Exporting archived Data from WinCC into a *.csv-File

WinCC/Connectivity Pack

Configuration Example • August 2009

Applikationen & Tools

Answers for industry.

SIEMENS

Warranty, Liability and Support

Note

The Application Examples are not binding and do not claim to be complete regarding the circuits shown, equipping and any eventuality. The application examples do not represent customer-specific solutions. They are only intended to provide support for typical applications. You are responsible for ensuring that the described products are used correctly. These application examples do not relieve you of the responsibility to use sound practices in application, installation, operation and maintenance. When using these application examples, you recognize that we cannot be made liable for any damage/claims beyond the liability clause described. We reserve the right to make changes to these application examples at any time without prior notice. If there are any deviations between the recommendations provided in these application examples and other Siemens publications – e.g. Catalogs – the contents of the other documents have priority.

We do not accept any liability for the information contained in this document.

Any claims against us – based on whatever legal reason – resulting from the use of the examples, information, programs, engineering and performance data etc., described in this Application Example shall be excluded. Such an exclusion shall not apply in the case of mandatory liability, e.g. under the German Product Liability Act (“Produkthaftungsgesetz”), in case of intent, gross negligence, or injury of life, body or health, guarantee for the quality of a product, fraudulent concealment of a deficiency or breach of a condition which goes to the root of the contract (“wesentliche Vertragspflichten”). However, claims arising from a breach of a condition which goes to the root of the contract shall be limited to the foreseeable damage which is intrinsic to the contract, unless caused by intent or gross negligence or based on mandatory liability for injury of life, body or health. The above provisions do not imply a change of the burden of proof to your detriment.

For questions about this document please use the following e-mail address:

online-support.automation@siemens.com

Industry Automation and Drives Technologies Service & Support Portal

This article is taken from the Service Portal of Siemens AG, Industry Automation and Drives Technologies. The following link takes you directly to the download page of this document.

http://support.automation.siemens.com/WW/view/en/38132261

Table of Contents

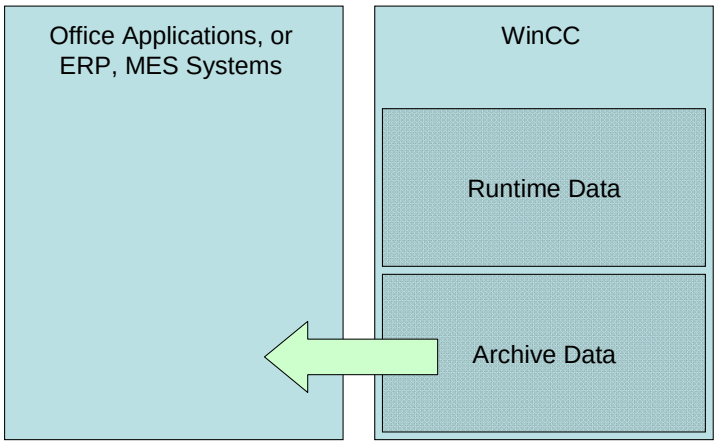
1 Automation Task

1.1 Overview

Overview of the automation task

The figure below gives an overview of the automation task.

Figure 1-1



Description of the automation problem

Using the example of a reverse-osmosis plant, the export of WinCC archive data into a *.csv-file shall be illustrated.

The following data shall be exported:

- Archived tags
- Archived messages

The document shows how to export via Script and via the SQL Manager.

2 Automation Solution

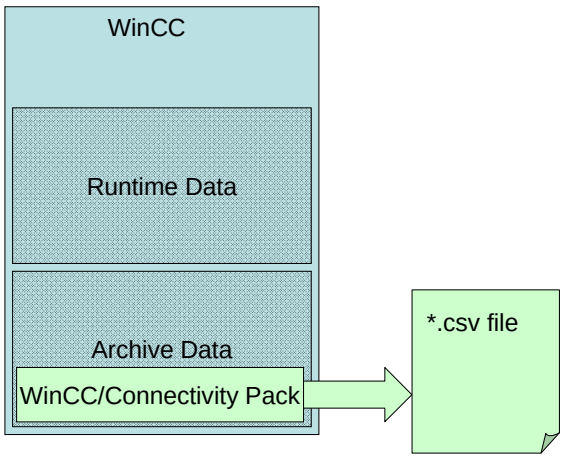
2.1 Overview of the overall solution

To solve the task, the WinCC option Connectivity Pack is used in this example.

Overview

The figure below shows the components of the solution:

Figure 2-1



Topics not covered by this application

This document does not contain any description of the following topics:

- Comparison of the various options and AddOns.
- Installation of the operating system, SIMATIC software, and the SQL server.
- Creating a WinCC project.
- Basics on Visual Basic
- Basics on Microsoft SQL
- Data exchange via the OPC interface of the Connectivity Pack.

Note

Since the document shall mainly provide the user with the basic knowledge on handling the options, many of the expanded standard procedures are not explained.

Further information in this is available in the respective documentations, which are contained in the delivery scope of the product.

2.2 Description of the core functionality

The example project illustrates a reverse-osmosis plant for treatment of process water, as used in the pharmaceutical industry or in breweries.

The represented aggregates and measuring points are simulated entirely by VB scripts in the operating states “Startup”, “Production”, “Shutdown”.

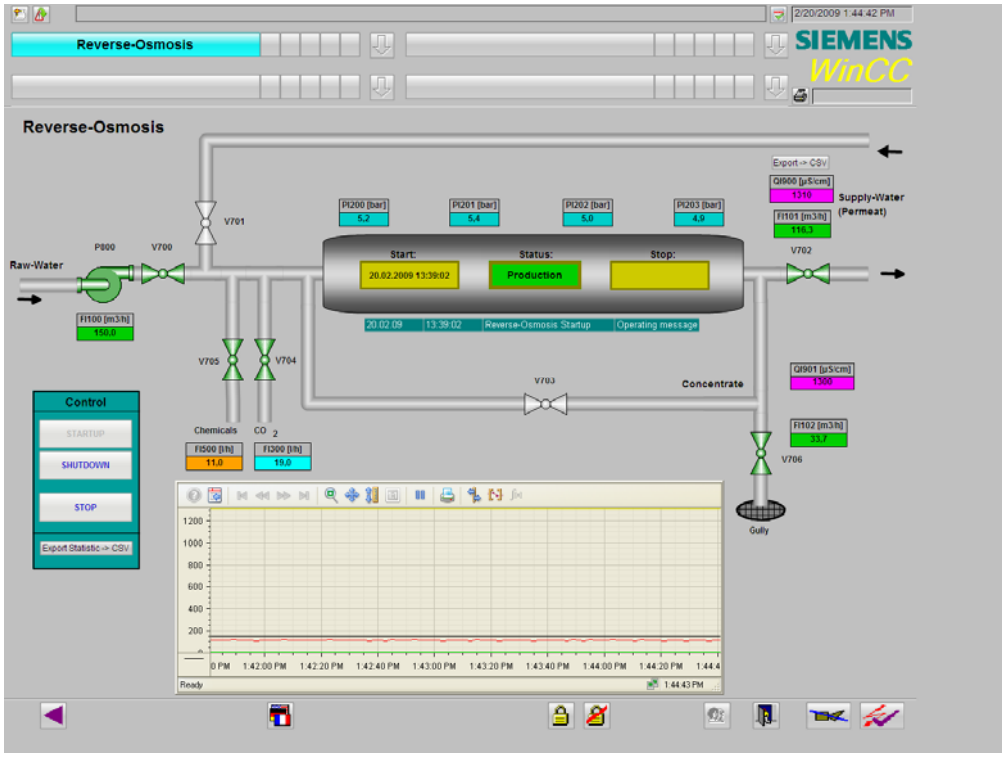
This document and example project describes the solution options, for exporting data archived with WinCC into a CSV-file using the WinCC OLE DB provider.

In four sections it is distinguished between:

- Archived measuring values
- Archived messages
- Export of the archived values with VB Script
- Export of the archived values with SQL Server Import/Export Wizard

Overview and description of the user interface

Figure 2-2



Sequence of the core functionality (of the osmosis plant)

Table 2-1

No	Operating state	Note
1	OFF	The valves and motors are in basic position, analog measuring points either have their start value or value “0”.
2	STARTUP	The valves and motors open/start step by step, all analog measuring points are written with simulated values.
3	PRODUCTION	Valves and motors are activated, the analog measuring points are further written with simulated and slightly varying values.

No	Operating state	Note
4	SHUTDOWN	The valves and motors close/stop step by step, technologically still active analog measuring points are written with simulated values.

2.3 Hardware and software components used

The application was generated with the following components:

Software components

Table 2-2

Component	Qty	MLFB / order number	Note
WinCC	1	6AV6381-2BM07-0AX0	128 Tags
WinCC/Connectivity Pack	1	6AV6371-1DR07-0AX0	

Example files and projects

The following list contains all files and projects used in this example.

Table 2-3

Component	Note
WinCC_Connectivity_Pack.zip	

2.4 Alternative solutions

Overview

Various options or AddOns are available for exchanging data between WinCC and other applications such as MES, ERP systems or other applications from the Office world.

If must be differentiated between the access to online values of WinCC (current values from the Tag Management) and historic data, such as archived measuring values or messages.

The table below provides an overview:

Table 2-4

Product	WinCC(1) integrated	Option	AddOn(2)	Function for data exchange
WinCC	x	-	-	Access options to online values (tags) via OPC DA - or OPC XML DA
ConnectivityPack	-	x	-	Access option to archived process values via OPC HDA Access option to messages via OPC A&E (message events and archived messages) Access option to archived process values and archived messages via WinCC OLE DB
IndustrialDataBridge	-	x	-	Configuration tool and runtime environment for data exchange between various data sources (providers) and data destinations (consumers), also standalone

Product	WinCC[1] integrated	Option	AddOn[2]	Function for data exchange
PM-OPEN EXPORT	-	-	x	Export of Online values and archive data (message values, process values, UserArchive) in user defined text files (ASCII, CSV, HTML/XML format)

(1) WinCC RT / RC license sufficient

(2) AddOns are no SIEMENS IA (Siemens Industrial Automation) products but products by partners

3 Basic Information

3.1 Connectivity Pack

The WinCC/Connectivity Pack enables a licensed access to online and archive data from WinCC. Access to process value archives and message archives is enabled via the WinCC OLE DB provider. Compressed data stored in the data base can be read in uncompressed form. Furthermore, the WinCC OLE DB provider offers analysis functions, such as minimum and maximum of archive tags.

The Connectivity Pack enables access via standardized interfaces:

- OPC HDA (Historical Data Access)
- OPC A&E (Alarm & Events)
- OPC XML DA (web-based data exchange, cross-platform)
- WinCC OLE DB (provider for direct access to process and message archives in the SQL server database on the WinCC RT machine and on a long-term archive server)

A further component of WinCC/Connectivity Pack is the “Archive Connector” tool. This enables connecting or disconnecting exported WinCC archive databases with the SQL server. An overview of the individual database segments is generated. The Archive Connector can monitor directories and automatically connect copied archives. WinCC OLE DB also enables access to archives connected with “Archive Connector”.

3.2 WinCC OLE DB Provider

In this document and the respective example project the benefits and applications of the WinCC OLE DB Provider are described in detail.

As of WinCC V6.0 the WinCC archive databases are segmented and partially stored in compressed binary format. The WinCC OLE DB Provider provides these data from the archive segments transparent in decoded form, the user needs not take any measures regarding data decoding or segmentation when accessing these data.

Via the WinCC OLE DB Provider the Connectivity Pack Client has direct access to the archive data. It is irrelevant whether the data is compressed or uncompressed.

The WinCC OLE DB Provider is available on the following systems:

- WinCC Station
- Connectivity Pack Client
- Connectivity Pack Server
- Connectivity Station.

4 Function Mechanisms of this Application

4.1 Functionality reverse-osmosis plant

The example project illustrates a reverse-osmosis plant for treatment of process water, as used in the pharmaceutical industry or in breweries.

The represented aggregates and measuring points are simulated entirely by VB scripts in the operating states “Startup”, “Production”, “Shutdown”.

The operation of the plant occurs exclusively in the “Operating” window of the WinCC project via the STARTUP, SHUTDOWN, and STOP button.

The current state of the plant is displayed in the “Status” output field, the plant can take on the states *STARTUP*, *PRODUCTION*, *SHUTDOWN* and *OFF*.

Buttons

STARTUP button

The STARTUP button can only be pressed if the plant is in *OFF* state.

Operation:

- The start time is determined and entered into output field “Start”
- The operating status display “Status” goes to STARTUP mode.
- The valves and motors open/start step by step, all analog measuring points are written with simulated values.
- Curve recording starts event-triggered.
- The startup process takes approx. 25 seconds until the plant goes to *PRODUCTION* mode.

SHUTDOWN button

The SHUTDOWN button can only be operated if the plant is in *PRODUCTION* state.

Operation:

- The operating status display “Status” goes to SHUTDOWN mode.
- The valves and motors close/stop in step by step mode, technologically still active analog measuring points continue being written with simulated values.
- The shutdown process takes approx. 25 seconds until the plant goes to *OFF* mode.
- Curve recording stops event-triggered.
- The stop time is determined and entered into output field “Stop”

STOP button

The STOP button can only be operated if the plant is in the state *STARTUP*, *PRODUCTION, or SHUTDOWN*.

Operation:

- The operating status display “Status” goes to *OFF* mode.
- Valves and motors close/stop immediately, analog measuring points are reset to their start value.

- Curve recording stops event-triggered.
- The stop time is determined and entered into output field “Off”

4.2 Functionality when exporting the data

In this document the focus is placed on the traceability for own projects.

The explanations are available in chapter 6 appropriate for the required configuration steps.

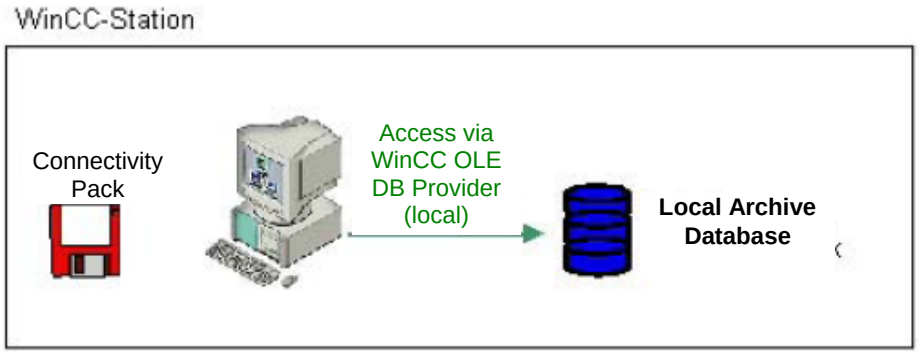
5 Installation

5.1 Installing the software

The example project was created for using the WinCC/Connectivity Pack in a standalone environment on a single-user system with WinCC Version 7.0 SP1 in Windows XP SP2.

The created CSV-files are filed locally on the hard drive of the WinCC station.

Figure 5-1



Software requirements

The WinCC station of the example project requires the following installations:

- WinCC Basis system V7.0 SP1

Installation of the WinCC/Connectivity Pack server or client is not necessary here, since the WinCC Basis system has already been installed.

The WinCC station of the example project requires the following licenses:

- License for the WinCC Basis system V7.0
- License for the WinCC option Connectivity Pack V7.0
- Further possible system configurations for using the WinCC/Connectivity Pack, as well as the appropriate software and licenses are described in chapter 3 of the online-help of the WinCC/Connectivity Pack.

Note

The Software WinCC/Connectivity Pack Client is installed on computers with remote access to WinCC and which receive no WinCC installation.

On such a computer (Connectivity Pack Client) without installed and licensed WinCC software, such as

- WinAC Basis
- WebNavigator Server
- DataMonitor Server
- Connectivity Pack Server

…a WinCC Client Access License (WinCC/CAL) must exist. After the WinCC software has been installed, licensing occurs via the WinCC licenses, WinCC/CAL is not required in this case.

Neither is the installation of the WinCC/Connectivity Pack Client required here.

Note The software WinCC/Connectivity Pack Server is installed on a WinCC long-term archive server.

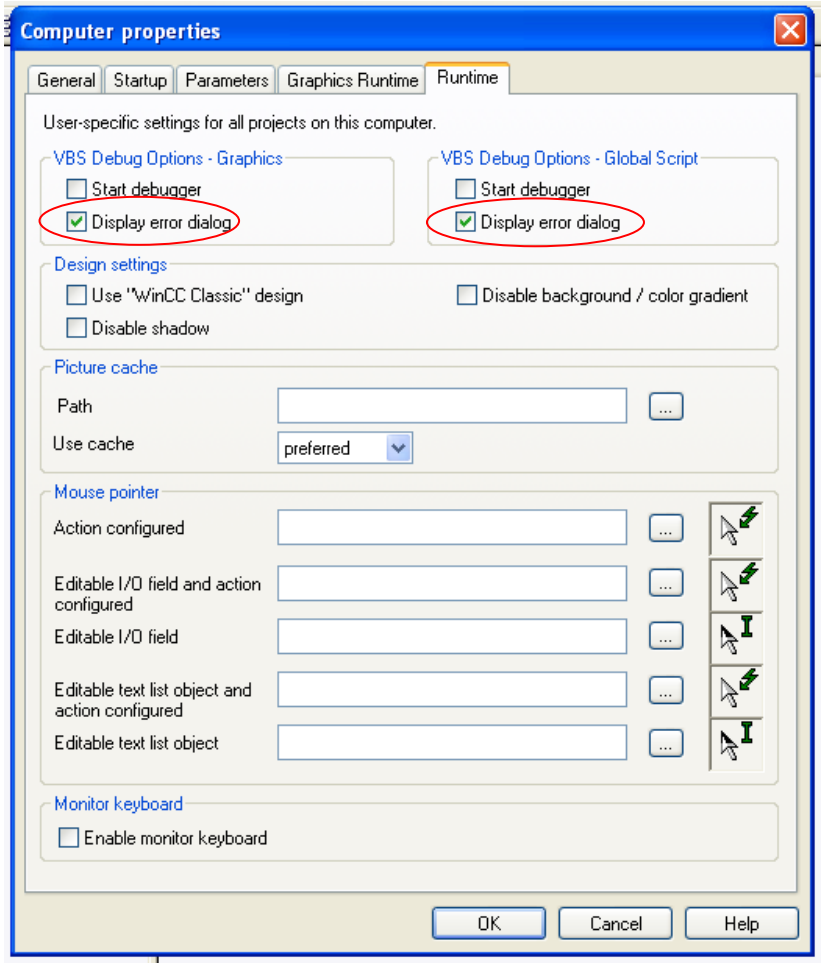
Script-Debugger

If you wish to debug your scripts, a script debugger is necessary which from WinCC V6.2 on is not contained in the delivery scope of WinCC for licensing reasons.

The Microsoft Script Debugger is available via the Microsoft website: http://www.microsoft.com/germany/downloads.

In the WinCC project the following setting is required in the Runtime tab to be able to activate the display and the debugging of script errors.

Figure 5-2:



Start debugger

If the "Start debugger" function has been activated, the debugger is started upon starting the first VB script. Prerequisite is an installed debugger for Visual Basic. The function serves for quick error search during the configuration phase.

The "Start debugger" function can be configured separately for VB scripts in pictures of the Graphics Designer and for VB scripts in Global Script. Activate the respective checkbox for this.

Display error dialog

Of the "Display error dialog" function has been activated, an error dialog with information on the occurred error is displayed upon and error occurring in a VB script. The debugger can be started via a button in the error dialog. Prerequisite is an installed debugger for Visual Basic.

The "Display error dialog" function can be configured separately for VB scripts in pictures of the Graphics Designer and for VB scripts in Global Script. Activate the respective checkbox for this.

5.2 Application software installation

The example project was created as WinCC project without integration into the SIMATIC Manager. The Runtime process simulation occurs via WinCC scripts and requires no further simulation software or process connection.

- Unzip the file named <WinCC_Connectivity_Pack.zip> into any WinCC project directory on your computer.
- Open the WinCC project named <WinCC_Connectivity_Pack.mcp> and in the project adjust the computer name in the Computer Properties, adopt the Windows computer name of your PC via the context menu in “Computer -> Properties”.
- Restart WinCC.

The example project has been configured as single-user system for WinCC Version 7.0 SP1 in Windows XP SP2, the functionalities of the Basic Process Controls (BPC) are used in the project.

The set resolution is 1280 x 1024 pixels.

The project is bilingual (alternatively German/English).

6 Configuration and Settings

6.1 Export of archive values into a CSV file with VB Script

After the end of a production cycle of the osmosis plant the archived measuring values shall be entered into a CSV-file between start and stop time.

Furthermore, all startup and shutdown times of the reverse-osmosis from the passed 24 hours shall be listed in a CSV file.

The solution of this task is displayed here using a VB script, which was generated with the Global Script Editor of WinCC and can be called event-controlled (by pressing a button or via tag trigger).

Note

The VB code for the “WriteArchiveValuesToCSV” functions and “WriteArchive MessagesToCSV” are completely listed at the end of the document.

The code is also stored in the subfolder of the <VBScriptTextfiles> project as text file.

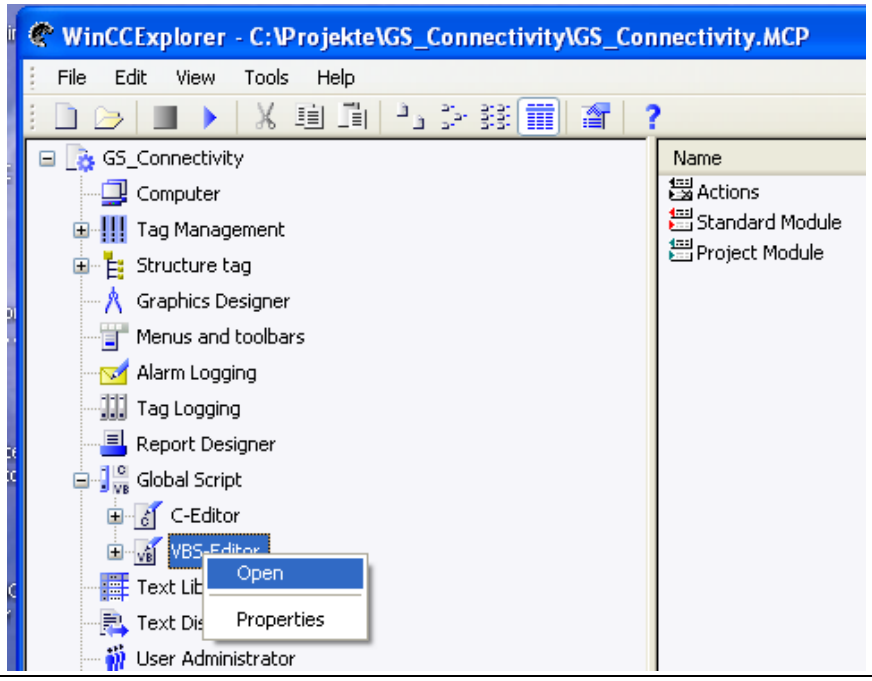
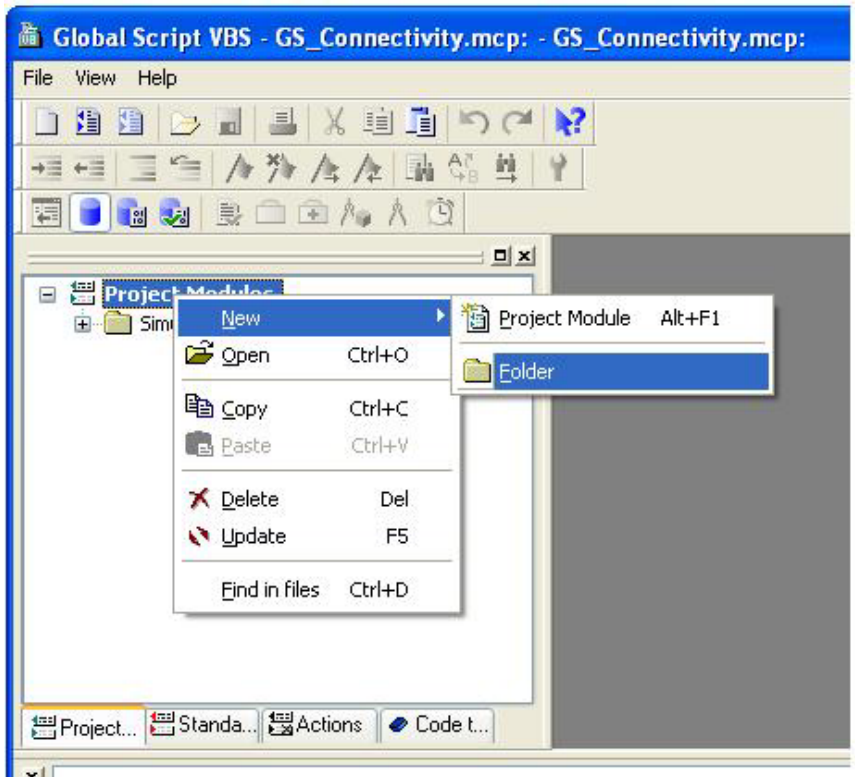
6.1.1 Export of archived tags into a CSV-file with VB Script

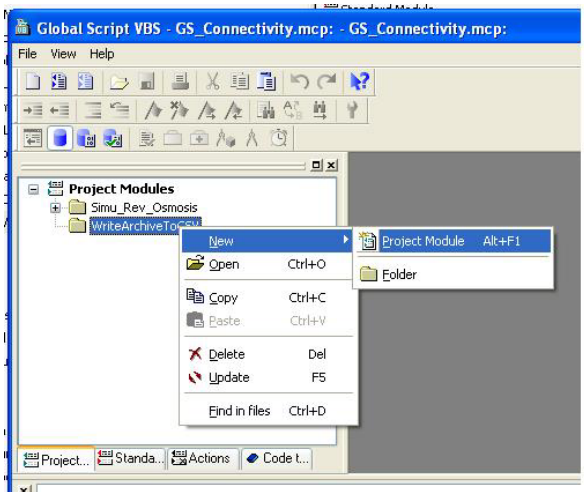
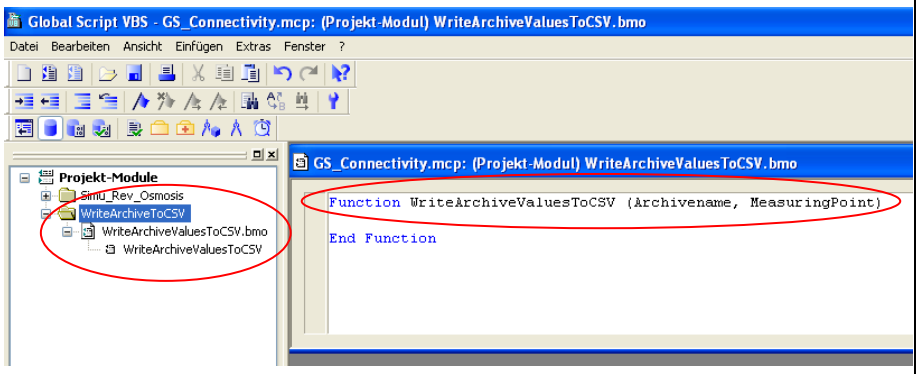
For a clearer representation only the measuring point QI900 (conductivity of the created process water) is considered in this example, whose archived values are written into a CSV-file via a pressed button.

The simultaneous query of several measuring values is alternatively possible without extensive adjustment workload in the VB code.

Creating the VB module

Table 6-1

No	Action
1.	In the WinCC example project you start the Global Script VBS Editor. 
2.	Select the “Project Module” tab and create a new folder named “WriteArchiveToCSV”. 

No	Action
3.	Create a new project module in this new folder. 
4.	Change the sub-procedure into a function. Supplement it by the parameters “Archivename” and “MeasuringPoint” and save the project module under the name “WriteArchiveValuesToCSV”.  Archive name (parameter Archivename) and measuring point name (parameter MeasuringPoint) shall later be transferred upon calling the function.

VB code: creating the CSV file

In the first step of the VB code the CSV-file is initially created on drive C:\. The file name shall be composed of the plant name “Osmosis”, the start time of the osmosis plant and the measuring point name of the measuring point to be archived:

e.g. C:\Osmosis 20.02.2009 10:30:00 QI900.csv

Table 6-2

No	Action

5.	Start the code with the declaration of the following tags:
	`Function WriteArchiveValuesToCSV (Archivename, MeasuringPoint)`
	
	`'////////////////////////////////////////////////////////////////`
	`'1. Step: Creating the CSV-File`
	`'////////////////////////////////////////////////////////////////`
	
	`'Declaration of local Tags`
	`Dim fso 'FileSystemObject`
	`Dim f 'File`
	`Dim ts 'TextStream`
	`Dim path 'Path`
	`Dim StartArchive'Starttime of Archiving`
	`Dim StopArchive 'Endtime of Archiving`
	`Dim TimeStamp 'Timestamp for building the filename`

6.	Subsequently, use the following code to read the start and stop time of the osmosis plant stored in the internal WinCC tag. If start and stop time are missing a message shall be output and the editing of the function be terminated.
	`'Read Start- and Stoptime of Osmosis:`
	`Set StartArchive = HMIRuntime.Tags("DateTime_LastStart")`
	`Set StopArchive = HMIRuntime.Tags("DateTime_LastStop")`
	
	`If StartArchive.read = " " or StopArchive.Read = " " Then`
	`MsgBox "Start- or Stoptime is missing !"`
	`Exit Function`
	`End If`

7.	At plant start and plant stop the respective times (Date/Time) are saved in an internal WinCC tag. The start time is used here for generating a CSV-file name and is brought to a suitable format beforehand.
	The “path” tag contains the complete file name in the form:
	C:\Osmosis + Start time + Measuring point name
	(e.g. C:\Osmosis 19.02.2009 16_11_33 QI900_Conductivity_Permeat.csv)
	`'Generate String for the CSV-Filename and replace ":" with "_"`
	`TimeStamp = FormatDateTime(StartArchive.Read,vbGeneralDate)`
	`TimeStamp = Replace(TimeStamp,":","_")`
	
	`'Path and name for the CSV-File:`
	`path= "C:\Osmosis " & TimeStamp & " " & MeasuringPoint & ".csv"`

No	Action

8.	Subsequently, you use the following code to generate the file system object (fso) and the file object (f) using the previously formed path and file name (path). Beforehand, it shall be checked whether the file already exists and in this case the editing of the function be terminated after output of a message.
	Finally, the CSV-file is opened with the TextStream object (ts) for writing.
	`'Create Filesystemobject and CSV-File if not exists:`
	`Set fso = CreateObject("Scripting.FilesystemObject")`
	`If Not fso.FileExists(path) Then`
	`  fso.CreateTextFile(path)`
	`Else`
	`  MsgBox "File already exits !"`
	`  Exit Function`
	`End If`
	
	`'Create File-Object and open this File for writing`
	`Set f = fso.GetFile(path)`
	`Set ts = f.OpenAsTextStream(2,-2)`

9.	The CSV-file is now created and opened for writing.

Connection to the database

In the second step of the VB code the connection with the database is created via the WinCC OLE DB Provider.

First, some explanations and basics on this.

The WinCC OLE DB Provider enables the access to uncompressed and compressed archive data. For data exchange with own applications, generated e.g. with Visual Basic, Visual Basic Script or VBA, the ADO-DB interface (Active-X data objects for data base access) is used.

Requirement for the connection is that

a) the archive to be read from is part of a WinCC Runtime project or

b) the archive to be read from is connected in the SQL Manager via “Attach Database” or

c) the archive to be read from is connected via “Archive-Connector”

The measured value archive data are entered into a RecordSet by the WinCC OLE DB Provider, the following fields are transferred per data record:

Table 6-3

Field No	Field Name	Type	Description
0	ValueID	Integer (4 bytes)	ID of the process value
1	TimeStamp	DateTime	Timestamp of the process value
2	RealValue	Real (8 bytes)	Process value
3	Quality	Integer (4 bytes)	QualityCode of the process value
4	Flags	Integer (4 bytes)	Flags of the process value (internal control parameter)

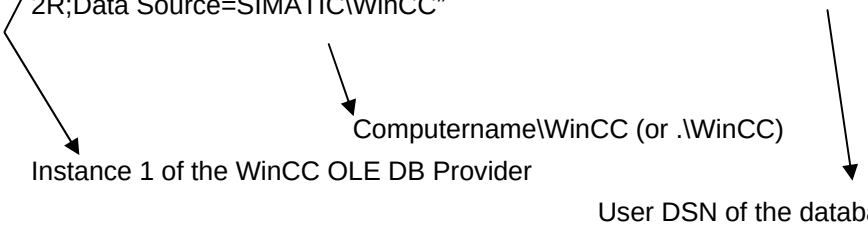
Connection setup

To setup the connection with the database a Connection object must be generated and transferred to the ConnectionString. In the ConnectionString the ProviderName (WinCCOLEDBProvider.1), User DSN (GUID + R of the WinCC project; e.g. CC_GS_Conne_08_12_10_09_05_02R) and as data source WinCC incl. the computer name (e.g. SIMATIC\WinCC) must be transferred.

Instead of the computer name, <.\WinCC> can also be specified, however, the access occurs with reduced performance.

Form of the ConnectionString

“Provider=WinCCOLEDBProvider.1;Catalog=CC_GS_Conne_08_12_10_09_05_02R;Data Source=SIMATIC\WinCC”



Note

For the transparent access for "Catalog" you specify the name of the WinCC project, e.g.: "Catalog=WinCC_Project_Name".

For transparent access of the Central Archive Server or for redundant servers via the OLE DB Provider for "Data Source" you enter the following:

<Symbolic Computer Name>::\WinCC.

When directly accessing an archive tag on the Central Archive Server "CAS", then you use the name of the archive variable. The Central Archive Server "CAS" returns the CAS-ID as ID and not the ID of the archive tag:

<SYMBOLIC COMPUTER NAME>\\<Achive_Tag_Name>.

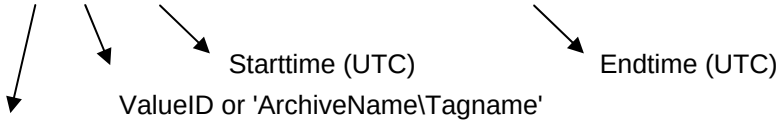
Data query

For a query of the data a Command object must be created which a CommandText is then transferred to for processing.

In the CommandText the ValueID or name of the archive tag and the start and stop time must be transferred.

Format of the CommandText

"Tag:R,1,'2009-01-20 11:15:23.000',"'2009-01-20 13:26:45.000"'



The times (Start time or End time) can be transferred absolute or relative:

Table 6-4

Start time	End time	Queried time range
2009-01-10 14:10:00:000	2009-01-10 14:30:00:000	10.01.2009 14:10 to 14:30
2009-01-10 14:10:00:000	0000-00-00 01:00:00:000	10.01.2009 14:10 to 15:10
0000-00-00 01:00:00:000	2009-01-10 14:30:00:000	10.01.2009 13:30 to 14:30
0000-00-00 00:00:00:000	0000-00-00 00:01:00:000	Archive start to archive end + 1 min.
0000-00-00 00:01:00:000	0000-00-00 00:00:00:000	Archive end - 1 min. to archive start

Note

The time stamp of the process values is stored in UTC (coordinated world time). Therefore, when entering the start and end times in **absolute form** the regional time zone and possibly summer/winter time must be taken into account. Information on converting local computer time into UTC is available in:

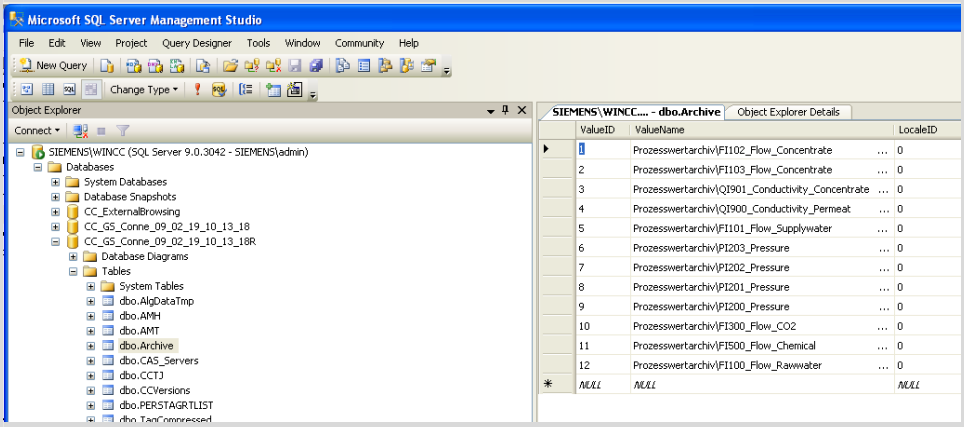
http://support.automation.siemens.com/WW/view/en/24201113

Notes on determining summer/winter time is available in:

http://support.automation.siemens.com/WW/view/en/26729459

Note

The query via ValueID has a higher performance than the query via Tagname. However, the ValueID is not identical with the sequence of the tag in the Tag-Logging editor. The ValueID of a tag can be determined via the SQL Manager. Open the SQL Server Management Studio and there you open the table “dbo.Archive” in the Runtime database of the project. The ValueID is listed in the first column of the table.



Creating the connection and querying the data

Table 6-5

No	Action

10.	Now you declare the further tags:
	`Dim Pro 'Provider`
	`Dim DSN 'Data Source Name`
	`Dim DS 'Data Source`
	`Dim ConnString 'Connection String`
	`Dim MachineNameRT 'Name of the PC from WinCC-RT`
	`Dim DSNRT 'Data Source Nane from WinCC-RT`
	
	`Dim Conn 'Connection to ADODB`
	`Dim RecSet 'RecordSet`
	`Dim Command 'Query`
	`Dim CommandText 'Command-Text`
	`Dim CommandTextStart 'Starttime for SQL-String`
	
	`Dim Duration 'Duration of Production-Cycle`
	`Dim DurationSec 'Duration of Production-Cycle`
	`Dim DurationMin 'Duration of Production-Cycle`
	`Dim DurationHour'Duration of Production-Cycle`
	`Dim DurationDay 'Duration of Production-Cycle`
	`Dim CurrLanguage'Current Language`

11.	Use the following code to read the computer name (@LocalMachineName) and the GUID (@DataSourceNameRT) of the project from the WinCC tags into the previously declared local tags.
	Using these tags you form the ConnectionString, required to create the connection with the database.
	
	Structure of the ConnectionString:
	„Provider=WinCCOLEDBProvider.1;Catalog=CC_GS_Conne_08_12_10_09_05_02R;Data Source=SIMATIC\WinCC“
	
	`'Read the name of the PC-Station and the DSN-Name from WinCC-RT`
	`Set MachineNameRT = HMIRuntime.Tags("@LocalMachineName")`
	`Set DSNRT = HMIRuntime.Tags("@DatasourceNameRT")`
	
	`'Preparing the Connection-String`
	`Pro="Provider=WinCCOLEDBProvider.1;" 'First instance of WinCCOLEDB`
	`DSN="Catalog=" & DSNRT.Read & ";" 'Name of Runtime-Database`
	`DS= "Data Source=" & MachineNameRT.Read & "\WinCC" 'Data Source`
	
	`'Build the complete String:`
	`ConnString = Pro + DSN + DS`

12.	Subsequently, you establish the connection with the database using the application of the ConnectionString:
	`'Make Connection`
	`Set Conn = CreateObject("ADODB.Connection")`
	`Conn.ConnectionString = ConnString`
	`Conn.CursorLocation = 3`
	`Conn.open`

No	Action

13.	The next task consists in forming the Command Text according to the conventions, i.e. the query of data in the database.
	In this example, the relative time is used for the query. The difference between start and end point is formed and this difference is entered as start time. The query comprises the time range “archive end minus time difference until archive end” (see table 4-4).
	The full time difference is initially formed in seconds and only then converted into days, hours, minutes, and seconds. During formatting the leading zeros for number values < 10 must also be created.
	`'Duration between Start an Stop in seconds:`
	`Duration = DateDiff ("s",StartArchive.Read,StopArchive.Read)`
	
	`'Split the Duration in days, hours, minutes and seconds:`
	`DurationMin=Fix(Duration/60)`
	`DurationSec= Duration - (DurationMin * 60)`
	`DurationHour= Fix (DurationMin/60)`
	`DurationMin= DurationMin -(DurationHour * 60)`
	`DurationDay= Fix(DurationHour / 24)`
	`DurationHour= DurationHour - (DurationDay * 24)`
	
	`'Creating leading zeros:`
	`DurationSec= Right("00" & DurationSec,2)`
	`DurationMin= Right("00" & DurationMin,2)`
	`DurationHour= Right("00" & DurationHour,2)`
	`DurationDay= Right("00" & DurationDay,2)`

14.	The complete query is depicted in the “CommandText” tag for further processing.
	The tag is in this example not addressed with the ValueID but with specification of the archive name and the archive tag.
	Format of the CommandText
	"Tag:R,Archivename\Archivetag,'2009-01-20 11:15:23.000','"2009-01-20 13:26:45.000'"
	`'Formating Starttime:`
	`CommandTextStart="'0000-00-" & DurationDay & " " & DurationHour & ":" & _`
	`DurationMin & ":" & DurationSec & ".000'"`
	
	`'Building the complete String:`
	`CommandText="Tag:R,'" & Archivename & "\" & MeasuringPoint & "'," & _`
	`CommandTextStart & ",'0000-00-00 00:00:00.000'"`

15.	Then the RecordSet object is created and the query executed with the previously created CommandText.
	The RecordSet is then set to the first data record in which the first recorded, hence oldest process value is stored.
	`'Create the redordset, read the records an set to first redcordset:`
	`Set RecSet = CreateObject("ADODB.Recordset")`
	`Set Command = CreateObject("ADODB.Command")`
	`Command.CommandType = 1`
	`Set Command.ActiveConnection = Conn`
	`Command.CommandText=CommandText`
	
	`Set RecSet = Command.Execute`
	`RecSet.MoveFirst`

Note When working with time and date in WinCC and Visual Basic it must be noted, that the date format depends on the current regional settings in Windows. The VB code should be designed so that data and time can be evaluated independent of these computer specific settings.

Note The simultaneous querying of several archived values is achieved by specifying the ValueID or Tag name in brackets and separated by semicolon e.g.

"TAG:R,('ValueName_1';'ValueName_2';'ValueName_x'), <TimeBegin>, <TimeEnd>".

Writing the CSV file

The data records from the previously opened RecordSet can then in a simple way be written to the CSV file.

Table 6-6

No	Action

1.	Writing of data records is achieved with the following loop, which writes the measuring point name, the ValueID, the time stamp (UTC) and the process value of each data record into the CSV-file. Beforehand a line with the column headers is generated once. It is considered here, whether the Runtime language is currently German or English (@CurrentLanguage; 1031=German, 1033=English).
	The RecordSet is set to the next data record upon each run:
	`'write recordsets to CSV-File`
	
	`Set CurrLanguage = HMIRuntime.Tags("@CurrentLanguage")`
	`Select Case CurrLanguage.Read`
	`Case 1031 'German`
	`  ts.WriteLine ("Tag-Name;ValueID;Datum/Zeit;Pozesswert") 'header`
	`Case 1033 'English`
	`  ts.WriteLine ("Tag-Name;ValueID;Date/Time;Process-Value") 'header`
	`end select`
	
	`'writing recordsets`
	`Do while Not RecSet.EOF`
	`  ts.WriteLine (MeasuringPoint & ";" & RecSet.Fields(0).Value & ";" & _`
	`  RecSet.Fields(1).Value & ";" & RecSet.Fields(2).Value)`
	`  RecSet.MoveNext`
	`Loop`

No	Action

2.	These instructions located at the end of the script must not be omitted, here the previously created VB objects are destroyed again:
	`ts.close`
	`RecSet.Close`
	`Set Recset=Nothing`
	`Set Command = Nothing`
	`conn.close 'Close connection`
	`Set Conn = Nothing`
	`Set fso = Nothing`
	`Set f = Nothing`
	`Set ts = Nothing`
	
	`End Function`

3.	Save this script.
4.	The script must then be called up e.g. at a button within a process picture, and when calling the archive name and the name of the archive tag to be recorded be transferred as parameter.

Calling the function in the process picture

The created function “WriteArchiveValuesToCSV” must now be called and supplied with both parameters “Archivename” and “MeasuringPoint”.

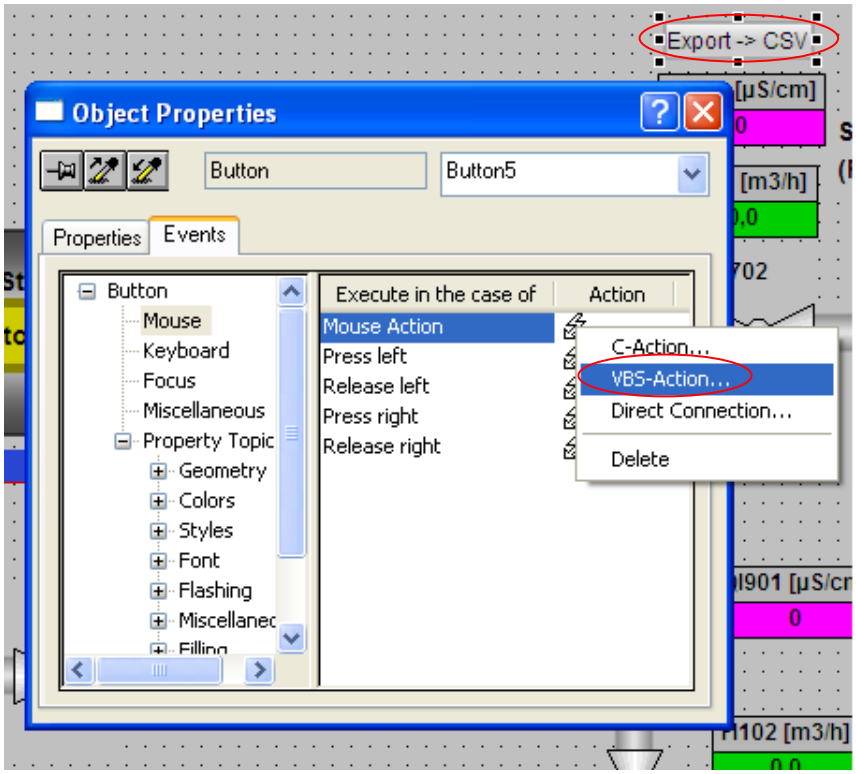
The “Archivename” corresponds to the name of the archive in which the tag to be written is located, in this example project this is the “Prozesswertarchiv” (process value archive).

The “MeasuringPoint” corresponds to the name of the archive tag to be written, as an example the “QI900_Condictivity_Permeat” tag shall be used.

In the “Reverse_Osmosis.pdl” picture a button has already been prepared, apart from measuring point QI900, at which the “WriteArchiveValuesToCSV” function shall be called.

Table 6-7

No	Action
1.	Open the “Reverse_Osmosis.pdl” picture in the Graphics Designer.

No	Action
2.	Open the Properties dialog of the button “Export -> CSV” and open the VBS editor at Events -> Mouse -> Mouse Action. 

3.	Create the following code for calling the previously created VB script:
	`Sub OnClick(Byval Item)`
	`  Dim PA`
	`  Dim MP`
	
	`  PA="Prozesswertarchiv"`
	`  MP="QI900_Conductivity_Permeat"`
	
	`  Call WriteArchiveValuesToCSV (PA, MP)`
	
	`End Sub`

4.	Save the picture. (The result can be followed in chapter 7.1.)

Note The “WriteArchiveValuesToCSV” function can alternatively also be called up automatically, by creating an action in the VBS editor and calling it tag triggered. The trigger could, for example, be the simulated internal tag “StopArchive”.

Note The “WriteArchiveValuesToCSV” function can be adjusted with little workload so that several tags are written to a common CSV-file.

6.1.2 Export of archived messages into a CSV-file with VB Script

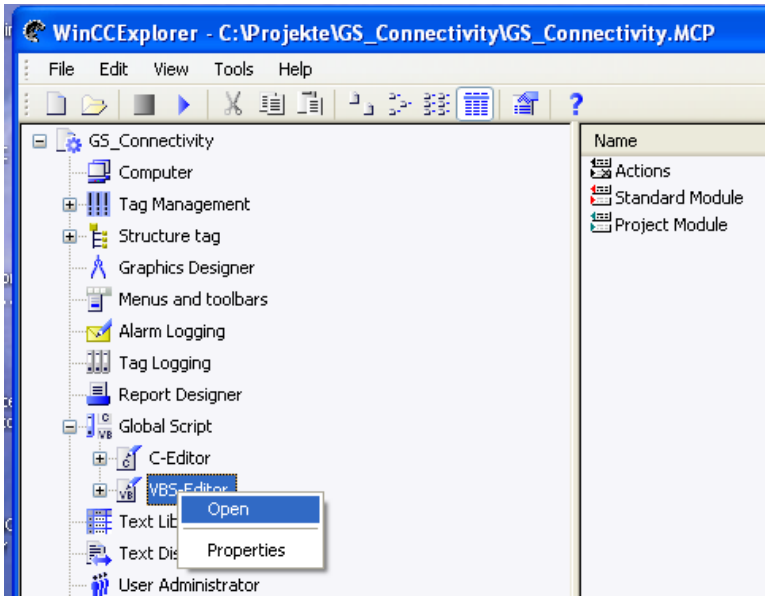
The startup and shutdown times of the osmosis plant are entered into the message archive as operational message. In this configuration example, the startup and shutdown messages of the last 24 hours are written into a CSV-file via pressing a button.

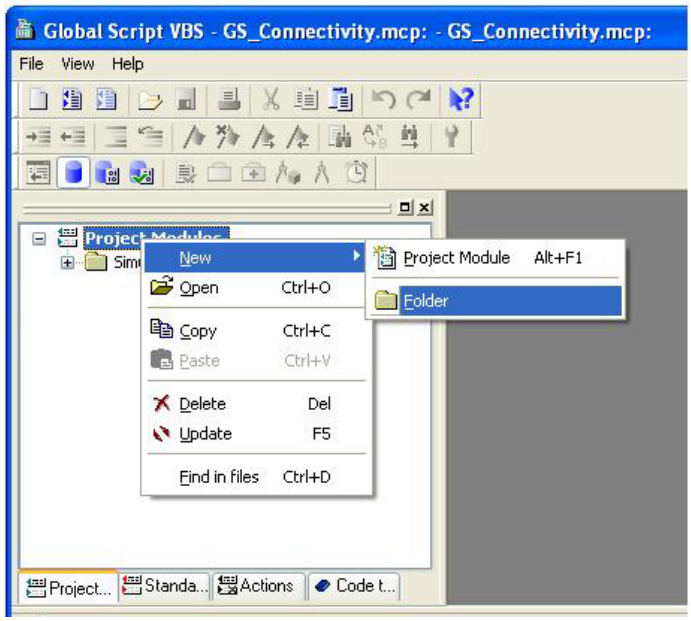
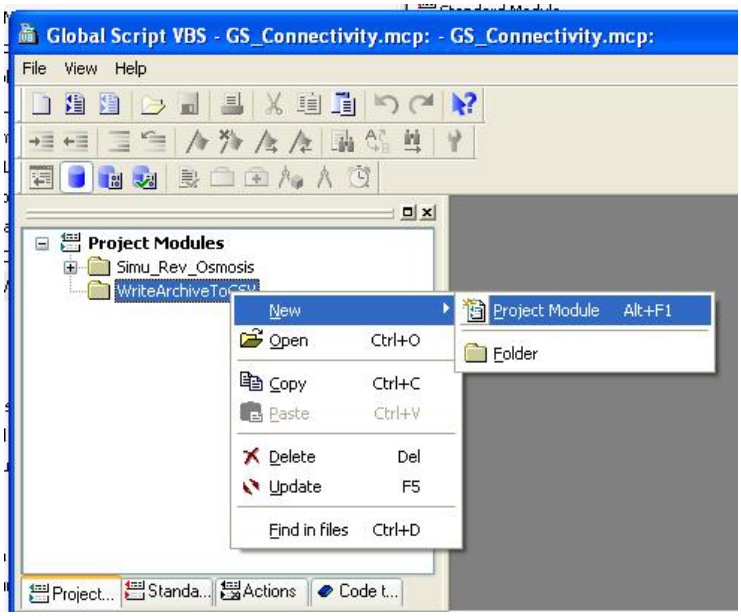
The procedure mainly corresponds to the writing of measuring value archives into a CSV-file.

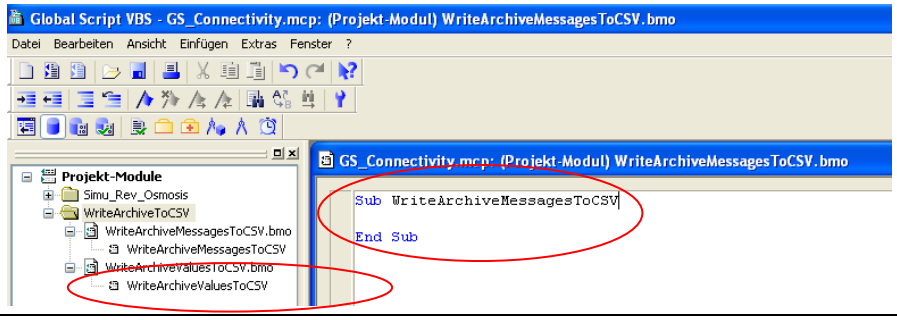
Differences lie in the setup of the record set and in the query of the data records.

Creating the VB module

Table 6-8

No	Action
1.	In the WinCC example project you start the Global Script VBS Editor. 

No	Action
2.	This step can be skipped if chapter 4.1.1 has already been processed. Otherwise, select the “Project Module” tab and create a new folder named “WriteArchiveToCSV”. 
3.	Create a new project module in this folder. 

No	Action
4.	Change the name of the procedure and save the project module under the name “WriteArchiveMessagesToCSV”. 

VB code: creating the CSV file

In the first step of the VB code the CSV-file is initially created on drive C:\. The file name shall be composed of the plant name “Osmosis”, the time at which creating the CSV-file was triggered and the attachment “Messages”:

e.g. C:\Osmosis 20.02.2009 10_30_00 Messages.csv

Table 6-9

No	Action

1.	Start the code with the declaration of the following tags:
	`'Declaration of local Tags`
	`Dim fso 'FileSystemObject`
	`Dim f 'File`
	`Dim ts 'TextStream`
	`Dim path 'Path`
	`Dim StartTime 'Date and time when writing is triggered`
	`Dim TimeStamp 'Timestamp for building the filename`

2.	Then read the current system time with the following code. Bring it into a suitable format and in the “path” tag depict the complete file name in the form:
	C:\Osmosis + Start time + Messages
	`StartTime = Now 'Date and Time when writing messages is triggerd`
	
	`'Generate String for the CSV-Filename:`
	`TimeStamp = FormatDateTime(StartTime,vbGeneralDate)`
	`TimeStamp = Replace(TimeStamp,":","_") 'Replace ":" with "_"`
	
	`'Path and name for the CSV-File:`
	`path= "C:\Osmosis " & TimeStamp & " Messages.csv"`

No	Action

3.	Subsequently, you use the following code to generate the file system object (fso) and the file object (f) using the previously formed path and file name (path). Beforehand, it shall be checked whether the file already exists and in this case the editing of the function be terminated after output of a message.
	Finally, the CSV-file is opened with the TextStream object (ts) for writing.
	`'Create Filesystemobject and CSV-File if not exists:`
	`Set fso = CreateObject("Scripting.FilesystemObject")`
	`If Not fso.FileExists(path) Then`
	`  fso.CreateTextFile(path)`
	`Else`
	`  MsgBox "File already exits !"`
	`  Exit Sub`
	`End If`
	
	`'Create File-Object and open this file for writing`
	`Set f = fso.GetFile(path)`
	`Set ts = f.OpenAsTextStream(2,-2)`

4.	The CSV-file is now created and opened for writing.

Connection to the database

In the second step of the VB code the connection with the database is created via the WinCC OLE DB Provider.

Notes and prerequisites for creating the connection were already described in chapter 4.1.1.3.

For the query of the message archive data a SQL command set must be created.

A message archive data record is structured according to table 4-10.

The field names can be directly addressed in the SQL query.

Table 6-10

Field No.	Field name	Type	Description
0	MsgNr	Integer (4 bytes)	Message number
1	State	Small Integer 2 bytes	Status of the message
2	DateTime	DateTime 8 bytes	Time stamp of the message (date, time without milliseconds)
3	Ms	Small Integer 2 bytes	Time stamp of the message (milliseconds)
4	Instance	VarChar(255)	Instance name of the message
5	Fleags1	Integer 4 bytes	(For internal use only)
6	PValueUsed	Integer 4 bytes	Used process values
7-16	PValue1 to PValue10	Real 8 bytes	Numerical process value 1 to 10
17-26	PText1 to PText10	VarChar(255)	Process value text 1 to 10
27	Computername	VarChar(255)	Computer name
28	Application	VarChar(255)	Application name
29	Comment	VarChar(255)	Comment
30	Username	VarChar(255)	User name
31	Counter	Integer 4 bytes	Continuous message counter

Field No.	Field name	Type	Description
32	TimeDiff	Integer 4 bytes	Time difference for state “coming”
33	Classname	VarChar(255)	Name of message class
34	Typename	VarChar(255)	Name of message type
35	Class	Small Integer 2 bytes	ID of message class
36	Type	Small Integer 2 bytes	ID of message type
37 bis 46	Text1 to Text10	VarChar(255)	Message text 1 to 10
47	AG_NR	Small Integer 2 bytes	Number of the AG
48	CPU_NR	Small Integer 2 bytes	Number of the CPU
49	CrComeFore	Integer 4 bytes	Foreground color for status “coming”
50	CrComeBack	Integer 4 bytes	Background color for status “coming”
51	CrGoFore	Integer 4 bytes	Foreground color for status “going”
52	CrGoBack	Integer 4 bytes	Background color for status “going”
53	CrAckFore	Integer 4 bytes	Foreground color for status “acknowledged”
54	CrAckBack	Integer 4 bytes	Background color for status “acknowledged”
55	LocaleID	Integer 4 bytes	Location of the alarm
56	Prioritiy	Integer 4 bytes	Priority
57	AP_type	Integer 4 bytes	Loop in alarm
58	AP_name	VarChar(255)	Loop in alarm function name
59	AP_PAR	VarChar(255)	Loop in alarm picture
60	InfoText	VarChar(255)	Infotext
61	TxtCame	VarChar(255)	Text coming
62	TxtWent	VarChar(255)	Text going
63	TxtCameNWent	VarChar(255)	Text coming and going
64	TxtAck	VarChar(255)	Text acknowledged
65	AlarmTag	Integer 4 bytes	Message tag
66	AckType	Small Integer 2 bytes	Acknowledgement type
67	Params	Integer 4 bytes	Parameter
68	Servername	VarChar(255)	Server name

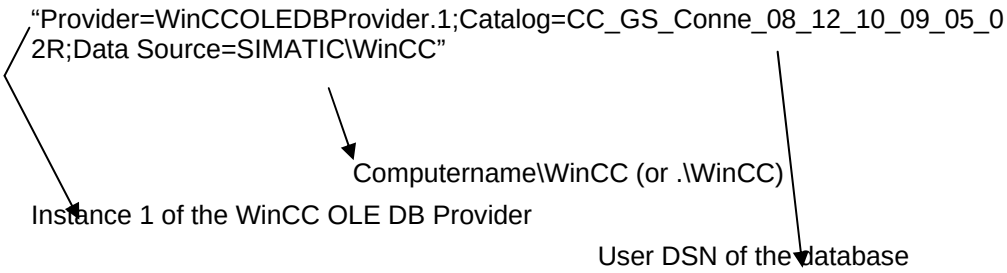
Connection setup

The setup of the connection occurs analog to the example for the access to the measured value archive.

To setup the connection with the database a Connection object must be generated and transferred to the ConnectionString. In the ConnectionString the ProviderName (WinCCOLEDBProvider.1), User DSN (GUID + R of the WinCC project; e.g. CC_GS_Conne_08_12_10_09_05_02R) and as data source WinCC incl. the computer name (e.g. SIMATIC\WinCC) must be transferred.

Instead of the computer name, <.\WinCC> can also be specified, however, the access occurs with reduced performance.

Form of the ConnectionString



Data query

An SQL query is used for querying the message archives.

For a query of the data a Command object must also be created here to which the SQL query is then transferred as CommandText for processing.

In the CommandText the name of the database table and one or several query conditions must be specified.

Format of the CommandText

ALARMVIEW:SELECT * FROM <ViewName> [WHERE <Condition> ……, optional]

Application of the parameters <Viewname> and <Condition>

Table 6-11

Parameter	Description
ViewName	Name of the database table. The table must be given in the desired language. The "ViewName" for the five European languages for example is: ALGVIEWDEU: German message archive data ALGVIEWENU: English message archive data ALGVIEWESP: Spanish message archive data ALGVIEWFRA: French message archive data ALGVIEWITA: Italian message archive data The "ViewName" for the five Asian languages for example is: ALGVIEWCHS: Chinese (simplified) message archive data ALGVIEWCHT: Chinese (traditional) message archive data ALGVIEWJPN: Japanese message archive data ALGVIEWKOR: Korean message archive data
Condition	Filter criterion e.g.: DateTime>'2003-06-01' AND DateTime<'2003-07-01' DateTime>'2003-06-01 17:30:00' MsgNr = 5 MsgNr in (4, 5) State = 2 For DateTime only absolute time values can be used.

Example:

Reads all German entries of message number 5 recorded after 01.01.2009:

"ALARMVIEW:SELECT * FROM ALGVIEWDEU WHERE MsgNr=5 and DateTime > 2009-01-01"

Note

Only the languages installed in the WinCC Basis system or configured in the WinCC Text Library are supported. Information on possible query languages or the respective "ViewName" is available in the SQL server in the connected message archives at "Views". All languages are displayed there with their identifiers e.g. "ALGVIEWENU". Supported in the respective archive.

Note

The time stamp of the messages is filed in UTC (coordinated world time). Therefore, when entering the start and end times in **absolute form**, the regional time zone and possibly summer/winter time must be taken into account.

Unlike for the query of the process value archives, a relative time is not possible here.

Creating the connection and querying the data

Table 6-12

No	Action

1.	Now you declare the further tags:
	`Dim Pro 'Provider`
	`Dim DSN 'Data Source Name`
	`Dim DS 'Data Source`
	`Dim ConnString 'Connection String`
	`Dim MachineNameRT 'Name of the PC from WinCC-RT`
	`Dim DSNRT 'Data Source Nane from WinCC-RT`
	
	`Dim Conn 'Connection to ADODB`
	`Dim RecSet 'RecordSet`
	`Dim Command 'Query`
	`Dim CommandText 'Command-Text`
	
	`Dim sqlSec 'Seconds for SQL`
	`Dim sqlMin 'Minutes for SQL`
	`Dim sqlHour 'Hours for SQL`
	`Dim sqlDay 'Day for SQL`
	`Dim sqlMonth 'Month for SQL`
	`Dim sqlYear 'Year for SQL`
	`Dim CurrLanguage 'Current Language`

<table>
<tr><th>No</th><th>Action</th></tr>
<tr><td>2.</td><td>Use the following code to read the computer name (@LocalMachineName) and the GUID (@DataSourceNameRT) of the project from the WinCC tags into the previously declared local tags.
Using these tags you form the ConnectionString, required to create the connection with the database.
<u>Structure of the ConnectionString:</u>
“Provider=WinCCOLEDBProvider.1;Catalog=CC_GS_Conne_08_12_10_09_05_02R;Data Source=SIMATIC\WinCC”
<pre>'Read the name of the PC-Station and the DSN-Name from WinCC-RT
Set MachineNameRT = HMIRuntime.Tags("@LocalMachineName")
Set DSNRT = HMIRuntime.Tags("@DatasourceNameRT")

'Preparing the Connection-String
Pro="Provider=WinCCOLEDBProvider.1;" 'First instance of WinCCOLEDB
DSN="Catalog=" & DSNRT.Read & ";" 'Name of Runtime-Database
DS= "Data Source=" & MachineNameRT.Read & "\WinCC" 'Data Source

'Build the complete String:
ConnString = Pro + DSN + DS</pre></td></tr>
<tr><td>3.</td><td>Subsequently, you establish the connection with the database using the application of the ConnectionString:
<pre>'Make Connection
Set Conn = CreateObject("ADODB.Connection")
Conn.ConnectionString = ConnString
Conn.CursorLocation = 3
Conn.open</pre></td></tr>
<tr><td>4.</td><td>The next task consists in forming the Command Text according to the conventions, i.e. the query of data in the database.
In this example the previously saved start time of the query (acknowledgement of the button) is detected and reduced by 24 hours. This time is reformatted into “YYYY-MM-DD hh:mm:ss”.
During formatting the leading zeros for number values < 10 must also be created here.
<pre>'Date and time 24 hours before
StartTime=DateAdd("h",-24,StartTime)

'Split in Years, months, days, hours, min.,seconds
sqlSec=second (StartTime)
sqlMin=Minute (StartTime)
sqlHour=Hour (StartTime)
sqlDay=Day (StartTime)
sqlMonth=Month (StartTime)
sqlYear=Year (StartTime)

'Creating leading zeros
sqlSec=Right("00" & sqlSec,2)
sqlMin=Right("00" & sqlMin,2)
sqlHour=Right("00" & sqlHour,2)
sqlDay=Right("00" & sqlDay,2)
sqlMonth=Right("00" & sqlMonth,2)</pre></td></tr>
</table>

No	Action

5.	The complete query is depicted in the “CommandText” tag for further processing.
	In this example, the messages with message number < 4 are selected from the time of the last 24 hours.
	Format of the CommandText
	"ALARMVIEW:SELECT * FROM <ViewName> [WHERE <Condition> ……, optional]"
	If the Runtime language has been set to German (@CurrenLanguage=1031), ViewName with “AlgViewDeu” is entered as parameter for active Runtime language English (@CurrenLanguage=1031) “AlgViewEnu” is selected.
	`'Formating Starttime fpr SQL-Statement:`
	`StartTime="'" & sqlYear & "-" & sqlMonth & "-" & sqlDay & " " _`
	`& sqlHour & ":" & sqlMin & ":" & sqlSec & "'"`
	
	`'Building the complete String:`
	
	`Set CurrLanguage = HMIRuntime.Tags("@CurrentLanguage")`
	`Select Case CurrLanguage.Read`
	`Case 1031 'German`
	`CommandText= "ALARMVIEW:Select * FROM AlgViewDeu WHERE DateTime>" & _`
	`StartTime & "AND MsgNr < 4 AND State = 1"`
	`Case 1033 'English`
	`CommandText= "ALARMVIEW:Select * FROM AlgViewEnu WHERE DateTime>" & _`
	`StartTime & "AND MsgNr < 4 AND State = 1"`
	`End Select`

6.	Then the RecordSet object is created and the query executed with the previously created CommandText.
	The RecordSet is then set to the first data record in which the first recorded hence oldest message is stored.
	`'Create the recordset, read the records and set to first recordset:`
	`Set RecSet = CreateObject("ADODB.Recordset")`
	`Set Command = CreateObject("ADODB.Command")`
	`Command.CommandType = 1`
	`Set Command.ActiveConnection = Conn`
	`Command.CommandText=CommandText`
	
	`Set RecSet = Command.Execute`
	`RecSet.MoveFirst`

Note

When working with time and date in WinCC and Visual Basic it must be noted, that the date format depends on the current regional settings in Windows. The VB code should be designed so that data and time can be evaluated independent of these computer specific settings.

Writing the CSV file

The data records from the previously opened RecordSet can then in a simple way be written to the CSV file.

Table 6-13

No	Action

1.	Writing of data records is performed with the following loop. In this example, the RecordSet fields with field number 2 (DateTime), field number 0 (MsgNr), field number 39 (message text 3) and field number 33 (name of the Class) are read from the data record and written to the CSV-file.
	Beforehand a line with the column headers is generated once. The currently set Runtime language is also considered here.
	The RecordSet is set to the next data record upon each run:
	`'write recordsets to CSV-File`
	
	`Select Case CurrLanguage.Read`
	`Case 1031 'German`
	`  ts.WriteLine ("Datum/Zeit;Meldenr.;Ereignis;Klasse")`
	`Case 1033 'Englisch`
	`  ts.WriteLine ("Date/Time;MsgNr.;Event;Class")`
	`End Select`
	
	`Do while Not RecSet.EOF`
	`  ts.WriteLine (RecSet.Fields(2).Value & ";" & RecSet.Fields(0).Value & ";" & _`
	`  RecSet.Fields(39).Value & _`
	`  ";" & RecSet.Fields(33).Value)`
	`  RecSet.MoveNext`
	`Loop`

2.	These instructions located at the end of the script must not be omitted, here the previously created VB objects are destroyed again:
	`ts.close`
	`RecSet.Close`
	`Set Recset=Nothing`
	`Set Command = Nothing`
	`conn.close  'Close connection`
	`Set Conn = Nothing`
	`Set fso = Nothing`
	`Set f = Nothing`
	`Set ts = Nothing`
	
	`End Function`

3.	Save this script.
4.	The script must then be called, for example, at a button within a process picture. No parameters are transferred.

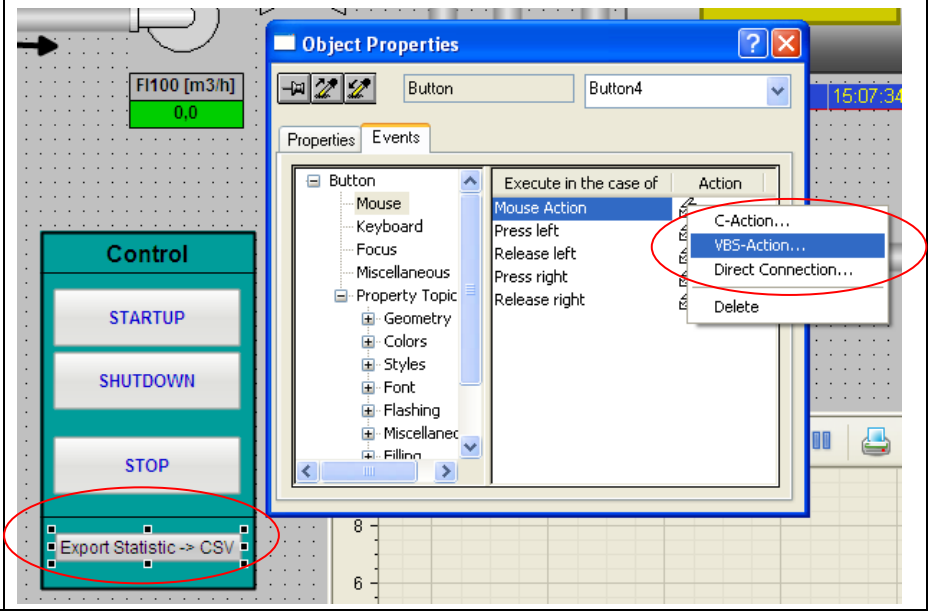
Calling the function in the process picture

The created function “WriteArchiveMessagesToCSV” must now be called.

In the “Reverse_Osmosis.pdl” picture a button has already been prepared in the operating window for the osmosis plant, at which the “WriteArchiveMessagesToCSV” function shall be called.

Table 6-14

No	Action
1.	Open “Reverse_Osmosis.pdl” picture in the Graphics Designer.

No	Action
2.	Open the Properties dialog of the button “Export Statistic -> CSV” and open the VBS editor with Events -> Mouse -> Mouse Action. 

3.	Create the following code for calling the previously created VB script:
	`Sub OnClick(Byval Item)`
	`WriteArchiveMessagesToCSV`
	`End Sub`

4.	Save the picture. (The result can be followed in chapter 7.2.)

Note The “WriteArchiveMessageToCSV” function can alternatively also be called up automatically, by creating an action in the VBS editor which is called time- or tag-triggered.

6.2 Export of archive values into a CSV file with SQL

The export of archive values from SQL is configured directly in the “SQL Server Management Studio” using the “SQL Server Import/Export Wizard” and can be triggered immediately from the Wizard following the parameterization.

Additionally, there is the option of storing the created parameterization as SSIS package (.DTSX-file) in the Windows file system and to trigger the export later by executing this file. The SQL query and hence the scope of the data to be executed is stored as default in the SSIS package.

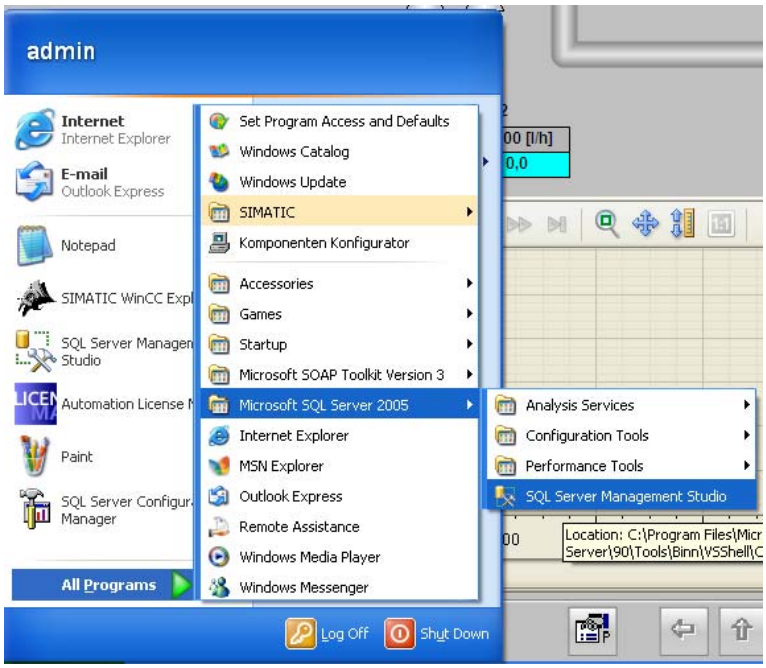
When accessing the WinCC archive data, WinCC OLE DB Provider must be specified as source, the data destination can be e.g. a newly created, separate SQL database or an Excel, Text or CSV-file.

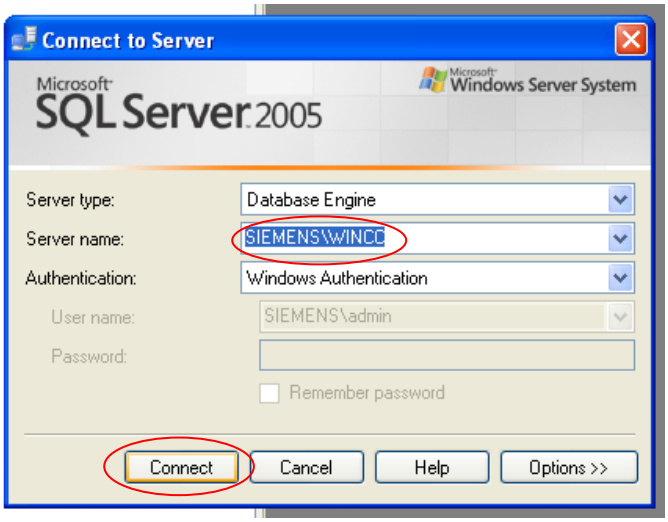
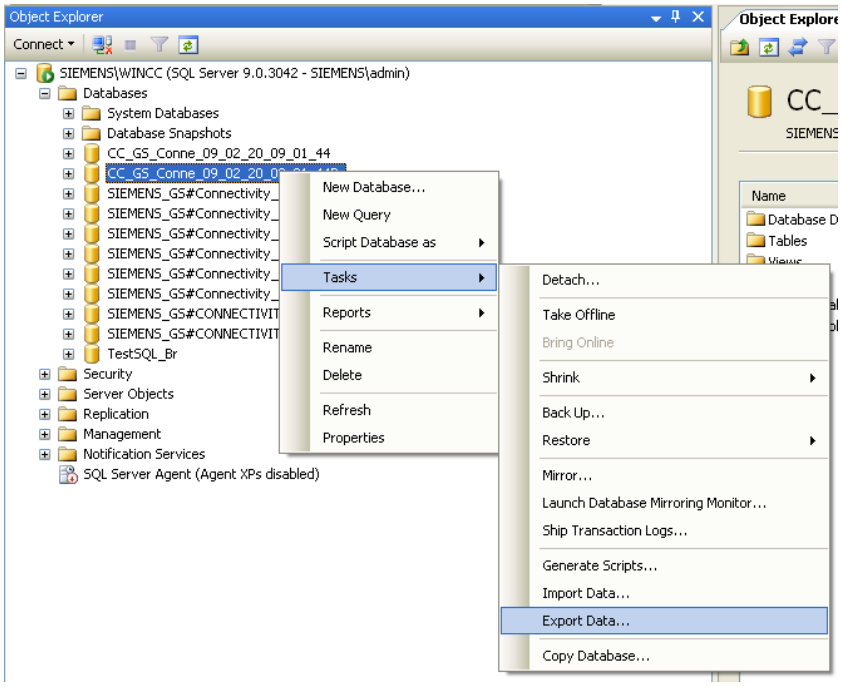
In the following instructions the configuration of the “SQL Server Import/Export Wizards” for the export of archived measured values and messages is written to a CSV-file.

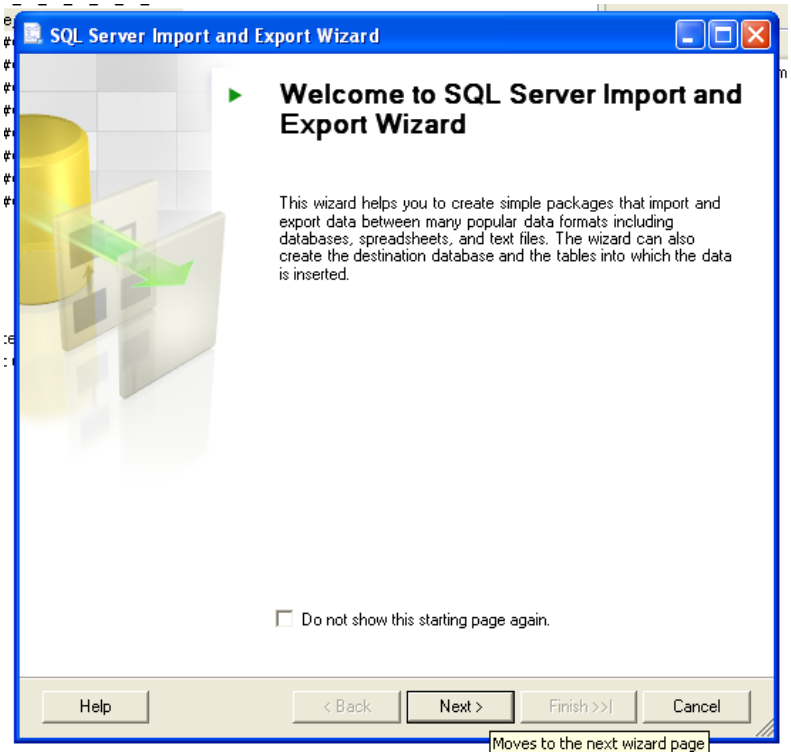
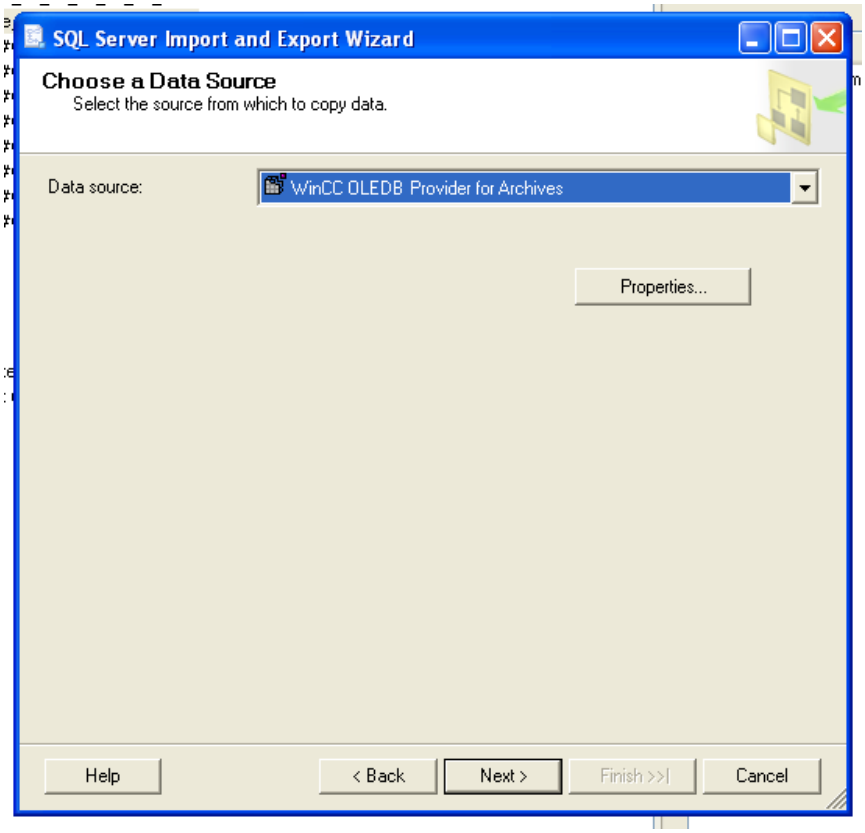
The instruction applies for the SQL Server 2005, which was applied first in WinCC V6.2. In WinCC Version 6.0 the SQL server 2000 was used for which the export of archive data was configured via the similar but not identically structured “DTS Import/Export Wizard”.

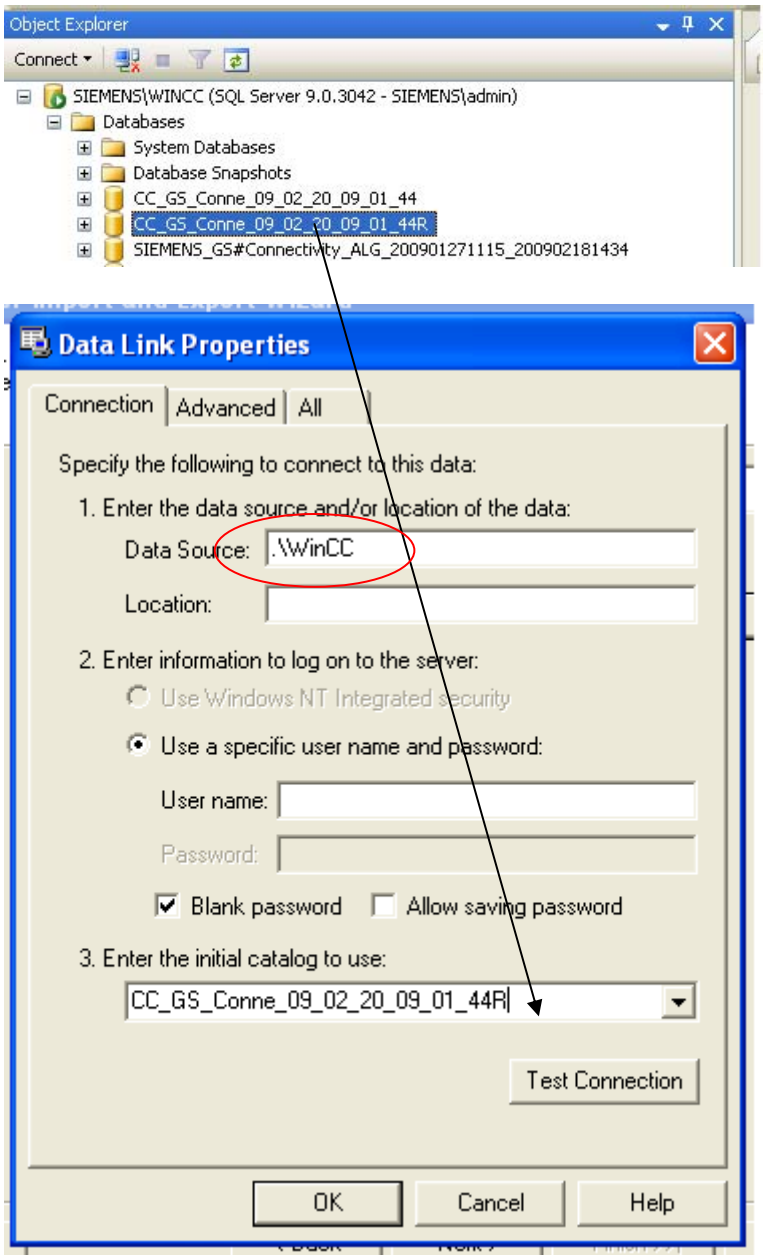
6.2.1 Export of archived measuring values into a CSV-file with SQL

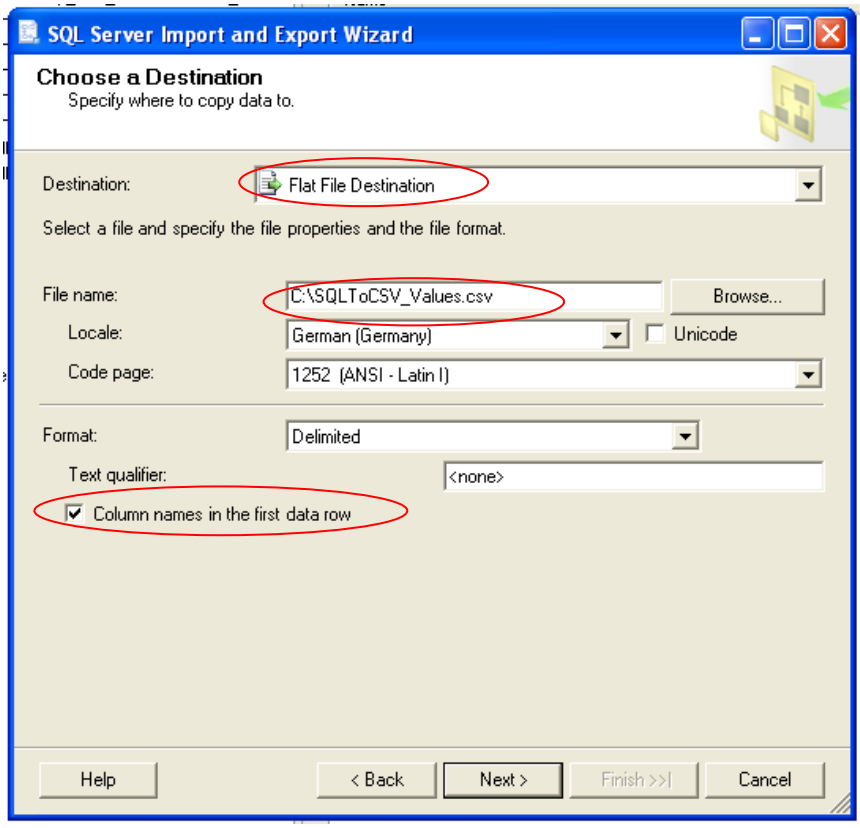
Table 6-15

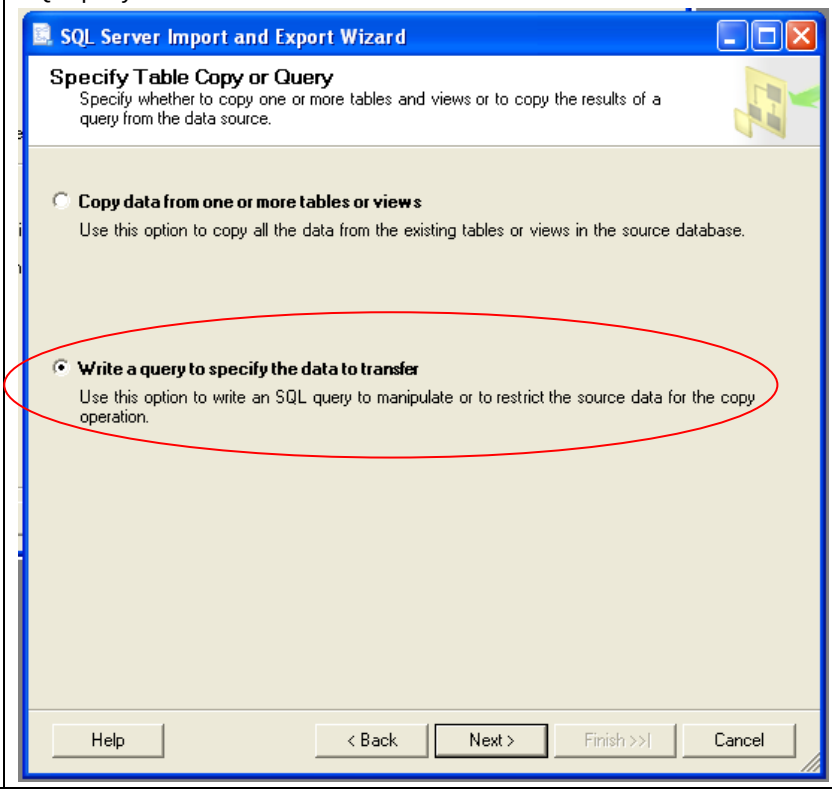
No	Action
1.	Start WinCC, open the WinCC project and activate Runtime. Here you go to the “Reverse-osmosis” picture and start the osmosis plant via the STARTUP button in the control window; wait until the plant has reached the “Production” status. After a short runtime you shut down the plant with the “SHUTDOWN” button.
2.	Start the SQL Server Management Studio: 

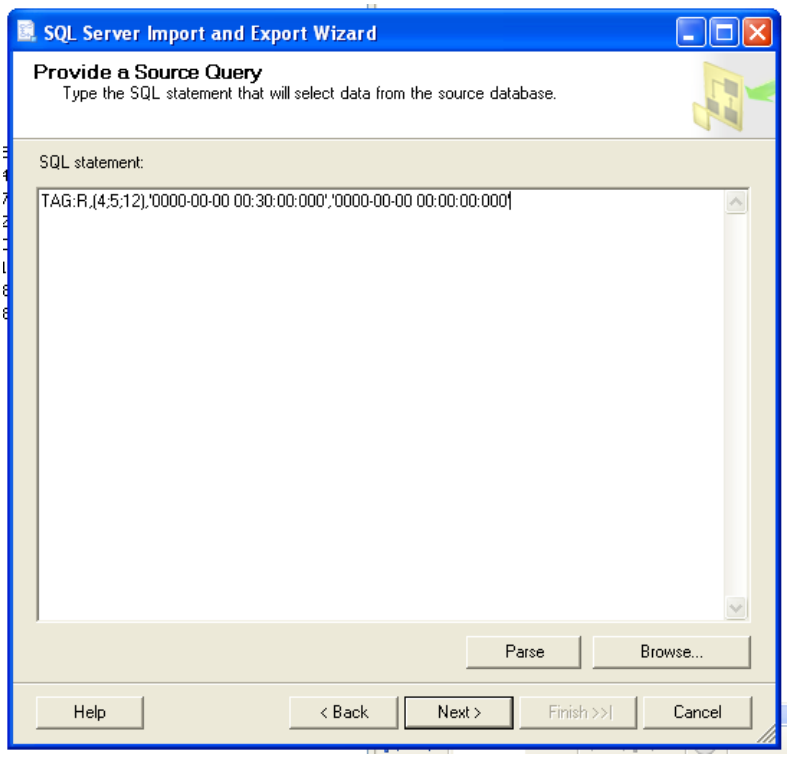
No	Action
3.	Generate the connection with WinCC: 
4.	Start the SQL Server Import/Export Wizard via the context menu of the WinCC Runtime database (CC_GS_Conne_09_02_20_09_01_44R): 

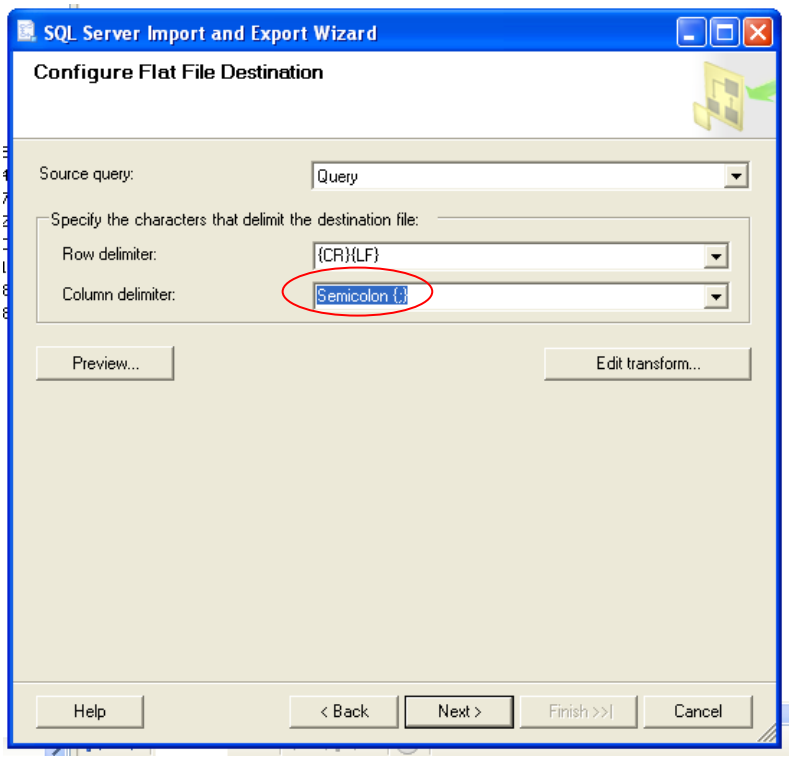
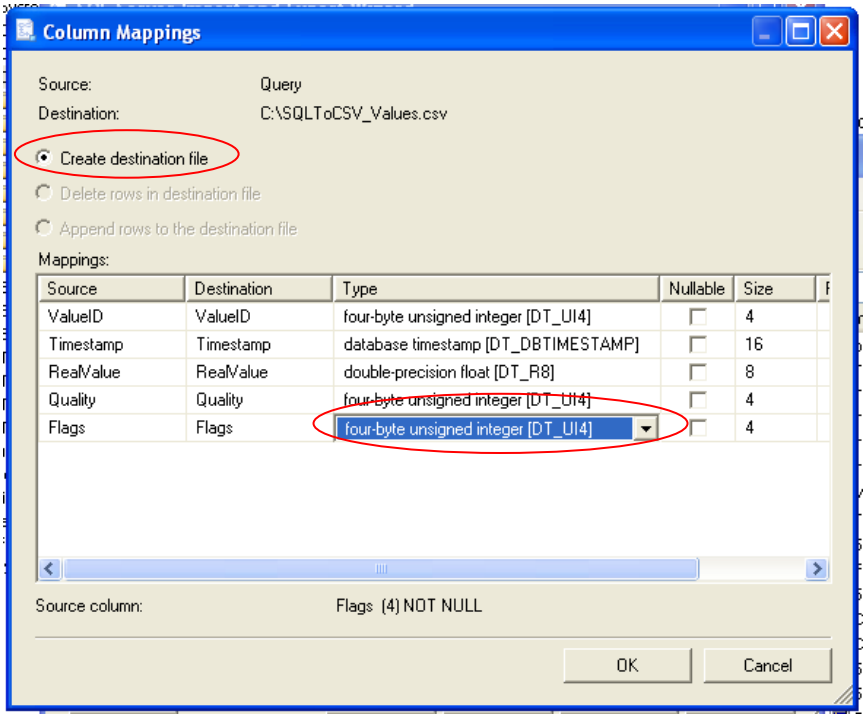
No	Action
5.	Exit the start screen with “Next” 
6.	As data source you select the WinCC OLE DB Provider for Archives and then acknowledge the “Properties” button 

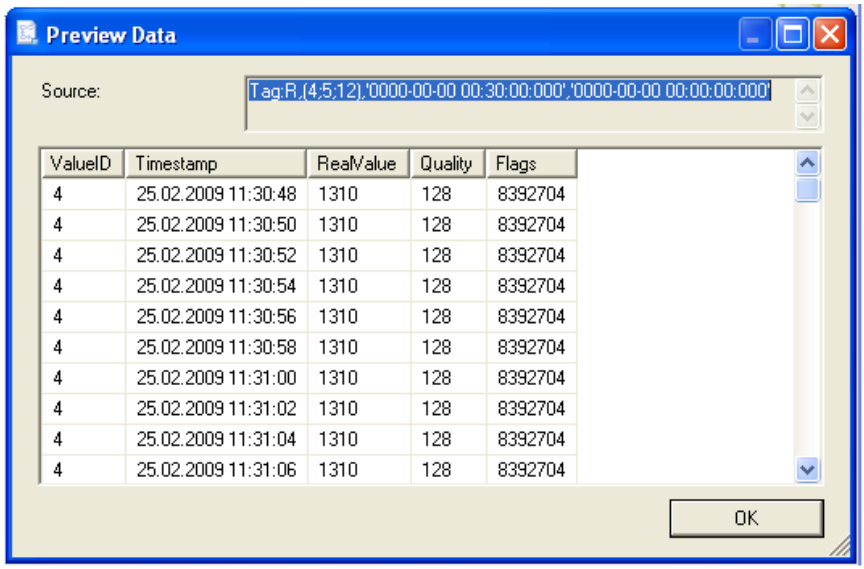
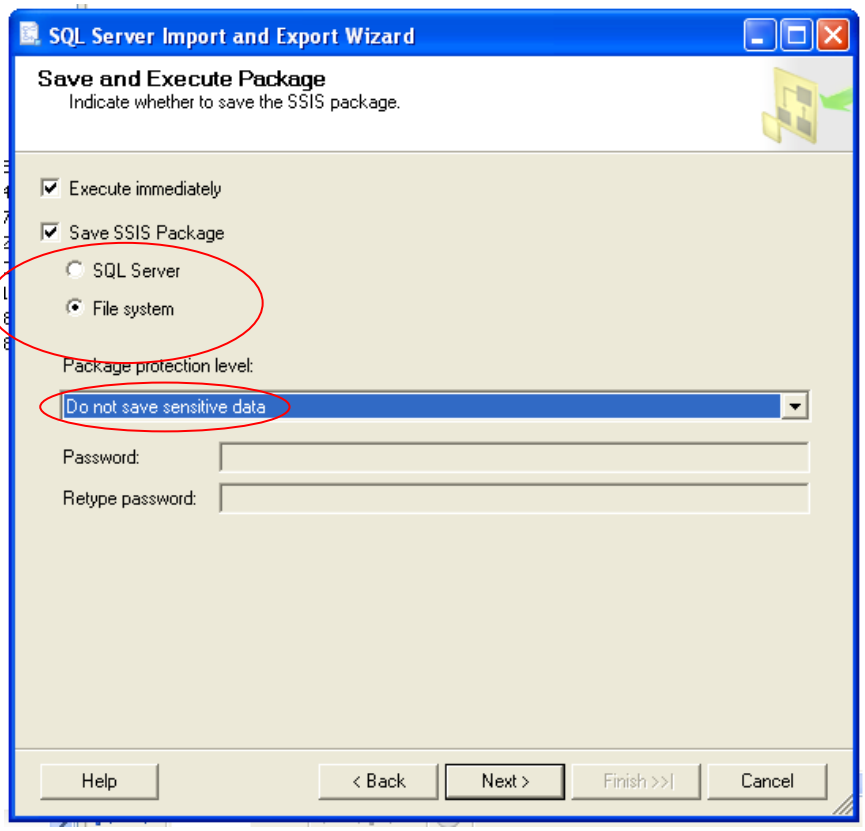
No	Action
7.	Here you set ".\WinCC" as data source and the name of the WinCC Runtime database as catalog. The name of the WinCC Runtime database is located in the Databases folder of the SQL Server Management Studios. Then quit the mask with the buttons "OK" and "Next".  Acknowledge the details via the OK button and then continue the Wizard via the "Next" button.

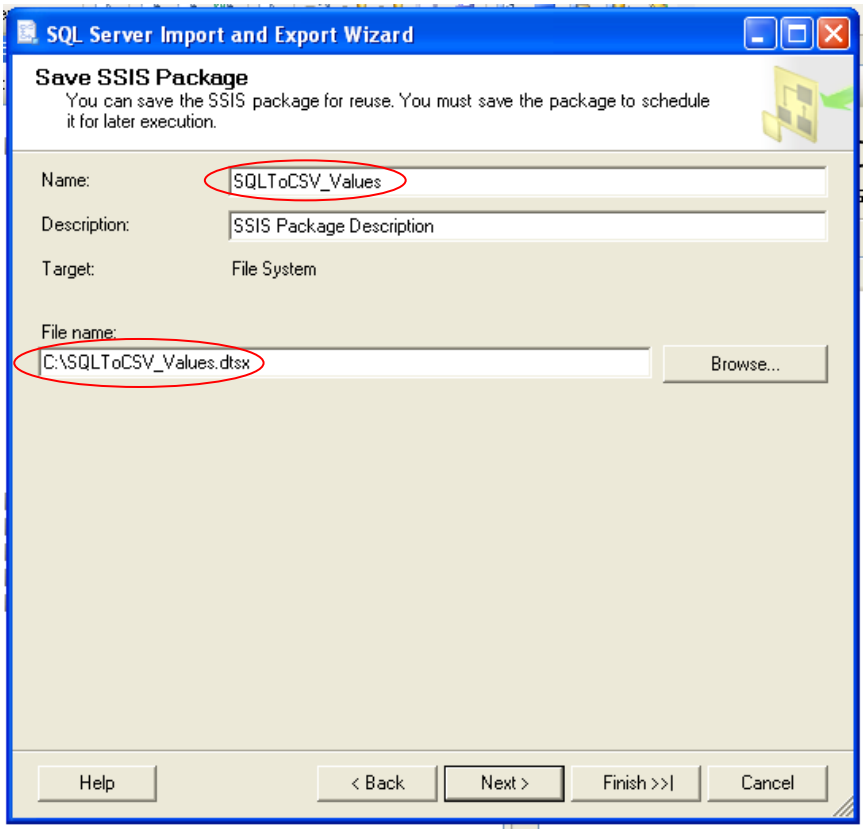
No	Action
8.	Select the data destination “Flat File Destination” and enter file location and name of the CSV-file. Activate the “Column names in first data row” checkbox so the column headers are written to the first line of the CSV-file.  Exit the mask by pressing the “Next” button.

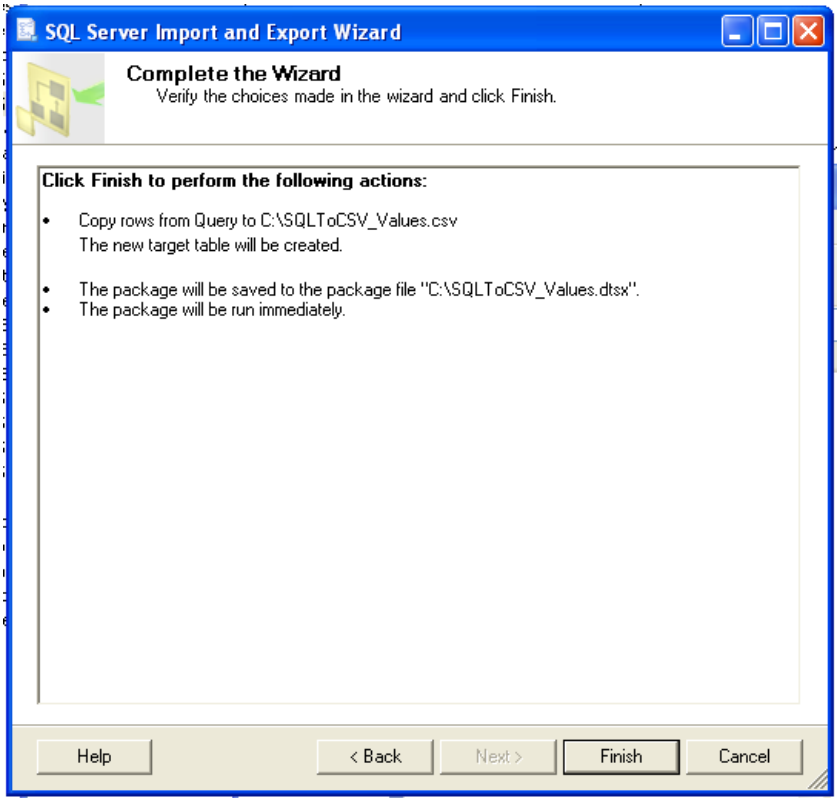
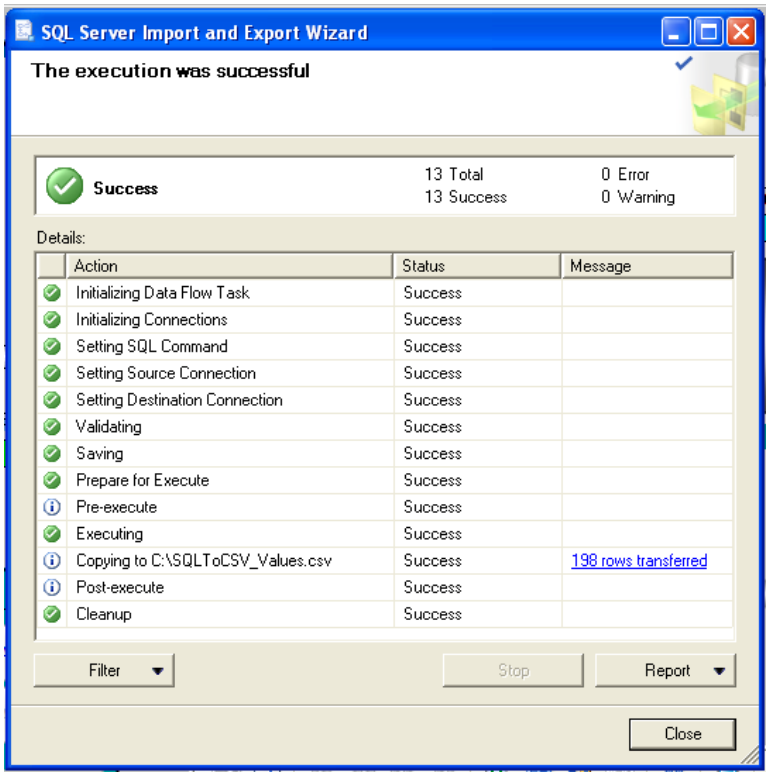
No	Action
9.	Activate the “Write a query to specify the data to transfer” checkbox to specify the SQL query in the next mask. Continue the Wizard with the “Next” button. 

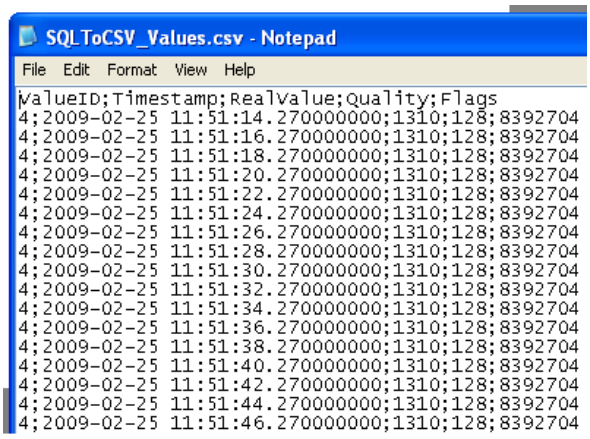
No	Action
10.	Here you enter the desired SQL query. In this example the archive tags with ValueID 4,5, and 12 are queried over a time period of the last 30 minutes. A list of ValueIDs in the SQL statement is put in brackets, the ValueIDs are separated with semicolon. The query structure corresponds to the description in chapter 4.1.1.3 . Exit the mask by pressing the “Next” button.  ValueID: See not at the end of the table.

No	Action
11.	Enter semicolon as the “Column delimiter” and press the “Edit transform” button to specify the data formats. 
12.	Set the data formats as follows:  Then quit the mask with “OK”. If the CSV-file to be created already exists in the destination path, “Create destination file” is not active and the data formats cannot be specified!

No	Action
13.	The result of the query can be checked via “Preview”:  Then quit the mask with “OK” and “Next”.
14.	The activate saving the SSIS package in the Windows file system without protection function and quit the mask with “Next”. 

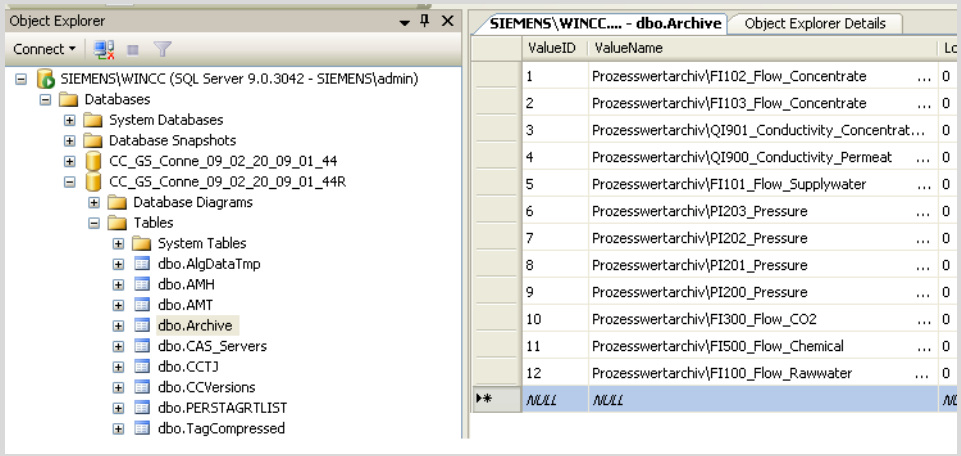
No	Action
15.	Enter the file name and storage location for the SSIS package and press the “Next” button. 

No	Action
16.	In the subsequent mask, the set parameters are summarized, press “Finish” to start the export. 
17.	The result of the export is then logged, via “Reports” you have the option to save the log as text file. 

No	Action
18.	Check the generated CSV-file: 

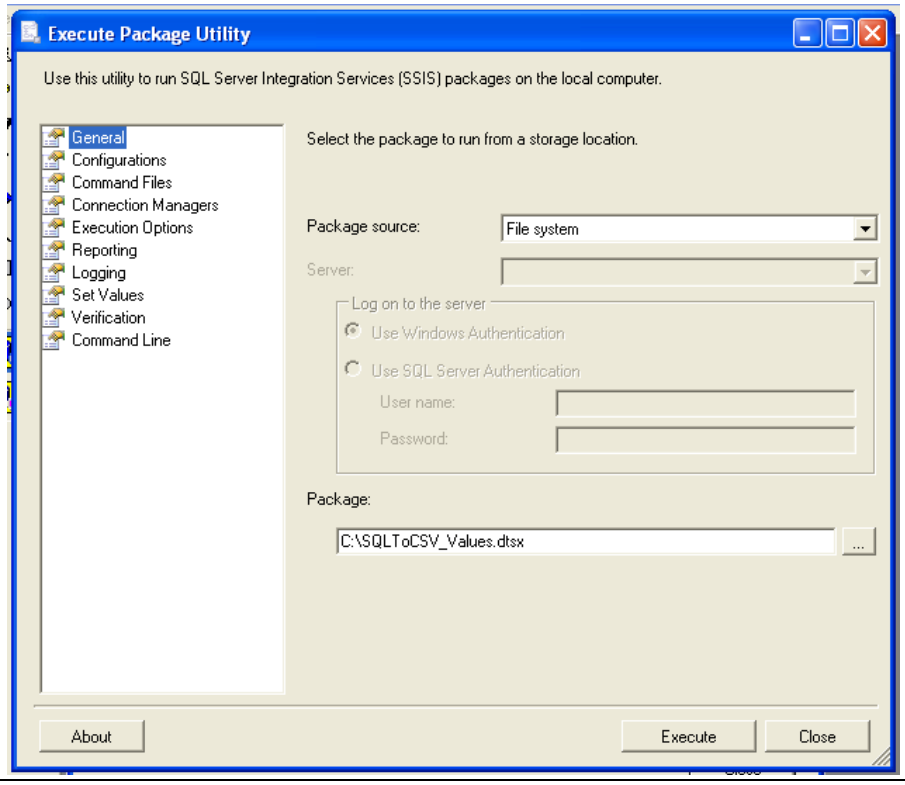
Note

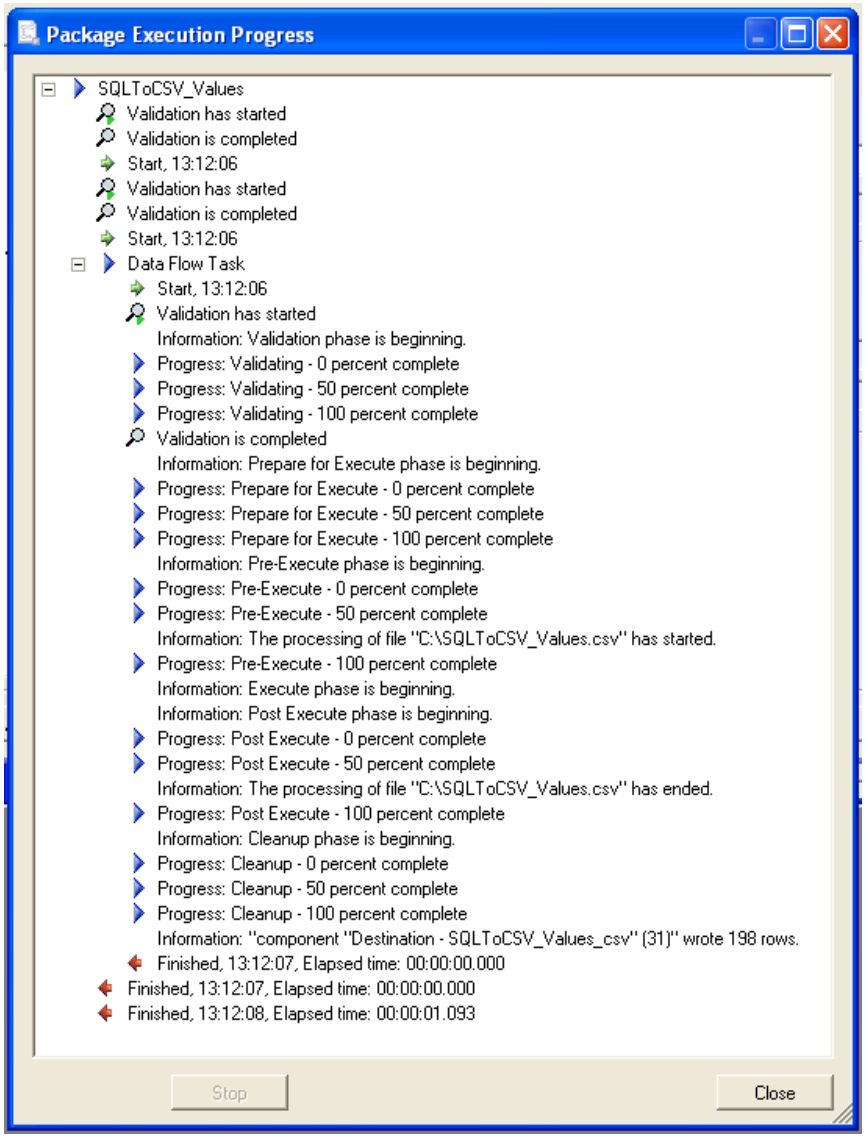
The ValueID of the archive tags can be seen in the “dbo.Archive” table which is accessible via the context menu:



Calling the DTSX package

Table 6-16

No	Action
1.	By executing (double-clicking) the created SSIS package (DTSX-file) and acknowledging the “Execute” you can repeat the export any time: 

No	Action
2.	The result is displayed as follows: 

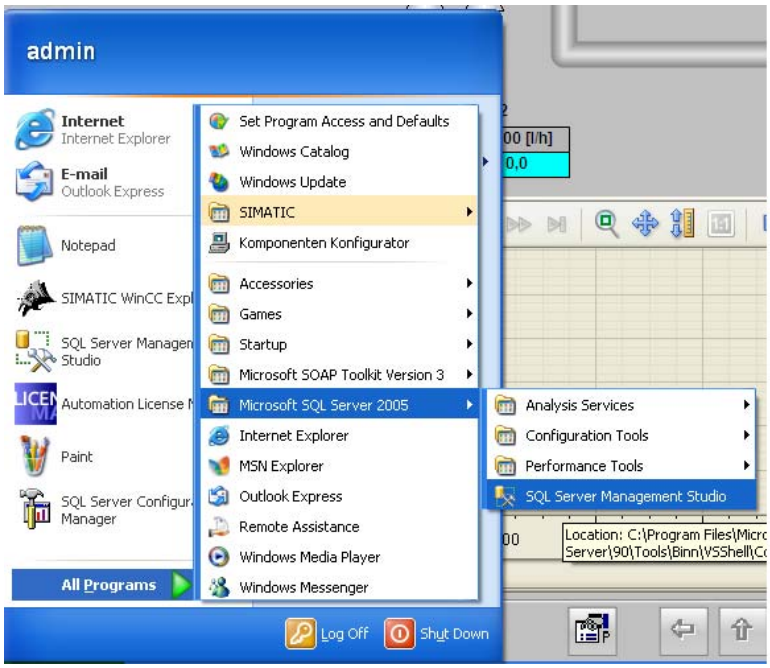
6.2.2 Export of archived messages into a CSV-file with SQL

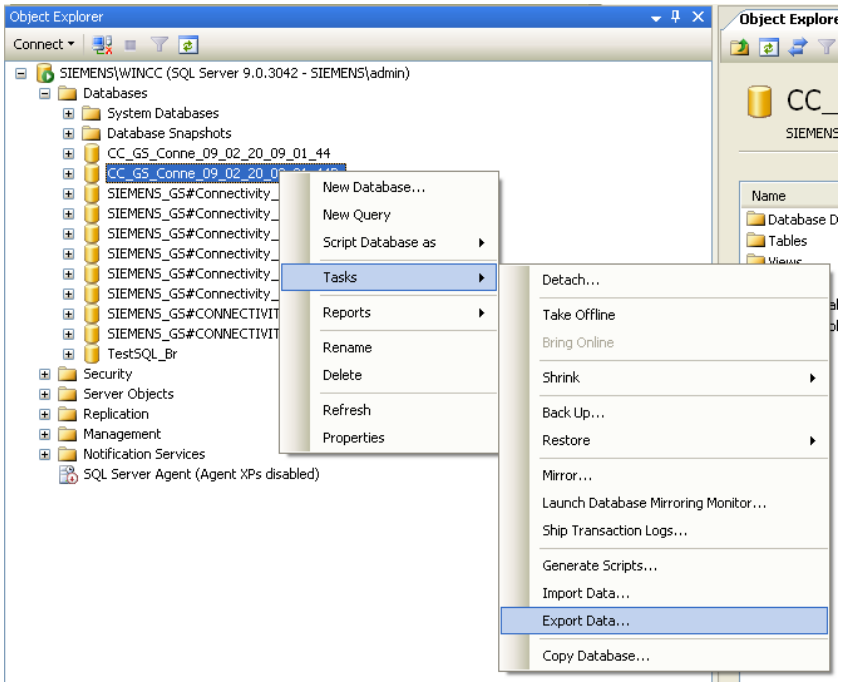
The procedure mainly corresponds the Export of archived measuring values described in chapter 4.2.1.

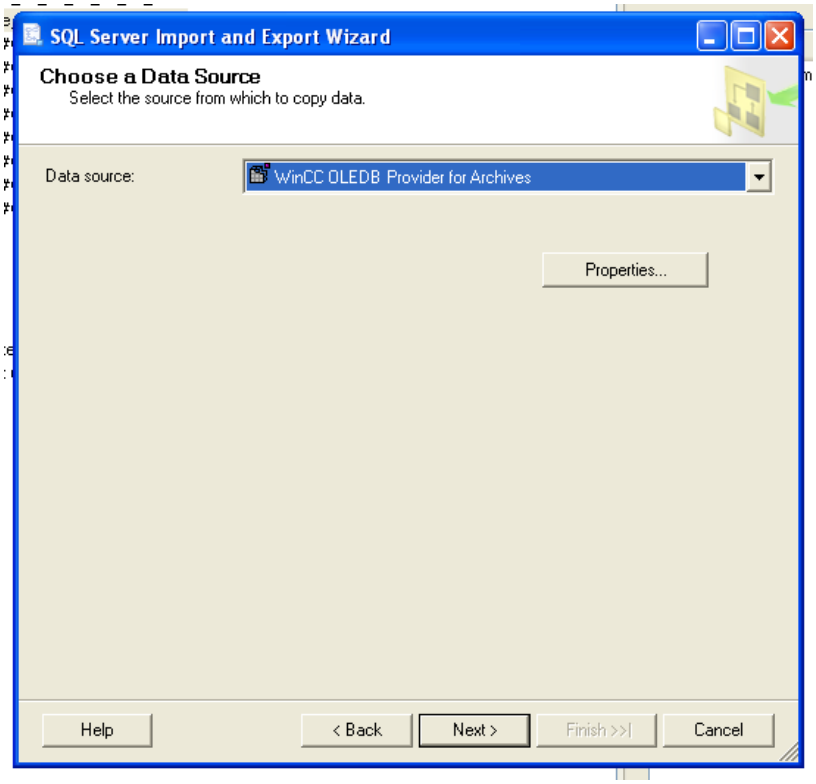
Differences only exist in the structure of the SQL query.

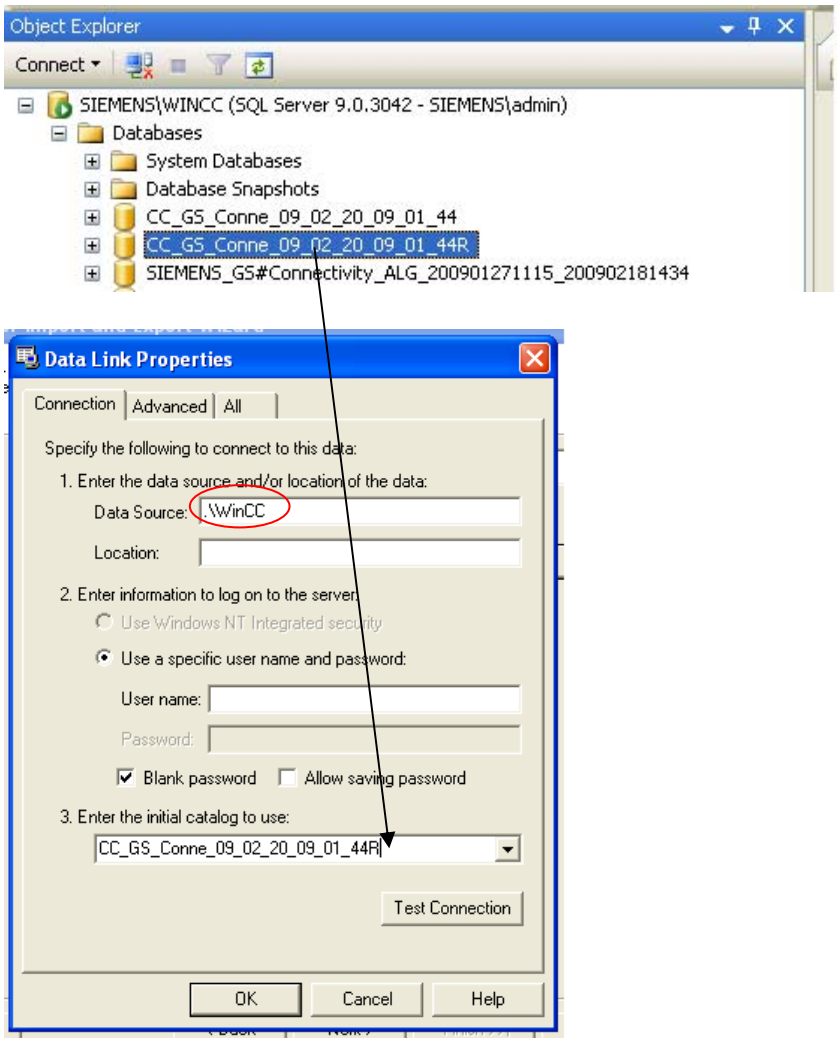
Table 6-17

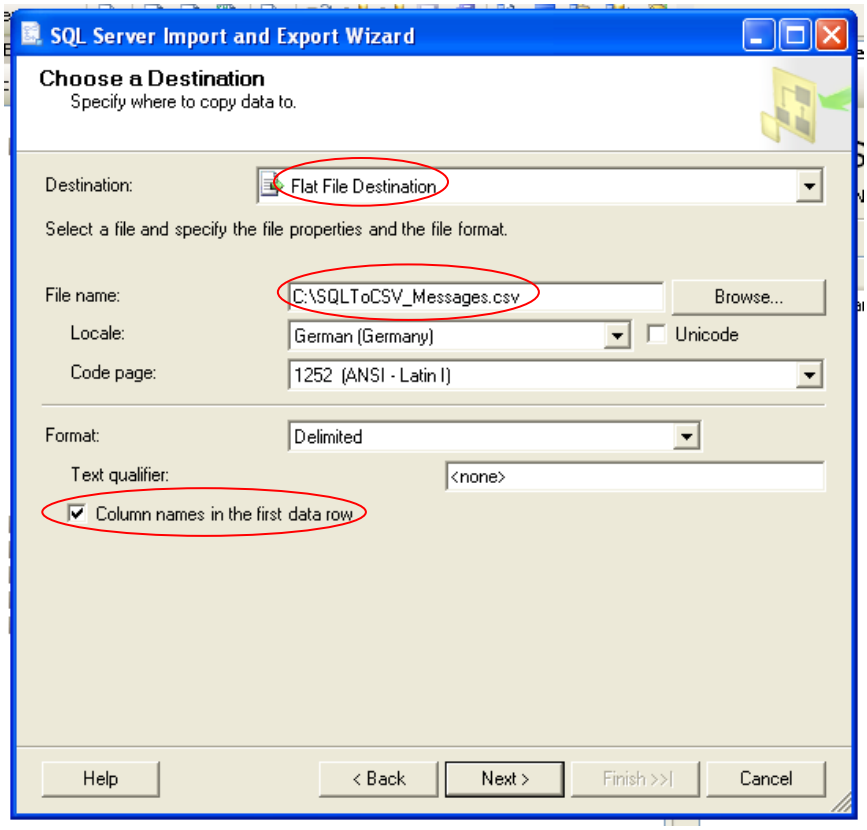
No	Action
1.	Start WinCC, open the WinCC project and activate Runtime. Here you go to the “Reverse-osmosis” picture and start the osmosis plant via the STARTUP button in the control window; wait until the plant has reached the “Production” status. After a short runtime you shut down the plant with the “SHUTDOWN” button.

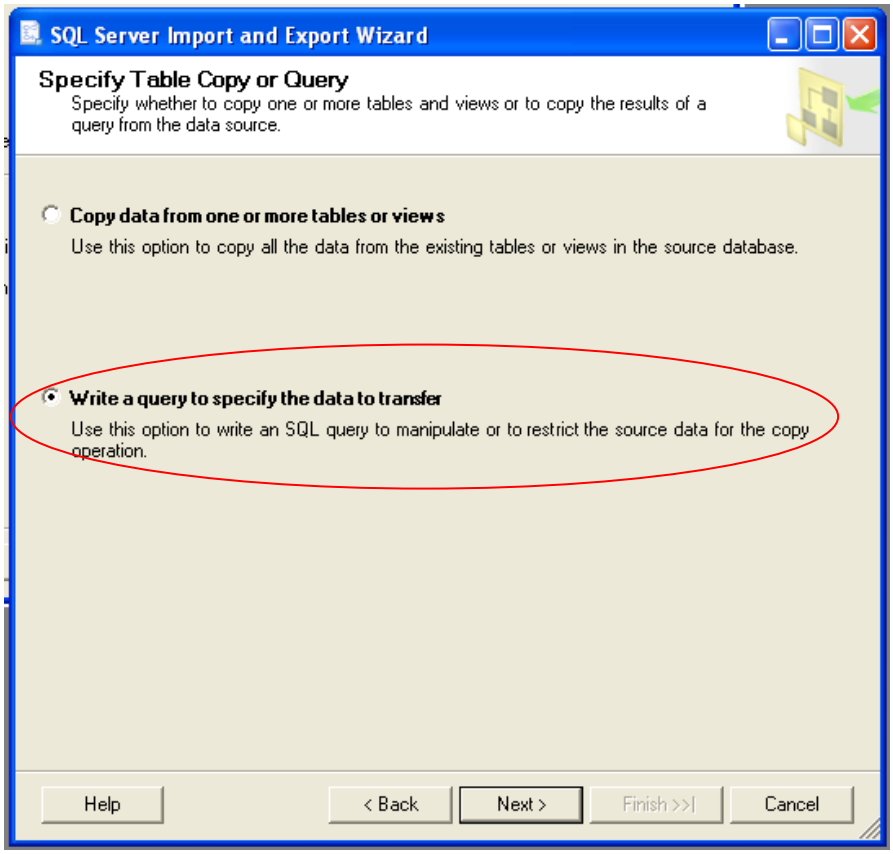
No	Action
2.	Start the SQL Server Management Studio: 
3.	Generate the connection with WinCC: 

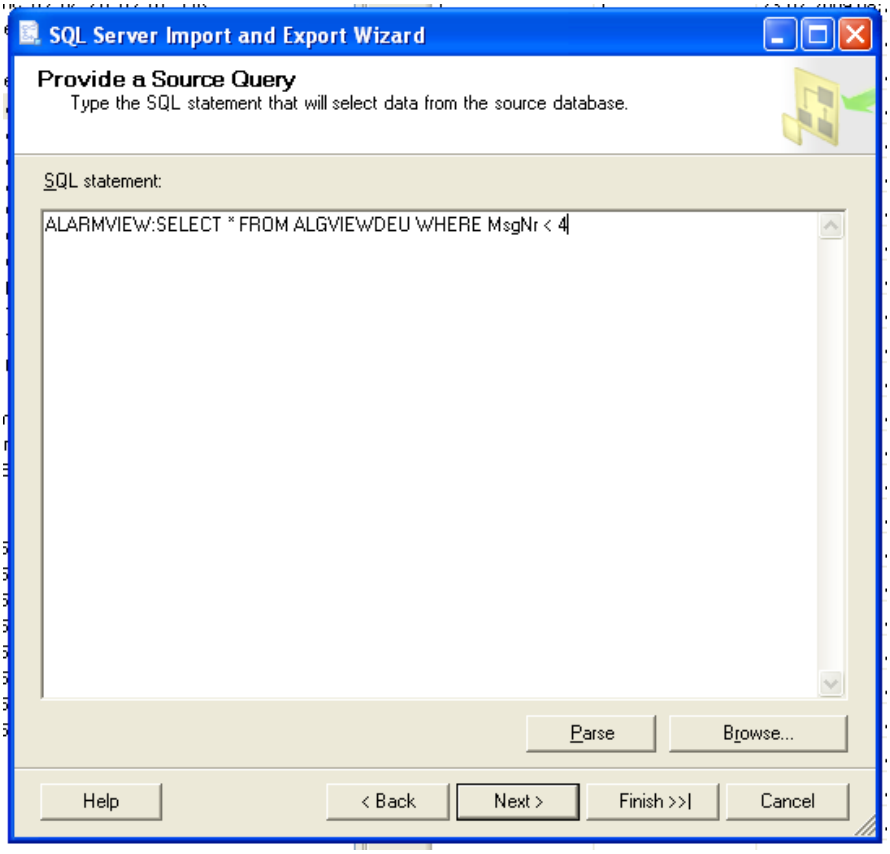
No	Action
4.	Start the SQL Server Import/Export Wizard via the context menu of the WinCC Runtime database (CC_GS_Conne_09_02_20_09_01_44R): 
5.	Exit the start screen with “Next” 

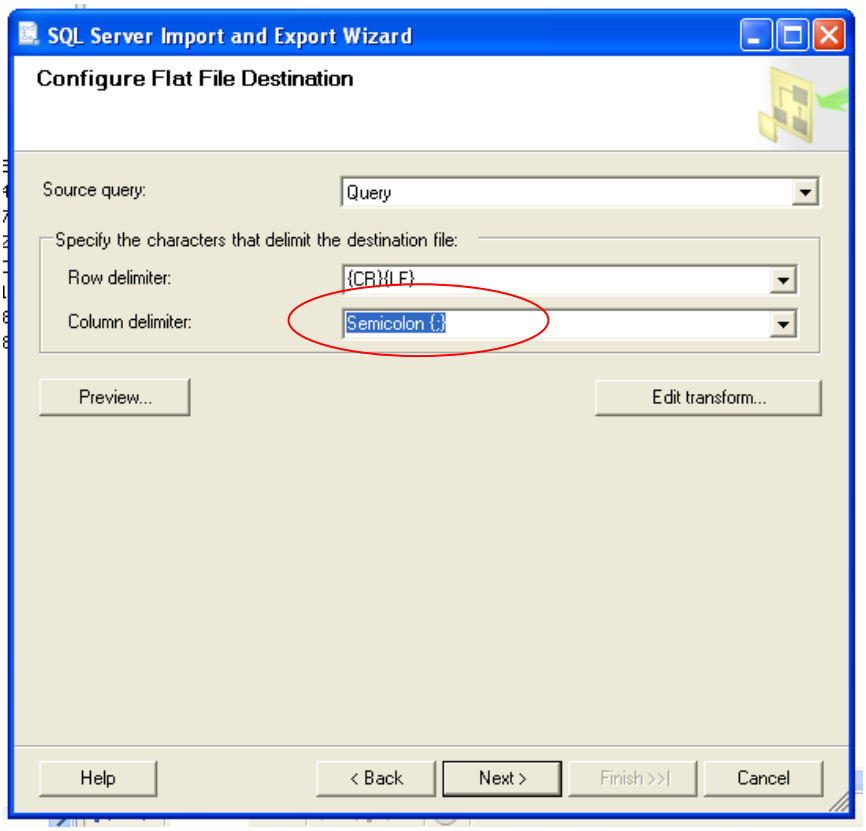
No	Action
6.	As data source you select the WinCC OLE DB Provider for Archives and then acknowledge the “Properties” button 

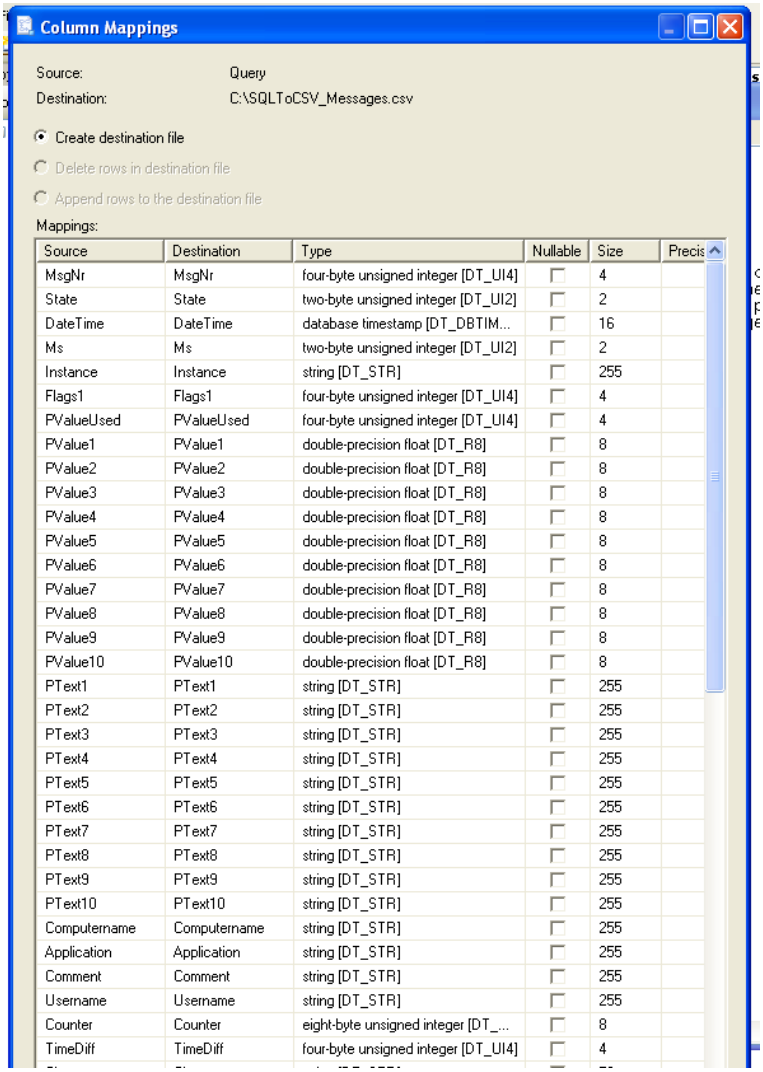
No	Action
7.	Here you set ".\WinCC" as data source and the name of the WinCC Runtime database as catalog. The name of the WinCC Runtime database is located in the Databases folder of the SQL Server Management Studios. Then quit the mask with the buttons "OK" and "Next".  Acknowledge the details via the OK button and then continue the Wizard via the "Next" button.

No	Action
8.	Select the data destination “Flat File Destination” and enter file location and name of the CSV-file. Activate the “Column names in first data row” checkbox so the column headers are written to the first line of the CSV-file.  Exit the mask by pressing the “Next” button.

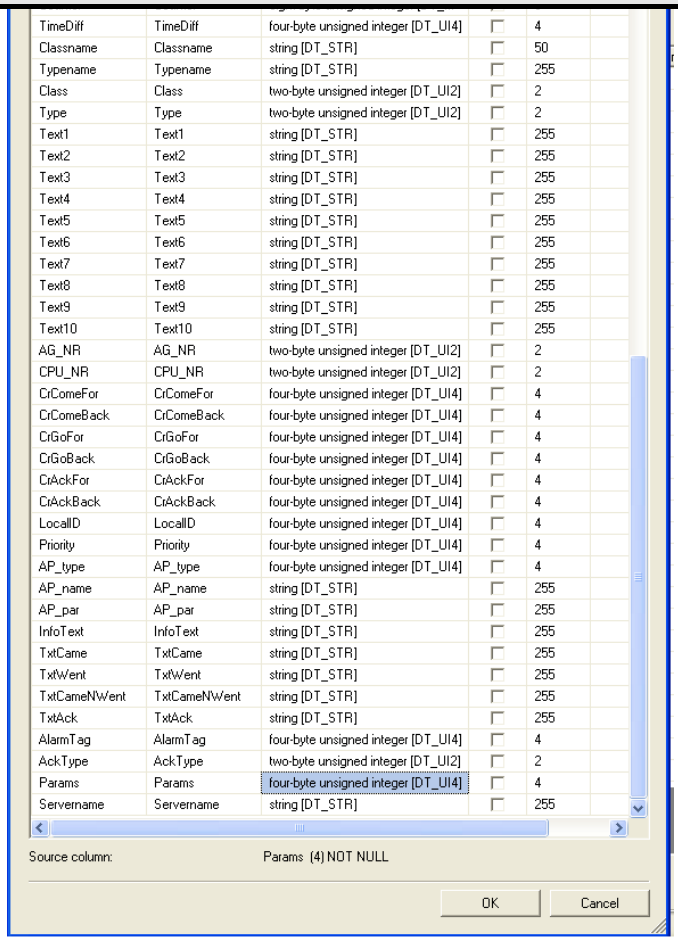
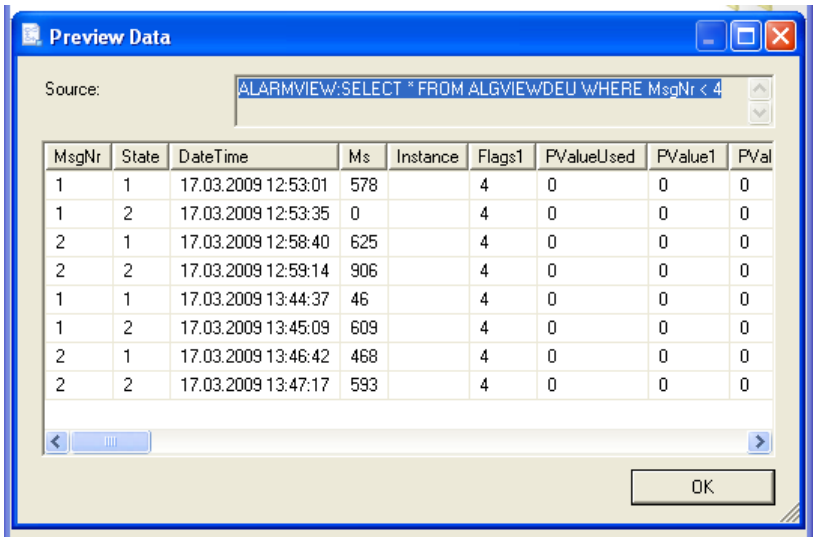
No	Action
9.	Activate the “Write a query to specify the data to transfer” checkbox to specify the SQL query in the next mask. Continue the Wizard with the “Next” button. 

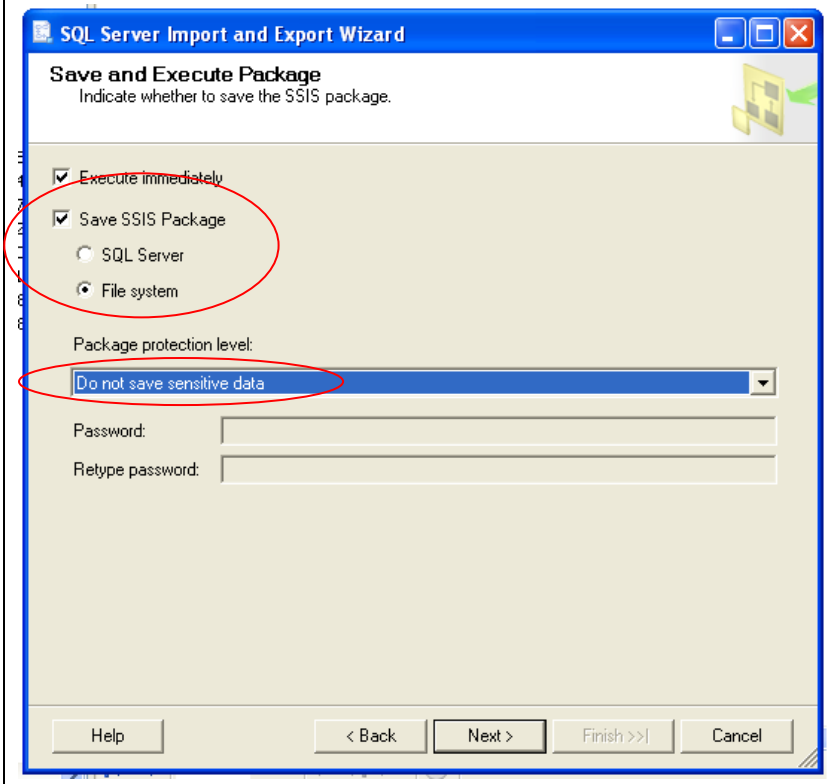
No	Action
10.	Here you enter the desired SQL query. In this example the messages with message number < 4 are queried. The query structure corresponds to the description in chapter 4.1.2.3 . Exit the mask by pressing the “Next” button. 

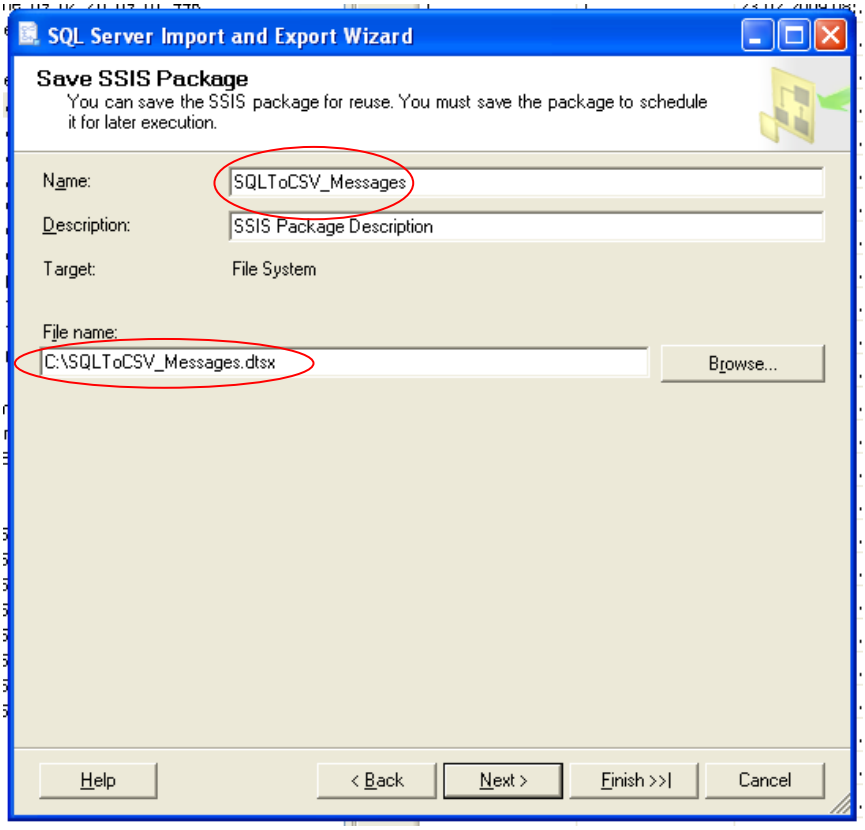
No	Action
11.	Enter semicolon as the “Column delimiter” and press the “Edit transform” button to specify the data formats. 

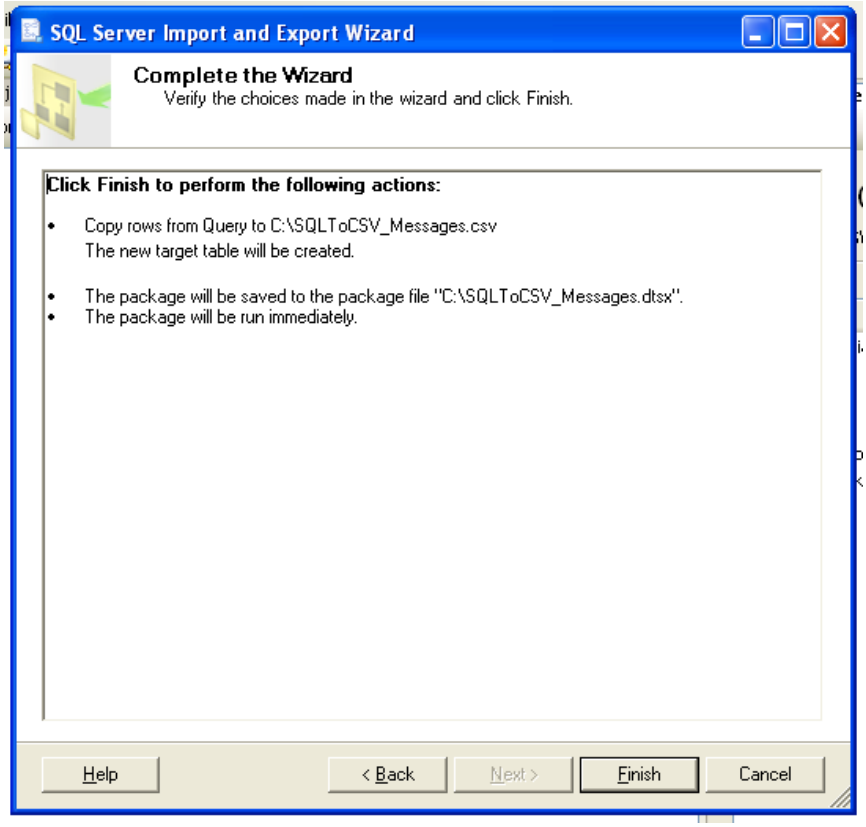
No	Action
12.	Set the data formats as follows: 

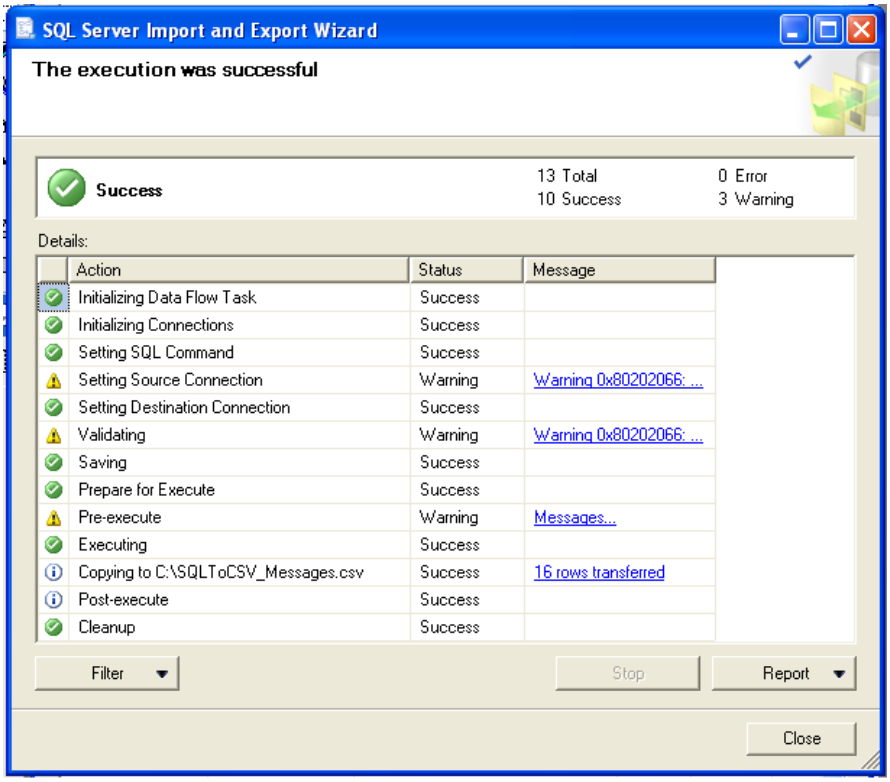
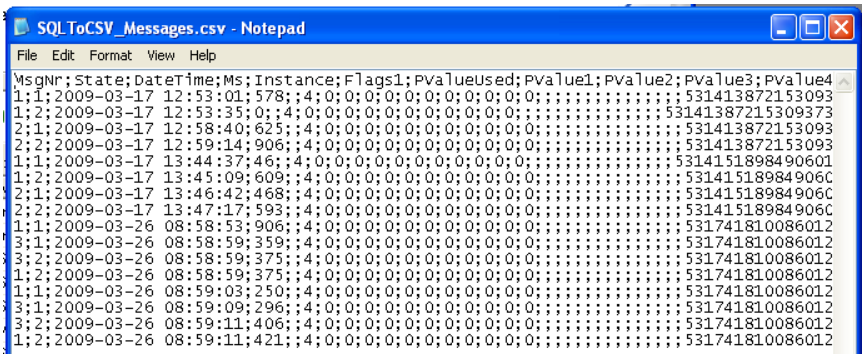
Source	Destination	Type	Nullable	Size	Precis
MsgNr	MsgNr	four-byte unsigned integer [DT_UI4]	☐	4	
State	State	two-byte unsigned integer [DT_UI2]	☐	2	
DateTime	DateTime	database timestamp [DT_DBTIM...	☐	16	
Ms	Ms	two-byte unsigned integer [DT_UI2]	☐	2	
Instance	Instance	string [DT_STR]	☐	255	
Flags1	Flags1	four-byte unsigned integer [DT_UI4]	☐	4	
PValueUsed	PValueUsed	four-byte unsigned integer [DT_UI4]	☐	4	
PValue1	PValue1	double-precision float [DT_R8]	☐	8	
PValue2	PValue2	double-precision float [DT_R8]	☐	8	
PValue3	PValue3	double-precision float [DT_R8]	☐	8	
PValue4	PValue4	double-precision float [DT_R8]	☐	8	
PValue5	PValue5	double-precision float [DT_R8]	☐	8	
PValue6	PValue6	double-precision float [DT_R8]	☐	8	
PValue7	PValue7	double-precision float [DT_R8]	☐	8	
PValue8	PValue8	double-precision float [DT_R8]	☐	8	
PValue9	PValue9	double-precision float [DT_R8]	☐	8	
PValue10	PValue10	double-precision float [DT_R8]	☐	8	
PText1	PText1	string [DT_STR]	☐	255	
PText2	PText2	string [DT_STR]	☐	255	
PText3	PText3	string [DT_STR]	☐	255	
PText4	PText4	string [DT_STR]	☐	255	
PText5	PText5	string [DT_STR]	☐	255	
PText6	PText6	string [DT_STR]	☐	255	
PText7	PText7	string [DT_STR]	☐	255	
PText8	PText8	string [DT_STR]	☐	255	
PText9	PText9	string [DT_STR]	☐	255	
PText10	PText10	string [DT_STR]	☐	255	
Computername	Computername	string [DT_STR]	☐	255	
Application	Application	string [DT_STR]	☐	255	
Comment	Comment	string [DT_STR]	☐	255	
Username	Username	string [DT_STR]	☐	255	
Counter	Counter	eight-byte unsigned integer [DT_...	☐	8	
TimeDiff	TimeDiff	four-byte unsigned integer [DT_UI4]	☐	4	

No	Action
	 Then quit the mask with “OK”. If the CSV-file to be created already exists in the destination path, “Create destination file” is not active and the data formats cannot be specified!
13.	The result of the query can be checked via “Preview”:  Then quit the mask with “OK” and “Next”.

No	Action
14.	The activate saving the SSIS package in the Windows file system without protection function and quit the mask with “Next”. 

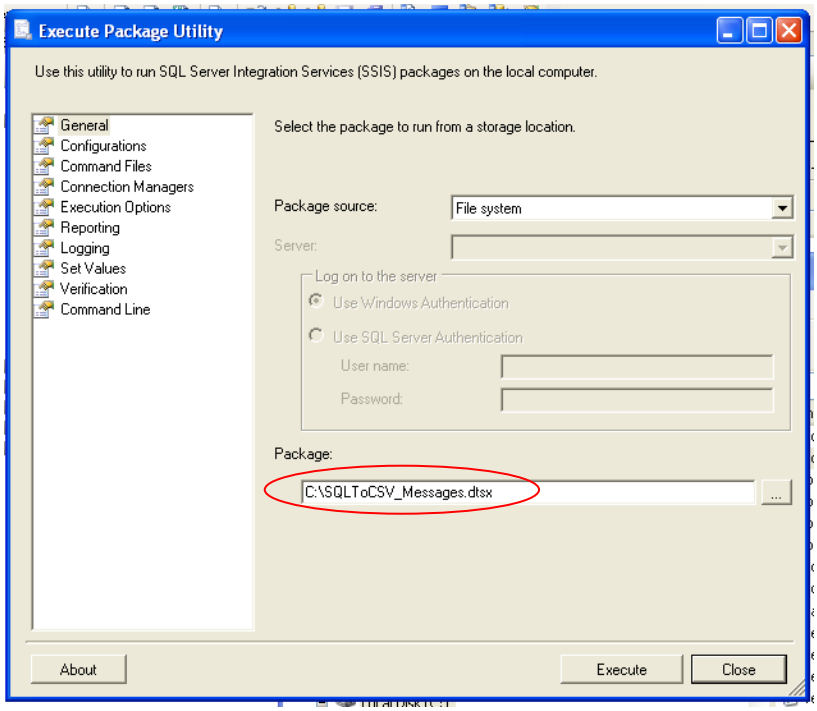
No	Action
15.	Enter the file name and storage location for the SSIS package and press the “Next” button. 

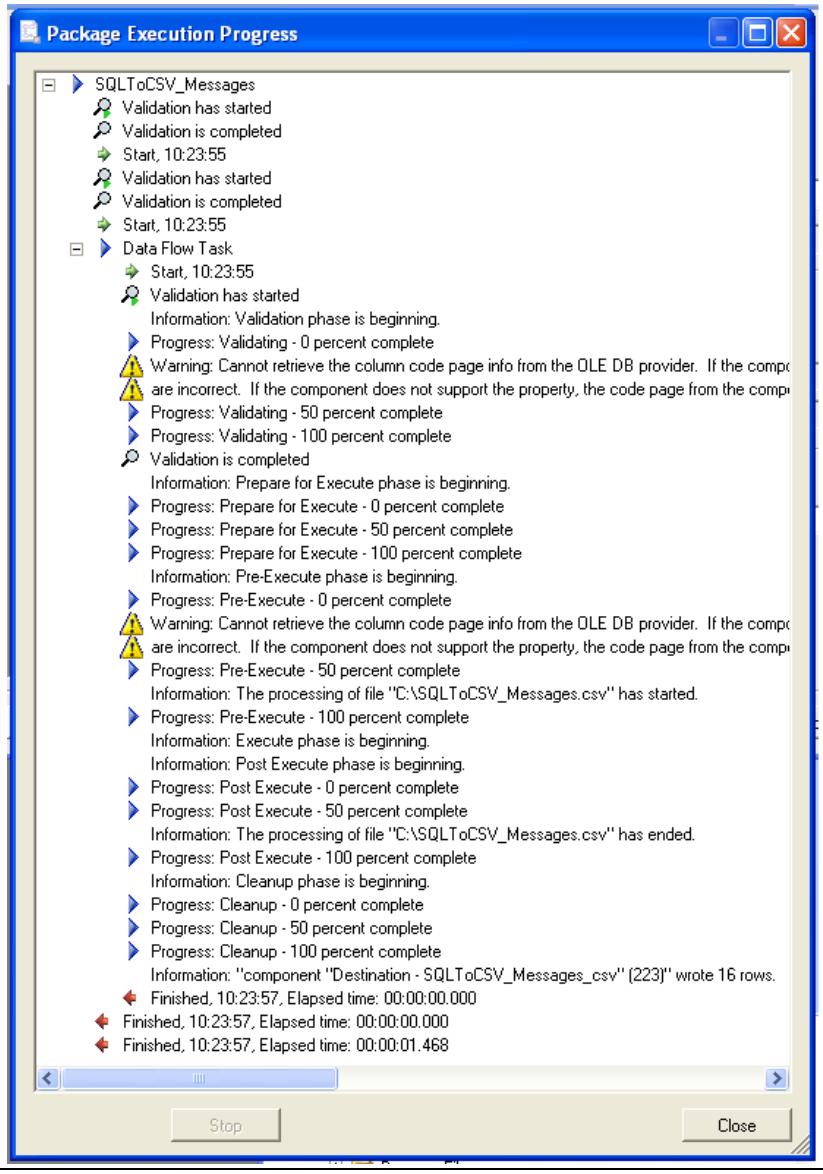
No	Action
16.	In the subsequent mask, the set parameters are summarized, press “Finish” to start the export. 

No	Action
17.	The result of the export is then logged, via “Reports” you have the option to save the log as text file.  Both warnings can be ignored here since they have no effect on the export functionality.
18.	Check the generated CSV-file: 

Calling the DTSX package

Table 6-18

No	Action
1.	By executing (double-clicking) the created SSIS package (DTSX-file) and acknowledging the “Execute” you can repeat the export any time: 

No	Action
2.	The result is displayed as follows: 

6.3 VBS Code

6.3.1 WriteArchiveValuesToCSV

```
Function WriteArchiveValuesToCSV (Archivename, MeasuringPoint)
'////////////////////////////////////////////////////////////
'1. Step: Creating the CSV-File
'////////////////////////////////////////////////////////////

'Declaration of local Tags
Dim fso          'FileSystemObject
Dim f            'File
Dim ts           'TextStream
Dim path         'Path
```

```
Dim StartArchive'Starttime of Archiving
Dim StopArchive      'Endtime of Archiving
Dim TimeStamp        'Timestamp for building the filename

'Read Start- and Stoptime of Osmosis:
Set StartArchive = HMIRuntime.Tags("DateTime_LastStart")
Set StopArchive = HMIRuntime.Tags("DateTime_LastStop")

If StartArchive.read = " " or StopArchive.Read = " " Then
        MsgBox "Start- or Stoptime is missing !"
        Exit Function

End If

'Generate String for the CSV-Filename and replace ":" with "_"
TimeStamp = FormatDateTime(StartArchive.Read,vbGeneralDate)
TimeStamp = Replace(TimeStamp,":","_")

'Path and name for the CSV-File:
path= "C:\Osmosis " & TimeStamp & " " & MeasuringPoint & ".csv"

'Create Filesystemobject and CSV-File if not exists:
Set fso = CreateObject("Scripting.FilesystemObject")
If Not fso.FileExists(path) Then

   fso.CreateTextFile(path)

Else

  MsgBox "File already exits !"
  Exit Function

End If

'Create File-Object and open this file for writing
Set f = fso.GetFile(path)
Set ts = f.OpenAsTextStream(2,-2)

'CSV-File is ready now for wrtiting !
'////////////////////////////////////////////////////

'------------------------------------------------------------

'////////////////////////////////////////////////////
'2. Connecting WinCC-Database
'////////////////////////////////////////////////////

Dim Pro              'Provider
Dim DSN                    'Data Source Name
Dim DS                     'Data Source
Dim ConnString       'Connection String
Dim MachineNameRT          'Name of the PC from WinCC-RT
Dim DSNRT            'Data Source Nane from WinCC-RT

Dim Conn             'Connection to ADODB
Dim RecSet           'RecordSet
Dim Command                'Query
Dim CommandText   'Command-Text
Dim CommandTextStart 'Starttime for SQL-String

Dim Duration    'Duration of Production-Cycle
Dim DurationSec 'Duration of Production-Cycle
Dim DurationMin 'Duration of Production-Cycle
Dim DurationHour'Duration of Production-Cycle
Dim DurationDay 'Duration of Production-Cycle
Dim CurrLanguage'Current Language

'Attention: Tag-Archiving is based on UTC, that means the timestamp
'of a Tag is in UTC !

'Read the name of the PC-Station and the DSN-Name from WinCC-RT
Set MachineNameRT = HMIRuntime.Tags("@LocalMachineName")
Set DSNRT = HMIRuntime.Tags("@DatasourceNameRT")

'Preparing the Connection-String
Pro="Provider=WinCCOLEDBProvider.1;" 'First instance of WinCCOLEDB
```

```
DSN="Catalog=" & DSNRT.Read & ";" 'Name of Runtime-Database
DS= "Data Source=" & MachineNameRT.Read & "\WinCC" 'Data Source

'Build the complete String:
ConnString = Pro + DSN + DS

'Make Connection
Set Conn = CreateObject("ADODB.Connection")
Conn.ConnectionString = ConnString
Conn.CursorLocation = 3
Conn.open

'Preparing query
'Format of Command.CommandText
'"Tag:R,1,'2009-01-20 11:15:23.000','"2009-01-20 13:26:45.000"'
'  | | |          |                                  |
'  | | |          ---- Starttime (UTC)           ------ Endtime (UTC)
'  | | ----- Value-ID or Tagname
'  | ------- Read
'  ---------- Command for a Tag
'

'Duration between Start an Stop in seconds:
Duration = DateDiff ("s",StartArchive.Read,StopArchive.Read)

'Split the Duration in days, hours, minutes and seconds:
DurationMin=Fix(Duration/60)
DurationSec= Duration - (DurationMin * 60)
DurationHour= Fix (DurationMin/60)
DurationMin= DurationMin -(DurationHour * 60)
DurationDay= Fix(DurationHour / 24)
DurationHour= DurationHour - (DurationDay * 24)

'Creating leading zeros:
DurationSec= Right("00" & DurationSec,2)
DurationMin= Right("00" & DurationMin,2)
DurationHour= Right("00" & DurationHour,2)
DurationDay= Right("00" & DurationDay,2)

'MsgBox "Tage: " & DurationDay & vbcrlf & "Stunden: "  & DurationHour & vbcrlf & "Minuten:
" & DurationMin & vbcrlf & _
'"Sekunden: " & DurationSec 'MBox for Diagnosis only

'Formating Starttime:
CommandTextStart="'0000-00-" & DurationDay & " " & DurationHour & ":" & _
DurationMin & ":" & DurationSec & ".000'"

'Building the complete String:
CommandText="Tag:R,'" & Archivename & "\" & MeasuringPoint & "'," & _
CommandTextStart & ",'0000-00-00 00:00:00.000'"

'MsgBox "Open Connection with:" & vbCr & ConnString & vbCr & CommandText 'MBox for
Diagnosis only

'Create the redordset, read the records an set to first redcordset:
Set RecSet = CreateObject("ADODB.Recordset")
Set Command = CreateObject("ADODB.Command")
Command.CommandType = 1
Set Command.ActiveConnection = Conn
Command.CommandText=CommandText

Set RecSet = Command.Execute
RecSet.MoveFirst

'write recordsets to CSV-File

Set CurrLanguage = HMIRuntime.Tags("@CurrentLanguage")
Select Case CurrLanguage.Read
Case 1031 'German
    ts.WriteLine ("Tag-Name;ValueID;Datum/Zeit;Pozesswert") 'header
Case 1033 'English
    ts.WriteLine ("Tag-Name;ValueID;Date/Time;Process-Value") 'header
```

```
end select

'writing recordsets
Do while Not RecSet.EOF
  ts.WriteLine (MeasuringPoint & ";" & RecSet.Fields(0).Value & ";" & _
  RecSet.Fields(1).Value & ";" & RecSet.Fields(2).Value)
  RecSet.MoveNext

Loop


' Please don't forget this..... !!!
ts.close
RecSet.Close
Set Recset=Nothing
Set Command = Nothing
conn.close                    'Close connection
Set Conn = Nothing
Set fso = Nothing
Set f = Nothing
Set ts = Nothing

End Function
```

6.3.2 WriteArchiveMessagesToCSV


```
Sub WriteArchiveMessagesToCSV

'////////////////////////////////////////////////////////////
'1. Step: Creating the CSV-File
'////////////////////////////////////////////////////////////

'Declaration of local Tags
Dim fso             'FileSystemObject
Dim f               'File
Dim ts              'TextStream
Dim path            'Path
Dim StartTime       'Date and time when writing is triggered
Dim TimeStamp       'Timestamp for building the filename

StartTime = Now 'Date and Time when writing messages is triggerd

'Generate String for the CSV-Filename:
TimeStamp = FormatDateTime(StartTime,vbGeneralDate)
TimeStamp = Replace(TimeStamp,":","_") 'Replace ":" with "_"

'Path and name for the CSV-File:
path= "C:\Osmosis " & TimeStamp & " Messages.csv"

'Create Filesystemobject and CSV-File if not exists:
Set fso = CreateObject("Scripting.FilesystemObject")
If Not fso.FileExists(path) Then
    fso.CreateTextFile(path)
Else
   MsgBox "File already exits !"
   Exit Sub
End If

'Create File-Object and open this file for writing
Set f = fso.GetFile(path)
Set ts = f.OpenAsTextStream(2,-2)


'CSV-File is ready now for wrtiting !
'////////////////////////////////////////////////////

'------------------------------------------------------------

'////////////////////////////////////////////////////
'2. Connecting WinCC-Database
```

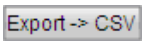
```
'////////////////////////////////////////////////
Dim Pro                 'Provider
Dim DSN                         'Data Source Name
Dim DS                          'Data Source
Dim ConnString      'Connection String
Dim MachineNameRT           'Name of the PC from WinCC-RT
Dim DSNRT           'Data Source Nane from WinCC-RT

Dim Conn                'Connection to ADODB
Dim RecSet              'RecordSet
Dim Command                     'Query
Dim CommandText     'Command-Text

Dim sqlSec      'Seconds for SQL
Dim sqlMin      'Minutes for SQL
Dim sqlHour     'Hours for SQL
Dim sqlDay      'Day for SQL
Dim sqlMonth    'Month for SQL
Dim sqlYear     'Year for SQL

Dim CurrLanguage'Current Language

'Attention: Tag-Archiving is based on UTC, that means the timestamp of a Tag is in UTC !

'Read the name of the PC-Station and the DSN-Name from WinCC-RT
Set MachineNameRT = HMIRuntime.Tags("@LocalMachineName")
Set DSNRT = HMIRuntime.Tags("@DatasourceNameRT")

'Preparing the Connection-String
Pro="Provider=WinCCOLEDBProvider.1;" 'First instance of WinCCOLEDB
DSN="Catalog=" & DSNRT.Read & ";" 'Name of Runtime-Database
DS= "Data Source=" & MachineNameRT.Read & "\WinCC" 'Data Source

'Build the complete String:
ConnString = Pro + DSN + DS

'Make Connection
Set Conn = CreateObject("ADODB.Connection")
Conn.ConnectionString = ConnString
Conn.CursorLocation = 3
Conn.open

'Preparing query
'Format needed for StartTime: '2009-01-20 13:26:45'"

'Date and time 24 hours before
StartTime=DateAdd("h",-24,StartTime)

'Split in Years, months, days, hours, min.,seconds
sqlSec=second (StartTime)
sqlMin=Minute (StartTime)
sqlHour=Hour (StartTime)
sqlDay=Day (StartTime)
sqlMonth=Month (StartTime)
sqlYear=Year (StartTime)

'Creating leading zeros
sqlSec=Right("00" & sqlSec,2)
sqlMin=Right("00" & sqlMin,2)
sqlHour=Right("00" & sqlHour,2)
sqlDay=Right("00" & sqlDay,2)
sqlMonth=Right("00" & sqlMonth,2)

'Formating Starttime fpr SQL-Statement:
StartTime="'" & sqlYear & "-" & sqlMonth & "-" & sqlDay & " " _
& sqlHour & ":" & sqlMin & ":" & sqlSec & "'"

'MsgBox "Jahr: " & sqlYear & vbcrlf & "Monat: " & sqlMonth & vbcrlf & "Tage: " & sqlDay &
vbcrlf & _
"'Stunden: "  & sqlHour & vbcrlf & "Minuten: " & sqlMin & vbcrlf & _
"'Sekunden: " & sqlSec 'MBox for Diagnosis only
```

```
'Building the complete String:

Set CurrLanguage = HMIRuntime.Tags("@CurrentLanguage")
Select Case CurrLanguage.Read
Case 1031 'German
    CommandText= "ALARMVIEW:Select * FROM AlgViewDeu WHERE DateTime>" & _
    StartTime & "AND MsgNr < 4 AND State = 1"
Case 1033 'English
    CommandText= "ALARMVIEW:Select * FROM AlgViewEnu WHERE DateTime>" & _
    StartTime & "AND MsgNr < 4 AND State = 1"
End Select

'MsgBox "Open Connection with: " & CommandText 'MBox for Diagnosis only

'Create the recordset, read the records and set to first recordset:
Set RecSet = CreateObject("ADODB.Recordset")
Set Command = CreateObject("ADODB.Command")
Command.CommandType = 1
Set Command.ActiveConnection = Conn
Command.CommandText=CommandText

Set RecSet = Command.Execute
RecSet.MoveFirst

'write recordsets to CSV-File

Select Case CurrLanguage.Read
Case 1031 'German
    ts.WriteLine ("Datum/Zeit;Meldenr.;Ereignis;Klasse")
Case 1033 'Englisch
    ts.WriteLine ("Date/Time;MsgNr.;Event;Class")
End Select

Do while Not RecSet.EOF
  ts.WriteLine (RecSet.Fields(2).Value & ";" & RecSet.Fields(0).Value & ";" & _
  RecSet.Fields(39).Value & _
  ";" & RecSet.Fields(33).Value)
  RecSet.MoveNext

Loop


' Please don't forget this..... !!!
ts.close
RecSet.Close
Set Recset=Nothing
Set Command = Nothing
conn.close                    'Close connection
Set Conn = Nothing
Set fso = Nothing
Set f = Nothing
Set ts = Nothing


End Sub
```

7 Operation of the Application

7.1 Scenario: “Export of archived tags into a CSV-file with VB Script”

Table 7-1

No	Action	Comment
1	Start the Runtime.	
2	Go to the “Reverse-osmosis” screen.	
3	Start the osmosis plant via the STARTUP button in the control window	
4	Wait until the plant has reached the “Production” status.	
5	After a short runtime you shut down the plant with the “SHUTDOWN” button.	
6	After the plant has reached the “Off” state and the stop time was entered automatically in “Stop”, you press the “Export -> CSV” button.	
7	Then look at the CSV-file created on drive C:\.	

7.2 Scenario: “Export of archived messages into a CSV-file with VB Script”

Table 7-2

No	Action	Comment
1	Start the Runtime.	
2	Go to the “Reverse-osmosis” screen.	
3	Start the osmosis plant via the STARTUP button in the control window	
4	Wait until the plant has reached the “Production” status.	
5	After a short runtime you shut down the plant with the “SHUTDOWN” button.	
6	After the plant has reached the “Off” state and the stop time was entered automatically in “Stop”, you press the “Export Statistic -> CSV” button.	
7	Then look at the CSV-file created on drive C:\.	

8 Bibliography

8.1 Bibliographic References

This list is not intended to be exhaustive and only represents a selection of the relevant literature.

Table 8-1

No	Subject	Title	Author / Publisher / ISBN
1.	Visual Basic	Visual Basic 6 Programmiertechniken, Datenbanken, Internet [Programming Techniques, Databases, Internet]	Autor: Michael Kofler Verlag: Adison-Wesley ISBN: 3-8273-1428-3
2.	Connectivity Pack	WinCC V7.0 WinCC/Connectivity Pack Documentation Printout of the Online help	
3.	Connectivity Pack	WinCC V7.0 Online Help Chapter “Interfaces” -> “WinCC Connectivity Pack”	

9 History

Table 9-1 History

Version	Date	Modifications
V1.0	31.08.2009	First issue