, but the next three states will have set this bit, which means that a start of Vision Q.400 will always fail without further actions
VISIONQ400_STATE_FATAL_ERROR	0x000000300	A fatal error has occurred
VISIONQ400_STATE_STOPPED_BY_ACTION	0x000000500	Vision Q.400 has been stopped by an ACTION
VISIONQ400_STATE_IS_REPETITIVE	0x000000900	Vision Q.400 is repetitive starting

Argument

none

Remarks

If the return value is zero, Vision Q.400 is not connected to the client.

If the loading of an application is started by the graphical user interface of Vision Q.400, getState() may return VISIONQ400_STATE_APPLICATION_LOAD, even if the loading of the application may not be finished yet.

If one of the methods of the ActiveX control fails, you should call one of the error handling methods to get further information, but not getState(). An exception is the method startApplication. If this method returns FALSE, you may call getState() and test for the state VISIONQ400_STATE_START_WILL_FAIL. This state signals that all further calls of startApplication() will return FALSE until the cause of the failure will be removed by an appropriate action.

In the setup mode VISIONQ400_STATE_READY is never set, in the run mode getState() returns the PCReady state of Vision Q.400. Be aware that getState() is not a substitution of the synchronisation mechanism of Vision Q.400. This synchronization mechanism is described in the Vision Q.400 reference manual in the chapter “Interfaces -> Introduction”. GetState() should only be used to retrieve the PCReady state of Vision Q.400 if a client is connected to Vision Q.400 in the run mode to initialize the ready state in the client correctly.

Visual Basic Example

```
Option Explicit
Const VISIONQ400_STATE_IN_SETUP_MODE = &H1
Const VISIONQ400_STATE_IN_RUN_MODE = &H2
Const VISIONQ400_STATE_APPLICATION_LOAD = &H4
Const VISIONQ400_STATE_START_WILL_FAIL = &H100
Const VISIONQ400_STATE_FATAL_ERROR = &H300
Const VISIONQ400_STATE_STOPPED_BY_ACTION = &H500
Const VISIONQ400_STATE_IS_REPETITIVE = &H900

Private Sub Command1_Click()

Dim lReturn As Long
Dim sErrorText As String

    lReturn = VisionQ400Control1.getState()

    If lReturn = 0 Or lReturn = -1 Then
                                                'If the return value is zero, Vision
                                                'Q400 is not connected to the client.
                                                'If the return value is -1, 'information could not be got
      sErrorText = VisionQ400Control1.getLastErrorText()
      MsgBox (sErrorText)
    Else
      If lReturn And VISIONQ400_STATE_IN_SETUP_MODE Then
            MsgBox ("Vision Q.400 is in setup mode")
      End If
    End If

End Sub
```

2.1.5 getProperty(name, parameter)

The method **getProperty** returns the current value(s) of a Vision Q.400 property.

Return

VARIANT

For the current type and meaning of this VARIANT please refer the description of the concerning property.
The method returns an empty VARIANT if it fails. In this case, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument		**Type**	**Description**
[IN]	name	BSTR	Name of the Vision Q.400 property e.g. “IsVisible”.
[IN]	parameter	VARIANT	Parameter which describes the property to get in detail. For the current type and meaning of this VARIANT please refer to the description of the concerning property.

Property Name	Parameter	Returned VARIANT Type	Comment
AllowedShutterTimes	VT_I2 The camera number for which all allowed shutter times will be got.	VT_BSTR | VT_ARRAY The format of the strings is the same as shown in the camera’s shutter property page of Vision Q.400.	The returned array contains one entry for each allowed shutter time. If no shutter times are allowed, the array contains one entry, the empty string.
ApplicationName	VARIANT Type:	VT_BSTR	Returns the path name of an

	VT_EMPTY or VT_I4 The number of an application, for which the path name should be returned. A number is assigned to an application in the menu Application -> Application Numbers…		application. If the given parameter type is VT_EMPTY, the parameter value 0 is used. Parameter Value == 0: The path name of the currently loaded application is returned. If no application is loaded, an empty VARIANT is returned. If the current application was never saved that means its path name does not exist, “” is returned. 1 <= Parameter Value <= 9999: If the given parameter value is assigned to an application, the path name of this application is returned, otherwise the empty string is returned.
ApplicationNumber	VARIANT Type: VT_EMPTY or VT_ BSTR Path name of the application, for which its assigned number should be returned. A number is assigned to an application in the menu Application -> Application Numbers	VT_I4	Returns the number of an application. If the given parameter type is VT_EMPTY, or if the parameter is the empty string, the path name of the currently loaded application is used. If the path name of the currently loaded application is used, and no application is loaded, an empty VARIANT is returned. If the current application was never saved that means a number cannot be assigned to it, - 1 is returned. If an application number is assigned to the given path name, this number is returned, otherwise – 1 is returned.
CheckerNames Without any modification, all checkers of a sequence are returned. To get only the checkers of a special type you can append a blank and one of the following strings to CheckerName:	VARIANT Type: VT_I2 and VT_ARRAY (one Dimensional, two values) [0]: camera number for which the checker names should be returned [1]: sequence number	VT_BSTR and VT_ARRAY (a one dimensional array, which contains the checker names)	Returns the ActiveX Names of all checkers, which belongs to the camera with the given number and to the sequence with the given number. If the sequence number is 0, the names of all checkers of all sequences belonging to the given camera are returned.

FE: feature extraction EA: exposure adjustment WI: window checker ED_B: edge detection binary ED_G: edge detection grey DI: difference checker CM: contour matching OCR: character recognition CR: code reader PRA: all types of position and rotation adjustment GEO: all types of geometric checkers ED_GP: edge detection grey protection OS: Object Shape IC: Indicator If user defined checkers are used, the short name of the Checker, which has to be used, is documented in the description of the appropriate dll. Types: the returned String array contains a list of the above listed strings, including the short names of all user defined checkers.	for which the checker names should be returned VT_EMPTY: The camera number and the sequence number are set to 0. VT_I2: The camera number is set to the given number, and the sequence number are set to 0. VT_EMPTY		The checker names are returned in the same sequence as they are shown in the sequence list. If the camera number is 0, the names of all checkers of all cameras belonging to the given sequence are returned. If the camera number and the sequence number are 0, the names of all checkers belonging to the currently loaded application are returned. If no checker is found, the returned array contains one entry, the empty string.
CurrentShutterTime	VT_I2 The camera number for which the currently set shutter time will be got.	VT_BSTR	Returns the currently set shutter time.
ExeGroupToStart	VT_EMPTY The parameter is ignored.	VT_UI2	At time only used together with the camera interface: If Vision Q.400 is started by a camera, Vision Q.400 can use this execution group number.

Graphic Update On	VT_BSTR “All” “Image” “Checkers” “Indicators” “Names”	VT_BOOL -1 and 1: the queried graphical update is switched on 0 : : the queried graphical update is switched off For “Names”: VT_BSTR | VT_ARRAY	The “Graphic Update” settings in the run mode, e.g. if Parameter is “Checkers” – 1 or 1 are returned if the grahic update for checkers is switched on, 0 if it is switched off. For “All”, -1 or 1 are only returned if all grahic update is swiched on, 0 if at least one is switched off. “Names” is an exception: it returns all other possible parameter values.
Graphic Update Off	VT_BSTR “All” “Image” “Checkers” “Indicators” “Names”	VT_BOOL -1 and 1: the queried graphical update is switched OFF 0 : : the queried graphical update is switched ON For “Names”: VT_BSTR | VT_ARRAY	The “Graphic Update” settings in the run mode, e.g. if Parameter is “Checkers” – 1 or 1 are returned if the grahic update for checkers is switched OFF, 0 if it is switched ON. For “All”, -1 or 1 are only returned if all grahic update is swiched OFF, 0 if at list one is swiched ON. “Names” is an exception: it returns all other possible parameter values.
IgnoreData	VARIANT Type: VT_EMPTY The parameter is ignored.	VT_BOOL -1 and 1: data AND images are not sent, 0: data AND images are sent	Returns if data and images are sent or not. For details see setProperty(IgnoreData, …)
ImageBufferDim	VT_I2 1 or 2: The dimension of the array, which is used in the event ImageAvaliable() and in the method getImage()to transfer the gray values of an image. Default: 2	VT_EMPTY Value is ignored.	The new dimension will be used for all images, which transfer will be enabled by setSendImage() or setSendImagePart() after the setting of the new dimension. That means that if the dimension of the array has to be changed for an image, the new dimesion has to be set and then setSendImage() or setSendImagePart() has to be called again. If VISUAL BASIC .Net is used, and the transferred image has to be inserted into a picture box, the dimension of the safe array has to be 1 for an fast insertion!
ImagePixelSizeInByte	VT_I2	VT_I2	Returns the pixel size of an

	The camera number for which the pixel size in Byte will be got.		image in Byte.
ImageType	VT_I2 The camera number for which the image type will be got.	VT_BSTR "Gray" for 8 bit (1Byte) gray value images "Color" for 24 bit (3 Byte) RGB color images	Returns the image type.
IsVisible	VT_EMPTY The parameter is ignored.	VT_BOOL -1 and 1 as visible, 0 as not visible (hidden)	Returns if Vision Q.400 is visible.
CheckClientLicense	VT_BSTR License string to check.	VT_UI1	This property can be used to to license an ActiveX client by a Vision Q.400 dongle. Returns the information if the client can be licensed by the provided license string. 0 = The client cannot be licensed by a valid dongle, and a valid time limited license file is not found. 1 = The client can be licensed by a valid dongle. 3 = The client can be licensed by a time limited license file. The client has to decide if it can be licensed. A not licensed simulation version of Vision Q.400.always returns 0.
LicenseType	VT_BSTR "Vision Q.400": The license type of Vision Q.400 will be got. "Application": The license type currently set for an application is got. "Application Original": The "original" license type of an application. VT_EMPTY or the empty string:	VT_BSTR	"Application": If an application is currently loaded the license type which is currently set for the application is returned. If an application is not loaded, the "default" license type of a newly created application. (This is the license type of Vision Q.400.)

	The same as VT_BSTR "Vision Q.400".		
LiveImage	VT_I2 The camera number for which the display mode will be got.	VT_BOOL -1 and 1: camera is in live mode 0 : camera is in memory mode	Returns the display mode of the camera. This property works only in the setup mode.
OLEDataTransferListInfo	VARIANT Type: VT_EMPTY The parameter is ignored.	VT_BSTR | VT_ARRAY	Returns the content of the OLE datatransfer container (name and datatype)
PropertyNames	VARIANT Type: VT_EMPTY The parameter is ignored.	VT_BSTR | VT_ARRAY	Returns the names of all properties, which are supported by Vision P400.
SpreadsheetRowNames	VT_BSTR "Results": The names of the checker result rows are returned. "Formulas": The names of the formula rows are returned.	VT_BSTR | VT_ARRAY The names of the requested rows. The name at the array index 0 belongs to row number 1, and so one.	If the spreadsheet does not contain rows of the requested type, the returned array contains exactly none entry, the empty string.
SwitchInterfaceSending Off	VT_BSTR Name of the interface: RS232, P I/O, File	VT_BOOL -1 or 1: The interface does not send anything. 0: The interface does normally work.	If the interface sending is switched off for an interface, this interface does not sent any data or events. To switch of an interface the SetProperty() function with the parameter "SwitchInterfaceSendingOff" must be used. For OLE clients, the sending cannot be switched off.
TransferredImage	VT_I2 The camera number for which information about the transferred image will be got.	VT_I4 | VT_ARRAY (two dimension, every dimension with two values)	Returns information of the size of the transferred image: [0, 0]: x coordinate of the top left corner of the transferred image (part) [0, 1]: y coordinate of the top left corner of the transferred image (part) [1, 0]: number of transferred columns [1, 1]: number of transferred rows

SendImage	VT_I2 The camera number for which information about the sending of an image will be got.	VT_BOOL -1 or 1: The image is sent automatically. 0: The image is not sent automatically.	Returns the information if an image which is set by the methods setSendImage() or setSendImagePart() is sent automatically after each execution of an application with the help of the event ImageAvaliable(), or if it is not sent, and can only be accessed with help of the method getImage().
UserDataNames	VT_EMPTY The parameter is ignored.	VT_BSTR | VT_ARRAY The user data names, under which user data is stored in the currently loaded application.	If user data is not stored in the currently loaded application, getProperty() returns TRUE, and the returned string array contains exactly one entry, the empty string.
Version	VT_BSTR "Version": The version type of the currently running Vision Q.400 version. "Number": The version number of the currently running Vision Q.400 version. VT_EMPTY or the empty string: The same as VT_BSTR "Version".	VT_BSTR	"Version" values: "Full": The full version of Vision Q.400 is currently running. "Simulation": The simulation version of Vision Q.400 is currently running. "Demo": The demo version of Vision Q.400 is currently running. The "Number" string is formatted as shown in the Vision Q.400 help about box.
VersionNumber	VT_EMPTY The parameter is ignored.	VT_I4 The version number of the currently running Vision Q.400 version. The returned version number has the format: **Mmmbbpp** with: **M**: major version **mm**: minor version **bb**: build version **pp**: private version	
BlackLevel	VT_I2 The camera number for which the black level, and it's limits, will be got.	VT_R8 and VT_ARRAY (one dimension, three values) [0] current value	This property is only supported by GigE Vision cameras.

		[1] minimum allowed value [2] maximum allowed value	
Name	VT_EMPTY	VT_BSTR	Returns a unique name for the OLE connection.
Gain	VT_I2 The camera number for which the gain, and it's limits, will be got.	VT_R8 and VT_ARRAY (one dimension, three values) [0] current value [1] minimum allowed value [2] maximum allowed value	This property is only supported by GigE Vision cameras.

2.1.5.1 Accessing the Parallel I/O Channnels

The method getProperty can be used to request the values of the Parallel I/O channels of Vision Q.400.

Property Name	Parameter	Returned VARIANT Type	Comment
PioBoardName	VT_EMPTY The parameter is ignored.	VT_BSTR	Returns the name of the used Parallel I / 0 card like shown under Vision Q.400 -> Interfaces -> Properties.
PioInputValues	VT_EMPTY		Same behaviour as: Parameter type: VT_BSTR Parameter name: “All”
	VT_BSTR Names	VT_BSTR | VT_ARRAY	Returns the names of all input channels as given in the table below.
	VT_BSTR All All Signals All Data	VT_I4	Parameter value is “All”: The values of all input channels. Parameter value is “All Signals”: The values of all input signal channels. In the returned result, the bits of the input data channels are cleared. Parameter value is “All Data”: The values of all input data channels. In the returned result, the bits of the input signal channels are cleared, and the bits of the input data channels are shifted into the rightmost byte of the result. The assignment of a bit value to an input channel is given in the table below.
	VT_BSTR	VT_BOOL	Parameter value is one of the

Property Name	Parameter	Returned VARIANT Type	Comment
		-1 and 1: channel is set 0 : the channel is not set	input channel names.
	VT_I4	VT_BOOL -1 and 1: channel is set 0 : the channel is not set	Parameter value is one of the input channel numbers. The behavior is the same as if an input channel name is given. The assignment of an input channel number to an input channel name is given in the table below. To be compatible between different Parallel I/O hardware types, it is strictly recommented to use the input channel names.
PioOutputValues	VT_EMPTY		Same behaviour as: Parameter type: VT_BSTR Parameter name: “All”
	VT_BSTR Names	VT_BSTR | VT_ARRAY	Returns the names of all output channels as given in the table below.
	VT_BSTR All All Signals All Data	VT_I4	Parameter value is “All”: The values of all output channels. Parameter value is “All Signals”: The values of all output signal channels. In the returned result, the bits of the output data channels are cleared. Parameter value is “All Data”: The values of all output data channels. In the returned result, the bits of the output signal channels are cleared, and the bits of the output data channels are shifted into the rightmost byte of the result. The assignment of a bit value to an output channel is given in the table below.
	VT_BSTR	VT_BOOL -1 and 1: channel is set 0 : the channel is not set	Parameter value is one of the output channel names.
	VT_I4	VT_BOOL -1 and 1: channel is set 0 : the channel is not set	Parameter value is one of the output channel numbers. The assignment of an output channel number to an output

Property Name	Parameter	Returned VARIANT Type	Comment
			channel name is given in the table below. To be compatible between different Parallel I/O hardware types, it is strictly recommented to use the ouput channel names.
PioOutputData	VT_EMPTY		Same behaviour as: Parameter type: VT_BSTR Parameter name: “All”
	VT_BSTR Names	VT_BSTR | VT_ARRAY	Returns the names of all output data channels as given in the table below.
	VT_BSTR All All Data	VT_I4	The values of all output data channels. In the returned result, the bits of the output data channels are shifted into the rightmost byte of the result. The assignment of a bit value to an output data channel is given in the table below.
	VT_BSTR	VT_BOOL -1 and 1: channel is set 0 : the channel is not set	Parameter value is one of the output data channel names.
	VT_I4	VT_BOOL -1 and 1: channel is set 0 : the channel is not set	Parameter value is one of the output data channel numbers. The assignment of an output data channel number to an output data channel name is given in the table below. To be compatible between different Parallel I/O hardware types, it is strictly recommented to use the ouput data channel names.

Parallel I / O Input Channels:

If the property name is PioInputValues and the parameter name is “All Data”, the data bits are shifted into the rightmost byte of the result. That means Data 1 has bit number 1, and Data 8 has bit number 8.

	ANPC850V2D		ANPC850V3D	
Channel Name	**Channnel Number**	**Bit Number**	**Channel Number**	**Bit Number**
Start	1	1	0	1
Ack	2	2	1	2
Change Application	3	3	2	3
Reset Statistics	4	4	3	4
Shut Down	5	5	4	5
Lock Grab	6	6	5	6

Start/Stop Run Mode	7	7	6	7
reserved	8	8	7	8
Data 1	9	9	8	9
Data 2	10	10	9	10
Data 3	11	11	10	11
Data 4	12	12	11	12
Data 5	13	13	12	13
Data 6	14	14	13	14
Data 7	15	15	14	15
Data 8	16	16	15	16

Parallel I / O Output Channels:

If the property name is PioOutputValues and the parameter name is “All Data”, or if the property name is PioOutputData and the parameter name is “All”, the data bits are shifted into the rightmost byte of the result. That means Data 1 has bit number 1, and Data 8 has bit number 8.

	ANPC850V2D		**ANPC850V3D**	
Channel Name	**Channel Number**	**Bit Number**	**Channel Number**	**Bit Number**
PCReady	1	2	16	1
REnd	2	3	17	2
Strobe	3	17	18	3
Application Switch Completed	4	4	19	4
Byte Overflow/Start Lost	5	5	20	5
Error	6	6	21	6
Execution Result Error	7	7	22	7
Action Error	8	8	23	8
Data 1	9	9	24	9
Data 2	10	10	25	10
Data 3	11	11	26	11
Data 4	12	12	27	12
Data 5	13	13	28	13
Data 6	14	14	29	14
Data 7	15	15	30	15
Data 8	16	16	31	16

2.1.5.2 User Rights Management

The method getProperty can be used to query information about the Vision Q.400 User Rights Management.

Property Name	Parameter	Returned VARIANT Type	Comment
IsUserRightsManagementActivated	VT_EMPTY The parameter is ignored.	VT_BOOL -1 and 1 as activated, 0 as not activated	Returns if the Vision Q.400 User Rights Management is activated ore not.
UserRightsManagementUsers	VT_EMPTY The parameter is ignored.	VT_VARIANT | VT_ARRAY The returned VARIANT array contains always an even number of VARIANTS: the even array elements are of type VT_BSTR, the odd array elements of type VT_UI4.	Returns information of all existing Vision Q.400 users. For every user, two VARIANTs are used: The first contains the user name name of the user as VT_BSTR, the second his user level as VT_UI4.
UserRightsManagementCurrentUser	VT_EMPTY The parameter is ignored.	VT_BSTR	The name of the currently logged in Q.400 user, or the empty string, if a user is not currently logged in.

2.1.6 setProperty(name, parameter, values)

The method **setProperty** set the new value(s) of a Vision Q.400 property.

Return Value

VARIANT_BOOL

The return value is TRUE if the methods succeeds, otherwise FALSE. If the return value is FALSE, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument		**Type**	**Description**
[IN]	name	BSTR	Name of the Vision Q.400 property e.g. "LiveImage".
[IN]	parameter	VARIANT	Parameter which describes the property to set in more details. For the current type and meaning of this VARIANT please refer the description of the concerning property.
[IN]	values	VARIANT	Values to be set. For the current type and meaning of this VARIANT please refer the description of the concerning property.

Remarks

The method setProperty does not switch the ready state of Vision Q.400 to off. That means that the execution of an application can be started before the method setProperty will return. It is strictly recommended not to start the execution of an application before the method will return. If this situation is possible, you should use the method interruptStartSignals() to forbid the execution of an application before the method setProperty is called. And you should use the method interruptStartSignals() afterwards to allow the execution of an application again.

Property Name	Parameter	Values	Comment
ComTimeout	VT_I4 Timeout value in milliseconds. Default: 5000	VT_EMPTY Values is ignored.	Determines how long the calling application waits for a response from the server before taking further action. **Attention:** You have to set this parameter before you call "connectToServer"!
NotRespondingDialog	VT_BOOL -1 and 1: enable the dialog 0 : disable the dialog Default: 0	VT_EMPTY Values is ignored.	When a COM timeout occurred the OLE "busy dialog box" is displayed so that the user can choose to cancel or retry the call. You can enable or disable this dialog. **Attention:** You have to set this parameter before you call "connectToServer"!
BusyDialog	VT_BOOL -1 and 1: enable the dialog 0 : disable the dialog Default: 0	VT_EMPTY Values is ignored.	When a keyboard or mouse message is pending during a call and the COM timeout occurred, the "not responding" dialog box is displayed. **Attention:** You have to set this parameter before you call "connectToServer"!
CancelGrab	VT_I2 The camera number for which the grab will be cancelled.	VT_EMPTY Values is ignored.	Cancels a started grab of the camera.
CurrentShutterTime	VT_I2	VT_BSTR	Sets the shutter time for the

Property Name	Parameter	Values	Comment
	The camera number for which the shutter time will be set.	The format of the string is the same as shown in the camera's shutter property page of Vision Q.400.	camera.
GrabNexImage	VT_I2 The camera number for which the next image will be set.	VT_BSTR Path of the image which should be used for the next execution of Vision Q.400. In case the string is empty or '-', the next image of the selected camera is used (same like F3 Key).	Sets the image for the next execution of Vision Q.400. **<u>Attention:</u>** This works only with a licensed simulation version of Vision Q.400.
ExeGroupToStart	VT_UI2 The execution group number to start with.	VT_EMPTY Values is ignored.	At time only used together with the camera interface: If Vision Q.400 is started by a camera, Vision Q.400 can use this execution group number.
Graphic Update On	VT_BSTR "All" "Image" "Checkers" "Indicators"	VT_EMPTY Values is ignored.	The "Graphic Update" setting in the run mode of the geviven parameter is swiched on. E.g. if Parameter is "Checkers" the grahic update for checkers is switched on. If "All" is given, all grahic update is swiched on.
Graphic Update Off	VT_BSTR "All" "Image" "Checkers" "Indicators"	VT_EMPTY Values is ignored.	The "Graphic Update" setting in the run mode of the geviven parameter is swiched off. E.g. if Parameter is "Checkers" the grahic update for checkers is switched off. If "All" is given, all grahic update is swiched off.
IgnoreData	VARIANT Type: VT_EMPTY The parameter is ignored.	VT_BOOL -1 and 1: data AND images are not sent, 0: data AND images are sent	If this function is called with the parameter values -1 or 1, data and images are not sent to the CALLING client furthermore: each client which not want to receive data or images have to call this function. By calling the function with the value 0, the sending is switched on again. The default value is sending data and images.

Property Name	Parameter	Values	Comment
			This property is useful if you want to use an asynchronous client: you use to threads, one thread for event handling and one thread for data handling. For the event handling thread you can switch off the sending of data and images to save time.
ImageBufferDim	VT_I2 1 or 2: The dimension of the array, which is used in the event ImageAvaliable() and in the method getImage()to transfer the gray values of an image. Default: 2	VT_EMPTY Values is ignored.	The new dimension will be used for all images, which transfer will be enabled by setSendImage() or setSendImagePart() after the setting of the new dimension. That means that if the dimension of the array has to be changed for an image, the new dimesion has to be set and then setSendImage() or setSendImagePart() has to be called again. If VISUAL BASIC .Net is used, and the transferred image has to be inserted into a picture box, the dimension of the safe array has to be 1 for an fast insertion!
LiveImage	VT_I2 The camera number for which the display mode will be set.	VT_BOOL -1 and 1: the camera will will be set to live mode 0 : the camera will will be set to memory mode	Sets the display mode of the camera. This property works only in the setup mode.
RefreshWhiteBalance	VT_I2 The camera number for which the white balance should be refreshed.	VT_EMPTY Values is ignored.	Refreshing of the white balance means that the reference values used for the white balance are recalculated: a new image without white balancing is grabbed to recalculate the reference values. These new reference values are used for further white balancing. If Vision Q.400 is not ready the white balance cannot be refreshed. Before refreshing the ready state of Vision Q.400 is set to off, and it is set to on again after refreshing. Attention: If external trigger is used for the affected camera, it has to be made sure that the new image used for the recalculation can be grabbed without any error , e.g. a timeout which is caused by a

Property Name	Parameter	Values	Comment
			missing external trigger. This property works only in the run mode.
DeleteNGImages	VT_EMPTY	-	Deletes the NG images which are stored by an Action in the memory.
SwitchInterfaceSendingOff	VT_BSTR Name of the interface: RS232, P I/O, File	VT_BOOL -1 or 1: The interface does not send anything. 0: The interface does normally work .	If the interface sending is switched off for an interface, this interface does not sent any data or events. For OLE clients, the sending cannot be switched off. If an OLE client switches the sending off, and it disconnects afterwards, the sending is NOT switched on again!
SendImage	VT_I2 The camera number for which the automatically sending of an image is switched off or on.	VT_BOOL -1 or 1: The image is sent automatically . 0: The image is not sent automatically.	TRUE: The image which is set by the methods setSendImage() or setSendImagePart() is sent automatically after each execution of an application with the help of the event ImageAvaliable(). FALSE It is not sent automatically, and can only be accessed with help of the method getImage(). The default value is TRUE.
BlackLevel	VT_I2 The camera number for which the black level will be set.	VT_R8 The new value of the black level.	This property is only supported by GigE Vision cameras
Gain	VT_I2 The camera number for which the gain will be set.	VT_R8 The new value of the gain.	This property is only supported by GigE Vision cameras

2.1.6.1 Accessing the Parallel I/O Output Data Channnels

The method setProperty can be used to set the Parallel I/O data channels of Vision Q.400. To do so, an OLE client has to request the access to these channels.

Property Name	Parameter	Values	Comment
PioRequestOutputData	VT_EMPTY The parameter is ignored.	VT_BOOL: -1 and 1: the client requests the setting of the output data channels of the parallel I/O card	If at least one OLE client has requested to set the output data, Vision Q.400 does not set any output data channel. Vision Q.400 behaves as if no data has to be sent to the Parallel I/O

Property Name	Parameter	Values	Comment
		0: the client will not set the output data.	interface. The output data channels of the Parallel I/O interface can only be set by an OLE client by the call of setProperty() with the parameter name "PioOuputData." Because Vision Q.400 behaves as if no data has to be sent to the Parallel I/O interface, Vision Q.400 sets the PCReady signal even if setProperty() is not called before. Therefore if the output data channels of the Parallel I/O interface are set by an OLE client, this client is responsible for any synchronization.
PioOutputData	VT_EMPTY		Same behaviour as: Parameter type: VT_BSTR Parameter name: "All"
	VT_BSTR All All Data	VT_UI1	All output data channels are set at once. For each data channel the value of it's bit number is used for the setting. The assignment of a bit number to an output data channel is given in the table below. If an application is loaded, the setting for the "Handshake" and the "Forced Reset" of the application are used. This setting are given in the Parallel I/O's "Objects for Data Transfer" property page of Vision Q.400. If no application is loaded, the data is transferred without "Handshake" and without "Forced Reset".
	VT_BSTR	VT_UI1	Parameter value is one of the output data channel names: the appropriate channel is set. The channel names are given in the table below. A channel is set if the given value is not equal to zero. Otherwise it is reset. The set value of a channel will not be changed until it will be explicitly changed by the client.
	VT_I4	VT_UI1	Parameter value is one of the

Property Name	Parameter	Values	Comment
			output data channel numbers. The behavior is the same as if a channel name is given. The assignment of an output data channel number to an output data channel name is given in the table below. To be compatible between different Parallel I/O hardware types, it is strictly recommented to use the ouput data channel names.

Parallel I / O Output Data Channels:

	ANPC850V2D		ANPC850V3D	
Channel Name	**Channel Number**	**Bit Number**	**Channel Number**	**Bit Number**
Data 1	9	1	24	1
Data 2	10	2	25	2
Data 3	11	3	26	3
Data 4	12	4	27	4
Data 5	13	5	28	5
Data 6	14	6	29	6
Data 7	15	7	30	7
Data 8	16	8	31	8

The bit number 1 is the rightmost bit in the ouput byte.

2.1.6.2 User Rights Management

The method setProperty can be used to log in and out Vision Q.400 users.

Property Name	Parameter	Values	Comment
UserRightsManagementUserLogin	VT_BSTR The name of the Q.400 user to log in.	VT_BSTR His password.	A Vision Q.400 cannot be logged in, if Vision Q.400 is in the run mode and the user does not have the right to step into the run mode, or he does not have the right to save an application. (The last right is required for stepping into run mode, because an application may be implicitly saved in the run mode.)
UserRightsManagementUserLogout	VT_BOOL or VT_EMPTY This parameter is	VT_BSTR or VT_EMPTY This parameter is	In the run mode the application is never saved, even if it is required by the parameter.

Property Name	Parameter	Values	Comment
	only used in the setup mode: -1 and 1 as save the current application on logout 0 as do not save. VT_EMPTY means do not save.	only used in the setup mode: The name of the file to which the application should be saved. If the name is the empty string, or Values is VT_EMPTY, the current application file name is used.	If the file name is the empty string, and the application was never saved, an error will occur, because an application file name does not exist. If the currently logged in user does not have the right to save an application, saving is ignored, an the application is closed. (This is done to avoid that application changes of the unatherized user may be saved later bay an autherized user.)

2.1.7 interruptStartSignals(inhibit)

The method **interruptStartSignals** inhibits or allows the start signal in the run mode. If the start signal is inhibited, it is not accepted from any Vision Q.400 interface, not either from the start application button in the GUI of Vision Q.400 or the <F5> key.

Return Value

VARIANT_BOOL

The return value is TRUE if the methods succeeds, otherwise FALSE. If the return value is FALSE, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument	Type	Description
[IN] inhibit	VARIANT_BOOL	TRUE: the start signals are inhibited. FALSE: the start signals are allowed.

Remarks

If start signals are inhibited, the start application button in the GUI of Vision Q.400 is greyed in the run mode. In the setup mode, the start signals from the start application button in the GUI and the key <F5> are accepted.

The method interruptStartSignals fails in setup mode.

Visual Basic Example

```
Private Sub Check1_Click()
    Dim bReturn As Boolean
    Dim bInhibit As Boolean

    If Check1.Value = 0 Then        'Check1 is deaktivated
      bInhibit = True                'Start not possible
      bReturn = VisionQ400Control1. _
                        interruptStartSignals(bInhibit)
    Else
      bInhibit = False               'Start possible
      bReturn = VisionQ400Control1. _
                        interruptStartSignals(bInhibit)
    End If

    If bReturn = False Then
      sErrorText = VisionQ400Control1.getLastErrorText()
      MsgBox (sErrorText)
    End If

End Sub
```

2.1.8 showServer(show)

The method **showServer** shows or hides Vision Q.400.

Return Value

VARIANT_BOOL

The return value is TRUE if the methods succeeds, otherwise FALSE. If the return value is FALSE, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument		Type	Description
[IN]	show	VARIANT_BOOL	TRUE: Vision Q.400 is shown.

Remarks

If Vision Q.400 is hidden, any visible window Of Vision Q.400 does not exist on the desktop.

If a fatal error occurred, Vision Q.400 cannot be hidden.

If Vision Q.400 is hidden in the setup mode, it may happen that a checker’s advanced settings dialog (e.g, The feature extraction checker’s “Object Filters” “Advanced …” dialog) will not be hidden if it is visible on hiding Vision Q.400.

Visual Basic Example

```
Private Sub Command3_Click() 'Call ShowServer()
    Dim bReturn As Boolean
    Dim sErrorText As String

    If bShow = False Then
      bShow = True
    Else
      bShow = False
    End If

    bReturn = VisionQ400Control1.showServer(bShow)
    If bReturn = False Then
      sErrorText = VisionQ400Control1.getLastErrorText()
      MsgBox (sErrorText)
    End If


End Sub
```

2.1.9 startRunMode()

The method **startRunMode** starts the run mode of Vision Q.400.

Return Value

VARIANT_BOOL

The return value is TRUE if the methods succeeds, otherwise FALSE. If the return value is FALSE, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument

none

Remarks

The function has no effect if Vision Q.400 is already in run mode.

If a password is set in Vision Q.400, the password is not asked for.

Visual Basic Example

```
Private Sub Command3_Click()
    Dim bReturn As Boolean
    Dim sErrorText As String

    bReturn = VisionQ400Control1.startRunMode()
    If bReturn = False Then
      sErrorText = VisionQ400Control1.getLastErrorText()
      MsgBox (sErrorText)
    End If

End Sub
```

2.1.10 stopRunMode()

The method **stopRunMode** stops the run mode of Vision Q.400 and enters the setup mode.

Return Value

VARIANT_BOOL

The return value is TRUE if the methods succeeds, otherwise FALSE. If the return value is FALSE, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument

none

Remarks

The function has no effect if Vision Q.400 is not in run mode.

If a password is set in Vision Q.400, the password is not asked for.

StopRunMode() stops the run mode of Vision Q.400 always, even if Vision Q.400 is not ready, or a fatal error has occurred.Therefore if the execution of an application should be correctly finished, make sure that Vision Q.400 is ready if it’s run mode is stopped.

Visual Basic Example

```
Private Sub Command3_Click()
    Dim bReturn As Boolean
    Dim sErrorText As String

    bReturn = VisionQ400Control1.stopRunMode()
    If bReturn = False Then
      sErrorText = VisionQ400Control1.getLastErrorText()
      MsgBox (sErrorText)
    End If

End Sub
```

2.2 Application Methods

2.2.1 changeApplication(appNumber)

The method **changeApplication** starts the changing of the current application to the application with the number appNumber.

Return Value

VARIANT_BOOL

The return value is TRUE if the methods succeeds, otherwise FALSE. If the return value is FALSE, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument	Type	Range	Description
[IN] appNumber	SHORT	1..9999	The number of the new application.

Remarks

The number of an application is set in the Vision Q.400 menu "Application -> Application Numbers...".

If there are current running starts of the "old" application, they are finished correctly and not interrupted before the current application is changed to the "new" application.

The old application is saved implicitly unless it is write protected. In this case it is not saved, and a warning may be written in the Vision Q.400 error log. (For details if a warning is written or not see the chapter error management in the Vision Q.400 manual.)

When the method returns, the change of an application is started, but not completed. The change is only completed when, on success, the signal VISIONQ400_SIG_CHANGE_NOTIFY ("Application switch completed") is received, or when, on error, the signal VISIONQ400_SIG_PCERROR ("PC Error") is sent. (For details for these synchronization matters, please have a look into the Vision Q.400 reference manual chapter "Interfaces -> Introduction".) Please do use this synchrinization mechanism and not the method getState() to determine if the change of an application is finished.

If appNumber is the number of the "old" application, the application switch is performed, too.

The changing of an application depends on the timeout value, which is set in Vision Q.400 in "Applications -> Vision Q.400 Settings... -> Timeouts -> Application Change." If the changing of an application takes longer than this timeout value, the changing is stopped, the application to be loaded is closed, and the signal VISIONQ400_SIG_PCERROR is sent. If getState() is called in this case, it will not return VISIONQ400_STATE_APPLICATION_LOAD.

The method changeApplication fails in setup mode.

Visual Basic Example

```
'Declaration of global variables
Option Explicit
    Dim bApplicationSwitched As Boolean
    Const VISIONQ400_SIG_CHANGE_NOTIFY = 64
    Const VISIONQ400_SIG_PCERROR = 256

'Call the change application function
Private Sub Command3_Click()
    Dim bReturn As Boolean
    Dim sErrorText As String
    Dim iAppNumber As Integer

    iAppNumber = 2
    bReturn = VisionQ400Control1. _
                     changeApplication(iAppNumber)

    If bReturn = False Then
      sErrorText = VisionQ400Control1.getLastErrorText()
      MsgBox (sErrorText)
      bApplicationSwitched = True
    Else
      bApplicationSwitched = False
    End If

End Sub


'Set the bApplicationSwitched Flag in the following sub
Private Sub VisionQ400Control1_SignalRecieved _
                    (ByVal signal As Long)

    If signal = VISIONQ400_SIG_CHANGE_NOTIFY Then
     bApplicationSwitched = True
    End If

End Sub


'Call StartApplication after receiving the
'Application switch signal
Private Sub Command3_Click()
    Dim bReturn As Boolean
    Dim bLockGrab As Boolean
    Dim sErrorText As String
    Dim iGroupNumber As Integer

    iGroupNumber = 0
    bLockGrab = False
    If bApplicationSwitched = True Then
      bReturn = VisionQ400Control1. _
                        startApplication(iGroupNumber, bLockGrab)
      If bReturn = False Then
          sErrorText = VisionQ400Control1. _
                        getLastErrorText()
          MsgBox (sErrorText)
        End If
    End If

End Sub
```

2.2.2 changeApplicationByName(applicationName)

The method **changeApplicationByName** starts the changing of the current application to the application applicationName.

Return Value

VARIANT_BOOL

The return value is TRUE if the methods succeeds, otherwise FALSE. If the return value is FALSE, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument	Type	Description
[IN] applicationName	BSTR	The file name of the new application.

Remarks

If there are current running starts of the “old” application, they are finished correctly and not interrupted before the current application is changed to the “new” application.

The old application is saved implicitly unless it is write protected. In this case it is not saved, and a warning may be written in the Vision Q.400 error log. (For details if a warning is written or not see the chapter error management in the Vision Q.400 manual.)

When the method returns, the change of an application is started, but not completed. The change is only completed when, on success, the signal VISIONQ400_SIG_CHANGE_NOTIFY (“Application switch completed”) is received, or when, on error, the signal VISIONQ400_SIG_PCERROR (“PC Error”) is sent. (For details for these synchronization matters, please have a look into the Vision Q.400 reference manual chapter “Interfaces -> Introduction”.) ”.) Please do use this synchrinization mechanism and not the method getState() to determine if the change of an application is finished.

If appNumber is the number of the “old” application, the application switch is performed, too.

The changing of an application depends on the timeout value, which is set in Vision Q.400 in “Applications -> Vision Q.400 Settings... -> Timeouts -> Application Change.” If the changing of an application takes longer than this timeout value, the changing is stopped, the application to be loaded is closed, and the signal VISIONQ400_SIG_PCERROR is sent. If getState() is called in this case, it will not return VISIONQ400_STATE_APPLICATION_LOAD.

The method changeApplicationByName fails in setup mode.

Visual Basic Example

```
Private Sub Command3_Click()  'Call Change Application
                                                                  'by Name
    Dim bReturn As Boolean
    Dim sErrorText As String
    Dim sApplicationName As String

    sApplicationName = _
                        "C:\Vision Q400\Applications\Application1.nav"

    bReturn = VisionQ400Control1. _
                         changeApplicationByName(sApplicationName)

    If bReturn = False Then
      sErrorText = VisionQ400Control1.getLastErrorText()
      MsgBox (sErrorText)
      bApplicationSwitched = True
    Else
      bApplicationSwitched = False
    End If

End Sub



'Set the bApplicationSwitched Flag in the following sub
Private Sub VisionQ400Control1_SignalRecieved _
                    (ByVal signal As Long)

    If signal = VISIONQ400_SIG_CHANGE_NOTIFY Then
      bApplicationSwitched = True
    End If

End Sub
```

2.2.3 getApplicationProperty(property)

The method **getApplicationProperty** returns the property given in the parameter property.

Return Value

BSTR

The desired property. If an error occurs, or if the desired property is not set, the empty string is returned. If this happens, you can call the method getLastErrorText()(or getLastErrorNumber()) to distinguish if an error occurred or if the given property is not set. If the property is not set, getLastErrorText() returns the empty string. If an error occurred, the returned string of getLastErrorText() is not empty. (getLastErrorNumber() returns zero if the property is not set. On error the returned value is not zero.)

Argument		Type	Description
[IN]	property	SHORT	See the Remarks section.

Remarks

The following properties are supported:

Name	Value	Meaning
VISIONQ400_PROP_PATH	1	the path of the currently loaded application file
VISIONQ400_PROP_NAME	2	the name of the currently loaded application
VISIONQ400_PROP_AUTHOR	3	the author of the currently loaded application
VISIONQ400_PROP_DESCRIPTION	4	the description of the currently loaded application
VISIONQ400_PROP_NUMBER	5	the number of the currently loaded application, if there is a number assigned (otherwise the result is -1)

The path of an application file does only exist if the application is saved at least one time.

The name, author, and description of an application are user defined properties. They can be defined in Vision Q.400 under “Application -> Properties… ->Description”.

The application number can be assigned under “Application -> Application Numbers…”.

Visual Basic Example

```
Option Explicit
Const VISIONQ400_PROP_PATH = 1
Const VISIONQ400_PROP_NAME = 2
Const VISIONQ400_PROP_AUTHOR = 3
Const VISIONQ400_PROP_DESCRIPTION = 4
Const VISIONQ400_PROP_NUMBER = 5


Private Sub Command3_Click() 'Call GetApplicationProperty()
    Dim sReturn As String
    Dim sErrorText As String
    Dim iProperty As Integer

    iProperty = VISIONQ400_PROP_PATH

    sReturn = VisionQ400Control1._
                        getApplicationProperty(iProperty)
    If TypeName(sReturn) = Empty Then
      sErrorText = VisionQ400Control1.getLastErrorText()
      MsgBox (sErrorText)
    Else
      Label1.Caption = sReturn
    End If

End Sub
```

2.2.4 getDependentFileNames(navFileName, resultValue)

The method **getDependentFileNames** returns a list of file names (e.g. the classifier file of an OCR checker), which are needed for the successful execution of the application given in navFileName.

The method works even if Vision Q.400 is not connected.

Return Value

VARIANT

This VARIANT contains the list of file names in an one Dimensional SAFEARRAY of type BSTR. The size of the array depends on the number of returned file names. The VARIANT is empty if the call of the method fails or if the application file does not contain dependent file information.

Argument		Type	Description
[IN]	navFileName	BSTR	The name of the Vision Q.400 application file.
[OUT]	resultValue	SHORT*	The state of the call of the method.

Remarks

resultValue can have the following values:

Value	Meaning
0	The call of the method failed.
-1	The call of the method succeeded, but the given application file does not contain dependent file information. To fix this the application has to be stored with Vision Q.400 at least once.
-2	The call of the method succeeded, but the given application file does not contain dependent file, because additional files are not needed for the successful execution of the application.
1	The call of the method succeeded and dependent file information was returned.

If resultValue is not equal to one, the returned VARIANT is empty.

If resultValue is not equal to one, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Visual Basic Example

```
Private Sub Command1_Click()

    Dim vFileNames As Variant
    Dim yReturn As Integer
    vFileNames = VisionQ400Control1.getDependentFileNames("ic-code.nav", yReturn)
    Select Case yReturn
        Case 0
            MsgBox ("Call of method failed")
        Case -1
            MsgBox ("The call of the method succeeded, but the given application" & _
            "file does not contain dependent file information, because it is" & _
            "created with a Vision Q.400 version which does not write this type" & _
            "of information (version number less than 1.7).")
        Case -2
            MsgBox ("The call of the method succeeded, but the given application" & _
            "file does not contain dependent file information, although it is" & _
            "created with a Vision Q.400 version which writes this type" & _
            "of information." & _
            "(Additional files are not needed for the" & _
            "successful execution of the application).")
    End Select
    MsgBox "Dependent File 1: & CStr(vFileNames(0))"

End Sub
```

2.2.5 openApplication(applicationName, saveLastOpen)

The method **openApplication** opens an application.

Return Value

VARIANT_BOOL

The return value is TRUE if the methods succeeds, otherwise FALSE. If the return value is FALSE, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument		Type	Description
[IN]	applicationName	BSTR	The name of the application to be opened.
[IN]	saveLastOpen	VARIANT_BOOL	TRUE: if an application is open when openApplication is called it is saved. FALSE: if an application is open when openApplication is called it is not saved.

Remarks

If the application, which is open on calling this method, has to be saved and was not saved earlier, a new name for the application is created. This name, e.g. „Application1.nav“, may not be unique. In this case, „Uqe“ is placed in front of the name, e.g. creating „UqeApplication1.nav“, until the name will become unique. A warning containing the created name is written into the error log file.

If openApplication returns, the opening of the application is already finished (or failed). This behaviour is in contrast to the behaviour of the methods changeApplication and changeApplicationByName, which return although the changing of the application is not finished yet. This difference is due to the fact that openApplication is a method of the setup mode of Vision Q.400 and the synchronisation mechanism of Vision Q.400 does not work in setup mode.

If several clients are connected to Vision Q.400, and one client calls openApplication, all other clients receive the signal VISIONQ400_SIG_APP_LOADED after Vision Q.400 has loaded the application. The client, which does call this method, does not receive the signal VISIONQ400_SIG_APP_LOADED.

The opening of an application depends on the timeout value, which is set in Vision Q.400 in “Applications -> Vision Q.400 Settings... -> Timeouts -> Application Change.” If the opening of an application takes longer than this timeout value, the opening is stopped, the application to be loaded is closed, the method returns FALSE, and the signal VISIONQ400_SIG_PCERROR is sent except to the client which called openApplication. If getState() is called in this case, it will not return VISIONQ400_STATE_APPLICATION_LOAD.

The method openApplication fails in run mode.

Visual Basic Example

```
Private Sub Command3_Click()
    Dim bReturn As Boolean
    Dim bSaveLastOpen As Boolean
    Dim sErrorText As String
    Dim sApplicationName As String

    bSaveLastOpen = True
    sApplicationName = _
            "C:\Vision Q400\Applications\Application1.nav"
    bReturn = _
          VisionQ400Control1.openApplication( _
          sApplicationName, bSaveLastOpen)
    If bReturn = False Then
      sErrorText = VisionQ400Control1.getLastErrorText()
      MsgBox (sErrorText)
    End If

End Sub
```

2.2.6 saveApplicationAs(fileName)

The method **saveApplicationAs** will save the current application under the name fileName.

Return Value

VARIANT_BOOL

The return value is TRUE if the methods succeeds, otherwise FALSE. If the return value is FALSE, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument		**Type**	**Description**
[IN]	filename	BSTR	The file name under which the current application will be saved.

Remarks

If saveApplicationAs returns, the saving of the application is already finished (or failed). This is due to the fact that saveApplicationAs is a method of the setup mode of Vision Q.400 and the synchronisation mechanism of Vision Q.400 does not work in setup mode.

The saving of an application depends on the timeout value, which is set in Vision Q.400 in “Applications -> Vision Q.400 Settings... -> Timeouts -> Application Change.” If the saving of an application takes longer than this timeout value, the method returns immedeatly returning FALSE, even if the saving is not completed.

The method saveApplicationAs fails in run mode.

Visual Basic Example

```
Private Sub Command3_Click()
    Dim bReturn As Boolean
    Dim sFileName As String
    Dim sErrorText As String

    sFileName = "C:\Test.nav"
    bReturn = VisionQ400Control1.saveApplicationAs(sFileName)
    If bReturn = False Then
      sErrorText = VisionQ400Control1.getLastErrorText()
      MsgBox (sErrorText)
    End If

End Sub
```

2.2.7 startApplication(groupNumber, lockGrab)

The method **startApplication** starts a new image grabbing and processing for the current application.

Return Value

VARIANT_BOOL

The return value is TRUE if the methods succeeds, otherwise FALSE. If the return value is FALSE, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument		**Type**	**Range**	**Description**
[IN]	groupNumber	SHORT	0..255	The number of an execution group.
[IN]	lockGrab	VARIANT_BOOL		Lock image grabbing.

Remarks

If groupNumber is 0, the whole application is started.

If groupNumber is greater than 0, and the execution group property “Allow group switching” is not activated, the group number is ignored, and the group given under the execution group property ”Use always this group” is executed. (Execution group matters are decribed in the Vision Q.400 reference manual in the chapter “Execution Groups”.)

If lockGrab is FALSE, the image grabbing is not locked, that means new images are grabbed before starting the processing.

If lockGrab is TRUE, the application is processed without image grabbing, that means, for the processing those images are used, which were grabbed by the last call of startApplication with the parameter lockGrab being FALSE.

If the method returns TRUE, the processing of the application is started, but not finished. For synchronisation, you have to set the sending and to wait for the appropriate signals. The sending of the approriate signals is set in Vision Q.400 “Interfaces -> OLE”. The synchronization matters of Vision Q.400 are described in the Vision Q.400 reference manual in the chapter “Interfaces -> Introduction”.

If the method returns FALSE, Vision Q.400 cannot accept a new start. This start is not delayed, but it is ignored. In this case, you may call getState() to get the value VISIONQ400_STATE_START_WILL_FAIL, which signals that all further calls of startApplication() will return FALSE until the cause of the failure will be removed by an approriate action.

If the method returns FALSE, the signal „Start Lost“ is sent to all Vision Q.400 interfaces, except to the client, which called the method.

The method startApplication fails in setup mode.

Visual Basic Example

```
Private Sub Command3_Click() 'Call StartApplication()
    Dim bReturn As Boolean
    Dim bLockGrab As Boolean
    Dim sErrorText As String
    Dim iGroupNumber As Integer

    iGroupNumber = 13
    bLockGrab = False
    bReturn = _
            VisionQ400Control1.startApplication( _
            iGroupNumber, bLockGrab)
    If bReturn = False Then
      sErrorText = VisionQ400Control1.getLastErrorText()
      MsgBox (sErrorText)
    End If
End Sub
```

2.2.8 stopAutoRestart()

The method **stopAutoRestart** stops the automatic restarting of the current application.

Return Value

VARIANT_BOOL

The return value is TRUE if the methods succeeds, otherwise FALSE. If the return value is FALSE, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument

none

Remarks

This method fails if Vision Q.400 is not in repetitive starting mode. The repetitiev starting mode is set in Vision Q.400 “Application -> Properties... -> Repetitive Start”.

The method stopAutoRestart fails in setup mode.

Visual Basic Example

```
Private Sub Command3_Click() 'Call StopAutoRestart
    Dim bReturn As Boolean
    Dim sErrorText As String

    bReturn = VisionQ400Control1.stopAutoRestart()
    If bReturn = False Then
      sErrorText = VisionQ400Control1.getLastErrorText()
      MsgBox (sErrorText)
    End If

End Sub
```

2.2.9 setUserData(userDataName, userData)

The method **setUserData** allows to store user data in the currently loaded application. Different user data can be stored under different user data names.

Return Value

VARIANT_BOOL

The return value is TRUE if the methods succeeds, otherwise FALSE. If the return value is FALSE, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument		Type	Description
[IN]	userDataName	BSTR	The user data name under which the current user data will be stored.
[IN]	userData	VARIANT	The data to be stored. For the allowed VARIANT types see the remarks section.

Remarks

The following VARIANT types are supported:

VT_EMPTY, VT_UI1, VT_UI2, VT_UI4, VT_I1, VT_I2, VT_I4, VT_R4, VT_R8, VT_UINT, VT_INT, VT_BOOL, and VT_BSTR.

For all supported VARIANT types (except of VT_EMPTY and VT_BSTR) , one or two dimensional arrays are allowed, too. For VT_EMPTY array are not allowed, for VT_BSTR only one dimensional arrays are allowed.

If the given VARIANT type is VT_EMPTY, the user data stored under userDataName will be cleared.
If the given VARIANT type is VT_EMPTY, and userDataName contains "*" (an asterix), all user data will be cleared.

If data is already stored under userDataName, this old data will be overwritten by the data given in userData.

The empty string, a "*" (the asterix), and strings starting with a '@' or not valid user data names.

2.2.10 getUserData(userDataName, userData)

The method **getUserData** allows to access user data, which is stored by a former call of setUserData() in the currently loaded application. Different user data may be stored under different user data names.

Return Value

VARIANT_BOOL

The return value is TRUE if the methods succeeds, otherwise FALSE. If the return value is FALSE, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument		Type	Description
[IN]	userDataName	BSTR	The data, which is stored under this user data name, will be returned.
[OUT]	userData	VARIANT*	The returned user data.

Remarks

If the method returns VARIANT_ FALSE, the VARIANT to which userData points to, is empty.

2.3 Spreadsheet Methods

2.3.1 changeResultName(oldName, newName)

The method **changeResultName** changes the name of a result in the spreadsheet. (This name is shown in the “Result name” column of the spreadsheet.)

Return Value

VARIANT_BOOL

The return value is TRUE if the methods succeeds, otherwise FALSE. If the return value is FALSE, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument		**Type**	**Description**
[IN]	oldName	BSTR	The name of the result which will be changed.
[IN]	newName	BSTR	The new name of the result.

Remarks

If the spreadsheet is updated in the run mode, the new name is shown after the next call of startApplication.

It the name of a formula is changed and one of the formula editors is visible, the editors may not correctly be updated to show the new name of the formula.

Visual Basic Example

```
Private Sub Command3_Click()
    Dim bReturn As Boolean
    Dim sErrorText As String
    Dim sOldName As String
    Dim sNewName As String

    sOldName = "WI[1;1]areaSize"
    sNewName = "AreaSize"
    bReturn = VisionQ400Control1. _
                            changeResultName(sOldName, sNewName)
    If bReturn = False Then
      sErrorText = VisionQ400Control1.getLastErrorText()
      MsgBox (sErrorText)
    End If

End Sub
```

2.3.2 getDataCount()

The method **getDataCount** returns the number of entries (rows) in the “Selected Spreadsheet Cells” list of the OLE interface. (To show this list in Vision Q.400, click on the OLE button of the spreadsheet.)

Return Value

LONG

The number entries (rows) in the “Selected Spreadsheet Cells” list of the OLE interface.

If an error occurs, the return value is –1. If this happens, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument

none

Visual Basic Example

```
Private Sub Command3_Click() 'Call GetDataCount()
    Dim sErrorText As String
    Dim vNamesOfSpreadColumn As Variant
    Dim lDataCount As Long

    lDataCount = VisionQ400Control1.getDataCount()
    If lDataCount = -1 Then
   sErrorText = VisionQ400Control1.getLastErrorText()
   MsgBox (sErrorText)
    Else
   Text1.Text = lDataCount
    End If

End Sub
```

2.3.3 getDataColoumnName(entryAt)

The method **getDataColoumnName** returns the “Column” name of the row with the index entryAt in the “Selected Spreadsheet Cells” list of the OLE interface. (To show this list in Vision Q.400, click on the OLE button of the spreadsheet.)

The indices start with one.

Return Value

BSTR

The desired column name. If an error occurs, the empty string is returned. If this happens, you can call the method getLastErrorText()(or getLastErrorNumber()) for further information.

Argument		**Type**	**Description**
[IN]	entryAt	LONG	The index of the desired column name in the Selected Spreadsheet Cells list.

Visual Basic Example

```
Private Sub Command3_Click() 'Call GetDataColoumnName()
    Dim sErrorText As String
    Dim sResultName As String
    Dim vNamesOfSpreadColumn As Variant
    Dim lDataCount As Long
    Dim lEntryAt As Long

    List1.Clear

    sResultName = VisionQ400Control1._
                               getDataColoumnName(lEntryAt)
    If sResultName = "" Then
      sErrorText = VisionQ400Control1.getLastErrorText()
      MsgBox (sErrorText)
    Else
      List1.AddItem (sResultName)
    End If

End Sub
```

2.3.4 getDataResultName(entryAt)

The method **getDataResultName** returns the “Result Name” of the row with the index entryAt in the “Selected Spreadsheet Cells” list of the OLE interface. (To show this list in Vision Q.400, click on the OLE button of the spreadsheet.)

The indices start with one.

Return Value

BSTR

The desired result name. If an error occurs, the empty string is returned. If this happens, you can call the method getLastErrorText() (or getLastErrorNumber()) for further information.

Argument		Type	Description
[IN]	entryAt	LONG	The index of the desired result name in the Selected Spreadsheet Cells list.

Visual Basic Example

```
Private Sub Command3_Click() 'Call GetDataResultName()
    Dim sErrorText As String
    Dim sResultName As String
    Dim vNamesOfSpreadColumn As Variant
    Dim lDataCount As Long
    Dim lEntryAt As Long

    List1.Clear
    LEntryAt = 2      'Get the name of the second
                                                      'entry in the OLE interface
                                                      'list
    sResultName = VisionQ400Control1._
                                   getDataResultName(lEntryAt)
    If sResultName = "" Then
      sErrorText = VisionQ400Control1.getLastErrorText()
      MsgBox (sErrorText)
    Else
      List1.AddItem (sResultName)
    End If
End Sub
```

2.3.5 getSpreadsheetColumnNames()

The method **getSpreadsheetColumnNames** returns the names of all spreadsheet columns.

Return Value

VARIANT

This VARIANT contains the list of column names in an one Dimensional SAFEARRAY of type BSTR. The size of the array depends on the number of spreadsheet columns. The first entry in the array is the name of column one (the “Result Name” column), the second the name of column two, and so on.

Argument

none

Remarks

The VARIANT is empty if the call of the method fails. If the VARIANT is empty, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Visual Basic Example

```
Private Sub Command3_Click()
                                                'Call GetSpreadsheetColumnNames()
    Dim sErrorText As String
    Dim vNamesOfSpreadColumn As Variant
    Dim lCounter As Long

    vNamesOfSpreadColumn = VisionQ400Control1. _
                                                         getSpreadsheetColumnNames()
    If TypeName(vNamesOfSpreadColumn) = "Empty" Then
      sErrorText = VisionQ400Control1.getLastErrorText()
      MsgBox (sErrorText)
    Else
    Text1.text = vNamesOfSpreadColumn(17)
                                                                'Text1.Text = Scans
                                                                'because the title of
                                                                'column number 17 is
                                                                '"Scans"

    End If

End Sub
```

2.3.6 getSpreadsheetData (rowName, colName, dataType, data)

The method **getSpreadsheetData** gets the data shown in a spreadsheet data cell.

Return Value

VARIANT_BOOL

The return value is TRUE if the methods succeeds, otherwise FALSE. If the return value is FALSE, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument		**Type**	**Description**
[IN]	rowName	BSTR	The name of the spreadsheet row for which the data is got.
[IN]	colName	BSTR	The name of the spreadsheet column for which the data is got.
[IN]	dataType	SHORT	The data type in which the data has to be returned.
[OUT]	data	VARIANT*	A pointer to an user supplied VARIANT in which the data of the desired data type is written.

Remarks

The parameter dataType can contain the following values:

Name	**Value**	**Meaning**
VISIONQ400_DATA_BOOLEAN	1	The data is returned as VT_BOOL
VISIONQ400_DATA_LONG	2	The data is returned as VT_I4
VISIONQ400_DATA_DOUBLE	4	The data is returned as VT_R8
VISIONQ400_DATA_STRING	8	The data is returned as VT_BSTR

The method getSpreadsheetData is not synchronised and it is not tested if the data to be returned is valid. For example if Vision400 is processing while this method is called, the value of an earlier execution of Vision Q.400 may be returned. Or if an error occurred during processing, a (not valid) value is returned nevertheless. The caller is responsible for synchronisation and validation of the data.

For synchronised data access and data validation, the appropriate ActiveX events have to be used, e.g. the event DataLongRecieved().

It is recommended to use getSpreadsheetData only for mostly “static” data, e.g. the lower and upper limits of a result.

Visual Basic Example

```
'Declaration of global variables
Option Explicit
    Const VISIONQ400_DATA_BOOLEAN = 1
    Const VISIONQ400_DATA_LONG = 2
    Const VISIONQ400_DATA_DOUBLE = 4
    Const VISIONQ400_DATA_STRING = 8


Private Sub Command3_Click() 'Call GetSpreadsheetData
    Dim bReturn As Boolean
    Dim sRowName As String
    Dim sColName As String
    Dim sErrorText As String
    Dim iDataType As Integer
    Dim vData As Variant

    sRowName = "WI[1;1]areaSize"
    sColName = "Lower Limit"
    iDataType = VISIONQ400_DATA_DOUBLE

    bReturn = VisionQ400Control1. _
      getSpreadsheetData(sRowName, sColName, iDataType, vData)
    If bReturn = False Then
      sErrorText = VisionQ400Control1.getLastErrorText()
      MsgBox (sErrorText)
    Else
      Label7.Caption = vData
    End If


End Sub
```

2.3.7 resetSpreadSheetStatistics()

The method **resetSpreadSheetStatistics** resets all spreadsheet statistics of the current application.

Return Value

VARIANT_BOOL

The return value is TRUE if the methods succeeds, otherwise FALSE. If the return value is FALSE, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument

none

Visual Basic Example

```
Private Sub Command3_Click() 'Call resetSpreadSheetStatistics()
    Dim bReturn As Boolean
    Dim sErrorText As String

    bReturn = VisionQ400Control1. _
                            resetSpreadsheetStatistics()
    If bReturn = False Then
      sErrorText = VisionQ400Control1.getLastErrorText()
      MsgBox (sErrorText)
    End If

End Sub
```

2.3.8 setCalibrationValue(attribName, valToSet)

The method **setCalibrationValue** changes the “Calibration” value of spreadsheet row with the “Result Name” attribName to the value valToSet in the spreadsheet of the current application. (The behaviour is the same as if the value is typed in in the appropriate spreadsheet cell.)

Return Value

VARIANT_BOOL

The return value is TRUE if the methods succeeds, otherwise FALSE. If the return value is FALSE, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument		Type	Description
[IN]	attribName	BSTR	The “Result Name” of a row in the spreadsheet.
[IN]	valToSet	DOUBLE	The new value of the calibration value.

Remarks

It is not possible to set the calibration value of an If Case formula.
It is not possible to set the calibration value of a Boolean formula.
It is not possible to set the the calibration value of a string type checker result, e.g. the result of an OCR checker.

If the spreadsheet is updated in the run mode, the new value is shown after the next call of startApplication.

Visual Basic Example

```
Private Sub Command3_Click()
    Dim bReturn As Boolean
    Dim sAttribName As String
    Dim sErrorText As String
    Dim dValToSet As Double

    sAttribName = "WI[1;1]areaSize"  'e.g. change limit
                                     'of window checker result
    dValToSet = 10
    bReturn = VisionQ400Control1. _
      setCalibrationValue(sAttribName, dValToSet)
    If bReturn = False Then
      sErrorText = VisionQ400Control1.getLastErrorText()
      MsgBox (sErrorText)
    End If

End Sub
```

2.3.9 setOcxReference(attribName, newRef)

The method **setOcxReference** sets the “String Reference” of the spreadsheet row with the “Result Name” attribName to the value newRef.

Return Value

VARIANT_BOOL

The return value is TRUE if the methods succeeds, otherwise FALSE. If the return value is FALSE, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument		Type	Description
[IN]	attribName	BSTR	The “Result Name” of a row in the spreadsheet.
[IN]	newRef	BSTR	The new value of the string reference.

Remarks

It is only possible to set the string reference of a string type checker result, e.g. the result of an OCR checker.

If the spreadsheet is updated in the run mode, the new reference is shown after the next call of startApplication.

Visual Basic Example

```
Private Sub Command3_Click() 'Call SetOcxReference
    Dim bReturn As Boolean
    Dim sAttribName As String
    Dim sErrorText As String
    Dim sNewRef As String

    sAttribName = "OCR[1;1]stringResult"
    sNewRef = "ABCD1234"

    bReturn = VisionQ400Control1. _
      setOcxReference(sAttribName, sNewRef)

    If bReturn = False Then
      sErrorText = VisionQ400Control1.getLastErrorText()
      MsgBox (sErrorText)
    End If


End Sub
```

2.3.10 setSpreadSheetLimit(attribName, valToSet, upper)

The method **setSpreadSheetLimit** sets the “Upper / Lower Limit” of the spreadsheet row with the “Result Name” attribName to the value valToSet in the spreadsheet of the current application.

Return Value

VARIANT_BOOL

The return value is TRUE if the methods succeeds, otherwise FALSE. If the return value is FALSE, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument		Type	Description
[IN]	attribName	BSTR	The “Result Name” of a row in the spreadsheet.
[IN]	valToSet	DOUBLE	The new value of the limit.
[IN]	upper	VARIANT_BOOL	TRUE: the upper limit is set. FALSE: the lower limit is set.

Remarks

It is not possible to set a new lower limit which is bigger then the current upper limit (or vice versa).

It is not possible to set a limit of an If Case formula.
It is not possible to set a limit of a Boolean formula.
It is not possible to set a limit of a string type checker result, e.g. the result of an OCR checker.

If the spreadsheet is updated in the run mode, the new limits are shown after the next call of startApplication.

Visual Basic Example

```
Private Sub Command3_Click()
    Dim bReturn As Boolean
    Dim sAttribName As String
    Dim sErrorText As String
    Dim dValToSet As Double
    Dim bUpper As Boolean

    sAttribName = "WI[1;1]areaSize"  'e.g. change limit
                                     'of window checker result
    bUpper = True      'TRUE: the upper limit is set
                       'FALSE: the lower limit is set
    dValToSet = 1000
    bReturn = VisionQ400Control1. _
            setSpreadsheetLimit(sAttribName, dValToSet, bUpper)
    If bReturn = False Then
      sErrorText = VisionQ400Control1.getLastErrorText()
      MsgBox (sErrorText)
    End If

End Sub
```

2.4 Image Methods

2.4.1 getImageSize(cameraNumber, numberOfCols, numberOfRows)

The method **getImageSize** returns the size of an image.

Return Value

VARIANT_BOOL

The return value is TRUE if the methods succeeds, otherwise FALSE. If the return value is FALSE, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument		Type	Range	Description
[IN]	cameraNumber	SHORT	[1..12]	The camera number for which the image the size is returned.
[OUT]	numberOfCols	SHORT*	.	The number of columns of the image
[OUT]	numberOfRows	SHORT*		The number of rows of the image.

Visual Basic Example

```
Private Sub Command3_Click() 'Call GetImageSize( . , . , . )
    Dim bReturn As Boolean
    Dim sErrorText As String
    Dim iCameraNumber As Integer
    Dim iNumberOfCols As Integer
    Dim iNumberOfRows As Integer

    iCameraNumber = Combo1.ListIndex + 1
    bReturn = VisionQ400Control1.getImageSize(iCameraNumber, _
                                     iNumberOfCols, iNumberOfRows)
    If bReturn = False Then
      sErrorText = VisionQ400Control1.getLastErrorText()
      MsgBox (sErrorText)
    Else
      Label1.Caption = iNumberOfCols
      Label2.Caption = iNumberOfRows
    End If

End Sub
```

2.4.2 setSendImage(cameraNumber)

The method **setSendImage** enables the automatic transfer of the image of the camera cameraNumber from Vision Q.400 to an OLE client by the event ImageAvaliable()., and it allows to access the image with the help of the method getImage().

After calling setSendImage, the automatic transfer of the image is enabled, and the image is automatically transferred at the end of every run mode execution of the current application.

The automatic transfer of the image can be switched off and on with the help of the property “SendImage” (see the methods setProperty() and getProperty()), but the image can be accessed always by the method getImage().

The method enables the transferring and the access of the **whole** image.

Return Value

VARIANT_BOOL

The return value is TRUE if the methods succeeds, otherwise FALSE. If the return value is FALSE, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument		Type	Range	Description
[IN]	cameraNumber	SHORT	[1..12]	The camera number for which the image is transferred or accessed.

Remarks

If you only want to transfer a part of the image, you have to use the method setSendImagePart() to enable the transferring and the access of a part of the image.

If the image should be zoomed, , the functions setZoomImageScale() or setZoomImageSize() have to be called **always** after calling setSendImage. (The call of setSendImage disables zooming.)

If the automatic transfer of the image should be switched off, the method setProperty() with the property "SendImage" and the value FALSE has to be called **always** after calling setSendImage. (The call of setSendImage enables the automatic transfer.)

If the camera cameraNumber is removed from the application , the image is not transferred and cannot be accessed furthermore.

If you change your application, the camera cameraNumber may not belong to the new application, and the image is not transferred and cannot be accessed furthermore.

If you change your application, the new image size (the size after changing the application) may be different from the image size in effect when calling setSendImage, and the image is not transferred and cannot be accessed furthermore. To overcame this behaviour, you have to call setSendImage again.

If the image is automatically transferred, you have to consider the following points:

- The image is only transferred in the run mode.
- To transfer the image, Vision Q.400 fires the event ImageAvaliable().
- If the camera cameraNumber does not belong to the currently executed execution group, the image of the camera is not transferred. (Execution group matters are described in the Vision Q.400 reference manual in the chapter "Execution Groups".)

In all cases, an error message is written into Vision Q.400's error log, and, if the image is automatically transferred, the event ImageAvailable() is sent with a negative camera number.

2.4.3 setSendImagePart(cameraNumber, startPointX, startPointY, width, height)

The method **setSendImagePart** enables the automatic transfer of the image of the camera cameraNumber from Vision Q.400 to an OLE client by the event ImageAvaliable()., and it allows to access the image with the help of the method getImage().

After calling setSendImagePart, the automatic transfer of the image is enabled, and the image is automatically transferred at the end of every run mode execution of the current application.

The automatic transfer of the image can be switched off and on with the help of the property "SendImage" (see the methods setProperty() and getProperty()), but the image can be accessed always by the method getImage().

The method enables the transferring and the access of **a part** of the image.

Return Value

VARIANT_BOOL

The return value is TRUE if the methods succeeds, otherwise FALSE. If the return value is FALSE, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument		**Type**	**Range**	**Description**
[IN]	cameraNumber	SHORT	[1..12]	The camera number for which the image is transferred or accessed.
[IN]	startPointX	SHORT	[> =0]	X coordinate of the top left point of the part to transfer.
[IN]	startPointY	SHORT	[> =0]	Y coordinate of the top left point of the part to transfer.
[IN]	width	SHORT	[> 0]	Number of columns to transfer starting at startPointX.
[IN]	height	SHORT	[> 0]	Number rows to transfer starting at startPointY.

Remarks

If you want to transfer an other part of the image, you can call the method with other parameter values again.

If the transferred image part should be zoomed, , the functions setZoomImageScale()or setZoomImageSize() have to be called **always** aftercalling setSendImagePart. (The call of setSendImagepart disables zooming.)

If the automatic transfer of the image should be switched off, the method setProperty() with the property "SendImage" and the value FALSE has to be called **always** after calling setSendImage. (The call of setSendImage enables the automatic transfer.)

If width is the number of columns of the image, and if height is the number of rows of the image, the method setSendImagePart behaves like the method setSendImage(). (But if you want to transfer the whole image, you should call the method setSendImage()).

If the camera cameraNumber is removed from the application , the image is not transferred and cannot be accessed furthermore.

If you change your application, the camera cameraNumber may not belong to the new application, and the image is not transferred and cannot be accessed furthermore.

If you change your application, the desired image part may not be totally inside the new image (the image after changing the application), and the image is not transferred and cannot be accessed furthermore. To overcame this behaviour, you have to call setSendImagePart again.

If the image is automatically transferred, you have to consider the following points:

- The image is only transferred in the run mode.
- To transfer the image, Vision Q.400 fires the event ImageAvaliable().
- If the camera cameraNumber does not belong to the currently executed execution group, the image of the camera is not transferred. (Execution group matters are described in the Vision Q.400 reference manual in the chapter "Execution Groups".)

In all cases, an error message is written into Vision Q.400's error log, and, if the image is automatically transferred, the event ImageAvailable() is sent with a negative camera number.

2.4.4 setZoomImageScale(cameraNumber, scaleX, scaleY, interpolaition)

The method **setZoomImageScale** enables the zooming of the transferred of image (part) of the camera cameraNumber.

You have to call setSendImage() or setSendImagePart() before you can call setZoomImageScale.

Return Value

VARIANT_BOOL

The return value is TRUE if the methods succeeds, otherwise FALSE. If the return value is FALSE, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument		Type	Range	Description
[IN]	cameraNumber	SHORT	[1..12]	The camera number for which the transferred image (part) is zoomed.
[IN]	scaleX	DOUBLE	[0.1..5.0]	Zoom factor in x direction.
[IN]	scaleY	DOUBLE	[0.1..5.0]	Zoom factor in y direction.
[IN]	interpolation	SHORT	[0..1]	Interpolation method for the zooming.

Remarks

If you want to change the values for the zooming, you can call the method with other parameter values again.

The parameter scaleX zooms the number of transferred columns into the range given by this parameter. E.g. if 100 columns of the image has to be transferred, and scaleX is 2.0, these 100 columns are zoomed into 200 columns. Or if scaleX is 0.5, these 100 columns are zoomed into 50 columns.

The parameter scaleY behaves the same for the transferred rows.

The parameter interpolation sets the method used for the zooming:

0: no interpolation, the grey values are skipped or doubled. This method is vary fast, but may be inaccurate.
1: an interpolation method is used. This method has a higher run time and a higher quality.

If the zooming is used Vision Q.400 adapts the number of transferred columns and rows set by setSendImage() or setSendImagePart() to the number needed after the zooming. You can get these new values by calling the method getProperty() with it's name parameter set to "TransferredImage".

2.4.5 setZoomImageSize(cameraNumber, width, height, interpolaition)

The method **setZoomImageSize** enables the zooming of the transferred of image (part) of the camera cameraNumber.

You have to call setSendImage() or setSendImagePart() before you can call setZoomImageSize.

Return Value

VARIANT_BOOL

The return value is TRUE if the methods succeeds, otherwise FALSE. If the return value is FALSE, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument		Type	Range	Description
[IN]	cameraNumber	SHORT	[1..12]	The camera number for which the transferred image (part) is zoomed.
[IN]	width	SHORT	[> 10]	Number of columns to be zoomed in (upper limit see Remarks).
[IN]	height	SHORT	[> 10]	Number of rows to be zoomed in (upper limit see Remarks).
[IN]	interpolation	SHORT	[0..1]	Interpolation method for the zooming.

Remarks

If you want to change the values for the zooming, you can call the method with other parameter values again.

The parameter width zooms the number of transferred columns into the range given by this parameter. E.g. if 100 columns of the image has to be transferred, and width is 200, these 100 columns are zoomed into 200 columns. Or if width is 50, these 100 columns are zoomed into 50 columns.

The upper limit of width is five times the width (the number of columns) of the image to be transferred.

The parameter height behaves the same for the transferred rows.

The parameter interpolation sets the method used for the zooming:

0: no interpolation, the grey values are skipped or doubled. This method is vary fast, but may be inaccurate.
1: an interpolation method is used. This method has a higher run time and a higher quality.

If the zooming is used Vision Q.400 adapts the number of transferred columns and rows set by setSendImage() or setSendImagePart() to the number needed after the zooming. You can get these new values by calling the method getProperty() with it's name parameter set to "TransferredImage".

2.4.6 getImage(cameraNumber)

With the method **getImage** the image of the camera cameraNumber can be accessed.

The method returns the last grabbed image of the camera.

In the run mode, the method fails if the grabbing of the accessed image is not finished yet.

In the setup mode, it is not tested if the grabbing of the accessed image is finished. Therefore if a grab of the camera occurs while this method is called, the returned image may be partly destroyed.

Return Value

VARIANT

The method returns an empty VARIANT if it fails. In this case, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument		**Type**	**Range**	**Description**
[IN]	cameraNumber	SHORT	[1..12]	The camera number for which the image is accessed.

Remarks

The method getImage can only be called if the access of an image is enabled by the methods setSendImage() or setSendImagePart().

If one of the methods setZoomImageScale()or setZoomImageSize() has been called, the returned image is appropriately zoomed.

The safe array in the VARIANT image can have one or two dimensions. (The dimension of the safe array can be changed by a call of the method setProperty() with the parameter name set to "ImageBufferDim".)

If the safe array has one dimension, it's lower bound is always 0, and it's upper bound is the number of columns multiplied with the number of minus 1. This means that the number of columns and the number of rows have to be known otherwise, e.g. by a call to the method getProperty() with it's name set to "TransferredImage".

If the safe array has two dimensions, the bounds of the first array dimension describe the columns, and the bounds of the second array dimension describe the rows of the image which are transferred. E.g. if the bounds of the first array dimension are [10, 90] and the bounds of the second dimension are [110, 190], only the image columns from 10 to 90 and the image rows from 100 to 190 are written. The desired image part must be totally inside the image. E.g. if the image has 512 columns, and the bounds of the first Dimension are [100, 550], an error is returned.

If the whole image is transferred, the lower bounds of the two array dimensions are always 0, and, without zooming, the upper bounds are the number of columns – 1 of the image, and the number of rows – 1 of the image, respectively.

If the transferred image (part) is zoomed, the upper bounds of the array dimensions may not be the values set by setSendImage() or setSendImagePart(): Vision Q.400 adapts the upper bounds to the values needed after the zooming. The lower bounds are left unchanged.

If the application has changed, and there was no grab with the current application, the image which was grabbed by the old application is returned.

2.4.7 removeSendImage(cameraNumber)

The method **removeSendImage** disables the automatic transfer of the image of the camera cameraNumber from Vision Q.400 to an OLE client: the image is not transferred furthermore.

Return Value

VARIANT_BOOL

The return value is TRUE if the methods succeeds, otherwise FALSE. If the return value is FALSE, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument		**Type**	**Range**	**Description**
[IN]	cameraNumber	SHORT	[1..12]	The camera number for which the transferred image is removed.

Visual Basic Example
```
Private Sub Command3_Click()   'Call RemoveImage()
    Dim bReturn As Boolean
    Dim sErrorText As String
    Dim iCameraNumber As Integer

    iCameraNumber = Combo3.ListIndex + 1

    bReturn = VisionQ400Control1.removeSendImage(iCameraNumber)
    If bReturn = False Then
      sErrorText = VisionQ400Control1.getLastErrorText()
      MsgBox (sErrorText)
    End If

End Sub
```

2.4.8 loadCameraImage(cameraNumber, fileName)

The method **loadCameraImage** loads an image for a selected camera of the Vision Q.400.

Return Value

VARIANT_BOOL

The return value is TRUE if the methods succeeds, otherwise FALSE. If the return value is FALSE, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument		**Type**	**Range**	**Description**
[IN]	cameraNumber	SHORT	[1..12]	The camera number for which the image is loaded.
[IN]	fileName	BSTR		The name of the file, which contains a 8 bit grey value bitmap.

Remarks

The method returns FALSE if the given file does not contain a valid 8 bit grey value bitmap.

If you want to apply the current checkers to the loaded image you have to avoid a new grab when executing a new start. Please refer to: startApplication()

If the camera cameraNumber   is in live mode, it will be set to memory mode before the image will be loaded.

This function only works if the Vision Q.400 is in setup mode.

Visual Basic Example

```
Private Sub Command3_Click()
    Dim bReturn As Boolean
    Dim sErrorText As String
    Dim iCameraNumber As Integer
    Dim sFileName As String

    iCameraNumber = 1
    sFileName = "C:\Temp\TestImage.bmp"

    bReturn = VisionQ400Control1.loadCameraImage(iCameraNumber, sFileName)
    If bReturn = False Then
      sErrorText = VisionQ400Control1.getLastErrorText()
      MsgBox (sErrorText)
    End If
End Sub
```

2.4.9 saveCameraImage(cameraNumber, fileName)

The method **saveCameraImage** is called to save the image of the selected camera to a bitmap file.

Return Value

VARIANT_BOOL

The return value is TRUE if the methods succeeds, otherwise FALSE. If the return value is FALSE, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument		**Type**	**Range**	**Description**
[IN]	cameraNumber	SHORT	[1..12]	The camera number for which the image is written.
[IN]	fileName	BSTR		The name of the file, which contains a 8 bit grey value bitmap.

Remarks

If the camera cameraNumber is in live mode, it will be set to memory mode before the image will be saved.

This function only works if the Vision Q.400 is in setup mode.

Visual Basic Example

```
Private Sub Command3_Click()
    Dim bReturn As Boolean
    Dim sErrorText As String
    Dim iCameraNumber As Integer
    Dim sFileName As String

    iCameraNumber = 1
    sFileName = "C:\Temp\ImageCam1.bmp"

    bReturn = VisionQ400Control1.saveCameraImage(iCameraNumber, sFileName)
    If bReturn = False Then
      sErrorText = VisionQ400Control1.getLastErrorText()
      MsgBox (sErrorText)
    End If
End Sub
```

2.5 Error Handling Methods

2.5.1 getLastErrorNumber()

The method **getLastErrorNumber** returns the number of the last error.

Return Value

LONG

The number of the last error, or if no error occurred, 0.

Argument

none

Remarks

You have to call this method immediately after an error occurred, otherwise a wrong (newer) error number may be returned.

This method has to be called before getLastErrorText, because getLastErrorText may clear the error number, but getLastErrorNumber does not.

Normally the explaining text got by getLastErrorText may be enough information, but sometimes the error number may be needed, e.g. if no application is loaded and the client wants to handle this situation. In this case, it is an good idea to get the error number first, to handle some of the numbers, and to call getLastErrorText for the not handled errors afterwards.

Important error numbers:

Value	**Meaning**
6606	No application is loaded
6655	The judgement of a result is “ERROR”.

Visual Basic Example

```
Private Sub Command3_Click()
    Dim bReturn As Boolean
    Dim lErrorNumber As Long

    'bReturn = VisionQ400Control1.__'Call of any Vision
                                                                        'Q.400 methode
    If bReturn = False Then
        lErrorNumber = VisionQ400Control1.getLastErrorNumber()
        Select Case lErrorNumber
            Case 6000
              MsgBox ("Vision Q.400 is not connected.")
            Case Else
        End Select
    End If

End Sub
```

2.5.2 getLastErrorText()

The method **getLastErrorText** returns an explaining text for the last error.

Return Value

BSTR

The explaining text of the last error, or if no error occurred, the empty string.

Argument

none

Remarks

You have to call this method immediately after an error occurred, otherwise the wrong (newer) error text may be returned.

This method may clear the error text on calling.

Visual Basic Example

```
Private Sub Command3_Click()
    Dim bReturn As Boolean
    Dim sErrorText As String

    'bReturn = VisionQ400Control1.__'Call of any Vision
                                                                          'Q.400 methode

    If bReturn = False Then
      sErrorText = VisionQ400Control1.getLastErrorText()
      MsgBox (sErrorText)
    End If

End Sub
```

2.6 getParameter(checkerName, parameterName)

The method **getParameter** returns the current value(s) of a checker parameter.

Return

VARIANT
For the current type and meaning of this VARIANT please refer the description of the concerning parameter.
The method returns an empty VARIANT if it fails. In this case, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument		**Type**	**Description**
[IN]	checkerName	BSTR	“ActiveX Name” of the checker e.g. "WI[1;1]".
[IN]	parameterName	BSTR	Name of the current parameter e.g. "Custom Slice Level".

Remarks

The “ActiveX Name” of an checker can be seen and changed on the checker’s general property page.

Visual Basic Example

The first example shows how to handle the ValueOfParameter if the received value is a single value. When we aim to get the value of the parameter e.g. Dynamic Slice Level the return value will be a data array where the upper and the lower limit will be stored. If we are trying to get the parameter value of the parameter Dynamic Slice Level Filter Size we will receive a single value which shows the filter size.

Example 1:

```
Private Sub Command3_Click()
    Dim sObjectName As String
    Dim sParameterName As String
    Dim vValueOfParameter As Variant
    Dim sErrorText As String
    Dim iReturn As Integer

    sObjectName = “WI[1;1]”
    sParameterName = “Dynamic Slice Level Filter Size”
    vValueOfParameter =  VisionQ400Control1. _
             GetParameter(sObjectName, sParameterName)

    sErrorText = VisionQ400Control1.getLastErrorText()
    If sErrorText <> "" Then
      MsgBox (sErrorText)
    Else
     iReturn = Cint(vValueOfParameter)
    End If
End Sub
```

Example 2:

```
Private Sub Command3_Click()
    Dim sObjectName As String
    Dim sParameterName As String
    Dim vValueOfParameter As Variant
    Dim sErrorText As String
    Dim iReturn1 As Integer
    Dim iReturn2 As Integer

    sObjectName = “WI[1;1]”
    sParameterName = “Dynamic Slice Level”
    vValueOfParameter =  VisionQ400Control1. _
             GetParameter(sObjectName, sParameterName)

    sErrorText = VisionQ400Control1.getLastErrorText()
    If sErrorText <> "" Then
      MsgBox (sErrorText)
    Else
      iReturn1 = CInt(vValueOfParameter(0))
      iReturn2 = CInt(vValueOfParameter(1))
    End If
End Sub
```

2.6.1 Checker

2.6.1.1 Common Parameters

The parameters, which are described here, are used by different checkers.

Parameter Name	Returned VARIANT Type	Comment
<The empty string>	VT_BSTR and VT_ARRAY	Returns the list of the names of all parameters, which are supported by the checker. If the checker does not support any parameters, the returned array contains one entry, the empty string.
Enable Execution	VT_BOOL -1 and 1 as ON, 0 as OFF	Returns if the execution of the checker is currently enabled (ON), or disabled (OFF).
Depends From Direct	VT_BSTR and VT_ARRAY	Returns the list of checker names, from which the checker checkerName directly depends. That means if one of the checkers in the list does depend on other checkers, these checkers are not in the list. If no checker is found, the returned array contains one entry, the empty string.
Depends From	VT_BSTR and VT_ARRAY	Returns the list of all checker names, from which the checker checkerName depends. If one of the checkers in the list does depend on other checkers in the list, it is inserted behind the checkers it depends from. If no checker is found, the returned array contains one entry, the empty string.

2.6.1.1.1 Thresholding

Parameter Name	Returned VARIANT Type	Comment
Custom Slice Level	VT_BOOL -1 and 1 as ON, 0 as OFF	Gets the current custom slice level setting.
Dynamic Slice Level	VT_I2 and VT_ARRAY (one Dimensional, two values) [0]: LowerLimit [1]: UpperLimit	Gets the current limits of the dynamic slice level It is not possible to access to this parameter if the static slice level is active.
Dynamic Slice Level Filter Size	VT_I2	Gets the filter current size of the dynamic slice level. It is not possible to access to this parameter if the static slice level is active.
Static Slice Level	VT_UI1 and VT_ARRAY (one Dimensional, two values) [0]: LowerLimit [1]: UpperLimit	Gets the current limits of the static slice level. It is not possible to access to this parameter if the dynamic slice level is active.
Use Static Slice Level	VT_BOOL -1 and 1 as Static Slice Level, 0 as Dynamic Slice Level	Gets the current static slice level setting.

2.6.1.1.2 Image Filters

The placeholder *Filter Name* in the following table can consist of:

- An image filter name, e.g. “Closing”
- An image filter name.
  If different image filters with the same name have to be distinguished in the image filter list, the name has to be modified by an index: name[*Index*]. The index starts with 1. E.g “Closing[2]” means the second closing filter in the list.

The placeholder *Parameter Name* in the following table can consist of:

- a parameter name, e.g. “Filter Size”

Parameter Name	Returned VARIANT Type	Comment	
Image Filter Gray List	VT_BSTR and VT_ARRAY	The list of gray filters currently selected for the checker.	If the requested list is empty, the returned array contains one entry, the empty string.
Image Filter Binary List		The list of binary filters currently selected for the checker.	
Image Filter Gray <*Filter Name* , Parameters>	VT_BSTR and VT_ARRAY	The names of all parameters of the image filter *Filter Name* . If the image filter does not have parameters, the returned array contains one entry, the empty string. To get the parameter names of an image filter, this filter has to be currently selected for the checker.	
Image Filter Binary <*Filter Name*, Parameters>			
Image Filter Gray <*Filter Name*, *Parameter Name* >	Depends on the requested parameter. If an enumeration parameter is requested, the current enumeration value is returned as VT_BSTR.	Current value of the requested parameter.	
Image Filter Binary <*Filter Name*, *Parameter Name* >			
Image Filter Gray <*Filter Name*, *Parameter Name* All>	VT_BSTR and VT_ARRAY	Only available for enumeration parameters. All possible values of the enumeration parameter *Parameter Name* of the image filter *Filter Name* . To get the values the Image filter filter *Filter Name* has to be currently selected for the checker.	
Image Filter Binary <*Filter Name*, *Parameter Name* All>			

2.6.1.2 Window Checker

Parameter Name	Returned VARIANT Type	Comment
Thresholding		All parameters described in Thresholding
Uses Image Filter Dll	VT_BOOL	If the return value is true, a customized image filter dll is used, false if not.
Image Filters		All parameters described in Image Filters

2.6.1.3 Feature Extraction Checker

Parameter Name	Returned VARIANT Type	Comment
Advanced Object Selection	VT_BOOL	If the return value is true, advanced object selection is used.
Advanced Object Selection Formula	VT_BSTR	The return value is the formula string used for the advanced object selection.
Boundary	VT_BOOL	If the return value is true, "Area Boundary" is set ON, false it is set OFF.
Thresholding		All parameters described in Thresholding
Uses Image Filter Dll	VT_BOOL	If the return value is true, a customized image filter dll is used, false if not.
Image Filters		All parameters described in Image Filters

2.6.1.4 Binary Edge Detection Checker

Parameter Name	Returned VARIANT Type	Comment
Width	VT_I4	The return value is the current “Width” setting of a binary edge detection checker. The edge width value is in the data range between 1 and 701.
Depth	VT_I4	The return value is the current “Depth” setting of a binary edge detection checker. The edge depth value is in the data range between 1 and 701.
Thresholding		All parameters described in Thresholding
Uses Image Filter Dll	VT_BOOL	If the return value is true, a customized image filter dll is used, false if not.
Image Filters		All parameters described in Image Filters

2.6.1.5 Gray Edge Detection Checker

Parameter Name	Returned VARIANT Type	Comment
Maximal Edge Width	VT_I4	The return value is the current minimum edge width of the gray edge detection checker. This Parameter is only available if the checker shape is a rectangle and if the edge type "connected" is selected.
Minimum Edge Length	VT_I4	The return value is the current minimum edge length of the gray edge detection checker.
Minimum Gradient	VT_I4	The return value is the current minimum gradient of the gray edge detection checker.

2.6.1.6 Difference Checker

Parameter Name	Returned VARIANT Type	Comment
Accuracy	VT_BSTR The return string is the current accuracy setting: "high", "medium" or "standard".	Gets the current setting of checker accuracy.
Template Rotation	VT_R8 and VT_ARRAY (one Dimensional, three values) [0] double Min Angle [1] double Max Angle [2] double Delta Angle	Gets the values for the pre-rotation of the template. Min Angle and Max Angle is between -180° and 360°. Maximum difference is 360°.
Slice Level	VT_I2 and VT_ARRAY (one dimensional, two values) [0]: short LowerLimit [1]: short UpperLimit	Gets the slice level settings of the checker. The limits are in the range between -254 and 254.
Selected Range	VT_BOOL	If the return value is TRUE "Process Selected Range" is active. If the return value is FALSE "Process not Selected Range" is active.
Uses Image Filter Dll	VT_BOOL	If the return value is true, a customized image filter dll is used, false if not.
Binary Image Filters		All binary parameters described in Image Filters

2.6.1.7 Contour Matching

Parameter Name	Returned VARIANT Type	Comment
Use Scale	VT_I4	If the return value is true, "Model Type" is set correctly. 0 = Standard 1 = Scale 2 = Anisotrope
Search Method	VT_R8	The return value is the current setting of the parameter “Search Method”
Maximum Overlap	VT_R8	The return value is the current setting of the parameter “Maximum Overlap”.
Angle Range	VT_R8 and VT_ARRAY (one Dimensional, two values) [0] double Start Angle [1] double Angle Range	This parameter retrieves the start angle, the angle extend of the checker. Sample: The result: Start: -30 Angle Extend: 60 means an angle range from -30 to 30 degrees.
Scale Range	VT_R8 and VT_ARRAY (one Dimensional, two values) [0] double MinScale [1] double MaxScale	Gets the current setting of the parameter “Scale Range”.
Scale Range Column	[VT_R8 and VT_ARRAY 0: Min Factor [0.1 .. 5.0] 1: Max Factor [0.1 .. 5.0]	Gets the current setting of the parameter “Scale Range Column” for ansiotrope scaling.

Parameter Name	Returned VARIANT Type	Comment
Object Appearance	VT_I4	This parameter retrieves the object appearance setting of the checker. 0 = like template 1 = like template or inverted 2 = dynamic
Minimum Correlation	VT_R8	Gets the current setting of the parameter “Minimum Correlation”.
Number Of Matches	VT_I4	Gets the current setting of the parameter number of matches.
Minimum Inspection Contrast	VT_I4	Gets the current setting of the parameter “Minimum Inspection Contrast” of the checker. (Former name of this parameter: "Minimum Contrast")
Maximum Model Contrast	VT_I4	Gets the current setting of the parameter “Maximum Model Contrast” of the checker.
Minimum Model Contrast	VT_I4	Gets the current setting of the parameter “Minimum Model Contrast” of the checker. (Former name of this parameter: "Contrast")
Last Level	VT_I4	Gets the current setting of the parameter “Last Compression Level”.
Sequences	VT_I4	Gets the current setting of the parameter “Compression Levels”
Area Boundary	VT_BOOL	If the return value is true, "Area Boundary" is set ON, if false it is set OFF.
Accuracy	VT_I4	Gets the current setting of the parameter “Accuracy”
Minimum Component Size	VT_I4	Gets the current setting of the parameter “Minimum Component Size” of the checker.
Optimize Large Models	VT_I4	Gets the current setting of the parameter “Optimize Large Models” of the checker.
Output Point	VT_BOOL	If the return value is true, " Output Point " is set to the center of the model, if false it is set to the center oft the template area.
Pregenerate Model	VT_BOOL	If the return value is true, " Pregenerate Model " is set ON, if false it is set OFF.
Maximal Allowed Deformation	VT_I4	Gets the current setting of the parameter “Maximal Allowed Deformation”.
Reference Point	[VT_R8 and VT_ARRAY 0: MinScale [0.0 .. 5000.0] 1: MaxScale [0.0 .. 5000.0]	Get the reference point of the checker.

2.6.1.8 Correlation Matching

Parameter Name	Returned VARIANT Type	Comment
Maximum Overlap	VT_R8	The return value is the current setting of the parameter “Maximum Overlap”.
Angle Range	VT_R8 and VT_ARRAY (one Dimensional, two values) [0] double Start Angle [1] double Angle Range	This parameter retrieves the start angle, the angle extend of the checker. Sample: The result: Start: -30 Angle Extend: 60 means an angle range from -30 to 30 degrees.

Parameter Name	Returned VARIANT Type	Comment
Object Appearance	VT_I4	This parameter retrieves the object appearance setting of the checker. 0 = like template 1 = like template or inverted
Minimum Correlation	VT_R8	Gets the current setting of the parameter “Minimum Correlation”.
Number Of Matches	VT_I4	Gets the current setting of the parameter number of matches.
Minimum Inspection Contrast	VT_I4	Gets the current setting of the parameter “Minimum Inspection Contrast” of the checker. (Former name of this parameter: "Minimum Contrast")
Last Level	VT_I4	Gets the current setting of the parameter “Last Compression Level”.
Sequences	VT_I4	Gets the current setting of the parameter “Compression Levels”
Area Boundary	VT_BOOL	If the return value is true, "Area Boundary" is set ON, if false it is set OFF.
Accuracy	VT_I4	Gets the current setting of the parameter “Accuracy”

2.6.1.9 OCR Checker

Parameter Name	Returned VARIANT Type	Comment
Use Classifier	VT_BOOL	If the return value is true, "Use Classifier" is set ON, if false it is set OFF.
Classifier Name	BSTR	Gets the name of the classifier of the selected checker.
Correlation Threshold	VT_I4	Gets the current “Correlation Threshold” setting of the checker.
Thresholding		All parameters described in Thresholding
Uses Image Filter Dll	VT_BOOL	If the return value is true, a customized image filter dll is used, false if not.
Image Filters		All parameters described in Image Filters

2.6.1.10Code Reader

Parameter Name	Returned VARIANT Type	Comment
Code Type	VT_BSTR	Gets the current “Code Type” setting of the checker.
Number Of Codes	VT_I4	Gets the current “Number of Codes” setting of the checker.
Number Of Loops	VT_I4	Gets the current “Number of Search Loops” setting of the checker. This setting is only valid for 2D code types.
Model Parameter Mode	VT_I4	Gets the current “Code Model Parameter” setting of the checker.
Pharma Code Reverse	VT_BOOL	If the return value is true, a pharma code is read in reverse direction, if false not.
Check Digit	VT_BOOL	Gets the current “Check Digit” setting of the checker.
2D Code Model Parameters – Only valid in “Manual” mode		
Min Contrast 2D	VT_I4	Gets the current “Minimal Contrast” setting of the checker.
Max Angle Variation ECC200 2D	VT_R8	Gets the current “Maximal Angle Variation” setting of the checker. This parameter is only valid for the ECC200 code.
Min Symbol Col 2D Max Symbol Col 2D	VT_I4	Gets the current “Minimal Symbol Columns” / “Maximal Symbol Columns” setting of the checker.
Min Symbol Row 2D Max Symbol Row 2D	VT_I4	Gets the current “Minimal Symbol Rows” / “Maximal Symbol Rows” setting of the checker.
Symbol Shape 2D	VT_BSTR	Gets the current “Symbol Shape” setting of the checker.
Mirrored 2D	VT_BSTR	Gets the current “Mirrored” setting of the checker.
Polarity 2D	VT_BSTR	Gets the current “Polarity” setting of the checker.
Module Size Min 2D Module Size Max 2D	VT_I4	Gets the current “Minimal Module Size” / “Maximal Module Size” setting of the checker.
Module Gap Col Max 2D Module Gap Col Min 2D	VT_BSTR	Gets the current “Minimal Module Gap” / “Maximal Module Gap” setting of the checker.
Module Gap Row Max 2D Module Gap Row Min 2D	VT_BSTR	Gets the current “Minimal Module Gap Row” / “Maximal Module Gap Row” setting of the checker.
Model Type QR 2D	VT_BSTR	Gets the current “QR Model Type” setting of the checker. This parameter is only valid for the QR code.
Version Min QR 2D Version Max QR 2D	VT_I4	Gets the current “Minimal Symbol Version” / Maximal Symbol Version” setting of the checker. This parameter is only valid for the QR code.
Persistance 2D	VT_I4	Gets the current “Persistence” setting of the checker.
Strict Model 2D	VT_BSTR	Gets the current “Strict Model” setting of the checker.
Small Module Robustness	VT_BSTR	Gets the current “Small Module Robustness” setting of the checker.

Module Width Max PDF417 2D Module Width Min PDF417 2D	VT_I4	Gets the current “Module Width Max” / Module Width Min” setting of the checker. This parameter is only valid for the PDF417 code.
Pattern Position Min QR 2D	VT_I4	Gets the current “Position Pattern Min” setting of the checker.
Module Aspect Min PDF417 2D Module Aspect Max PDF417	VT_I4	Gets the current “Module Aspect Max” / Module Aspect Min” setting of the checker. This parameter is only valid for the PDF417 code.

2D		
1D Code Model Parameters – Only valid in “Manual” mode		
Gray Image Filters		All gray parameters described in Image Filters
Min Size Element 1D	VT_R8	Gets the current value of the parameter “Minimal Element Size”.
Max Size Element 1D	VT_R8	Gets the current value of the parameter “Maximal Element Size”.
Min Element Height 1D	VT_I4	Gets the current value of the parameter “Minimum Element Height”.
Angle Range 1D	VT_I4	Gets the current value of the parameter “Angle Range”.
Measure Threshold 1D	VT_R8	Gets the current value of the parameter “Segmentation Threshold”.
Orientation 1D	VT_R8	Gets the current value of the parameter “Element Orientation”.
Orientation Tolerance 1D	VT_R8	Gets the current value of the parameter “Element Orientation Tolerance”.
Composite Code 1D	VT_BSTR	Gets the current value of the parameter “1D Composite Code”.

2.6.1.11 Edge Detection Gray Value Projection

Parameter Name	Returned VARIANT Type	Comment
Transition	VT_I4	Gets the current “Edge Transition” setting of the checker. All = 0 Dark - Light = 1 Light – Dark = 2

Select Edges	VT_I4	Gets the current “Edge Selection” setting of the checker.
		All = 0
		First = 1
		Last = 2
		First - Last = 3
		Maximum Gradient = 4

Execution Mode	VT_I4	Gets the current “Execution Mode” setting of the checker. Edge Pairs = 0, Edge Position = 1
Gradient	VT_I4	Gets the current “Minimal Gradient” setting of the checker.
Noise Level	VT_I4	Gets the current “Noise Level” setting of the checker.
Number Of Edges	VT_I4	Gets the current “Number of edges” setting of the checker.
Gray Image Filters		All gray parameters described in Image Filters

2.6.1.12 Identifier Checker

Parameter Name	Returned VARIANT Type	Comment
All Sample Images Directories	VT_BSTR and VT_ARRAY.	The currently available sample images directories. If a sample image directory is not available, the returned array contains one entry, the empty string.
Sample Images Directory	VT_BSTR	The currently selected sample images directory.

Parameter Name	Returned VARIANT Type	Comment
		If a sample image directory is not selected, “None” is returned.
Use Color Information	VT_BOOL -1 and 1 as “is used” 0 as “is not used”	Gets the current “Use Color Information” setting of the checker
Image Resize Method	VT_BSTR	Gets the current “Image Resize Method” setting of the checker.
Resulting Image Size	VT_R8	Gets the current “Resulting Image Size” setting of the checker. Can only be got if the “Image Resize Method” is set to “Resulting Image Size”.
Scaling Factor	VT_R8	Gets the current “Scaling Factor” setting of the checker. Can only be got if the “Image Resize Method” is set to “Scaling Factor”.
Subsampling Step	VT_R8	Gets the current “Subsampling Step” setting of the checker. Can only be got if the “Image Resize Method” is set to “Subsampling Step”.
Maximum Number of Result	VT_I4	Gets the current “Maximum Nuber Of Results” setting of the checker.
Rating Method	VT_BSTR	Gets the current “Rating Method” setting of the checker.
Rating Threshold	VT_R8	Gets the current “Rating Threshold” setting of the checker.

2.6.2 Shapes

If you change some shape parameters by calling the method setParameter, all other shape parameters may be changed implicitly, too. E.g. if you change the center point of a shape, all other shape points are moved, too. You can get the changed values by calling the method getParameter. (You can test the behaving of setParameter by changing some parameters in a checkers shape property page.)

The method setParameter fails if only one of the (implicitly) changed parameters lays outside it’ s range, e.g. all changed image points have to lay inside the image.

Some checkers, e.g. the Contour Mactching checker, have more than one shape. Please refer to the appropriate checker description for details.

Visual Basic Example

The first example shows how to get shape parameters from a checker and the second example shows how to set shape parameters.

Example 1:

```
Private Sub Command3_Click()
    Dim sObjectName As String
    Dim sParameterName As String
    Dim vShapeParameter As Variant
    Dim lStartEdge_X As Long
    Dim lStartEdge_Y As Long
    Dim lEndEdge_X As Long
    Dim lEndEdge_Y As Long

    sObjectName = "WI[1;1]"   'Rectangle Shape
    sParameterName = "Shape Points"
    vShapeParameter =  VisionQ400Control1. _
                GetParameter (sObjectName, sParameterName)
    If TypeName(vShapeParameter) = "Empty" Then
      MsgBox (VisionQ400Control1.getLastErrorText)
    Else
      lStartEdge_X = vShapeParameter(0, 0)
      lStartEdge_Y = vShapeParameter(0, 1)
      lEndEdge_X = vShapeParameter(1, 0)
      lEndEdge_Y = vShapeParameter(1, 1)
    End If

End Sub
```

Example 2:

```
    Dim sObjectName As String
    Dim sParameterName As String
    Dim lStartEdge_X As Long
    Dim lStartEdge_Y As Long
    Dim lEndEdge_X As Long
    Dim lEndEdge_Y As Long
    Dim lShapeParameter(0 To 2, 0 To 1) As Long
    Dim lEdge_1_X As Long
    Dim lEdge_1_Y As Long
    Dim lEdge_2_X As Long
    Dim lEdge_2_Y As Long
    Dim lEdge_3_X As Long
    Dim lEdge_3_Y As Long

    lEdge_1_X = 1
    lEdge_1_Y = 1
    lEdge_2_X = 500
    lEdge_2_Y = 1
    lEdge_3_X = 250
    lEdge_3_Y = 400

    lShapeParameter(0, 0) = lEdge_1_X
    lShapeParameter(0, 1) = lEdge_1_Y
    lShapeParameter(1, 0) = lEdge_2_X
    lShapeParameter(1, 1) = lEdge_2_Y
    lShapeParameter(2, 0) = lEdge_3_X
    lShapeParameter(2, 1) = lEdge_3_Y

    sObjectName = "WI[1;1]"   'Poligon Shape with 3 Points
    sParameterName = "Shape Points"
    bReturn = VisionQ400Control1. _
      setParameter(sObjectName, sParameterName, vShapeParameter)
    If bReturn = False Then
      MsgBox (VisionQ400Control1.getLastErrorText)
    End If

End Sub
```

2.6.2.1 Common Parameters

The parameters described in the following table are used by all shapes.

Parameter Name	Returned VARIANT Type	Comment
Shape Type	VT_I4	The type of the shape: 0x00: Rectangle 0x01: Ellipse (Circle) 0x03: Polygon 0x07: Doughnut

		0x0A: Object Shape 0x10: Rectangle -> 0x11: Ellipse -> (Circle ->) 0x12: Line -> 0x13: Doughnut ->

2.6.2.2 Shape Line ->

Parameter Name	Returned VARIANT Type	Comment
Shape Points	VT_I4 | VT_ARRAY 2 Points	Point 1: Start Point Point 2: End Point
Shape Start Point	VT_I4 | VT_ARRAY 1 Point	Start Point
Shape End Point	VT_I4 | VT_ARRAY 1 Point	End Point
Shape Center Point	VT_I4 | VT_ARRAY 1 Point	Center Point
Shape Direction	VT_R4	Angle of the line Range: 0 <= angle <= 360

Visual Basic Example: Shape Points

```
Private Sub Command1_Click()

    Dim bReturn As Boolean
    Dim sObjectName As String
    Dim sParameterName As String
    Dim vShapeParameter(0 To 1, 0 To 1) As Long
    Dim lStartEdge_X As Long
    Dim lStartEdge_Y As Long
    Dim lEndEdge_X As Long
    Dim lEndEdge_Y As Long
        lStartEdge_X = 5
        lStartEdge_Y = 5
        lEndEdge_X = 400
        lEndEdge_Y = 400
        vShapeParameter(0, 0) = lStartEdge_X
        vShapeParameter(0, 1) = lStartEdge_Y
        vShapeParameter(1, 0) = lEndEdge_X
        vShapeParameter(1, 1) = lEndEdge_Y
        sObjectName = "ED_B[1;1]" 'Line Shape
        sParameterName = "Shape Points"
        bReturn = VisionQ400Control1. _
        setParameter(sObjectName, sParameterName, vShapeParameter)
    If bReturn = False Then
        MsgBox (VisionQ400Control1.getLastErrorText)
    End If

End Sub
```

Visual Basic Example: Center Point

```
Private Sub Command1_Click()

    Dim bReturn As Boolean
    Dim sObjectName As String
    Dim sParameterName As String
    Dim lShapeParameter(0, 0 To 1) As Long
    Dim lCenterPoint_x As Long
    Dim lCenterPoint_y As Long
        lCenterPoint_x = 150    'x-Coordinate
        lCenterPoint_y = 150    'y-Coordinate
        lShapeParameter(0, 0) = lCenterPoint_x
        lShapeParameter(0, 1) = lCenterPoint_y
        sObjectName = "ED_B[1;1]" 'Line Shape
        sParameterName = "Shape Center Point"
        bReturn = VisionQ400Control1. _
        setParameter(sObjectName, sParameterName, lShapeParameter)
    If bReturn = False Then
        MsgBox (VisionQ400Control1.getLastErrorText)
    End If

End Sub
```

Visual Basic Example: Direction

```
Private Sub Command1_Click()

    Dim bReturn As Boolean
    Dim sObjectName As String
    Dim sParameterName As String
    Dim lShapeParameter As Long
        lShapeParameter = 180    ' Angle
        sObjectName = "ED_B[1;1]" ' Line Shape
        sParameterName = "Shape Direction"
        bReturn = VisionQ400Control1. _
        setParameter(sObjectName, sParameterName, lShapeParameter)
    If bReturn = False Then
        MsgBox (VisionQ400Control1.getLastErrorText)
    End If

End Sub
```

2.6.2.3 Shape Rectangle

Parameter Name	Returned VARIANT TYPE	Comment
Shape Start Point	VT_I4 | VT_ARRAY 1 Point	Start Point The Start Point equals to the top left point of the rectangle.
Shape End Point	VT_I4 | VT_ARRAY 1 Point	End Point The End Point equals to the bottom right point of the rectangle.
Shape Center Point	VT_I4 | VT_ARRAY 1 Point	Center Point

Visual Basic Example

```
Private Sub Command1_Click()

    Dim bReturn As Boolean
    Dim sObjectName As String
    Dim sParameterName As String
    Dim lShapeParameter(0 To 1, 0 To 1) As Long
    Dim lStartEdge_X As Long
    Dim lStartEdge_Y As Long
    Dim lEndEdge_X As Long
    Dim lEndEdge_Y As Long
        lStartEdge_X = 5
        lStartEdge_Y = 5
        lEndEdge_X = 400
        lEndEdge_Y = 400
        lShapeParameter(0, 0) = lStartEdge_X
        lShapeParameter(0, 1) = lStartEdge_Y
        lShapeParameter(1, 0) = lEndEdge_X
        lShapeParameter(1, 1) = lEndEdge_Y
        sObjectName = "WI[1;1]" 'Rectangle Shape
        sParameterName = "Shape Points"
        bReturn = VisionQ400Control1. _
        setParameter(sObjectName, sParameterName, lShapeParameter)
    If bReturn = False Then
        MsgBox (VisionQ400Control1.getLastErrorText)
    End If

End Sub
```

2.6.2.4 Shape Rectangle ->

Parameter Name	Returned VARIANT TYPE	Comment
Shape Start Point	VT_I4 | VT_ARRAY 1 Point	Start Point The Start Point equals to the top left point of the rectangle.
Shape End Point	VT_I4 | VT_ARRAY 1 Point	End Point The End Point equals to the bottom right point of the rectangle.
Shape Center Point	VT_I4 | VT_ARRAY 1 Point	Center Point
Shape Direction	VT_R4	Direction of the rectangle Range: 0 <= angle <= 360
Shape Direction Point	VT_I4 | VT_ARRAY 1 Point	Point to change the direction of the rectangle. This point is not supported in a checkers shape properties page.

2.6.2.5 Shape Ellipse (Circle)

Parameter Name	Returned VARIANT TYPE	Comment
Shape Points	VT_I4 | VT_ARRAY 2 Points	Point 1: Top Left Point Point 2: Bottom Right Point
Shape Top Left Point	VT_I4 | VT_ARRAY 1 Point	Top Left Point of the surrounding rectangle
Shape Bottom Right Point	VT_I4 | VT_ARRAY 1 Point	Bottom Right Point of the surrounding Rectangle.
Shape Center Point	VT_I4 | VT_ARRAY 1 Point	Center Point

2.6.2.6 Shape Ellipse -> (Circle ->)

Parameter Name	Returned VARIANT TYPE	Comment
Shape Points	VT_I4 | VT_ARRAY 2 Points	Point 1: Top Left Point Point 2: Bottom Right Point
Shape Top Left Point	VT_I4 | VT_ARRAY 1 Point	Top Left Point of the surrounding rectangle
Shape Bottom Right Point	VT_I4 | VT_ARRAY 1 Point	Bottom Right Point of the surrounding Rectangle.
Shape Center Point	VT_I4 | VT_ARRAY 1 Point	Center Point
Shape Start Point	VT_I4 | VT_ARRAY 1 Point	Start point of the part of the shape The given point has not to lay exactly on the shape. If not the intersection of the shape and the line given by the new point and the center point is the new Start Point.
Shape End Point	VT_I4 | VT_ARRAY 1 Point	End point of the part of the shape See remark of Start Point.
Shape Direction	VT_BOOL	Direction for the shape 0: clockwise 1: counterclockwise
Shape Direction Point	VT_I4 | VT_ARRAY 1 Points	Point to change the direction of the shape This point is not supported in a checkers shape properties page.

2.6.2.7 Shape Doughnut

Parameter Name	Returned VARIANT TYPE	Comment	
Shape Outer Circle Point	VT_I4 | VT_ARRAY 1 Point	Top Left point of the surrounding rectangle of the outer circle. The Bottom Right point of the surrounding rectangle is the point, which is point-symmetric to the given point at the Center Point.	
Shape Inner Circle Point	VT_I4 | VT_ARRAY 1 Point	Top Left point of the surrounding rectangle of the inner circle. See comment of Shape Outer Circle Point.	
Shape Center Point	VT_I4 | VT_ARRAY 1 Point	Center Point	
Shape Start Point	VT_I4 | VT_ARRAY 1 Points	Start point of the part of the shape. The given point has not to lay exactly on the shape. If not the intersection of the shape and the line given by the new point and the center point is the new Start Point.	
Shape End Point	VT_I4 | VT_ARRAY 1 Points	End point of the part of the shape. See remark of Start Point.	

Visual Basic Example: Shape Center Point

```
Private Sub Command1_Click()

    Dim bReturn As Boolean
    Dim sObjectName As String
    Dim sParameterName As String
    Dim lShapeParameter(0, 0 To 1) As Long
    Dim lCenterPoint_x As Long
    Dim lCenterPoint_y As Long
        lCenterPoint_x = 250    'x-Coordinate
        lCenterPoint_y = 250    'y-Coordinate
        lShapeParameter(0, 0) = lCenterPoint_x
        lShapeParameter(0, 1) = lCenterPoint_y
        sObjectName = "WI[1;1]" 'Doughnut Shape
        sParameterName = "Shape Center Point"
        bReturn = VisionQ400Control1. _
        setParameter(sObjectName, sParameterName, lShapeParameter)
    If bReturn = False Then
        MsgBox (VisionQ400Control1.getLastErrorText)
    End If

End Sub
```

Visual Basic Example: Shape Outer Circle Point

```
Private Sub Command1_Click()

    Dim bReturn As Boolean
    Dim sObjectName As String
    Dim sParameterName As String
    Dim lShapeParameter(0, 0 To 1) As Long
    Dim lOuterCirclePoint_x As Long
    Dim lOuterCirclePoint_y As Long
        lOuterCirclePoint_x = 200    'x-Coordinate
        lOuterCirclePoint_y = 200    'y-Coordinate
        lShapeParameter(0, 0) = lOuterCirclePoint_x
        lShapeParameter(0, 1) = lOuterCirclePoint_y
        sObjectName = "WI[1;1]" 'Doughnut Shape
        sParameterName = "Shape Outer Circle Point"
        bReturn = VisionQ400Control1. _
        setParameter(sObjectName, sParameterName, lShapeParameter)
    If bReturn = False Then
        MsgBox (VisionQ400Control1.getLastErrorText)
    End If

End Sub
```

2.6.2.8 Shape Doughnut ->

Parameter Name	Returned VARIANT TYPE	Comment
Shape Outer Circle Point	VT_I4 | VT_ARRAY 1 Point	Top Left point of the surrounding rectangle of the outer circle. The Bottom Right point of the surrounding rectangle is the point, which is point-symmetric to the given point at the Center Point.
Shape Inner Circle Point	VT_I4 | VT_ARRAY 1 Point	Top Left point of the surrounding rectangle of the inner circle. See comment of Shape Outer Circle Point.
Shape Center Point	VT_I4 | VT_ARRAY 1 Point	Center Point
Shape Start Point	VT_I4 | VT_ARRAY 1 Points	Start point of the part of the shape. The given point has not to lay exactly on the shape. If not the intersection of the shape and the line given by the new point and the center point is the new Start Point.
Shape End Point	VT_I4 | VT_ARRAY 1 Points	End point of the part of the shape. See remark of Start Point.

Parameter Name	Returned VARIANT TYPE	Comment
Shape Direction	VT_BOOL	Direction for the shape 0: clockwise 1: counterclockwise

2.6.2.9Shape Polygon

Parameter Name	Returned VARIANT Type	Comment
Shape Points	VT_I4 | VT_ARRAY n Points	A list of points which describes the polygon. A maximum of 256 points is allowed.
Shape Center Point	VT_I4 | VT_ARRAY 1 Points	Center Point

Visual Basic Example

```
Private Sub Command1_Click()

    Dim bReturn As Boolean
    Dim sObjectName As String
    Dim sParameterName As String
    Dim lShapeParameter(0 To 2, 0 To 1) As Long
    Dim lPoint1_x As Long
    Dim lPoint1_y As Long
    Dim lPoint2_x As Long
    Dim lPoint2_y As Long
    Dim lPoint3_x As Long
    Dim lPoint3_y As Long
        lPoint1_x = 100    'x-Coordinate
        lPoint1_y = 100    'y-Coordinate
        lPoint2_x = 200
        lPoint2_y = 350
        lPoint3_x = 300
        lPoint3_y = 300
        lShapeParameter(0, 0) = lPoint1_x
        lShapeParameter(0, 1) = lPoint1_y
        lShapeParameter(1, 0) = lPoint2_x
        lShapeParameter(1, 1) = lPoint2_y
        lShapeParameter(2, 0) = lPoint3_x
        lShapeParameter(2, 1) = lPoint3_y
        sObjectName = "WI[1;1]" 'Shape Polygon
        sParameterName = "Shape Points"
        bReturn = VisionQ400Control1. _
        setParameter(sObjectName, sParameterName, lShapeParameter)
    If bReturn = False Then
        MsgBox (VisionQ400Control1.getLastErrorText)
    End If

End Sub
```

2.6.2.10Object Shape

Parameter Name	Returned VARIANT Type	Comment
Shape Points	VT_ARRAY | VT_I4 (3 * number of runs) + 1 points a[0] : 3 * number of runs 0 <= i < number of runs: a[3*i+1]: row number of the run i a[3*i+2]: start column number of the run i a[3*i+3]: end column number of the run i	The shape is returned in so called runs. A run is a triple with the following meaning: run[0]: row number of the run run[1]: start column number of the run run[2]: end column number of the run

2.6.2.11 Additional Shapes

2.6.2.11.1 Template Shape

The contour matching and the correlation matching checkers have two shapes. To get the values of the “Search Area” shape, the shape parameter names are used. E.g. the parameter name “Shape Center Point” returns the center point of the Search Area.

To get the values of the “Template” shape, the string “Template “ has to be inserted before the shape parameter name. E.g. the parameter name “Template Shape Center Point” returns the center point of the Template.

2.7 setParameter(checkerName, parameterName, parametervalues)

The method **setParameter** sets the current value(s) of a checker parameter.

Return

VARIANT_BOOL

If the return value is TRUE setting the parameter was successful.
If the return value is FALSE any error occurred. In this case, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument		**Type**	**Description**
[IN]	checkerName	BSTR	ActiveX name of the checker e.g. "WI[1;1]".
[IN]	parameterName	BSTR	Name of the current parameter e.g. "Custom Slice Level".
[IN]	parameterValues	VARIANT	For the current type and meaning please refer the description of the concerning parameter.

Remarks

The “ActiveX Name” of an checker can be seen and changed on the checker’s general property page.

If possible, the method setParameter() converts the data given in parameterValues automatically in the requested data type. For further information about this please refer to Appendix A:VARIANT Type Conversion in Vision Q.400.

The method setParameter does not switch the ready state of Vision Q.400 do off. That means that Vision Q.400 can be started before the method setParameter will return. It is strictly recommended not to start Vision Q.400 before the method will return. If this situation is possible, you should use the method interruptStartSignals() to forbid the starting of Vision Q.400 before the method setParameter is called. And you should use the method interruptStartSignals() afterwards to allow the starting of Vision Q.400 again.

Visual Basic Example

The first example shows how to handle the method if the parameter is a single value (e.g. “Dynamic Slice Level Filter Size” + filter value). The second example shows how to handle the function if more than one additional parameter is sent to the function (e.g. "Static Slice Level" + Lower Limit + Upper Limit)

Example 1:

```
Private Sub Command3_Click()
    Dim sObjectName As String
    Dim sParameterName As String
    Dim bReturn As Boolean
    Dim sErrorText As String
    Dim lFilterValue As Long

    sObjectName = “WI[1;1]”
    sParameterName = “Dynamic Slice Level Filter Size”
    lFilterValue = 501
    bReturn =  VisionQ400Control1. _
      SetParameter(sObjectName, sParameterName, lFilterValue)
    If bReturn = False Then
      SErrorText = VisionQ400Control1.getLastErrorText()
      MsgBox (sErrorText)
    End If

End Sub
```

Example 2:

```
Private Sub Command3_Click()
    Dim sObjectName As String
    Dim sParameterName As String
    Dim bReturn As Boolean
    Dim sErrorText As String
    Dim yLowUpSliceLevel(0 To 1) As Byte

    sObjectName = “WI[1;1]”
    sParameterName = "Static Slice Level"
    yLowUpSliceLevel(0) = 10
    yLowUpSliceLevel(1) = 200
    bReturn =  VisionQ400Control1. _
      SetParameter(sObjectName, sParameterName, yLowUpSliceLevel)
    If bReturn = False Then
      SErrorText = VisionQ400Control1.getLastErrorText()
      MsgBox (sErrorText)
    End If

End Sub
```

2.7.1 General Remarks

If the data range of a parameter is limited, the limitation is given in the column "Parameter VARIANT Type" of the following tables.

If the parameter is only limited on "both" sides, the format is given in the following table:

Format	Meaning
[lower limit .. upper limit]	lower limit <= allowed value <= upper limit
(lower limit .. upper limit]	lower limit < allowed value <= upper limit
[lower limit .. upper limit)	lower limit <= allowed value < upper limit
(lower limit .. upper limit)	lower limit < allowed value < upper limit

If the parameter is only limited on "one" side, the format is given in the following table:

Format	Meaning
(>= allowed value)	
(> allowed value)	
(<= allowed value)	
(< allowed value)	

2.7.2 Checker

2.7.2.1 Common Parameters

The parameters, which are described here, are used by different checkers.

Parameter Name	Parameter VARIANT Type	Comment
Enable Execution	VT_BOOL -1 and 1 as ON, 0 as OFF	Sets if the execution of the checker should be enabled (ON), or disabled (OFF).
Execute	No parameter value	With this parameter you can execute a checker. This works only if Vision Q.400 is in the setup mode.

2.7.2.1.1 Thresholding

Parameter Name	Parameter VARIANT Type	Comment
Custom Slice Level	VT_BOOL -1 and 1 as ON, 0 as OFF	With this parameter you can activate and deactivate the usage of custom slice level.
Dynamic Slice Level	VT_I2 and VT_ARRAY (one Dimensional, two values) [0]: LowerLimit [- 254 .. 254] [1]: UpperLimit [- 254 .. 254]	Changes the dynamic slice level values. This parameter cannot be accessed if the static slice level is active.
Dynamic Slice Level Filter Size	VT_I2 [3..501] The value has to be odd.	The filter size of the dynamic slice level. The filter size should be at least twice as big as the maximal size of the target object is on the X-axis

Parameter Name	Parameter VARIANT Type	Comment
		and the Y-axis. This parameter cannot be accessed if the static slice level is active.
Static Slice Level	VT_UI1 and VT_ARRAY (one Dimensional, two values) [0]: LowerLimit [0 .. 255] [1]: UpperLimit [0 .. 255]	Set the limits of the static slice level. This parameter cannot be accessed if the dynamic slice level is active.
Use Static Slice Level	VT_BOOL -1 and 1 as Static Slice Level, 0 as Dynamic Slice Level	With this parameter you can between the usage of the static and the dynamic slice level.

2.7.2.1.2 Image Filters

The placeholder *Filter Name* in the following table can consist of:

- An image filter name, e.g. "Closing"
- An image filter name.
  If different image filters with the same name have to be distinguished in the image filter list, the name has to be modified by an index: name[*Index*]. The index starts with 1. E.g "Closing[2]" means the second closing filter in the list.

The placeholder *Parameter Name* in the following table can consist of:

- a parameter name, e.g. "Filter Size"

Parameter Name	Parameter VARIANT Type	Comment
Image Filter Gray <*Filter Name*, *Parameter Name*>	Depends on the parameter. If an enumeration parameter is set, the current enumeration value has to be VT_BSTR.	Sets the current value of the given parameter.
Image Filter Binary <*Filter Name*, *Parameter Name*>		

2.7.2.2 Window Checker

Parameter Name	Parameter VARIANT Type	Comment
Thresholding		All parameters described in Thresholding
Image Filters		All parameters described in Image Filters

2.7.2.3 Feature Extraction Checker

Parameter Name	Parameter VARIANT Type	Comment
Advanced Object Selection Formula	VT_BSTR	The formula string used for the advanced object selection.
Boundary	VT_BOOL -1 and 1 as ON, 0 as OFF	Use this parameter to define whether or not an object to be processed will be allowed to touch the checker shape.
Thresholding		All parameters described in Thresholding

Parameter Name	Parameter VARIANT Type	Comment
Image Filters		All parameters described in Image Filters

2.7.2.4 Binary Edge Detection Checker

Parameter Name	Parameter VARIANT Type	Comment
Width	VT_I4 [1.. **Upper Limit**]	Sets the current “Width” of a binary edge detection checker. The **Upper Limit** depends on the current image size.
Depth	VT_I4 [1.. **Upper Limit**]	Sets the current “Depth” of a binary edge detection checker. The **Upper Limit** depends on the current image size.
Thresholding		All parameters described in Thresholding
Image Filters		All parameters described in Image Filters

2.7.2.5 Gray Edge Detection Checker

Parameter Name	Parameter VARIANT Type	Comment
Maximal Edge Width	VT_I4 (>=0)	This Parameter is only available if the checker shape is a rectangle and if the edge type "connected" is selected.
Minimum Edge Length	VT_I4 (>= 0)	This Parameter is only available if the checker shape is a rectangle. Minimum Edge Length defines the minimal length which an edge piece must have in order to be considered in the calculation of the edge representative.
Minimum Gradient	VT_I4 [5 .. 254]	Minimum Gradient defines how high the edge gradient (gray-value difference between a pixel and its neighbours) needs to be at a given pixel in order for it to be accepted as an edge point.

2.7.2.6 Difference Checker

Parameter Name	Parameter VARIANT Type	Comment
Rotate Template	VT_BOOL (BOOL) -1 and 1 as TRUE	In order to speed up the checker calculation, all template rotations brought about by position and rotation adjustments may be calculated and saved in advance during the initial checker definition. With this command you can execute this calculation. Use this method after setParameter "Refresh Template" or setParameter "Template Rotation".
Accuracy	VT_BSTR "high",	The accuracy parameters High, Medium, and Standard let you determine whether pseudo

Parameter Name	Parameter VARIANT Type	Comment
	"medium" "standard"	differences will influence the image processing to a lesser or greater extent. Please note that setting a high accuracy factor slows down the algorithms.
Template Rotation	VT_R8 and VT_ARRAY 0: Min Angle [-180° .. 360°] 1: Max Angle [-180° .. 360°] 2: Delta Angle	The confines of the rotation range are defined in the parameters Min Angle and Max Angle, the increment itself in "Delta Angle". The maximum difference between Max Angle and Min Angle is 360°. After changing one of these values, the method setParameter() with the parameter "Rotate Template" has to be executed.
Selected Range	VT_BOOL (BOOL) -1 and 1 as "Process Selected Range", 0 as "Process Not Selected Range"	When "Process Selected Rang" is set as true (1 or -1) , the differences within the range of the threshold values will determine the objects to continue the process. If you select this parameter as false (0), the differences outside the selected range are the objects.
Slice Level	VT_I2 and VT_ARRAY 0: LowerLimit [- 254 .. 254] 1: UpperLimit [- 254 .. 254]	Threshold which determines the range of valid differences.
Refresh Template	VT_BOOL -1 and 1 as TRUE	With this command you can force an update of the template. If the template is rotated the method setParameter() with the parameter "Rotate Template" must be executed. This parameter can only be set in the setup mode.
Binary Image Filters		All binary parameters described in Image Filters

2.7.2.7 Contour Matching

Parameter Name	Parameter VARIANT Type	Comment
Use Scale	VT_I4 [0..3]	This parameter the model type: 0 = Standard 1 = Scale 2 = Anisotrope
Search Method	VT_R8 [0.0 .. 1.0]	The parameter influences the search heuristic of the checker. A value of "0.0" means safe and slower. "1.0" means faster and maybe objects will not be found.
Maximum Overlap	VT_R8 [0.0 .. 1.0]	The parameter „Maximum Overlap, lets you specify the extent to which two matches may overlap (in percent).
Angle Range	VT_R8 and VT_ARRAY 0: StartAngle [– 360.0 .. 360.0] 1: AngleExtend [– 360.0 .. 360.0]	If the object's rotation may vary in the search images you can specify the allowed range in the parameters "Start Angle" and "End Angle" (StartAngle + AngleExtend).
Scale Range	[VT_R8 and VT_ARRAY 0: MinScale [0.0 .. 5.0]	Similarly to the range of orientation, you can specify an allowed range of scale with the

Parameter Name	Parameter VARIANT Type	Comment
	1: MaxScale [0.0 .. 5.0]	parameters „Min Scale“ and „Max Scale”.
Scale Range Column	[VT_R8 and VT_ARRAY 0: MinScale [0.0 .. 5.0] 1: MaxScale [0.0 .. 5.0]	Similarly to the range of orientation, you can specify an allowed range of scale (column) for ansiotrope scaling with the parameters „Min Scale“ and „Max Scale”.
Refresh Template	VT_BOOL -1 and 1 as TRUE	This parameter reads a new template image for the model to be calculated. This parameter can only be set in the setup mode.
Object Appearance	VT_I4 [0 .. 2] 0 = like template 1 = like template or inverted 2 = dynamic	The parameter „Object Appearance“ lets you specify whether the polarity, i.e., the direction of the contrast, must be observed.
Recalculate Model	VT_BOOL -1 and 1 as TRUE	This parameter lets you recalculate the model with the current template.
Minimum Correlation	VT_R8 [0.0 .. 1.0]	With the parameter „Minimum Contrast“ you can specify the contrast a pixel in a search image must have in order to be compared with the model.
Number Of Matches	VT_I4 [0..128]	Sets the current value of the parameter number of matches.
Minimum Inspection Contrast	VT_I4 [0..255]	Sets the current value of the parameter “Minimum Inspection Contrast” of the checker. (Former Name of this parameter: "Minimum Contrast")
Maximum Model Contrast	VT_I4 [0..255]	Sets the current setting of the parameter “Maximum Model Contrast” of the checker.
Minimum Model Contrast	VT_I4 [0..255]	Sets the current value of the parameter “Minimum Model Contrast” of the checker. (Former name of this parameter: "Contrast ")
Last Level	VT_I4	Sets the current value of the parameter “Last Compression Level”.
Export Contour Model	VT_BSTR	The name of the file, in which the contour model wll be exported, e.g. “C:\Temp\Model.bmp”
Area Boundary	VT_BOOL -1 and 1 as ON 0 as OFF	Sets the current value of the parameter “Area Boundary”.
Accuracy	VT_I4 [0…4] 0 = Pixel 1 = Subpixel Standard 2 = Subpixel Advanced 3 = Subpixel High 4 = Subpixel Very High	Sets the current setting of the parameter “Accuracy”
Minimum Component Size	VT_I4 [0..255]	Sets the current setting of the parameter “Minimum Component Size” of the checker.
Optimize Large Models	VT_I4 0 = none 1 = low 2 = medium 3 = high	Sets the current setting of the parameter “Optimize Large Models” of the checker.
Output Point	VT_BOOL	Sets the current setting of the parameter “Output

Parameter Name	Parameter VARIANT Type	Comment
	-1 and 1 as Center of Model 0 as Center of Template Area	Point" of the checker.
Pregenerate Model	VT_BOOL -1 and 1 as ON 0 as OFF	Sets the current setting of the parameter "Pregenerate Model" of the checker.
Maximal Allowed Deformation	VT_I4	Sets the current setting of the parameter "Maximal Allowed Deformation".
Reference Point	[VT_R8 and VT_ARRAY 0: MinScale [0.0 .. 5000.0] 1: MaxScale [0.0 .. 5000.0]	Set the reference point of the checker.

2.7.2.8 Correlation Matching

Parameter Name	Parameter VARIANT Type	Comment
Maximum Overlap	VT_R8 [0.0 .. 1.0]	The parameter „Maximum Overlap, lets you specify the extent to which two matches may overlap (in percent).
Angle Range	VT_R8 and VT_ARRAY 0: StartAngle [– 360.0 .. 360.0] 1: AngleExtend [– 360.0 .. 360.0]	If the object's rotation may vary in the search images you can specify the allowed range in the parameters "Start Angle" and "End Angle" (StartAngle + AngleExtend).
Refresh Template	VT_BOOL -1 and 1 as TRUE	This parameter reads a new template image for the model to be calculated. This parameter can only be set in the setup mode.
Object Appearance	VT_I4 [0 .. 1] 0 = like template 1 = like template or inverted	The parameter „Object Appearance“ lets you specify whether the polarity, i.e., the direction of the contrast, must be observed.
Recalculate Model	VT_BOOL -1 and 1 as TRUE	This parameter lets you recalculate the model with the current template.
Minimum Correlation	VT_R8 [0.0 .. 1.0]	With the parameter „Minimum Contrast“ you can specify the contrast a pixel in a search image must have in order to be compared with the model.
Number Of Matches	VT_I4 [0..128]	Sets the current value of the parameter number of matches.
Last Level	VT_I4	Sets the current value of the parameter "Last Compression Level".
Export Contour Model	VT_BSTR	The name of the file, in which the contour model wll be exported, e.g. "C:\Temp\Model.bmp"
Sequences	VT_I4	Sets the current setting of the parameter "Pyramid Levels". Setting to 0 results in the maximum available being used.
Area Boundary	VT_BOOL -1 and 1 as ON 0 as OFF	Sets the current value of the parameter "Area Boundary".
Accuracy	VT_I4 [0..1] 0 = Pixel 1 = Subpixel Standard	Sets the current setting of the parameter "Accuracy"

2.7.2.9 OCR Checker

Parameter Name	Parameter Value	Comment
Use Classifier	VT_BOOL -1 and 1 as ON, 0 as OFF	
Classifier Name	VT_BSTR	name of the new classifier
Correlation Threshold	VT_R8 [0.0 .. 1.0]	
Thresholding		All parameters described in Thresholding
Image Filters		All parameters described in Image Filters

2.7.2.10 Code Reader

Parameter Name	Parameter Value	Comment
Code Type	VT_BSTR	Sets the current "Code Type" setting of the checker. Following code types are valid: "2/5 Industrial", "2/5 Interleaved", "Codabar", "Code 39", "Code 93", "Code 128", "EAN-13", "EAN-13 Add-On 2", "EAN-13 Add-On 5", "EAN-8", "EAN-8 Add-On 2", "EAN-8 Add-On 5", "UPC-A", "UPC-A Add-On 2", "UPC-A Add-On 5", "UPC-E", "UPC-E Add-On 2", "UPC-E Add-On 5", "PharmaCode", "RSS-14", "RSS-14 Truncated", "RSS-14 Stacked", "RSS-14 Stacked Omnidir", "RSS Limited", "RSS Expanded", "RSS Expanded Stacked"
Number Of Codes	VT_I4	Sets the current "Number of Codes" setting of the checker.
Number Of Loops	VT_I4	Sets the current "Number of Search Loops" setting of the checker. This setting is only valid for 2D code types.
Model Parameter Mode	VT_I4	Sets the current "Code Model Parameter" setting of the checker.
Pharma Code Reverse	VT_BOOL -1 and 1 as TRUE, 0 as FALSE	If set to TRUE a pharma code is read in reverse direction, if FALSE not.
Check Digit	VT_BOOL -1 and 1 as TRUE, 0 as FALSE	Sets the current value of the parameter "Check Digit".
2D Code Model Parameters – Only valid in "Manual" mode		
Teach 2D	VT_BOOL -1 and 1 as TRUE	With this command you can execute the teaching of 2D code parameters. After processing the teaching of the 2D code the processing time of the checker will be faster. This parameter can only be set in the setup mode.
Min Contrast 2D	VT_I4	Sets the current value of the parameter "Minimal Contrast".
Max Angle Variation ECC200 2D	VT_R8	Sets the current value of the parameter "Maximal Angle Variation". This parameter is only valid for the ECC200 code.

Parameter Name	Parameter Value	Comment
Min Symbol Col 2D	VT_I4	Sets the current value of the parameter “Minimal Symbol Columns” / “Maximal Symbol Columns”.
Max Symbol Col 2D		
Min Symbol Row 2D	VT_I4	Sets the current value of the parameter “Minimal Symbol Rows” / “Maximal Symbol Rows”.
Max Symbol Row 2D		
Symbol Shape 2D	VT_BSTR	Sets the current value of the parameter “Symbol Shape”.
Mirrored 2D	VT_BSTR	Sets the current value of the parameter “Mirrored”.
Polarity 2D	VT_BSTR	Sets the current value of the parameter “Polarity”.
Module Size Min 2D	VT_I4	Sets the current value of the parameter “Minimal Module Size” / “Maximal Module Size”.
Module Size Max 2D		
Module Gap Col Max 2D	VT_BSTR	Sets the current value of the parameter “Minimal Module Gap” / “Maximal Module Gap”.
Module Gap Col Min 2D		
Module Gap Row Max 2D	VT_BSTR	Sets the current value of the parameter “Minimal Module Gap Row” / “Maximal Module Gap Row”.
Module Gap Row Min 2D		
Model Type QR 2D	VT_BSTR	Sets the current value of the parameter “QR Model Type”. This parameter is only valid for the QR code.
Version Min QR 2D	VT_I4	Sets the current value of the parameter “Minimal Symbol Version” / Maximal Symbol Version”. This parameter is only valid for the QR code.
Version Max QR 2D		
Persistance 2D	VT_I4	Sets the current value of the parameter “Persistence”.
Small Module Robustness	VT_BSTR	Sets the current value of the parameter Small Module Robustness
Strict Model 2D	VT_BSTR	Sets the current value of the parameter “Strict Model”.
Module Width Max PDF417 2D	VT_I4	Sets the current “Module Width Max” / Module Width Min” setting of the checker. This parameter is only valid for the PDF417 code.
Module Width Min PDF417 2D		
Pattern Position Min QR 2D	VT_I4	Sets the current “Position Pattern Min” setting of the checker.
Maximal Symbol Columns PDF417 2D	VT_I4	Sets the current “Maximal Symbol Columns” / “Minimal Symbol Columns” setting of the checker. This parameter is only valid for the PDF417 code.
Minimal Symbol Columns PDF417 2D		
Maximal Symbol Rows PDF417 2D	VT_I4	Sets the current “Maximal Symbol Rows” / “Minimal Symbol Rows” setting of the checker. This parameter is only valid for the PDF417 code.
Minimal Symbol Rows PDF417 2D		
Module Aspect Min PDF417 2D	VT_I4	Sets the current “Module Aspect Max” / Module Aspect Min” setting of the checker. This parameter is only valid for the PDF417 code.
Module Aspect Max PDF417 2D		
1D Code Model Parameters – Only valid in “Manual” mode		
GrayImage Filters 1D		All gray parameters described in Image Filters
Min Size Element 1D	VT_R8	Sets the current value of the parameter “Minimal Element Size”.
Max Size Element 1D	VT_R8	Sets the current value of the parameter “Maximal Element Size”.
Min Element Height 1D	VT_I4	Sets the current value of the parameter “Minimum Element Height”.
Angle Range 1D	VT_I4	Sets the current value of the parameter “Angle Range”.
Measure Threshold 1D	VT_R8	Sets the current value of the parameter

Parameter Name	Parameter Value	Comment
		“Segmentation Threshold”.
Orientation 1D	VT_R8	Sets the current value of the parameter “Element Orientation”.
Orientation Tolerance 1D	VT_R8	Sets the current value of the parameter “Element Orientation Tolerance”.
Composite Code 1D	VT_BSTR	Sets the current value of the parameter “1D Composite Code”.

2.7.2.11Edge Detection Gray Value Projection

Parameter Name	Parameter Value	Comment
Transition	VT_I4	Sets the current “Edge Transition” setting of the checker. All = 0 Dark - Light = 1 Light – Dark = 2

Select Edges	VT_I4	Sets the current “Edge Selection” setting of the checker.
		All = 0
		First = 1
		Last = 2
		First - Last = 3
		Maximum Gradient = 4

Execution Mode	VT_I4	Sets the current “Execution Mode” setting of the checker. Edge Pairs = 0 Edge Position = 1
Gradient	VT_I4	Sets the current “Minimal Gradient” setting of the checker. (0-255)
Noise Level	VT_I4	Sets the current “Noise Level” setting of the checker. (0-100)
Number Of Edges	VT_I4	Sets the current “Number of edges” setting of the checker. (0-255)
GrayImage Filters		All gray parameters described in Image Filters

2.7.2.12Identifier Checker

Parameter Name	Parameter Value	Comment
Sample Images Directory	VT_BSTR	Sets the sample images directory. “None”, deselect any sample images directory, cannot be set.
Use Color Information	VT_BOOL -1 and 1 as “is used” 0 as “is not used”	Sets the “Use Color Information” of the checker
Image Resize Method	VT_BSTR	Sets the “Image Resize Method” of the checker.
Resulting Image Size	VT_R8	Sets the “Resulting Image Size” of the checker. Can only be set if the “Image Resize Method” is set to

Parameter Name	Parameter Value	Comment
		“Resulting Image Size”.
Scaling Factor	VT_R8	Sets the “Scaling Factor” of the checker. Can only be set if the “Image Resize Method” is set to “Scaling Factor”.
Subsampling Step	VT_R8	Sets the “Subsampling Step” of the checker. Can only be set if the “Image Resize Method” is set to “Subsampling Step”.
Maximum Number of Result	VT_I4	Sets the “Maximum Nuber Of Results” of the checker.
Rating Method	VT_BSTR	Sets the “Rating Method” of the checker.
Rating Threshold	VT_R8	Sets the “Rating Threshold” of the checker.
Train	No parameter value	Trains the identifier with the currently selected settings. Attention: If some settings are changed by the GUI, and “Apply” is not pressed, these new settings will not be used for the training. If the identifier is trained by the ActiveX control, settings should only be changed by the ActiveX control

2.7.2.13Position and Rotation Adjustment Checker

Parameter Name	Parameter VARIANT Type	Comment
Set Base Position	Not used	Same as the button "Copy Current Values" on the checker's dependencies property page: the current values are used as the new reference values.

2.7.3 Shapes

If you change some shape parameters by calling the method setParameter, all other shape parameters may be changed implicitly, too.
E.g. if you change the center point of a shape, all other shape points are moved, too. You can get the changed values by calling the method getParameter. (You can test the behaving of setParameter by changing some parameters in a checkers shape property page.)

The method setParameter fails if only one of the (implicitly) changed parameters lays outside it’ s range, e.g. all changed image points have to lay inside the image.

Visual Basic Example

The first example shows how to get shape parameters from a checker and the second example shows how to set shape parameters.

Example 1:

```
Private Sub Command3_Click()
    Dim sObjectName As String
    Dim sParameterName As String
    Dim vShapeParameter As Variant
    Dim lStartEdge_X As Long
    Dim lStartEdge_Y As Long
    Dim lEndEdge_X As Long
    Dim lEndEdge_Y As Long

    sObjectName = "WI[1;1]"   'Rectangle Shape
    sParameterName = "Shape Points"
    vShapeParameter =  VisionQ400Control1. _
                GetParameter (sObjectName, sParameterName)
    If TypeName(vShapeParameter) = "Empty" Then
      MsgBox (VisionQ400Control1.getLastErrorText)
    Else
      lStartEdge_X = vShapeParameter(0, 0)
      lStartEdge_Y = vShapeParameter(0, 1)
      lEndEdge_X = vShapeParameter(1, 0)
      lEndEdge_Y = vShapeParameter(1, 1)
    End If

End Sub
```

Example 2:

```
    Dim sObjectName As String
    Dim sParameterName As String
    Dim lStartEdge_X As Long
    Dim lStartEdge_Y As Long
    Dim lEndEdge_X As Long
    Dim lEndEdge_Y As Long
    Dim lShapeParameter(0 To 2, 0 To 1) As Long
    Dim lEdge_1_X As Long
    Dim lEdge_1_Y As Long
    Dim lEdge_2_X As Long
    Dim lEdge_2_Y As Long
    Dim lEdge_3_X As Long
    Dim lEdge_3_Y As Long

    lEdge_1_X = 1
    lEdge_1_Y = 1
    lEdge_2_X = 500
    lEdge_2_Y = 1
    lEdge_3_X = 250
    lEdge_3_Y = 400

    lShapeParameter(0, 0) = lEdge_1_X
    lShapeParameter(0, 1) = lEdge_1_Y
    lShapeParameter(1, 0) = lEdge_2_X
    lShapeParameter(1, 1) = lEdge_2_Y
    lShapeParameter(2, 0) = lEdge_3_X
    lShapeParameter(2, 1) = lEdge_3_Y

    sObjectName = "WI[1;1]"   'Poligon Shape with 3 Points
    sParameterName = "Shape Points"
    bReturn = VisionQ400Control1. _
      setParameter(sObjectName, sParameterName, vShapeParameter)
    If bReturn = False Then
      MsgBox (VisionQ400Control1.getLastErrorText)
    End If

End Sub
```

2.7.3.1 Shape Line ->

Parameter Name	Parameter VARIANT Type	Comment
Shape Points	VT_I4 | VT_ARRAY 2 Points	Point 1: Start Point Point 2: End Point If the shape is a horizontal line, the y coordinates of the to points has to be identical. If the shape is a vertical line, the x coordinates of the to points has to be identical.
Shape Start Point	VT_I4 | VT_ARRAY 1 Point	Start Point If the shape is a horizontal or a vertical line, only the x or the y coordinate is used.

Parameter Name	Parameter VARIANT Type	Comment
Shape End Point	VT_I4 | VT_ARRAY 1 Point	End Point If the shape is a horizontal or a vertical line, only the x or the y coordinate is used.
Shape Center Point	VT_I4 | VT_ARRAY 1 Point	Center Point
Shape Direction	VT_R4	Angle of the line Range: 0 <= angle <= 360 If the shape is a horizontal or vertical line, the angle is converted to a horizontal or vertical direction, and the Start and End Point may be exchanged.

Visual Basic Example: Shape Points

```
Private Sub Command1_Click()

    Dim bReturn As Boolean
    Dim sObjectName As String
    Dim sParameterName As String
    Dim vShapeParameter(0 To 1, 0 To 1) As Long
    Dim lStartEdge_X As Long
    Dim lStartEdge_Y As Long
    Dim lEndEdge_X As Long
    Dim lEndEdge_Y As Long
        lStartEdge_X = 5
        lStartEdge_Y = 5
        lEndEdge_X = 400
        lEndEdge_Y = 400
        vShapeParameter(0, 0) = lStartEdge_X
        vShapeParameter(0, 1) = lStartEdge_Y
        vShapeParameter(1, 0) = lEndEdge_X
        vShapeParameter(1, 1) = lEndEdge_Y
        sObjectName = "ED_B[1;1]" 'Line Shape
        sParameterName = "Shape Points"
        bReturn = VisionQ400Control1. _
        setParameter(sObjectName, sParameterName, vShapeParameter)
    If bReturn = False Then
        MsgBox (VisionQ400Control1.getLastErrorText)
    End If

End Sub
```

Visual Basic Example: Center Point

```
Private Sub Command1_Click()

    Dim bReturn As Boolean
    Dim sObjectName As String
    Dim sParameterName As String
    Dim lShapeParameter(0, 0 To 1) As Long
    Dim lCenterPoint_x As Long
    Dim lCenterPoint_y As Long
        lCenterPoint_x = 150    'x-Coordinate
        lCenterPoint_y = 150    'y-Coordinate
        lShapeParameter(0, 0) = lCenterPoint_x
        lShapeParameter(0, 1) = lCenterPoint_y
        sObjectName = "ED_B[1;1]" 'Line Shape
        sParameterName = "Shape Center Point"
        bReturn = VisionQ400Control1. _
        setParameter(sObjectName, sParameterName, lShapeParameter)
    If bReturn = False Then
        MsgBox (VisionQ400Control1.getLastErrorText)
    End If

End Sub
```

Visual Basic Example: Direction

```
Private Sub Command1_Click()

    Dim bReturn As Boolean
    Dim sObjectName As String
    Dim sParameterName As String
    Dim lShapeParameter As Long
        lShapeParameter = 180    ' Angle
        sObjectName = "ED_B[1;1]" ' Line Shape
        sParameterName = "Shape Direction"
        bReturn = VisionQ400Control1. _
        setParameter(sObjectName, sParameterName, lShapeParameter)
    If bReturn = False Then
        MsgBox (VisionQ400Control1.getLastErrorText)
    End If

End Sub
```

2.7.3.2 Shape Rectangle

Parameter Name	Parameter VARIANT Type	Comment
Shape Start Point	VT_I4 | VT_ARRAY 1 Point	Start Point The Start Point equals to the top left point of the rectangle.
Shape End Point	VT_I4 | VT_ARRAY 1 Point	End Point The End Point equals to the bottom right point of the rectangle.
Shape Center Point	VT_I4 | VT_ARRAY 1 Point	Center Point

Visual Basic Example

```
Private Sub Command1_Click()

    Dim bReturn As Boolean
    Dim sObjectName As String
    Dim sParameterName As String
    Dim lShapeParameter(0 To 1, 0 To 1) As Long
    Dim lStartEdge_X As Long
    Dim lStartEdge_Y As Long
    Dim lEndEdge_X As Long
    Dim lEndEdge_Y As Long
        lStartEdge_X = 5
        lStartEdge_Y = 5
        lEndEdge_X = 400
        lEndEdge_Y = 400
        lShapeParameter(0, 0) = lStartEdge_X
        lShapeParameter(0, 1) = lStartEdge_Y
        lShapeParameter(1, 0) = lEndEdge_X
        lShapeParameter(1, 1) = lEndEdge_Y
        sObjectName = "WI[1;1]" 'Rectangle Shape
        sParameterName = "Shape Points"
        bReturn = VisionQ400Control1. _
        setParameter(sObjectName, sParameterName, lShapeParameter)
    If bReturn = False Then
        MsgBox (VisionQ400Control1.getLastErrorText)
    End If

End Sub
```

2.7.3.3 Shape Rectangle ->

Parameter Name	Parameter VARIANT Type	Comment
Shape Start Point	VT_I4 | VT_ARRAY 1 Point	Start Point The Start Point equals to the top left point of the rectangle.
Shape End Point	VT_I4 | VT_ARRAY 1 Point	End Point The End Point equals to the bottom right point of the rectangle.
Shape Center Point	VT_I4 | VT_ARRAY 1 Point	Center Point
Shape Direction	VT_R4	Direction of the rectangle Range: 0 <= angle <= 360
Shape Direction Point	VT_I4 | VT_ARRAY 1 Point	Point to change the direction of the rectangle. This point is not supported in a checkers shape properties page.

2.7.3.4 Shape Ellipse (Circle)

Parameter Name	Parameter VARIANT Type	Comment
Shape Points	VT_I4 | VT_ARRAY 2 Points	Point 1: Top Left Point Point 2: Bottom Right Point
Shape Top Left Point	VT_I4 | VT_ARRAY 1 Point	Top Left Point of the surrounding rectangle
Shape Bottom Right Point	VT_I4 | VT_ARRAY 1 Point	Bottom Right Point of the surrounding Rectangle.
Shape Center Point	VT_I4 | VT_ARRAY 1 Point	Center Point

2.7.3.5 Shape Ellipse -> (Circle ->)

Sometimes the given point cannot be set directly, but has to be modified (see e.g. the comment of "Shape Start Point"). In this case the modified value is set, the last error number is set to 7779, and the method returns FALSE. Therefore if the method resturns FALSE, it has to be checked if the last error number is 7779. If yes the error can be ignored, and the corrected point can be got with the method getParameter().

Parameter Name	Parameter VARIANT Type	Comment
Shape Points	VT_I4 | VT_ARRAY 2 Points	Point 1: Top Left Point Point 2: Bottom Right Point
Shape Top Left Point	VT_I4 | VT_ARRAY 1 Point	Top Left Point of the surrounding rectangle
Shape Bottom Right Point	VT_I4 | VT_ARRAY 1 Point	Bottom Right Point of the surrounding Rectangle.
Shape Center Point	VT_I4 | VT_ARRAY 1 Point	Center Point
Shape Start Point	VT_I4 | VT_ARRAY 1 Point	Start point of the part of the shape The given point has not to lay exactly on the shape. If not the intersection of the shape and the line given by the new point and the center point is the new Start Point.
Shape End Point	VT_I4 | VT_ARRAY 1 Point	End point of the part of the shape See remark of Start Point.
Shape Direction	VT_BOOL	Direction for the shape 0: clockwise 1: counterclockwise
Shape Direction Point	VT_I4 | VT_ARRAY 1 Points	Point to change the direction of the shape This point is not supported in a checkers shape properties page.

2.7.3.6 Shape Doughnut

Sometimes the given point cannot be set directly, but has to be modified (see e.g. the comment of "Shape Start Point"). In this case the modified value is set, the last error number is set to 7779, and the method returns FALSE. Therefore if the method resturns FALSE, it has to be checked if the last error number is 7779. If yes the error can be ignored, and the corrected point can be got with the method getParameter().

Parameter Name	Parameter VARIANT Type	Comment	
Shape Outer Circle Point	VT_I4 | VT_ARRAY 1 Point	Top Left point of the surrounding rectangle of the outer circle. The Bottom Right point of the surrounding rectangle is the point, which is point-symmetric to the given point at the Center Point.	
Shape Inner Circle Point	VT_I4 | VT_ARRAY 1 Point	Top Left point of the surrounding rectangle of the inner circle. See comment of Shape Outer Circle Point.	
Shape Center Point	VT_I4 | VT_ARRAY 1 Point	Center Point	
Shape Start Point	VT_I4 | VT_ARRAY 1 Points	Start point of the part of the shape. The given point has not to lay exactly on the shape. If not the intersection of the shape and the line	

Parameter Name	Parameter VARIANT Type	Comment
		given by the new point and the center point is the new Start Point.
Shape End Point	VT_I4 | VT_ARRAY 1 Points	End point of the part of the shape. See remark of Start Point.

Visual Basic Example: Shape Center Point

```
Private Sub Command1_Click()

    Dim bReturn As Boolean
    Dim sObjectName As String
    Dim sParameterName As String
    Dim lShapeParameter(0, 0 To 1) As Long
    Dim lCenterPoint_x As Long
    Dim lCenterPoint_y As Long
        lCenterPoint_x = 250    'x-Coordinate
        lCenterPoint_y = 250    'y-Coordinate
        lShapeParameter(0, 0) = lCenterPoint_x
        lShapeParameter(0, 1) = lCenterPoint_y
        sObjectName = "WI[1;1]" 'Doughnut Shape
        sParameterName = "Shape Center Point"
        bReturn = VisionQ400Control1. _
        setParameter(sObjectName, sParameterName, lShapeParameter)
    If bReturn = False Then
        MsgBox (VisionQ400Control1.getLastErrorText)
    End If

End Sub
```

Visual Basic Example: Shape Outer Circle Point

```
Private Sub Command1_Click()

    Dim bReturn As Boolean
    Dim sObjectName As String
    Dim sParameterName As String
    Dim lShapeParameter(0, 0 To 1) As Long
    Dim lOuterCirclePoint_x As Long
    Dim lOuterCirclePoint_y As Long
        lOuterCirclePoint_x = 200    'x-Coordinate
        lOuterCirclePoint_y = 200    'y-Coordinate
        lShapeParameter(0, 0) = lOuterCirclePoint_x
        lShapeParameter(0, 1) = lOuterCirclePoint_y
        sObjectName = "WI[1;1]" 'Doughnut Shape
        sParameterName = "Shape Outer Circle Point"
        bReturn = VisionQ400Control1. _
        setParameter(sObjectName, sParameterName, lShapeParameter)
    If bReturn = False Then
        MsgBox (VisionQ400Control1.getLastErrorText)
    End If

End Sub
```

2.7.3.7 Shape Doughnut ->

Parameter Name	Parameter VARIANT Type	Comment
Shape Outer Circle Point	VT_I4 | VT_ARRAY 1 Point	Top Left point of the surrounding rectangle of the outer circle. The Bottom Right point of the surrounding rectangle is the point, which is point-symmetric to the given point at the Center Point.
Shape Inner Circle Point	VT_I4 | VT_ARRAY 1 Point	Top Left point of the surrounding rectangle of the inner circle. See comment of Shape Outer Circle Point.
Shape Center Point	VT_I4 | VT_ARRAY 1 Point	Center Point
Shape Start Point	VT_I4 | VT_ARRAY 1 Points	Start point of the part of the shape. The given point has not to lay exactly on the shape. If not the intersection of the shape and the line given by the new point and the center point is the new Start Point.
Shape End Point	VT_I4 | VT_ARRAY 1 Points	End point of the part of the shape. See remark of Start Point.

Parameter Name	Parameter VARIANT Type	Comment
Shape Direction	VT_BOOL	Direction for the shape 0: clockwise 1: counterclockwise

2.7.3.8 Shape Polygon

Parameter Name	Parameter VARIANT Type	Comment
Shape Points	VT_I4 | VT_ARRAY n Points	A list of points which describes the polygon. A maximum of 256 points is allowed.
Shape Center Point	VT_I4 | VT_ARRAY 1 Points	Center Point

Visual Basic Example

```
Private Sub Command1_Click()

    Dim bReturn As Boolean
    Dim sObjectName As String
    Dim sParameterName As String
    Dim lShapeParameter(0 To 2, 0 To 1) As Long
    Dim lPoint1_x As Long
    Dim lPoint1_y As Long
    Dim lPoint2_x As Long
    Dim lPoint2_y As Long
    Dim lPoint3_x As Long
    Dim lPoint3_y As Long
        lPoint1_x = 100     'x-Coordinate
        lPoint1_y = 100     'y-Coordinate
        lPoint2_x = 200
        lPoint2_y = 350
        lPoint3_x = 300
        lPoint3_y = 300
        lShapeParameter(0, 0) = lPoint1_x
        lShapeParameter(0, 1) = lPoint1_y
        lShapeParameter(1, 0) = lPoint2_x
        lShapeParameter(1, 1) = lPoint2_y
        lShapeParameter(2, 0) = lPoint3_x
        lShapeParameter(2, 1) = lPoint3_y
        sObjectName = "WI[1;1]" 'Shape Polygon
        sParameterName = "Shape Points"
        bReturn = VisionQ400Control1. _
        setParameter(sObjectName, sParameterName, lShapeParameter)
    If bReturn = False Then
        MsgBox (VisionQ400Control1.getLastErrorText)
    End If

End Sub
```

2.7.3.9 Shape Object Shape

Parameter Name	Parameter VARIANT Type	Comment
Refresh Object	VT_I4	The row index in the dependencies list of the object shape. The object in this row will be refreshed. This parameter cannot be set for a dynamic object shape.
Refresh All Objects	Not used	All objects in the dependencies list of the object shape are refreshed. This parameter cannot be set for a dynamic object shape.

2.7.3.10 Additional Shapes

2.7.3.10.1 Template Shape

The contour matching and the correlation matching checkers have two shapes. To get the values of the “Search Area” shape, the shape parameter names are used. E.g. the parameter name “Shape Center Point” returns the center point of the Search Area.

To get the values of the “Template” shape, the string “Template “ has to be inserted before the shape parameter name. E.g. the parameter name “Template Shape Center Point” returns the center point of the Template.

2.8 Shape Methods

2.8.1 getShapeAdjusted(CheckerName)

The method **getShapeAdjusted** returns a variant value which includes the coordinates of the moved shape. A checker shape can be moved by a position and rotation adjustment.

Return Value

VARIANT

For more details please refer to the table below.
The method returns an empty VARIANT if it fails. In this case, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument		**Type**	**Description**
[IN]	checkerName	BSTR	ActiveX Name of the checker e.g. "WI[1;1]".

Remarks

This method is very helpful when adding a checker shape to a received camera image.

Shape	Type	Meaning
Line ->	VT_I4 | VT_ARRAY 2 Points	Point 1: Start Point rotated Point 2: End Point rotated
Rectangle	VT_I4 | VT_ARRAY 4 Points	Point 1: Top Left Point rotated Point 2: Top Right Point rotated Point 3: Bottom Right Point rotated Point 3: Bottom Left Point rotated
Rectangle ->	VT_I4 | VT_ARRAY 4 Points	Point 1: Top Left Point rotated Point 2: Top Right Point rotated Point 3: Bottom Right Point rotated Point 3: Bottom Left Point rotated
Ellipse / Circle	VT_I4 | VT_ARRAY n Points	A list of points, which describe the rotated part of the ellipse /circle
Ellipse / Circle ->	VT_I4 | VT_ARRAY n Points	A list of points, which describe the rotated part of the ellipse /circle ->
Doughnut	VT_I4 | VT_ARRAY n Points	A list of points, which describe the rotated part of the doughnut.
Polygon	VT_I4 | VT_ARRAY n Points	A list of points, which describe the rotated polygon
Object Shape	VT_ARRAY | VT_I4 (3 * number of runs) + 1 points a[0] : 3 * number of runs 0 <= i < number of runs: a[3*i+1]: row number of the run i a[3*i+2]: start column number of the run i a[3*i+3]: end column number of the run i	The shape is returned in so called runs. A run is a triple with the following meaning: run[0]: row number of the run run[1]: start column number of the run run[2]: end column number of the run

Visual Basic Example

```
Private Sub Command1_Click()

    Dim sObjectName As String
    Dim vShapeAdjusted As Variant
    Dim lStartEdge_X As Long
    Dim lStartEdge_Y As Long
    Dim lEndEdge_X As Long
    Dim lEndEdge_Y As Long
        sObjectName = "ED_B[1;1]"
        vShapeAdjusted = VisionQ400Control1.getShapeAdjusted(sObjectName)
    If TypeName(vShapeAdjusted) = "Empty" Then
        MsgBox (VisionQ400Control1.getLastErrorText)
    Else
        lStartEdge_X = vShapeAdjusted(0, 0)
        lStartEdge_Y = vShapeAdjusted(0, 1)
        lEndEdge_X = vShapeAdjusted(1, 0)
        lEndEdge_Y = vShapeAdjusted(1, 1)
        Debug.Print "Start (x/y): " & lStartEdge_X & " / " & lStartEdge_Y
        Debug.Print "End (x/y):   " & lEndEdge_X & " / " & lEndEdge_Y
    End If

End Sub
```

2.9 Getting Checker Results

Some of the checkers of Vision Q.400, e.g. the feature extraction checker, generate a list of “objects”, and calculate results of these objects. Because a result of an object can only be inserted into the spreadsheet if an object exists, it may not be easy to insert results if the number of objects can vary, or is unknown at the time an application is created. Because only results, which are inserted into the spreadsheet, can be accessed by the interfaces, e.g. by the method getSpreadsheetData (), it may not be possible to access all results of all objects by an interface.

To overcome this problem, the methods **getResult**, **getResultsObject**, and **getResultsObjects** are introduced. At time, these methods are only implemented for checkers, which generate a list of objects at their execution. E.g. they are not implemented for the window checker.

The three methods are not synchronised and it is not tested if the data to be returned is valid. For example if a checker is executed while one of these methods is called, the value(s) of an earlier execution of the checker may be returned. The caller is responsible for synchronisation.

The method getResult is used to return results which belong directly to a checker, e.g. the result string of an OCR checker, or the number of judged objects of a feature exctraction checker.

The method returns an VARIANT. The current type and meaning of this VARIANT depend on the concerning checker and the result. The possible types of the VARIANT are:

VARIANT Type	Meaning
VT_EMPTY	Error
VT_R8	One double value
VT_BSTR	One string value

The result of a ckecker is accessed by the result name displayed in the result property page of a checker. To achieve the language independence of the ActiveX control, always the English names of the results have to be used.

To get all result names, which are supported by a checker type, you can call the method with it’s parameter resultName set to “Result Names”. In this case the returned VARIANT contains a BSTR array, with all supported result names.

The methods getResultsObject, and getResultsObjects are used to return results which belong to an object of the checker, e.g. the area size of an object, or all objects, of a feature exctraction checker.

Both methods return an VARIANT. The current type and meaning of this VARIANT depend on the concerning checker and the result. The possible types of the VARIANT are listed in the table below.

VARIANT Type	Meaning
VT_EMPTY	Error
VT_R8 | VT_ARRAY	An array of double value
VT_BSTR | VT_ARRAY	An array of string value
VT_VARIANT | VT_ ARRAY	An array of VARIANTs. At time, this can only happen for user defined checker dlls.

The methods getResultsObject, and getResultsObjects return always an array, even if this array contains only one entry.

The results of an object are accessed by the column headers displayed in the result property page of an checker. Because one column header can "contain" more than one result , e.g. the result "Gravity" of the feature extraction checker contains a X – and a Y – coordinate, more than one value may be returned by one call of the above methods.

If a column header contains more than one result, the returned VT_ARRAY contains the values as they are listed from left to right in the property page of a checker. E.g. if the result "Gravity" of the feature extraction checker is accessed, the value at the index zero of the array is the X – coordinate, and the value at the index one is the Y – Coordinate.

To achieve the language independence of the ActiveX control, always the English names of the column headers has to be used to access results of an object.

To get all column headers, which are supported by a checker type, you can call both methods with their parameter colName set to "Column Names". In this case the returned VARIANT contains a BSTR array, with all supported column headers.

2.9.1 getResult(checkerName, resultName)

The method **getResult** returns the current value of the object independent checker result **resultName**. "Object independent" means that the result is not calculateted from an checker object, but belongs to the checker itself. An example of such an result is the result "Total Objects" of the feature extraction checker.

Return Value

VARIANT

The current type and meaning of this VARIANT depend on the concerning checker and the concerning result name. The possible types of the VARIANT are listed in the table in the chapter Getting Checker Results.

The method returns an empty VARIANT if it fails. In this case, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument		**Type**	**Range**	**Description**
[IN]	checkerName	BSTR		ActiveX name of the checker e.g. "FE[1;1]"
[IN]	resultName	BSTR		Name of the result which is accessed

Remarks

To get all result names, which are supported by a checker type, you can call the method with it's parameter resultName set to "Result Names". In this case the returned VARIANT contains a BSTR array, with all supported result names.

2.9.2 getResultsObject(checkerName, objectNumber, colName, numberResults)

The method **getResultsObject** returns the current value(s) under the column header **colName** of the object **objectNumber**. If more than one value is listed under the column header colName, all these values are returned.

Return Value

VARIANT

The current type and meaning of this VARIANT depend on the concerning checker and the concerning header. The possible types of the VARIANT are listed in the table in the chapter Getting Checker Results. If the requested column header of the object contains more than one value, the returned array contains more than one value.

The method returns an empty VARIANT if it fails. In this case, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument		**Type**	**Range**	**Description**
[IN]	checkerName	BSTR		ActiveX name of the checker e.g. "FE[1;1]".
[IN]	objectNumber	SHORT	> 0	Number of the object, for which the result is accessed.
[IN]	colName	BSTR		Column header of the object, for which the result is accessed.
[OUT]	numberResults	SHORT*		Number of result values of the column header colName.

Remarks

This method is very helpful to access the results of an checker object, which does not exist at the creation time of an application.

To get all column headers, which are supported by a checker type, you can call the method with the parameter colName set to “Column Names”. In this case the returned VARIANT contains a BSTR array with all supported column headers, and numberResults contains the number of column headers.

For the feature extraction checker, and (most) user defined checkers, two special results can be got, “Contour” and “Region”:

Result Name	**Description**	**VARIANT Type**
Contour	The **outermost** contour of the object objectNumber is returned. Outermost means, that holes are not contained in the contour. Countours are given by the coordinates of the contour points, and the firts and the last point of a contour are identical.	VT_ARRAY | VT_I4 a[0] : 2 * number of contour points 0 <= i < number of contour points: a[2*i+1]: X-coordinate of contour point i a[2*i+2]: Y-coordinate of contour point i
Region	The region of the object objectNumber is returned. Regions are given in so called runs. A run is a triple with the following meaning: run[0]: row number of the run run[1]: start column number of the run run[2]: end column number of the run	VT_ARRAY | VT_I4 a[0] : 3 * number of runs 0 <= i < number of runs: a[3*i+1]: row number of the run i a[3*i+2]: start column number of the run i a[3*i+3]: end column number of the run i

For the results “Contour” and “Region”, the value returned in **numberResults** is always 0.

For user defined checkers it depends on the definition of the checker if it supports the results “Contour” and/ or “Region”.

2.9.3 getResultsObjects(checkerName, colName, numberObjects, numberResultsObject)

The method **getResultsObjects** returns the current value(s) under the column header **colName** of all objects of the checker **checkerName**.

Return Value

VARIANT

The current type and meaning of this VARIANT depend on the concerning checker and the concerning header. The possible types of the VARIANT are listed in the table in the chapter Getting Checker Results. The VARIANT does always contain an array, even if only one object with one result value does exist. The array contains the result values of the objects in the following order: all result values of object number one, all result values of object number two, and so on.

The method returns an empty VARIANT if it fails. In this case, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument		**Type**	**Range**	**Description**
[IN]	checkerName	BSTR		ActiveX name of the checker e.g. "FE[1;1]".
[IN]	colName	BSTR		Column header of the objects, for which the result is accessed.
[OUT]	numberObjects	SHORT*		Number of objects.
[OUT]	numberResultsObject	SHORT*		Number of result values of the column header colName.

Remarks

This method is very helpful to access the checker results of all objects of an checker at once.

To get all column headers, which are supported by a checker type, you can call the method with the parameter colName set to “Column Names”. In this case the returned VARIANT contains a BSTR array with all supported column headers,
numberObjects is one, and numberResultsObjects contains the number of column headers.

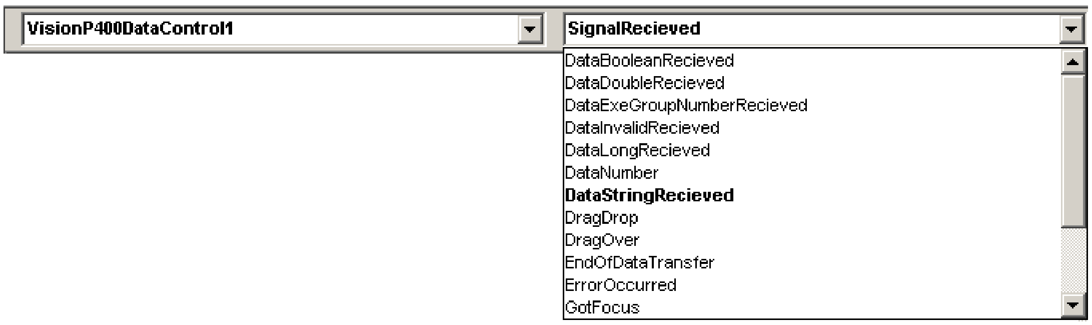
The number of entries in the array, which the returned VA RIANT contains, is numberResultsObject * numberObjects.

3 Events of the ActiveX Control VisionQ400Control

Visual Basic Example

When using Visual Basic you can select an event and Visual Basic automatically writes the first and the last line of the body in your code editor window.

Neues Bild



The following example shows how to handle an event in Visual Basic based on the SignalReceived event.

```
Option Explicit
' Vision Q.400 Signals
Const VISIONQ400_SIG_UNKNOWN = 0
Const VISIONQ400_SIG_PCREND = 1
Const VISIONQ400_SIG_PCREND_OFF = 2
Const VISIONQ400_SIG_PCREADY = 4
Const VISIONQ400_SIG_PCREADY_OFF = 8
Const VISIONQ400_SIG_PCDATAREADY = 16
Const VISIONQ400_SIG_PCDATAREADY_OFF = 32
Const VISIONQ400_SIG_CHANGE_NOTIFY = 64
Const VISIONQ400_SIG_PCERROR = 256
Const VISIONQ400_SIG_STOP = 1024
Const VISIONQ400_SIG_START_LOST = 8192
Const VISIONQ400_SIG_START_LOST_OFF = 16384
Const VISIONQ400_SIG_START_RUNMODE = 65536
Const VISIONQ400_SIG_STOP_RUNMODE = 131072
Const VISIONQ400_SIG_ACTION_ERROR = 16777218
Const VISIONQ400_SIG_EXITING = 16777220


Private Sub VisionQ400Control1_SignalRecieved _
                                                            (ByVal signal As Long)

    Select Case signal
        Case VISIONQ400_SIG_PCREADY
            Label1.BackColor = vbGreen
        Case VISIONQ400_SIG_PCREADY_OFF
            Label1.BackColor = vbRed
        Case VISIONQ400_SIG_PCREND
            Label2.BackColor = vbGreen
        Case VISIONQ400_SIG_PCREND_OFF
            Label2.BackColor = vbRed
        Case VISIONQ400_SIG_PCDATAREADY
            Label3.BackColor = vbGreen
        Case VISIONQ400_SIG_PCDATAREADY_OFF
            Label3.BackColor = vbRed
        Case VISIONQ400_SIG_CHANGE_NOTIFY
            Label4.BackColor = vbGreen
        Case VISIONQ400_SIG_PCERROR
            Label4.BackColor = vbGreen
        Case VISIONQ400_SIG_START_LOST
            Label5.BackColor = vbGreen
        Case VISIONQ400_SIG_START_RUNMODE
            Label6.BackColor = vbGreen
        Case VISIONQ400_SIG_STOP_RUNMODE
            Label6.BackColor = vbRed
        Case VISIONQ400_SIG_ACTION_ERROR
            Label7.BackColor = vbGreen
        Case VISIONQ400_SIG_EXITING
            Label8.BackColor = vbGreen
        Case Else
    End Select

End Sub
```

3.1 SignalRecieved(signal)

The event **SignalRecieved** is fired if the client has to process a signal.

Return Value

none

Argument		**Type**	**Description**
[IN]	signal	LONG	The signal which has to be processed.

Remarks

The signals can have the following values:

Name	**Value**	**Meaning**
VISIONQ400_SIG_UNKNOWN	0x00000000	unknown signal
VISIONQ400_SIG_PCREND	0x00000001	PC Rend
VISIONQ400_SIG_PCREND_OFF	0x00000002	PC Rend Off
VISIONQ400_SIG_PCREADY	0x00000004	PC Ready
VISIONQ400_SIG_PCREADY_OFF	0x00000008	PC Ready Off
VISIONQ400_SIG_PCDATAREADY	0x00000010	PC Data Ready
VISIONQ400_SIG_PCDATAREADY_OFF	0x00000020	PC Data Ready Off
VISIONQ400_SIG_CHANGE_NOTIFY	0x00000040	application change notify
VISIONQ400_SIG_PCERROR	0x00000100	PC error
VISIONQ400_SIG_START_LOST	0x00002000	a start signal is lost
VISIONQ400_SIG_START_RUNMODE	0x00010000	Vision Q.400 steps into run mode
VISIONQ400_SIG_STOP_RUNMODE	0x00020000	Vision Q.400 steps into setup mode
VISIONQ400_SIG_ACTION_ERROR	0x01000002	Vision Q.400 ACTION error
VISIONQ400_SIG_EXITING	0x01000004	Vision Q.400 will exit
VISIONQ400_SIG_APP_LOADED	0x01000008	Vision Q.400 has loaded an application
VISIONQ400_SIG_APP_CLOSED	0x01000010	the current application in Vision Q.400 has been closed

Remarks

VISIONQ400_SIG_PCERROR:
If you receive this signal (the PC error signal), we do recommend to call getLastErrorText() to get information about the error.

VISIONQ400_SIG_START_LOST:
If Vision Q.400 tried to start an application, because a client called the method startApplication(), this signal is not sent to client which called the method.

VISIONQ400_SIG_EXITING:
If you recieve this signal, the ActiveX Control disconnected from Vision Q.400, because Vision Q.400 will exit.
You have not to call the method disconnectFromServer by yourself.

If Vision Q.400 will exit, because a client called the method exitServer(), this signal is not sent to client which called the method.

VISIONQ400_SIG_APP_LOADED:
If you recieve this signal, Vision Q.400 has created or loaded a new application.

If Vision Q.400 loaded an application, because a client called the method openApplication(), this signal is not sent to client which called the method.

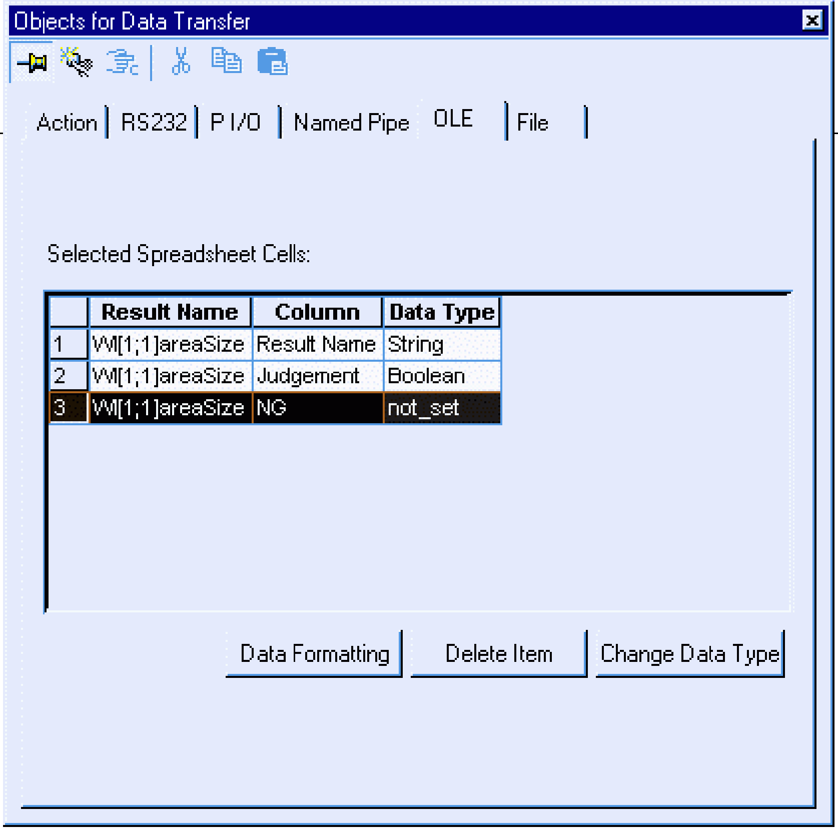
VISIONQ400_SIG_APP_CLOSED:
If you recieve this signal, Vision Q.400 has closed the current application, and not loaded or created a new one. All calls to Vision Q.400 will fail until a new application is loaded or created.

Visual Basic Example

```
Private Sub VisionQ400Control1_SignalRecieved(ByVal signal As Long)

    Select Case signal
        Case VISIONQ400_SIG_PCREADY
            cb_StartApp.Enabled = True
        Case VISIONQ400_SIG_PCREADY_OFF
            cb_StartApp.Enabled = False
        Case VISIONQ400_SIG_START_RUNMODE
            cb_StartRunMode.Enabled = False
            cb_StopRunMode.Enabled = True
            Debug.Print "Run-Mode"
        Case VISIONQ400_SIG_STOP_RUNMODE
            cb_StartRunMode.Enabled = True
            cb_StopRunMode.Enabled = False
            Debug.Print "Setup-Mode"
    End Select

End Sub
```



Selected Spreadsheet Cells:

	Result Name	Column	Data Type
1	WI[1;1]areaSize	Result Name	String
2	WI[1;1]areaSize	Judgement	Boolean
3	WI[1;1]areaSize	NG	not_set

3.2 DataNumber(numberOfBlocks)

The event **DataNumber** is fired to inform how many data items are sent by the current execution of the application.

Return Value

none

Argument	**Type**	**Description**
[IN] numberOfBlocks	SHORT	The number of sent data items.

Remarks

If an execution group is executed, the number of sent data items depends on the executed execution group, because only data items, which “belong” to the executed group, are sent. (Execution group matters are decribed in the Vision Q.400 reference manual in the chapter “Execution Groups”.)

If the whole application is executed, or execution groups are not used, the number of sent data items is the number of entries (rows) in the “Selected Spreadsheet Cells” list of the OLE interface. (To show this list in Vision Q.400, click on the OLE button of the spreadsheet.)

If data is not transferred (number of data items to be sent is zero), the event is not fired, except that the execution group number is transferred (see DataExeGroupNumberRecieved()).

Visual Basic Example

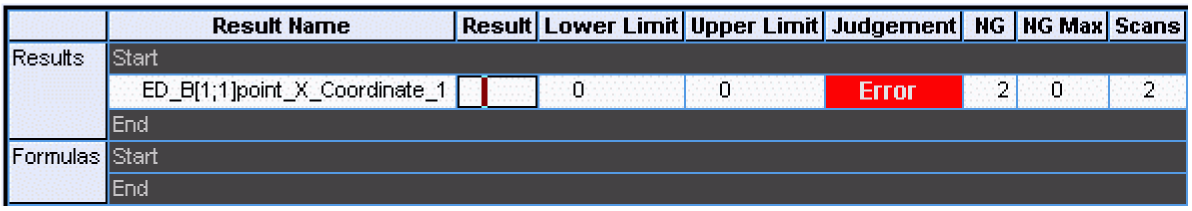
```
Private Sub VisionQ400Control1_DataNumber(ByVal numberOfBlocks As Integer)

    Debug.Print numberOfBlocks

End Sub
```

3.3 DataInvalidRecieved(entryAt)

The event **DataInvalidRecieved** is fired if the client has to process invalid data. Invalid data occurs if data has to be transferred, which belonging checker or formula produced an error:



	Result Name	Result	Lower Limit	Upper Limit	Judgement	NG	NG Max	Scans
Results	Start							
	ED_B[1;1]point_X_Coordinate_1		0	0	Error	2	0	2
	End							
Formulas	Start							
	End							

If invalid data occurs, DataInvalidRecieved is sent **instead** of one of the other events DataLongRecieved(), DataDoubleRecieved(), DataStringRecieved(), or DataBooleanRecieved().

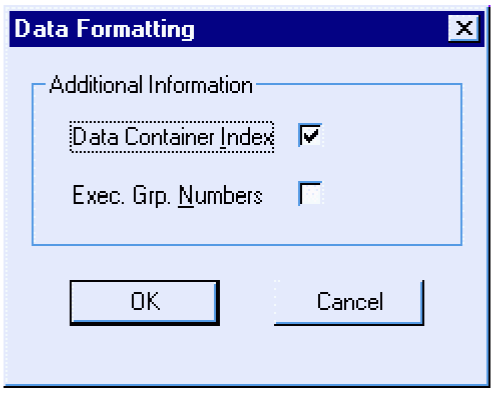
Return Value

none

Argument		Type	Description
[IN]	entryAt	LONG	The index of the entry (row) in the "Selected Spreadsheet Cells" list which belongs to the Invalid data. (To show this list in Vision Q.400 click on the OLE button of the spreadsheet.)

Remarks

The parameter entryAt is only valid if in Format Data of the OLE interface "Data Container Index" is selected. Otherwise entryAt is zero.


Data Formatting

Additional Information

Data Container Index ✓

Exec. Grp. Numbers ☐

Visual Basic Example

```
Private Sub VisionQ400Control1_DataInvalidRecieved(ByVal entryAt As Long)

    Debug.Print entryAt

End Sub
```

3.4 DataLongRecieved(dataLong, entryAt)

The event **DataLongRecieved** is fired if the client has to process a long data.

Return Value

none

Argument		Type	Description
[IN]	dataLong	LONG	The long value, which has to be processed.
[IN]	entryAt	LONG	The index of the entry in the "Selected Spreadsheet Cells" list, which belongs to dataLong.(To show this list in Vision Q.400, click on the OLE button of the spreadsheet.)

Remarks

The parameter entryAt is only valid if in Format Data of the OLE interface "Data Container Index" is selected. Otherwise entryAt is zero. (See DataInvalidRecieved())

If data has to be transferred, which belonging checker or formula produced an error, the event DataLongRecieved is not sent, but the event DataInvalidRecieved() is sent.

Visual Basic Example

```
Option Explicit

Dim LongData(0 To 4) As Long

Private Sub VisionQ400Control1_DataLongRecieved(ByVal dataLong As Long, ByVal entryAt As Long)
    LongData(entryAt) = dataLong

End Sub
```

3.5 DataDoubleRecieved(dataDouble, entryAt)

The event **DataDoubleRecieved** is fired if the client has to process a double data.

Return Value

none

Argument		Type	Description
[IN]	dataDouble	DOUBLE	The double value, which has to be processed.
[IN]	entryAt	LONG	The index of the entry in the “Selected Spreadsheet Cells” list, which belongs to dataDouble.(To show this list in Vision Q.400, click on the OLE button of the spreadsheet.)

Remarks

The parameter entryAt is only valid if in Format Data of the OLE interface “Data Container Index” is selected. Otherwise entryAt is zero. (See DataInvalidRecieved())

If data has to be transferred, which belonging checker or formula produced an error, the event DataDoubleRecieved is not sent, but the event DataInvalidRecieved() is sent.

Visual Basic Example

```
Private Sub VisionQ400Control1_DataLongRecieved(ByVal dataLong As Long, ByVal entryAt As Long)

    Debug.Print "Value  " & dataLong
    Debug.Print "Index  " & entryAt

End Sub
```

3.6 DataStringRecieved(dataString, entryAt)

The event **DataStringRecieved** is fired if the client has to process a string data.

Return Value

none

Argument		Type	Description
[IN]	dataString	LONG	The string value, which has to be processed.
[IN]	entryAt	LONG	The index of the entry in the “Selected Spreadsheet Cells” list, which belongs to dataString.(To show this list in Vision Q.400, click on the OLE button of the spreadsheet.)

Remarks

The parameter entryAt is only valid if in Format Data of the OLE interface “Data Container Index” is selected. Otherwise entryAt is zero.

If data has to be transferred, which belonging checker or formula produced an error, the event DataStringRecieved is not sent, but the event DataInvalidRecieved() is sent.

Visual Basic Example

```
Private Sub VisionQ400Control1_DataStringRecieved(ByVal dataString As String, ByVal entryAt As Long)

    Debug.Print "Value  " & dataString
    Debug.Print "Index  " & entryAt

End Sub
```

3.7 DataBooleanRecieved(dataBoolean, entryAt)

The event **DataBooleanRecieved** is fired if the client has to process a boolean data.

Return Value

none

Argument		Type	Description
[IN]	dataBoolean	VARIANT_BOOL	The Boolean value, which has to be processed.
[IN]	entryAt	LONG	The index of the entry in the “Selected Spreadsheet Cells” list, which belongs to dataBoolean.(To show this list in Vision Q.400, click on the OLE button of the spreadsheet.)

Remarks

The parameter entryAt is only valid if in Format Data of the OLE interface “Data Container Index” is selected. Otherwise entryAt is zero. (See DataInvalidRecieved())

If data has to be transferred, which belonging checker or formula produced an error, the event DataBooleanRecieved is not sent, but the event DataInvalidRecieved() is sent.

Visual Basic Example

```
Private Sub VisionQ400Control1_DataBooleanRecieved(ByVal dataBoolean As Boolean, ByVal entryAt As Long)

    Debug.Print "Value  " & dataBoolean
    Debug.Print "Index  " & entryAt

End Sub
```

3.8 DataExeGroupNumberRecieved(exeGroupNumber)

The event **DataExeGroupNumberRecieved** is fired if the client has to process an execution group number.

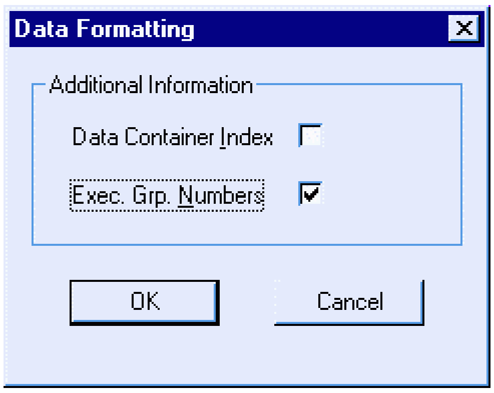
Return Value

none

Argument		Type	Description
[IN]	exeGroupNumber	LONG	The execution group number, which was processed by the last execution of the application.

Remark

If the event DataNumber() is not fired, this event is not fired, too.
Execution group numbers are only transferred if in Format Data of the OLE interface “Exec.Grp. Numbers” is selected.


Data Formatting
Additional Information
Data Container Index [ ]
Exec. Grp. Numbers [✓]

Even if no other data is transferred, the events DataNumber with numberOfBlocks equal to zero and EndOfDataTransfer are fired.

Visual Basic Example

```
Private Sub VisionQ400Control1_DataExeGroupNumberRecieved(ByVal exeGroupNumber As Long)

    Debug.Print "Number of Execution Group: " & exeGroupNumber

End Sub
```

3.9 EndOfDataTransfer()

The event **EndOfDataTransfer** is fired if all data of the last start application is transferred.

Return Value

none

Argument

None

Remark

If data is not transferred (number of data items to be sent is zero), the signal is not fired, except that the execution group number is transferred (see DataExeGroupNumberRecieved()).

Visual Basic Example

```
Private Sub VisionQ400Control1_EndOfDataTransfer()

    Debug.Print "End of data transfer"

End Sub
```

3.10 ImageAvaliable(cameraNumber, image)

The event **ImageAvailable** is fired to transfer the image of an camera.

Return Value

None

Argument		**Type**	**Range**	**Description**
[IN]	cameraNumber	SHORT	[1..12]	The camera number for which the image (or a part of the image) is transferred. May be less than or equal to zero.
[IN]	image	VARIANT		VARIANT which contains the grey values of the image in a safe array of type VT_UI1.

Remarks

ImageAvailable is only fired if the transfer of an image is enabled by the methods setSendImage() or setSendImagePart().

ImageAvailable is a synchronised call, that mean, Vision Q.400 waits until ImageAvailable returns. Therefore it may be better to store the image data and return immediately before a very time consuming processing is performed with the data.

If cameraNumber is less than or equal to zero, the image with the appropriate positive number should be transferred but could not. If this happens (see in the description of the the methods setSendImage() or setSendImagePart()), the parameter image contains an empty VARIANT, and you can call the method getLastErrorText() (or getLastErrorNumber()) for further information.

If cameraNumber is greater than zero, the transfer of the image has succeeded. In this case the safe array in the VARIANT image can have one or two dimensions. (The dimension of the safe array can be changed by a call of the method setProperty() with the parameter name set to “ImageBufferDim”.)

If the safe array has one dimension, it’s lower bound is always 0, and it’s upper bound is (number of columns * image pixel size in Byte * number of rows) - 1.

The number of columns and the number of rows of the image, which should be transferred, can be querried by a call to the method getProperty() with the name set to “TransferredImage”. The image pixel size in Byte can be querried by a call to the method getProperty() with the name set to “ImagePixelSizeInByte”.

If the safe array has two dimensions, it has to be distinguished between the number of columns of the image, which should be transferred, and the number of columns in the safe array: this number is “pixel size in Byte” times bigger than the number of image columns. Only for an eight bit gray value image both values are the same. E.g. if for a RGB color image 100 columns have to be transferred, the safe array contains 300 columns, because every image pixel consists if three Byte.

The bounds of the first array dimension describe the number of Byte which are transferred for every column, and the bounds of the second array dimension describe the number of rows which are transferred.

The lower bound of the first dimension contains the number of the first transferred image column, and the lower bound of the second dimension contains the number of the first transferred image row. E.g if they are 10, and 20 respectively, the transferred image part starts at [10, 20].

The upper bound of the second dimension, which describes the transferred image rows, is the coordinate of the last transferred row. E.g. if it is 45, the image rows 20..45 are transferred.

The upper bound of the first dimension depends not only on the number of transferred image columns, but also on the size in Byte if an image pixel. Therefore, only for eight bit gray value images, this upper bound is the number of the last transferred image column.

For not eight bit gray value images, the number of the last transferred column can be calculated as follows:
Number of transferred image columns = (upper bound – lower bound + 1) / image pixel size in Byte.
Number of last transferred column = (number of transferred image columns + number of first transferred column – 1).

E.g. if a RGB color image is transferred and the bounds of the first dimension are [20, 109], the number of the last transferred image coloumn is 49:
Number of transferred image columns = (upper bound – lower bound + 1) / image pixel size in Byte
= (109 - 20 + 1) / 3 = 90.
Number of last transferred column = (number of transferred image columns + number of first transferred column – 1)
= 30 + 20 -1 = 49.

If the whole image is transferred, the lower bounds of the two array dimensions are always 0. The upper bounds of the first dimension is the (number of image columns – 1) * image pixel size in Byte. The upper bounds of the second dimension is the number of image rows – 1.

If the transferred image (part) is zoomed, the upper bounds of the array dimensions belong to the zoomed image and may not be the values set by setSendImage() or setSendImagePart(): Vision Q.400 adapts the upper bounds to the values needed after the zooming. The lower bounds are left unchanged.

If the transferred image is of type "Color" (see getProperty() with name ImageType), the three Byte values of a pixel are inserted in the order <blue value>, <green value>, <red value> into the safe array.

3.11 ErrorOccurred(errorText)

The event **ErrorOccurred** may be fired if an error occurred in Vision Q.400. This event is independent from the methods of VisionQ400Control and may occur at any time.

Return Value

none

Argument		Type	Description
[IN]	errorText	BSTR	Text which describes the error.

Remarks

We strongly recommend to implement the event ErrorOccurred. Otherwise it can happen that Vision Q.400 does not work anymore, but the client does not know this.

If you display a message box (or an other window) inside the event ErrorOccurred, the event does not return until the message box is closed. But the ActiveX control VisionQ400Control cannot process other events or methods until an event returns. Additionally, it may lead to a timeout error of Vision Q.400 if an event does not return.

If one of the methods of control VisionQ400Control fails, getLastErrorText() or getLastErrorNumber() have to be used to get the correct error information, because the event ErrorOccurred is normally not sent in this case.

Visual Basic Example

```
Private Sub VisionQ400Control1_ErrorOccurred(ByVal errorText As String)

    Debug.Print "Error: " & errorText

End Sub
```

4 Functionality: ActiveX Control VisionQ400Goodies Control

The ActiveX Control VisionQ400Goodies is a collection of useful methods. In contrast to the control VisionQ400Control it can be used without Vision Q.400.

VisionQ400Goodies offers methods for

- writing a bitmap to a file.

The chapters Naming Convention and A Note on Visual Studio of the ActiveX Control VisionQ400Control apply to the the ActiveX Control VisionQ400Goodies, too.

To use the control in Visual Studio on a 64 Bit system, the file VisionQ400Goodies_32.ocx found in the installation directory of Vision Q.400 has to be registered.

5 Methods of the ActiveX Control VisionQ400Goodies Control

5.1 savePixelValuesAsBitmap1(pixelValues, numberOfColumns, pixelType, fileName, comment)

The method **savePixelValuesAsBitmap1** writes an image, given as gray value or color image, to a bitmap file.

Return Value

VARIANT_BOOL

The return value is TRUE if the methods succeed, otherwise FALSE. If the return value is FALSE, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument		**Type**	**Description**
[IN]	pixelValues	VARIANT	A one dimensional safe array which contains the pixel values to be written.
[IN]	numberOfColumns	LONG	The number of columns of the image. The number of rows is internally calculated.
[IN]	pixelType	BSTR	The type of the pixels: “Gray” for 8 Bit gray value images. “Color” for 24 Bit color images.
[IN]	filename	BSTR	The name of the file to which the image has to be written.
[IN]	comment	BSTR	An additional comment to be written into the file. Comment can be the empty string.

Remarks

The VARIANT pixelValues must contain a safe array of type VT_UI1. For gray images, it has to contain one byte for every pixel. For color images, it has to contain three byte for every pixel, in the sequence blue, red, green byte.

The number of rows of the image is calculated from the size of the safe array and the value of the parameter numberOfColoumns.

If filename contains an extension, it has to be “.bmp, otherwise an error will occur. If filename does not contain an extension, the extension “.bmp” will be appended to it before the image is written.

If the length of comment exceeds 4000 characters, comment is truncated to 4000 characters.

5.2 savePixelValuesAsBitmap2(pixelValues, pixelType, fileName, comment)

The method **savePixelValuesAsBitmap2** writes an image, given as gray value or color image, to a bitmap file.

Return Value

VARIANT_BOOL

The return value is TRUE if the methods succeed, otherwise FALSE. If the return value is FALSE, you can call the methods getLastErrorText() (or getLastErrorNumber()) for further information.

Argument		**Type**	**Description**
[IN]	pixelValues	VARIANT	A two dimensional safe array which contains the pixel values to be written.
[IN]	pixelType	BSTR	The type of the pixels: “Gray” for 8 Bit gray value images. “Color” for 24 Bit color images.
[IN]	filename	BSTR	The name of the file to which the image has to be written.
[IN]	comment	BSTR	An additional comment to be written into the file. Comment can be the empty string.

Remarks

The VARIANT pixelValues must contain a safe array of type VT_UI1. It must have two dimensions. For gray images, it has to contain one byte for every pixel. For color images, it has to contain three byte for every pixel, in the sequence blue, red, green byte.

The number of columns of the image is calculated from the size of the first dimension of the safe array and the size of one pixel value in byte.

The number of rows of the image is the size of the second dimension of the safe array.

If filename contains an extension, it has to be “.bmp, otherwise an error will occur. If filename does not contain an extension, the extension “.bmp” will be appended to it before the image is written.

If the length of comment exceeds 4000 characters, comment is truncated to 4000 characters.

5.3 getLastErrorText():

The method **getLastErrorText** returns an explaining text for the last error.

Return Value

BSTR

The explaining text of the last error, or if no error occurred, the empty string.

Argument

none

Remarks

You have to call this method immediately after an error occurred, otherwise a wrong (newer) error text may be returned.

5.4 getLastErrorNumber():

The method **getLastErrorNumber** returns the number of the last error.

Return Value

LONG

The number of the last error, or if no error occurred, 0.

Argument

none

Remarks

You have to call this method immediately after an error occurred, otherwise a wrong (newer) error number may be returned.

This method has to be called before getLastErrorText, because getLastErrorText may clear the error number, but getLastErrorNumber does not.

Normally the explaining text got by getLastErrorText may be enough information, but sometimes the error number may be needed. In this case, it is an good idea to get the error number first, to handle some of the numbers, and to call getLastErrorText for the not handled errors afterwards.

Appendix A: VARIANT Type Conversion in Vision Q.400

If a Vision Q.400 method needs a VARIANT as input parameter, in the description of the method is given, which data type the VARIANT should contain. If possible, Vision Q.400 converts the data given in the VARIANT in the requested data type.

The following tables describe the possible conversions.

Requested Data Type	Given Data Type	Remarks
VT_BOOL	VT_BOOL	
	VT_UI1 VT_UI2 VT_UI4 VT_UI8 VT_I1 VT_I2 VT_I4 VT_I8 VT_R4 VT_R8 VT_UINT VT_INT	A value of 0 or VARIANT_FALSE is converted to FALSE. A value if 1 or VARIANT_TRUE is converted to TRUE. Otherwise an error will occur.
	VT_BSTR	A value of “false” or “FALSE” is converted to FALSE. A value of “true” or “TRUE” is converted to TRUE. All other values are converted into a VT_R8 value, and than converted like a VT_R8 value. If they cannot be converted into a VT_R8 value, an error will occur.

Requested Data Type	Given Data Type	Remarks
VT_BSTR	VT_BSTR	
	VT_BOOL	VARIANT_FALSE -> “FALSE” VARIANT_TRUE -> “TRUE”
	VT_UI1 VT_UI2 VT_UI4 VT_UI8 VT_I1 VT_I2 VT_I4 VT_I8 VT_R4 VT_R8 VT_UINT VT_INT	The given value is converted into a VT_R8 value, and this value is converted into a string. If the given value cannot be converted into a VT_R8 value, an error will occur.

Requested Data Type	Given Data Type	Remarks
VT_UI1 VT_UI2 VT_UI4 VT_UI8 VT_I1 VT_I2 VT_I4 VT_I8 VT_UINT VT_INT	VT_UI1 VT_UI2 VT_UI4 VT_UI8 VT_I1 VT_I2 VT_I4 VT_I8 VT_UINT VT_INT	The given value has to fit in the data range of the requested data type. Otherwise an error will occur.
	VT_BOOL	VARIANT_FALSE -> 0 VARIANT_TRUE -> 1
	VT_BSTR	If the given string can be converted in a valid double value this conversion is done. The converted value has to fit in the data range of the requested data type, too. Otherwise an error will occur.
	VT_R4 VT_R8	The given value has to fit in the data range of the requested data type, and it has to be an integer value. (Its fractional part has to be zero.) Otherwise an error will occur.

Requested Data Type	Given Data Type	Remarks
VT_R4 VT_R8	VT_UI1 VT_UI2 VT_UI4 VT_UI8 VT_I1 VT_I2 VT_I4 VT_I8 VT_UINT VT_INT	The given value has to fit in the data range of the requested data type. Otherwise an error will occur.
	VT_BOOL	VARIANT_FALSE -> 0.0 VARIANT_TRUE -> 1.0
	VT_BSTR	If the given string can be converted in a valid double value this conversion is done. The converted value has to fit in the data range of the requested data type, too. Otherwise an error will occur.