Access to WinCC Data with independent Windows Application

"WinCC_CopackCsharp"

Application Description • October 2009

Applications & Tools

Answers for industry.

SIEMENS

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/35840700

If you have any questions regarding this document, please send us an e-mail to the following address:

online-support.automation@siemens.com

SIEMENS

SIMATIC
WinCC_CopackCsharp

Access to WinCC Data with independent Windows application

Warranty and Liability

Note

The application examples are not binding and do not claim to be complete regarding configuration, equipment 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 this application example and other Siemens publications – e.g. Catalogs – the contents of the other documents have priority.

We accept no liability for 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 in the burden of proof to your detriment.

It is not permissible to transfer or copy these Application Examples or excerpts thereof without express authorization from Siemens Industry Sector.

Table of Contents

1 Automation Task

Introduction

As of WinCC V6.0 the WinCC Runtime database is segmented, i.e. data are filed in several archive segments (several databases). The data is partially filed in compressed binary form. The WinCC option “**WinCC Connectivity Pack**” provides the **WinCC-OleDBProvider**, which enables reading the Runtime data of Tag Logging and Alarm Logging directly. The WinCC OLE DB Provider provides the data from the respective archive segments in decompressed, decrypted form. The user of the WinCC Connectivity Pack does not have to worry about the segmentation of the archives and their encryption when accessing the Tag Logging and Alarm Logging data.

Description of the automation task

With a separate C-Sharp Windows application and using the WinCC Connectivity Pack the archived WinCC Runtime data of the tag logging (process data archiving), of the alarm logging (archived messages and alarms) and the user archive can be accessed.

It shall be described additionally how the Runtime data of the tag logging, the alarm logging and the user archives can be read, displayed and output via Crystal reports or into a CSV-file.

Particular attention is not given here to creating and writing a C# Windows application, bit to the required mechanisms for accessing the WinCC archive data.

2 Automation Solution

2.1 Description

Structure of the database connection

- Preparing the data (adjusting the time format; local time and universal time)
- Using the MS OleDB interface for reading the WinCC archive configuration and the WinCC user archive
- Using the WinCC OleDBProviders for reading the archived process values (WinCC Tag Logging) and alarms and messages (WinCC Alarm Logging).
- Table display of data with the “DataGrid” control element
- Output of the file into a csv-file
- Output of the data via Crystal reports

This entry contains a complete Visual C# example program which illustrates the above access mechanisms on a runnable Windows application.

Note This example only uses read access to the data. When using the MS OleDB interface the respective SQL commands (e.g. update, insert, delete ...) technically also enable write access to the data.

ATTENTION **Write accesses are only tested and enabled for data in the user archive.**

Note Entry http://support.automation.siemens.com/WW/view/en/22578952 gives an overview of further options to access the WinCC archives.

2.2 Hardware and software components used

For this example two separate computers (WinCC-Server and Connectivity Pack Client) were used. The WinCC server performs archiving in the WinCC Runtime database. The Connectivity Pack Client reads the data of the WinCC Runtime database. The following configurations were used for the systems:

WinCC-Server:

Table 2-1

Hardware	Software
Intel Pentium 4 CPU 2,4 GHz, 2GB RAM	MS Windows XP Professional SP2
	SIMATIC WinCC V6.2 (contains SQL Server 2005 SP1) **or** SIMATIC WinCC V7.0

Connectivity Pack Client:

Table 2-2

Hardware	Software
Intel Pentium 4 CPU 2,4 GHz, 1GB RAM	MS Windows XP Professional SP2 **Note**: The Windows component “Microsoft Message Queuing” must have been installed. In “Control Panel > Software > Add/Remove Windows components > Message Queuing” these components can be installed.
	When using the WinCC-OleDBProvider: • WinCC/ConnectivityPack V6.2 (Client) **or** • SIMATIC WinCC/ConnectivityPack V7.0 (Client)
	Optional: MS Visual Studio 2005 Professional with Visual C#

Example files and projects

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

Table 2-3

Component	Note
WinCCcopack.zip	Contains the C# project created with Visual Studio 2005

3 Installation

Download and unzip

Download the example program available as download and unzip the received zip-archive. Folder “WinCCcopack” is created. Subfolder “WinCCcopack > appCopack” contains the C#-project created with Visual Studio 2005.

Depending on whether the development environment MS Visual Studio has been installed on your computer, you can use the example program as follows:

Development environment Visual Studio has been installed

If Visual Studio has been installed on your computer, you can open the project by double-clicking on the **“appCopack.sln”** file. After the project has been opened in Visual Studio, you can edit the sources, compile the program and execute it with the menu command “Debug” > “Start without debugging”.

Development environment Visual Studio has not been installed

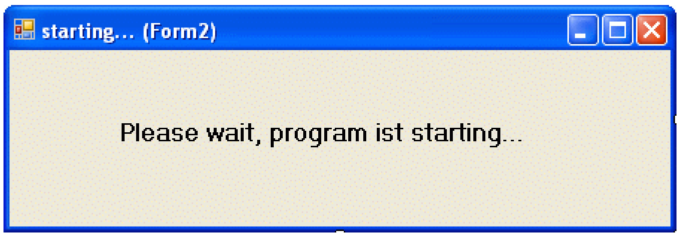
If Visual Studio has not been installed on your computer, you can execute the program by double-clicking on the **“.WinCCcopack > appCopack\obj\Debug\appCopack.exe”** file.

4 Operating the Application

4.1 Detailed description of the screen objects

After the Windows application has been started, the following program window “starting… (Form2)”. This window is started in a separate thread and terminated as soon as the program has been downloaded.

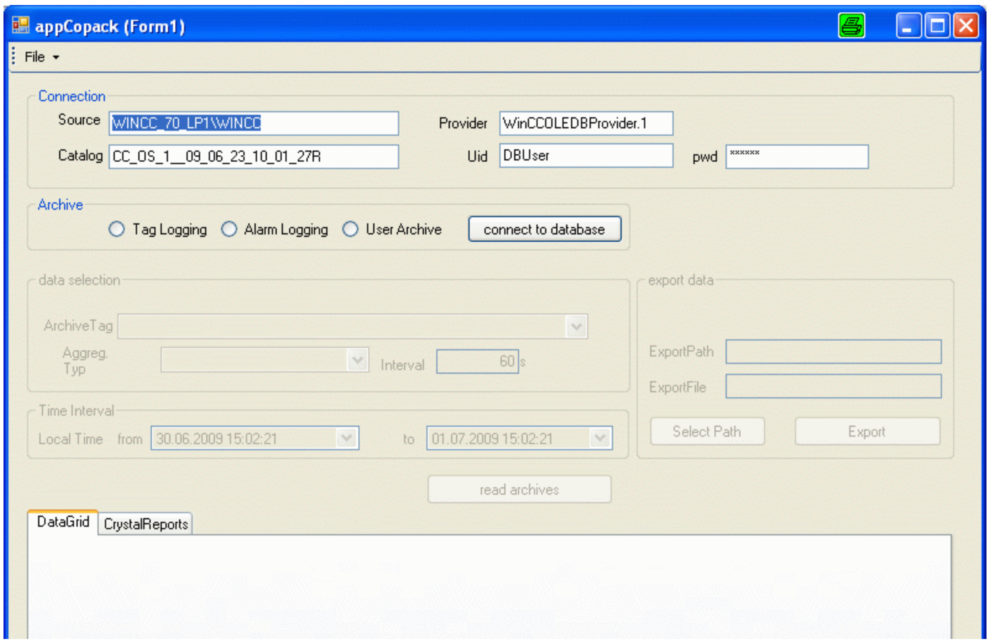
Figure 4-1


starting… (Form2)

Please wait, program ist starting...

As soon as the program has been downloaded, the program window “appCopack (Form1)” appears.

Figure 4-2


appCopack (Form1)

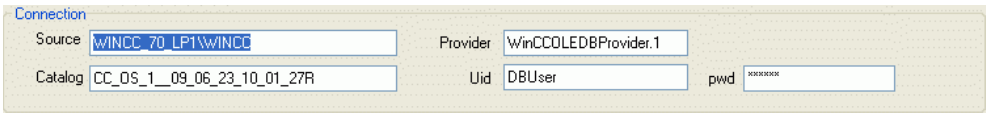
Source	WINCC_70_LP1\WINCC
Catalog	CC_OS_1__09_06_23_10_01_27R
Provider	WinCCOLEDBProvider.1
Uid	DBUser
pwd	******
Interval	60 s
Local Time from	30.06.2009 15:02:21
to	01.07.2009 15:02:21

The program consists of a program window which is described below.

4.1.1 “Connection” group

Type: GroupBox
Name: grpConnection

Figure 4-3



Source: WINCC_70_LP1\WINCC
Catalog: CC_OS_1__09_06_23_10_01_27R
Provider: WinCCOLEDBProvider.1
Uid: DBUser
pwd: ******

The input fields of this group are used to configure the connection with the data source. When starting the program the fields are preassigned. The user can change the connection parameter during runtime and execute the data query.

Note

After pressing the “connect to database” button this group can no longer be operated.
As soon as you change the selection in the “archive” group, the “Connection” group can be operated again.

Table 4-1

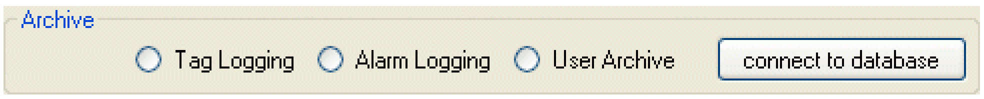
Screen objects (Object name)	Description
Input field “**Source**” Type: TextBox Name: txtSource	This input field contains the name of the WinCC server followed by the name of the instance of the SQL server whose runtime data is to be accessed. <ComputerName>\WINCC This field is preassigned with “WINCC_70_LP\WINCC” at the start of the program.
Input field “**Catalog**” Type: TextBox Name: lblCatalog	In this input field the user must assign the Data Source Name (DSN) of the Runtime database whose data shall be accessed. **Notes:** • For WinCC Runtime the internal WinCC tag **“@DatasourceNameRT”** contains the “Data Source Name” of the WinCC Runtime database. You can read this tag to determine the desired Data Source Name. Entry http://support.automation.siemens.com/WW/view/en/9061684 contains detailed information. • During program start this field is assigned with the value “CC_OS_1__09_06_23_10_01_27R”.

Screen objects (Object name)	Description
Input field “**Provider**” Type: TextBox Name: txtProvider	In this input field the user must assign the name of the WinCC-OleDBProviders. **Notes**: • The WinCC-OleDBProvider is provided by the Connectivity Pack and is used for reading the Runtime data of the tag and alarm logging. • To access WinCC Runtime data (e.g. archive configuration or user archive) with the MS OleDB interface the “SQLOLEDB” provider is always used in the program. • During program start this field is assigned with the value “WinCCOLEDBProvider.1”.
Input field “**Uid**” Type: TextBox Name: txtUid	In this input field the user must assign the name of the data base user for the access to the Runtime database. **Notes**: • This user name is only used for database access with the MS OleDB interface. This field is not used for database access with the WinCC-OleDBProvider. • In the WinCC Runtime database you create a separate user for the MS OleDB access and assign password and user rights. Entry http://support.automation.siemens.com/WW/view/en/27147643 contains detailed information on how to create a user. • During program start this field is assigned with the value “DBUser”.
Input field “**Pwd**” Type: TextBox Name: txtPwd	In this input field the user must assign the password of the database user for access to the Runtime database. The password input is hidden, i.e. stars “ *** ” are displayed. **Notes**: • The password is only used for database access with the MS OleDB interface. This field is not used for database access with the WinCC-OleDBProvider. • During program start this field is assigned with the value “123456”.

4.1.2 “Archive” group

Type: GroupBox
Name: grpArchive

Figure 4-4



These radio buttons can be used to select the data source. The following options are available (data source):

- “Tag Logging”
- “Alarm Logging”
- “User Archive” (user archive)

Selecting a data source the user can decide which type of data to read. Furthermore, the groups “data selection”, “Time Interval”, “export data” and the “read archives” button can no longer be operated by the selection.

Clicking the “connect to database” button loads the data required for further settings from the database, therefore the “Connection” group can no longer be operated after the click.

Note

This group can always be operated.

Changing a data source also enables operating the “Connection” group again.

Table 4-2

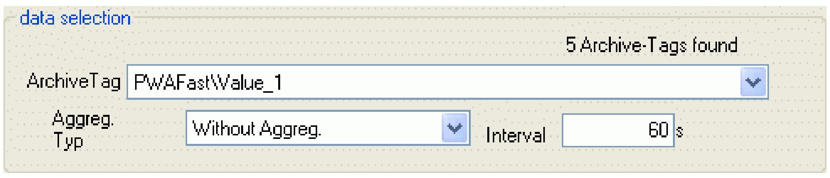
Screen objects (Object name)	**Description**
Option “**Tag Logging**” Type: RadioButton Name: rbtTagLogging	After pressing the “Tag Logging” option, “WinCC Tag Logging” is used as data source for the subsequent data query. **Notes**: After clicking the “connect to database” button the **“Data selection”, “Export Data”, “Time Interval”** groups and the **“read archives”** button can be operated.

Screen objects (Object name)	Description
Option “**Alarm Logging**” Type: RadioButton Name: rbtAlarmLogging	After pressing the “Alarm Logging” option, “WinCC Alarm Logging” is used as data source for the subsequent data query. **Note:** After clicking the “connect to database” button the **“Time Interval”** group and the **“read archives”** button can be operated.
Option “**User Archives**” Type: RadioButton Name: rbtAlarmLogging	After pressing the menu item “User Archives”, a “WinCC user archive” is used as data source for the subsequent data query. **Note**: After clicking the “connect to database” button the **“Time Interval”** group and the **“read archives”** button can be operated.

4.1.3 “data selection” group

Type: GroupBox
Name: grpDataSelection

Figure 4-5


data selection
5 Archive-Tags found
ArchiveTag: PWAFastValue_1
Aggreg. Typ: Without Aggreg.
Interval: 60 s

The objects in this group are only used for configuring the access to the runtime data of the **Tag Logging**. In this group an available archive tag of the WinCC Tag Logging can be selected and special parameters for compressing the data be specified.

Note This group can only be operated if the “Tag Logging” option has been selected in the “Archive” group and the “connect to database” button has been pressed.

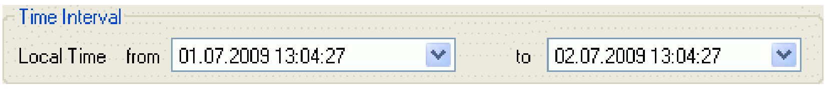
Table 4-3

Screen objects (Object name)	Description
Drop-down list **“Archive Tag”** Type: ComboBox Name: cmbTags	The drop-down list “Archive Tag” contains the archive tags configured in the Runtime database. You can open the drop-down list via mouse-click and select an archive tag whose values shall be read from the Runtime database. **Notes:** • When actually polling the data the WinCC-OleDBProvider is given the archive tag ID instead of the archive tag name for performance reasons. • Via the drop-down list the number of detected archive tags is displayed. • After selecting the “Data-Grid” control element and the CrystalReportViewer are deactivated.
Button **“Aggreg. Typ”** Type: ComboBox Name: cmbInterpol	The drop-down list “Aggreg. Typ” contains the aggregate types supported by the WinCC Connectivity Pack. You can select an aggregate type to summarize several successive archive values of the **Tag Logging** Runtime in the given time interval during data query (compressing). During program start the “Without Aggreg.” value is entered in the drop-down list. If this value is selected the values during this query are not combined with the WinCC-OleDBProvider.
Input field “**Interval**” Type: TextBox Name: txtStep	Here you enter the time interval in seconds in which the values are combined (compressed). The value entered in this field is only significant if a value unequal “Without Aggreg.” has been selected in the “Aggreg Typ” drop-down list. During program start this field is assigned with the value 60 (seconds).

4.1.4 “Time Interval” group

Type: GroupBox
Name: grpTimeInterval

Figure 4-6



01.07.2009 13:04:27
02.07.2009 13:04:27

The objects of this group are used for giving a time-interval which is used as filter criterion for requesting the Runtime data of the **Tag Logging** and **Alarm Logging**.

Note

This group can only be operated if the “Tag Logging” option has been selected in the “Archive” group and the “connect to database” button has been pressed.

The time is given in the local time format. When querying the time interval given here is transformed to UTC time and then transferred to the WinCC-OleDBProvider as a filter criterion.

When polling the user archives the time interval is not sent.

At program start the time interval is set to the last 24 hours.

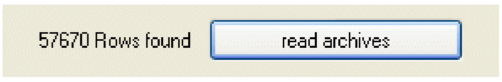
Table 4-4

Screen objects (Object name)	**Description**
DateTime selection box **“Local Time from”** Type: DateTimePicker Name: dtpFrom	In this selection box you specify the start time for the time interval of the query.
DateTime selection box **“Local Time to”** Type: DateTimePicker Name: dtpTo	In this selection box you specify the end time for the time interval of the query.

4.1.5 “read archives” button

Type: GroupBox
Name: grpExport

Figure 4-7



Click the “read archives” button to read the previously set archive from the database, this can take several minutes. While reading the data the program cannot be operated.

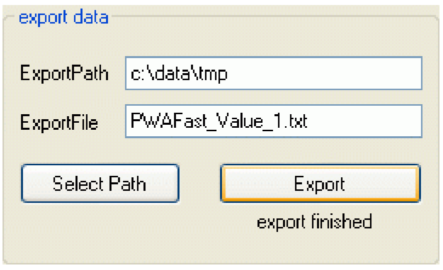
After the data have been read the volume of the read data is displayed in the text next to the button.

Note This button can only be operated if a radio button has been selected in the “Archive” group and the “connect to database” button has been pressed.

4.1.6 “Export data” group

Type: GroupBox
Name: grpExport

Figure 4-8



ExportPath: c:\data\tmp
ExportFile: PWAFast_Value_1.txt
export finished

The objects in this group are only used for configuring the Export of the read Runtime data of the Tag Logging, Alarmlogin.

Note This group can only be operated if the “read archives” button has been pressed.

The data displayed in the “DataGrid” spreadsheet are displayed (including performed sorting).

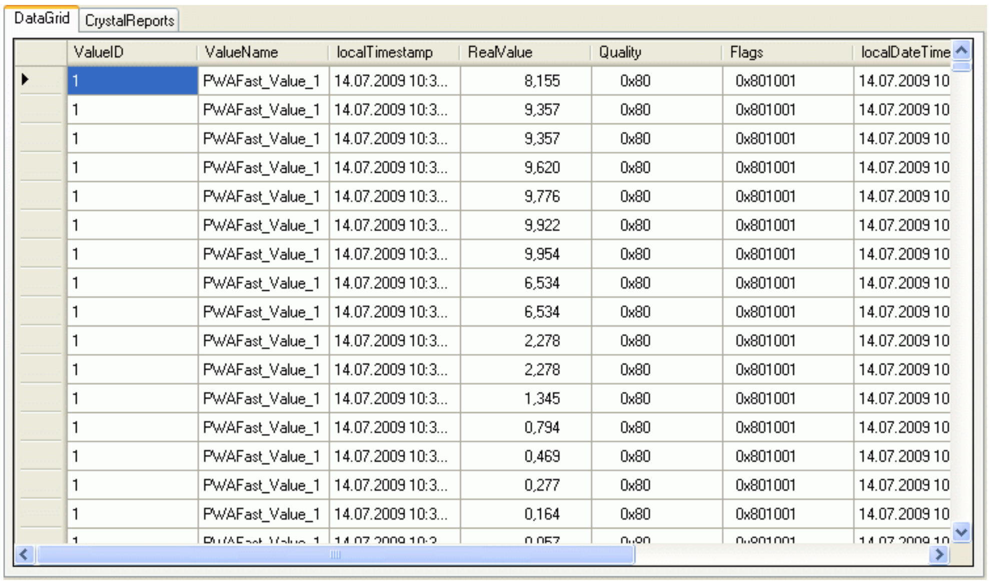
Table 4-5

Screen objects (Object name)	Description
Input field “**ExportPath**” Type: TextBox Name: txtExportPath	In this field the path is displayed in which the data are exported. The path can be changed by input into this field or by clicking the “Select Path” button. **Note:** At program start the path “**C:\data\tmp**” is created and the default entry for this field
Input field “**ExportFile**” Type: TextBox Name: txtExportFile	In this field you enter the file name. A .csv-compatible .txt-file is created. When selecting an archive tag or during program start the default entry for this field is the name of the selected archive tag followed by the characters “.txt”. “<ARCHIV_TAGNAME>.txt”
Button **“Select Path”** Type: Button Name: btnPath	Clicking this button opens the folder selection dialog. The selected or created folder is transferred to the “ExportPath” path.
Button **“Export”** Type: Button Name: btnExport	By clicking this button the data of the “DataGrid” control element are exported. After terminating the export the text “export finished” appears below the button.

4.1.7 Spreadsheet for displaying the runtime data

Type: TabControl
Name: tabView

Figure 4-9



ValueID	ValueName	localTimestamp	RealValue	Quality	Flags	localDateTime
1	PWAFast_Value_1	14.07.2009 10:3...	8,155	0x80	0x801001	14.07.2009 10
1	PWAFast_Value_1	14.07.2009 10:3...	9,357	0x80	0x801001	14.07.2009 10
1	PWAFast_Value_1	14.07.2009 10:3...	9,357	0x80	0x801001	14.07.2009 10
1	PWAFast_Value_1	14.07.2009 10:3...	9,620	0x80	0x801001	14.07.2009 10
1	PWAFast_Value_1	14.07.2009 10:3...	9,776	0x80	0x801001	14.07.2009 10
1	PWAFast_Value_1	14.07.2009 10:3...	9,922	0x80	0x801001	14.07.2009 10
1	PWAFast_Value_1	14.07.2009 10:3...	9,954	0x80	0x801001	14.07.2009 10
1	PWAFast_Value_1	14.07.2009 10:3...	6,534	0x80	0x801001	14.07.2009 10
1	PWAFast_Value_1	14.07.2009 10:3...	6,534	0x80	0x801001	14.07.2009 10
1	PWAFast_Value_1	14.07.2009 10:3...	2,278	0x80	0x801001	14.07.2009 10
1	PWAFast_Value_1	14.07.2009 10:3...	2,278	0x80	0x801001	14.07.2009 10
1	PWAFast_Value_1	14.07.2009 10:3...	1,345	0x80	0x801001	14.07.2009 10
1	PWAFast_Value_1	14.07.2009 10:3...	0,794	0x80	0x801001	14.07.2009 10
1	PWAFast_Value_1	14.07.2009 10:3...	0,469	0x80	0x801001	14.07.2009 10
1	PWAFast_Value_1	14.07.2009 10:3...	0,277	0x80	0x801001	14.07.2009 10
1	PWAFast_Value_1	14.07.2009 10:3...	0,164	0x80	0x801001	14.07.2009 10
1	PWAFast_Value_1	14.07.2009 10:3...	0,057	0x80	0x801001	14.07.2009 10

The result of the data query is displayed within a spreadsheet. The “**DataGrid**” tab for a table display as well as the “**CrystalReports**” for a formatted display (e.g. print version) are available.

Table 4-6

Screen objects (Object name)	Description
“DataGrid” tab Type: TabPage Name: tabPageDataGrid *contains* DataGrid control element Type: DataGridView Name: myGrid	The “DataGrid” control element is used for displaying the data in a table.

Screen objects (Object name)	**Description**
“CrystalReports” tab Type: TabPage Name: tabPageCrystalReports *contains* Report control element Type: CrystalReportViewer Name: crystalReportViewer1	The CrystalReportViewer is used for formatted data display.
“Rows found” display text Type: Label Name: lblAnz	This display field shows the number of result data records provided by the database query.
“**read archives**” button Type: Button Name: btnRead	Pressing the “read archives” button executes the database query. The delivered data are displayed in a table or formatted. When displaying the Tag Logging Runtime data the csv-file is written. If the csv-file already exists it is overwritten.

4.2 Reading, displaying and exporting the WinCC process value archives

The following description shows how the Runtime data of the Tag Logging is displayed in the “DataGrid” control element or in the Crystal Reports Viewer and output into a csv-file.

Table 4-7

No.	Action
1.	Configure the connection with the database.
2.	Select the “Tag Logging” radio button and click on the “connect to database” button.
3.	In the “ArchiveTag” drop-down list you select an archive tag and set the compression of the archive tag (Aggreg.Typ and Interval).
4.	Select the time interval

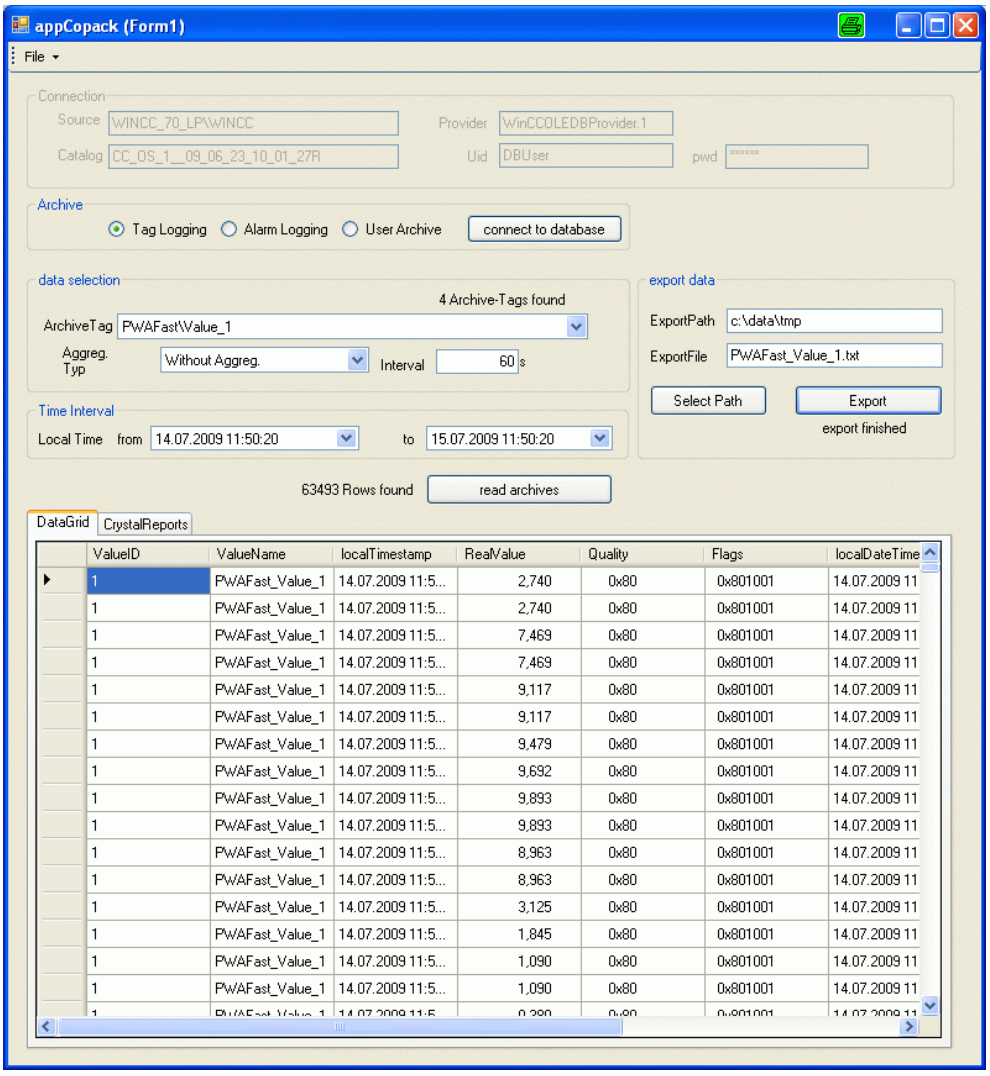
Note

The WinCC-OleDBProvider provides the data as a standard with a time stamp in UTC-format. To display the data this time stamp is transformed into local time.

The values “Quality” and “Flags” are displayed as hexadecimal values.

The following figure shows the table display of the Tag Logging archive data.

Figure 4-10



Connection

Source WINCC_70_LP\WINCC

Catalog CC_OS_1__09_06_23_10_01_27R

Provider WinCCOLEDBProvider.1

Uid DBUser

pwd ******

Archive

(● Tag Logging) ○ Alarm Logging ○ User Archive

data selection

4 Archive-Tags found

ArchiveTag PWAFast\Value_1

Aggreg. Typ Without Aggreg.

Interval 60 s

Time Interval

Local Time from 14.07.2009 11:50:20 to 15.07.2009 11:50:20

export data

ExportPath c:\data\tmp

ExportFile PWAFast_Value_1.txt

export finished

63493 Rows found

	ValueID	ValueName	localTimestamp	RealValue	Quality	Flags	localDateTime
▶	1	PWAFast_Value_1	14.07.2009 11:5...	2,740	0x80	0x801001	14.07.2009 11
	1	PWAFast_Value_1	14.07.2009 11:5...	2,740	0x80	0x801001	14.07.2009 11
	1	PWAFast_Value_1	14.07.2009 11:5...	7,469	0x80	0x801001	14.07.2009 11
	1	PWAFast_Value_1	14.07.2009 11:5...	7,469	0x80	0x801001	14.07.2009 11
	1	PWAFast_Value_1	14.07.2009 11:5...	9,117	0x80	0x801001	14.07.2009 11
	1	PWAFast_Value_1	14.07.2009 11:5...	9,117	0x80	0x801001	14.07.2009 11
	1	PWAFast_Value_1	14.07.2009 11:5...	9,479	0x80	0x801001	14.07.2009 11
	1	PWAFast_Value_1	14.07.2009 11:5...	9,692	0x80	0x801001	14.07.2009 11
	1	PWAFast_Value_1	14.07.2009 11:5...	9,893	0x80	0x801001	14.07.2009 11
	1	PWAFast_Value_1	14.07.2009 11:5...	9,893	0x80	0x801001	14.07.2009 11
	1	PWAFast_Value_1	14.07.2009 11:5...	8,963	0x80	0x801001	14.07.2009 11
	1	PWAFast_Value_1	14.07.2009 11:5...	8,963	0x80	0x801001	14.07.2009 11
	1	PWAFast_Value_1	14.07.2009 11:5...	3,125	0x80	0x801001	14.07.2009 11
	1	PWAFast_Value_1	14.07.2009 11:5...	1,845	0x80	0x801001	14.07.2009 11
	1	PWAFast_Value_1	14.07.2009 11:5...	1,090	0x80	0x801001	14.07.2009 11
	1	PWAFast_Value_1	14.07.2009 11:5...	1,090	0x80	0x801001	14.07.2009 11
	1	PWAFast_Value_1	14.07.2009 11:5	0,390	0x80	0x801001	14.07.2009 11

4.3 Reading, displaying and exporting the WinCC message archive

The following description shows how the Runtime data of the Alarm Logging is displayed in the “DataGrid” control element or in the Crystal Reports Viewer.

Table 4-8

No.	Action
1.	Configure the connection with the database.
2.	Select the “Alarm Logging” radio button and click on the “connect to database” button.
3.	Select the time interval
4.	Click the “read archives” button.
5.	If necessary you change the Export folder and the name of the Export file and then click the “Export” button

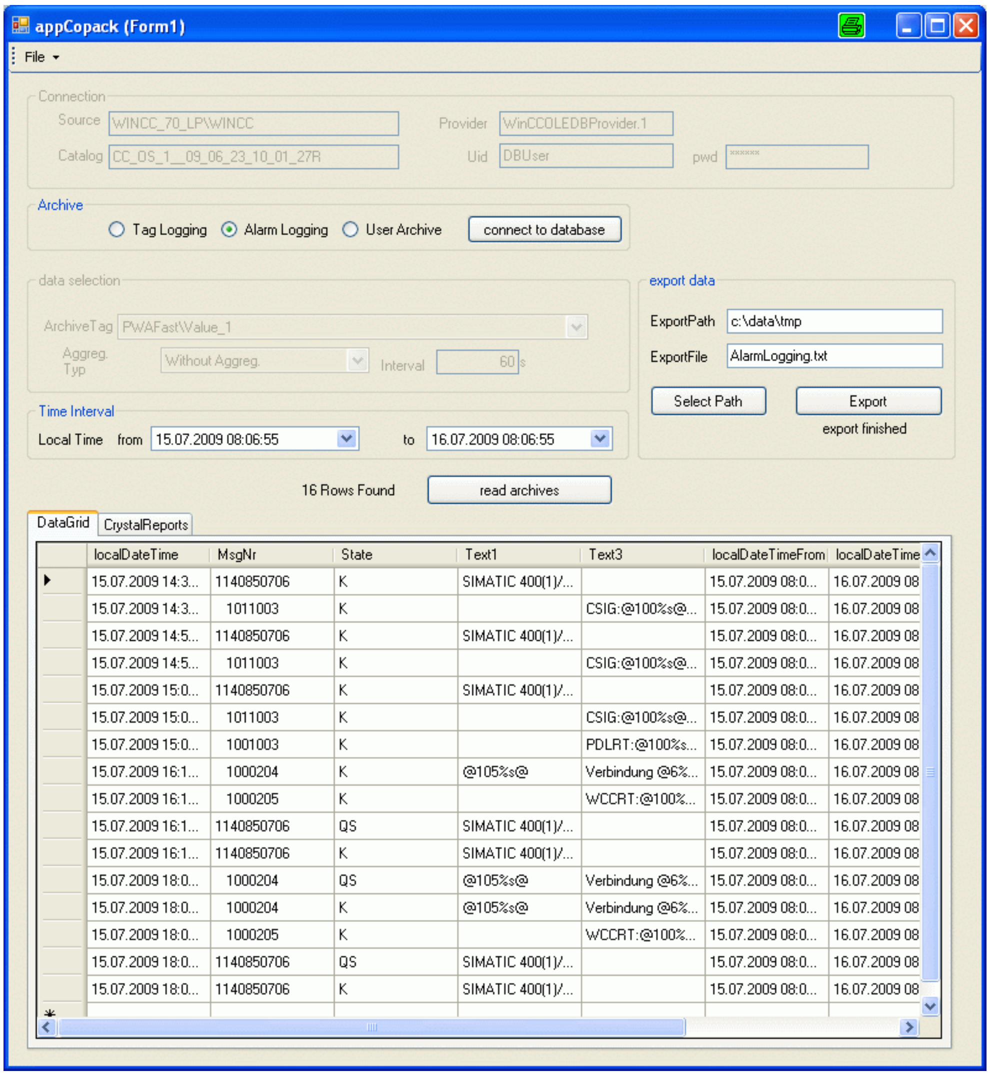
Note

The WinCC-OleDBProvider provides the data as a standard with a time stamp in UTC-format. To display the data this time stamp is transformed into local time.

The message status “State” is provided by the WinCC-OleDBProvider as a default decimal value. The message status is transformed into a character chain for display. The characters for a message class in the Alarm Logging editor are used.

The following figure shows the table display of the Alarm Logging archive data.

Figure 4-11



Source: WINCC_70_LP\WINCC
Catalog: CC_OS_1__09_06_23_10_01_27R
Provider: WinCCOLEDBProvider.1
Uid: DBUser
pwd: ******

Archive: ○ Tag Logging [✓ Alarm Logging] ○ User Archive

ArchiveTag: PWAFast\Value_1
Aggreg. Typ: Without Aggreg.
Interval: 60 s

Local Time from 15.07.2009 08:06:55 to 16.07.2009 08:06:55

ExportPath: c:\data\tmp
ExportFile: AlarmLogging.txt

export finished

16 Rows Found

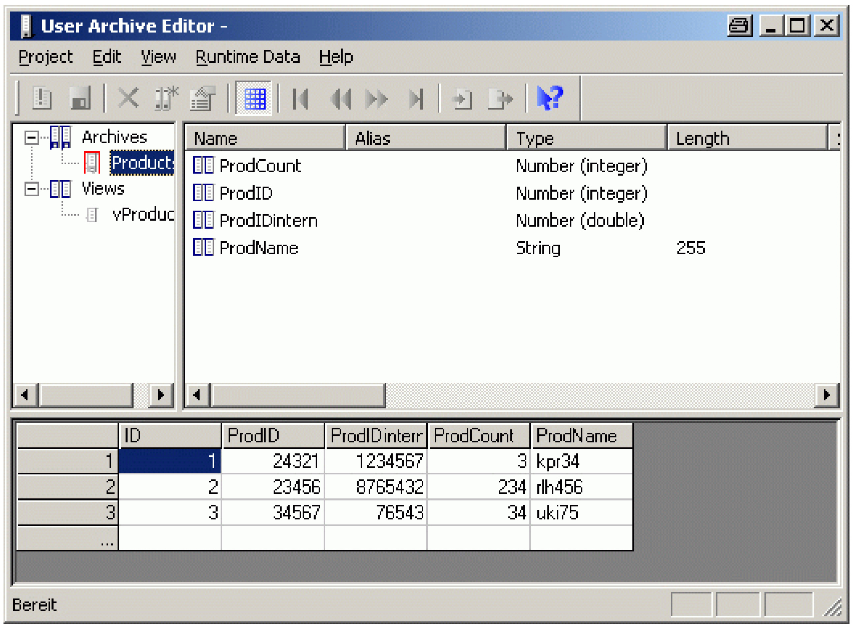
localDateTime	MsgNr	State	Text1	Text3	localDateTimeFrom	localDateTime
15.07.2009 14:3...	1140850706	K	SIMATIC 400(1)/...		15.07.2009 08:0...	16.07.2009 08
15.07.2009 14:3...	1011003	K		CSIG:@100%s@...	15.07.2009 08:0...	16.07.2009 08
15.07.2009 14:5...	1140850706	K	SIMATIC 400(1)/...		15.07.2009 08:0...	16.07.2009 08
15.07.2009 14:5...	1011003	K		CSIG:@100%s@...	15.07.2009 08:0...	16.07.2009 08
15.07.2009 15:0...	1140850706	K	SIMATIC 400(1)/...		15.07.2009 08:0...	16.07.2009 08
15.07.2009 15:0...	1011003	K		CSIG:@100%s@...	15.07.2009 08:0...	16.07.2009 08
15.07.2009 15:0...	1001003	K		PDLRT:@100%s...	15.07.2009 08:0...	16.07.2009 08
15.07.2009 16:1...	1000204	K	@105%s@	Verbindung @6%...	15.07.2009 08:0...	16.07.2009 08
15.07.2009 16:1...	1000205	K		WCCRT:@100%...	15.07.2009 08:0...	16.07.2009 08
15.07.2009 16:1...	1140850706	QS	SIMATIC 400(1)/...		15.07.2009 08:0...	16.07.2009 08
15.07.2009 16:1...	1140850706	K	SIMATIC 400(1)/...		15.07.2009 08:0...	16.07.2009 08
15.07.2009 18:0...	1000204	QS	@105%s@	Verbindung @6%...	15.07.2009 08:0...	16.07.2009 08
15.07.2009 18:0...	1000204	K	@105%s@	Verbindung @6%...	15.07.2009 08:0...	16.07.2009 08
15.07.2009 18:0...	1000205	K		WCCRT:@100%...	15.07.2009 08:0...	16.07.2009 08
15.07.2009 18:0...	1140850706	QS	SIMATIC 400(1)/...		15.07.2009 08:0...	16.07.2009 08
15.07.2009 18:0...	1140850706	K	SIMATIC 400(1)/...		15.07.2009 08:0...	16.07.2009 08

4.4 Reading, displaying and exporting the WinCC user archive

The following figure shows how the Runtime data of a user archive is displayed in the “DataGrid” control element or in the Crystal Reports Viewer.

This application shows the data of the user archive “Products”. This requires that the user archive in the WinCC project has been configured as follows:

Figure 4-12



Name	Alias	Type	Length
ProdCount		Number (integer)	
ProdID		Number (integer)	
ProdIDintern		Number (double)	
ProdName		String	255

	ID	ProdID	ProdIDintern	ProdCount	ProdName
1	1	24321	1234567	3	kpr34
2	2	23456	8765432	234	rlh456
3	3	34567	76543	34	uki75
...					

To display the “Products” user archive proceed as follows:

Table 4-9

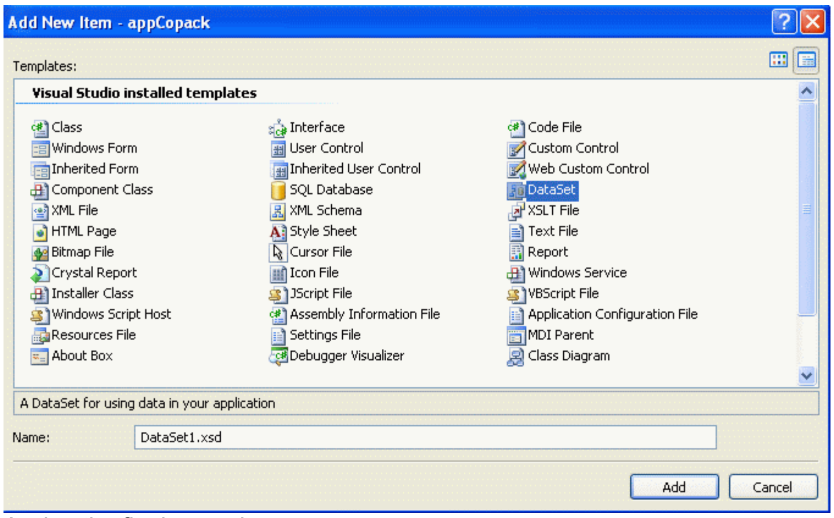
No.	Action
1.	Configure the connection with the database.
2.	Select the “Alarm Logging” radio button and click on the “connect to database” button.
3.	Click the “read archives” button.
4.	If necessary you change the Export folder and the name of the Export file and then click the “Export” button

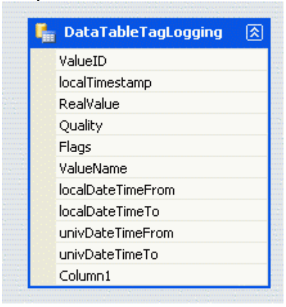
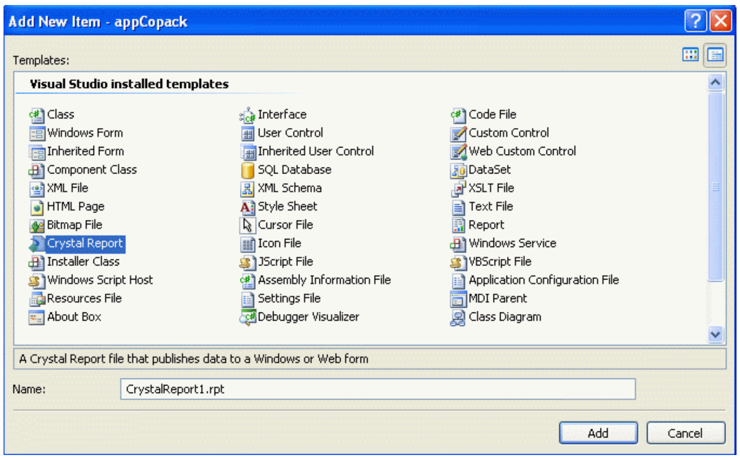
5 Further Notes, Tips and Tricks, etc.

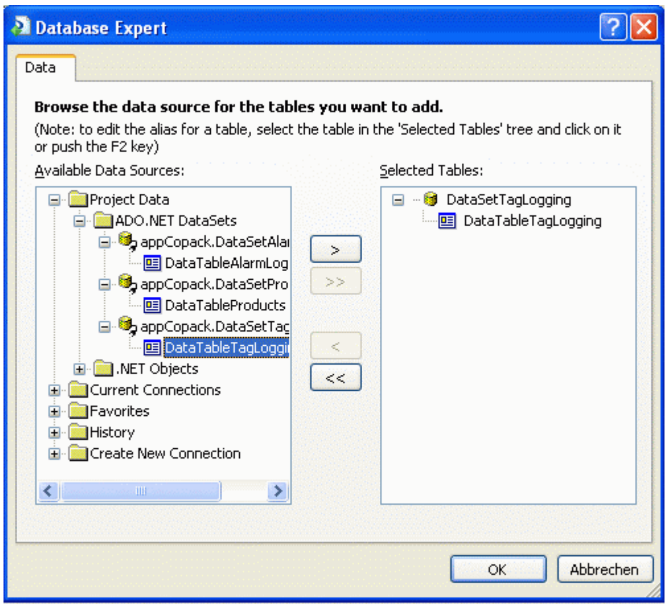
5.1 Creating a report in Crystal Reports

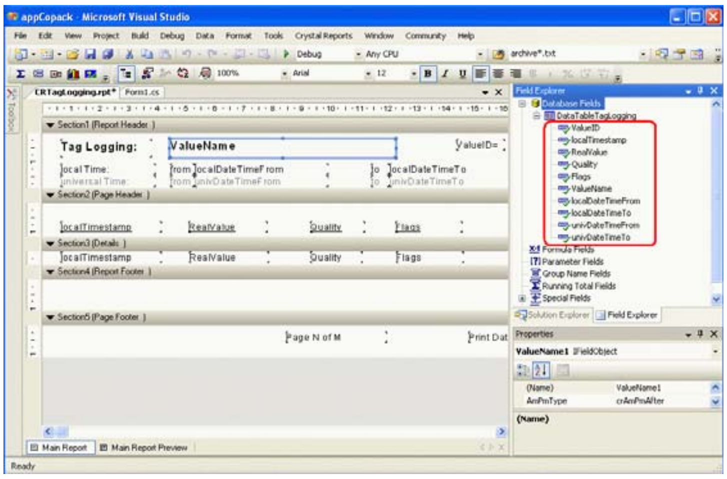
The following section describes how to create a crystal report in a form in MS Visual Studio 2005.

Table 5-1

No.	Procedure
1	**Add DataSet** In this step you create the **Dataset**, via which the read data are supplied to the report: • In the project folder Solution Explorer you right-click to open the context menu of the project and select the menu item ”Add > New Item” A window with a templates list opens. • Select the “DataSet” entry.  Add New Item - appCopack Templates: Visual Studio installed templates Class, Windows Form, Inherited Form, Component Class, XML File, HTML Page, Bitmap File, Crystal Report, Installer Class, Windows Script Host, Resources File, About Box, Interface, User Control, Inherited User Control, SQL Database, XML Schema, Style Sheet, Cursor File, Icon File, JScript File, Assembly Information File, Settings File, Debugger Visualizer, Code File, Custom Control, Web Custom Control, DataSet, XSLT File, Text File, Report, Windows Service, VBScript File, Application Configuration File, MDI Parent, Class Diagram A DataSet for using data in your application Name: DataSet1.xsd Add Cancel Assign the final name here. • Press the “Add” button to create the DataSet in the project.

No.	Procedure
2	**DataSet > Add DataTable** • Double-click the previously created DataSet in the Solution Explorer. The Dataset-Designer opens. • Right-click the “free” area within the Dataset-Designer. The context menu opens. Press the menu option “Add > DataTable” An empty DataTable is provided. • Adjust the name of the created DataTable.
3	**DataSet > Add DataTable > Add Column** • With the right mouse button you select the head of the DataTable within the DataSet. A context menu opens. Select the menu option “Add >Column”. A new column is added. Adjust the column name according to the archive data to be read later on. Add a respective DataTable column of the DataSet for each data table column.  DataTableTagLogging ValueID localTimestamp RealValue Quality Flags ValueName localDateTimeFrom localDateTimeTo univDateTimeFrom univDateTimeTo Column1
4	**Add a crystal report** • In the Solution Explorer you right-click to open the context menu of the project and select the menu command “Add > New Item”. A window with a list of templates opens. • Select the “Crystal Report” item.  Add New Item - appCopack Templates: Visual Studio installed templates Class, Interface, Code File Windows Form, User Control, Custom Control Inherited Form, Inherited User Control, Web Custom Control Component Class, SQL Database, DataSet XML File, XML Schema, XSLT File HTML Page, Style Sheet, Text File Bitmap File, Cursor File, Report Crystal Report, Icon File, Windows Service Installer Class, JScript File, VBScript File Windows Script Host, Assembly Information File, Application Configuration File Resources File, Settings File, MDI Parent About Box, Debugger Visualizer, Class Diagram A Crystal Report file that publishes data to a Windows or Web form Name: CrystalReport1.rpt Add Cancel • Assign the final name for the report here. (This name is used for creating the report class.) • Press the “Add” button to create the report in the project.

No.	Procedure
5	**Crystal Report with DataSet > Connect DataTable** • **Open report template** Open a report in the report designer. You can double-click a report "**<ReportName>.rpt**" in the Solution Explorer. • **Open the Database Expert** In a free area of the report you right-click the report. The context menu opens. Select the option "Database > Database Expert". To open the Database Expert you can also use the menu option "Crystal Reports > Database > Database Expert". The Database Expert opens. • **DataSet > Add DataTable** The "Available Data Sources" list contains the previously created DataSet in the "ADO.NET (XML)" item. Click the desired DataSet, then the contained DataTable becomes visible. Via double-click or the ">" button you add the DataTable to the "Selected Tables" list.  Database Expert Browse the data source for the tables you want to add. (Note: to edit the alias for a table, select the table in the 'Selected Tables' tree and click on it or push the F2 key) Available Data Sources: Project Data > ADO.NET DataSets > appCopack.DataSetAlar > DataTableAlarmLog; appCopack.DataSetPro > DataTableProducts; appCopack.DataSetTag > DataTableTagLoggi; .NET Objects; Current Connections; Favorites; History; Create New Connection Selected Tables: DataSetTagLogging > DataTableTagLogging From this time on the table columns are ready to use in the Crystal Reports Designer.

No.	Procedure
6	**Configure Crystal Reports with available database fields** Click on a free area of the report. The context menu opens. Select the menu option “Field Explorer”. You can also open the Field Explorer via menu command “Crystal Reports > Field Explorer”. The Field Explorer lists the previously connected DataTable with the previously created columns. You can now use the mouse to “drag” the columns into the report. A column used in the report is marked with a green checkmark in “Field Explorer > Database Fields”  appCopack - Microsoft Visual Studio CRTagLogging.rpt* Field Explorer Database Fields DataTableTagLogging ValueID localTimestamp RealValue Quality Flags ValueName localDateTimeFrom localDateTimeTo univDateTimeFrom univDateTimeTo Formula Fields Parameter Fields Group Name Fields Running Total Fields Special Fields Tag Logging: ValueName Properties Main Report Main Report Preview
7	**Add CrystalReportViewer into a form** To use a Crystal Report in the application you add the CrystalReportViewer into a form of your application. You find the CrystalReportViewer in the Toolbox at “Crystal Reports > CrystalReportViewer”. During runtime this enables previewing, printing and exporting the report. **Notes:** • The class of the report to be displayed is only assigned to the “.ReportSource” property during runtime. (see for example Crystal Report data connection) • Toolbar and status bar are not displayed in the standard setting. To display toolbar and statusbar, you set the properties **“DisplayStatusbar”** and **“DisplayToolbar”** to the value **“true”**. The toolbar (menu bar in the top area) provides the functions Export/Print/Scroll/Zoom and Search. The status bar (displayed at the bottom) provides information on the page and the zoom factor.

5.2 C#-Code to evaluate the process value archive

5.2.1 Definition for the connection process

The “string” tab “myConnectionString” is initialized with the required information for the connection established with the archive database. The principle setup of the string for the connection process is displayed below:

```
string myConnectionString =
 "Provider = WinCCOLEDBProvider.1; ////WinCC OleDBProvider
  Data Source = <Rechnername>\WINCC>;
  Catalog = <Data Source Name>“;
```

In the program the objects of the “Connection” group (“txtSource”, “txtCatalog” and “txtProvider”) are used to initialize the string for the connection process.

5.2.2 Definition for the data selection

The “string” tab “**mySelectQuery**” is initialized with the required information for the actual SQL data query. The setup of the string for the data selection is displayed below:

```
string mySelectQuery = "TAG:R,(id1;id2;idn),   //id=Ident. Process
 value archive
 'yyyy-mm-dd hh:mm:ss',     //Start time stamp
 'yyyy-mm-dd hh:mm:ss',     //End time stamp
 'TIMESTEP=n,Typ'";         //n=increment in seconds
                     //Type=compression type (e.g.AVG
                     //for the mean value)
```

In the program the objects of the **“Data selection”** group (“cmbTags”, “cmbInterpol” and “txtStep”) and **“Time Interval”** (“dtpFrom” and “dtpTo”) are used for initializing the string for the data selection.

Notes:

- The WinCCOLEDBProvider supports the specification of several archive tags in one query. The archive tags can be given with name or archive tag ID.

  This example program reads the data of only one archive tag. The values of the archive tag selected in the “cmbTags” drop-down list are selected.
- The archive tags are saved in the Runtime database with the universal time stamp (UTC). The query time period must be given to the WinCCOLEDBProvider in universal time code (UTC) so the delivered data have no time lag with the local time. The time stamp given in the “dtpFrom” and “dtpTo” objects are therefore transformed from local time into universal time before they are used for a data query. The following program code shows the transformation of the time stamp into universal time code as well as the preparation of the time tstamp for the data selection. The “string” tag “tfrom” and “tto” are used for setting up the string for the data selection.

```
//covert to universal time (utc)
localDateTimeFrom = dtpFrom.Value;
localDateTimeFrom =
  System.DateTime.Parse(localDateTimeFrom.ToString());
univDateTimeFrom = localDateTimeFrom.ToUniversalTime();
string tfrom = dtpFrom.Value.Year.ToString() + "-"
  + string.Format("{0:MM}",univDateTimeFrom.Month.ToString())
  + "-"
  + string.Format("{0:dd}",univDateTimeFrom.Day.ToString())
  + " "
  + string.Format("{0:HH}",univDateTimeFrom.Hour.ToString())
  + ":"
  +
string.Format("{0:mm}",univDateTimeFrom.Minute.ToString())
  + ":"
  +
string.Format("{0:ss}",univDateTimeFrom.Second.ToString());

//covert to universal time (utc)
localDateTimeTo = dtpTo.Value;
localDateTimeTo =
  System.DateTime.Parse(localDateTimeTo.ToString());
univDateTimeTo = localDateTimeTo.ToUniversalTime();
string tto = dtpTo.Value.Year.ToString() + "-"
  + string.Format("{0:MM}",univDateTimeTo.Month.ToString())
  + "-"
  + string.Format("{0:dd}",univDateTimeTo.Day.ToString())
  + " "
  + string.Format("{0:HH}",univDateTimeTo.Hour.ToString())
  + ":"
  + string.Format("{0:mm}",univDateTimeTo.Minute.ToString())
  + ":"
  + string.Format("{0:ss}",univDateTimeTo.Second.ToString());
```

5.2.3 Connecting with the database and reading data

The following program code shows the connecting process with the database and the access to the data.

```
OleDbConnection myConnection;
OleDbCommand myCommand;
OleDbDataAdapter myAdapter;

.
.
.
// Connection Archive-Database
myConnection=new OleDbConnection(myConnectionString);
myCommand = new OleDbCommand(mySelectQuery)
myCommand.Connection = myConnection;
myAdapter = new OleDbDataAdapter (myCommand); //connect and access
```

Note:
In this example data is read via OleDbDataAdapter. OleDbDataReader can also be used. This does not affect the actual transmission process of the SQL query but affects further data processing. Using OleDbDataAdapter the information in the DataGridView can be provided directly without having to deal with the lines and columns.

5.2.4 Providing data for DataGrid and/or Crystal Report:

The example program uses the DataGridView control element for representation of the data in a table and the CrystalReportViewer for formatted data output. For both displays an object of type DataTable is used as the data source. The OleDBDataAdapter supplies the DataTable object **"myTableTags"** with the read data of the SQL query using the **".Fill()"** method.

```
DataTable myTableTags;
.
.
.
myTableTags = new DataTable();
.
.
.
myTableTags.TableName = "myTableTags";
myAdapter.Fill(myTableTags);
```

Data of the DataTable **"myTableTags"** are read line by line (data record by data record), prepared for display and written to a further DataTable **"myTableTagsModify"**. Data of the modified DataTable "myTableTagsModify" are used for display.

The following program code shows the preparation of the data:

The structure (columns) of DataTable "myTableTagsModify" must in this case be created "manually". The following code shows creating the first three columns of DataTable "myTableTagsModify".

```
//=====================================================
//
//Adding Columns and Rows to Data Table myTableTagsModify
//
//=====================================================
DataColumn newColumn = new DataColumn ("localTimestamp",
  System.Type.GetType("System.String"));
newColumn.Caption = "localTimestamp";
newColumn.DefaultValue = string.Empty;
myTableTagsModify.Columns.Add(newColumn);
//
newColumn = new DataColumn ("RealValue",
  System.Type.GetType("System.String"));
newColumn.Caption = "RealValue";
newColumn.DefaultValue = string.Empty;
myTableTagsModify.Columns.Add(newColumn);
//
newColumn = new DataColumn ("Quality",
  System.Type.GetType("System.String"));
newColumn.Caption = "Quality";
newColumn.DefaultValue = string.Empty;
myTableTagsModify.Columns.Add(newColumn);
.
.
```

The following code shows “filling” the DataTable “myTableTagsModify”. The following adjustments from DataTable “myTableTags” are made:

For display the time stamp is transformed from universal time code (UTC) to the local time code.

The value of the archive tags is displayed with 3 digits.

The Quality code and the tag status are displayed as hexadecimal numbers.

In this section the columns “ValueName”, “localDateTimeFrom”, “localDateTimeTo”, “univDateTimeFrom” and “univDateTimeTo” are created and supplied with values. This column is accessed in the report.

-

```
//modify DataTable
myTableTagsModify.Clear();
foreach (DataRow row in myTableTags.Rows)
{
DataRow newRow = myTableTagsModify.NewRow();
//covert to local time
localDateTime =
System.DateTime.Parse(row["Timestamp"].ToString());
localDateTime = localDateTime.ToLocalTime();
newRow["localTimestamp"] = localDateTime.ToString();
newRow["RealValue"] =
  (String.Format("{0:F3}",row["RealValue"])).PadLeft(20);
newRow["Quality"] = String.Format("0x{0:X}",
  row["Quality"]).PadLeft(10);
newRow["Flags"] = String.Format("0x{0:X}",row["Flags"]).PadLeft(10);
newRow["ValueID"] = row["ValueID"];
newRow["ValueName"] = szValueName;
newRow["localDateTimeFrom"] = localDateTimeFrom;
newRow["localDateTimeTo"] = localDateTimeTo;
newRow["univDateTimeFrom"] = univDateTimeFrom;
newRow["univDateTimeTo"] = univDateTimeTo;
myTableTagsModify.Rows.Add(newRow);
}//foreach(DataRow)

myGrid.DataSource = myTableTagsModify;
```

5.2.5 DataGrid data connection:

The name of the DataTable to be displayed is assigned to the property **".DataSource"** of the DataGridView control element.

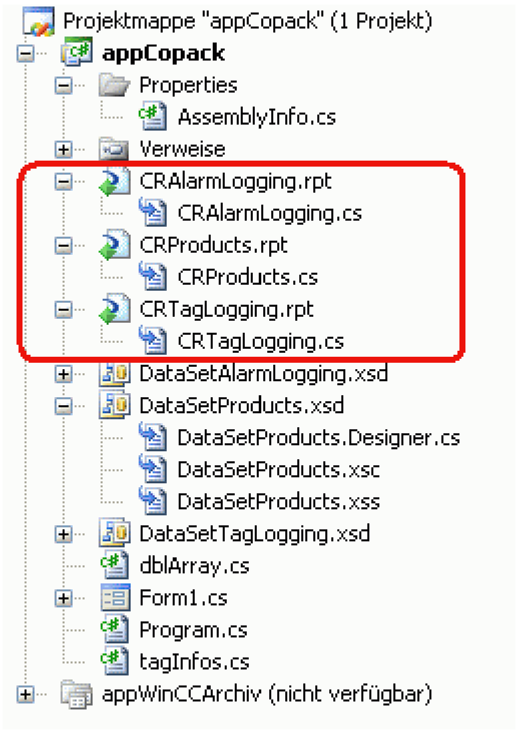
```
myGrid.DataSource = myTableTagsModify;
```

5.2.6 Crystal Report data connection:

In this example a separate report was created for each report (Tag Logging, Alarm Logging and Archive values). For each report Visual Studio creates a report class with the same name.

The following figure displays the reports created in this project as well as the existing report classes.

Figure 5-1



```
Projektmappe "appCopack" (1 Projekt)
appCopack
    Properties
        AssemblyInfo.cs
    Verweise
    CRAlarmLogging.rpt
        CRAlarmLogging.cs
    CRProducts.rpt
        CRProducts.cs
    CRTagLogging.rpt
        CRTagLogging.cs
    DataSetAlarmLogging.xsd
    DataSetProducts.xsd
        DataSetProducts.Designer.cs
        DataSetProducts.xsc
        DataSetProducts.xss
    DataSetTagLogging.xsd
    dblArray.cs
    Form1.cs
    Program.cs
    tagInfos.cs
appWinCCArchiv (nicht verfügbar)
```

The connection of the data pool of the created instance to the DataTable occurs via .SetDataSource.
In out example there is only one Crystal Report Viewer. The .ReportSource connection and the desired report instance determines which report it shall display here myDataReportAlarms.

```
//   Activating Crystal-Report
CRTagLogging myDataReportTags = new CRTagLogging();
myDataReportTags.SetDataSource(myTableTagsModify);
crystalReportViewer1.ReportSource = myDataReportTags;
```

5.2.7 Closing the connection with archive

```
myConnection.Close();
```

5.2.8 Exporting the archive values into a csv-file

The DataTable “myTableTagsModify” is read line by line (data record by data record) and the content written to a csv-file.

```
StreamWriter streamTagLogging = null;
string strLine = "";
string strExportFile = "";


//
//Loop through DataTable by DataRow
//
//text file open
strExportFile = String.Format("{0}\\{1}", txtExportPath.Text,
  txtExportFile.Text);
streamTagLogging = File.CreateText(strExportFile);
strLine = "strExportFile=" + txtExportFile.Text;
streamTagLogging.WriteLine(strLine);
streamTagLogging.WriteLine(myConnectionString);
strLine = String.Format("mySelectQuery=\"{0}\"", mySelectQuery);
streamTagLogging.WriteLine(strLine);
strLine = "localTimestamp; RealValue; Quality; Flags";
streamTagLogging.WriteLine(strLine);

foreach (DataRow row in myTableTagsModify.Rows)
{
  strLine = String.Format("{0};  {1}; {2}; {3}",
    row["localTimestamp"], row["RealValue"], row["Quality"],
    row["Flags"]);
  streamTagLogging.WriteLine(strLine);
}//DataRow
if (streamTagLogging != null)streamTagLogging.Close();
```

5.3 C#-Code for evaluating the alarms and messages

5.3.1 Definition for the connection process

Please proceed as in section 5.2.1.

5.3.2 Definition for the data selection

The “string” tab “**mySelectQuery**” is initialized with the required information for the actual SQL data query. The principle setup of the string for the data selection is displayed below:

```
string mySelectQuery = "ALARMVIEW:SELECT * FROM AlgViewDeu
  Where DateTime>'2007-08-10 12:00:00'
  AND DateTime<'2007-08-10 14:00:00'";
```

In the program the objects of the “**Time Interval**” group (“dtpFrom” and “dtpTo”) are used to initialize the string for the data selection.

Note

The messages archive are saved in the Runtime database with the **universal time stamp (UTC)**. As for data selection for process value archives (see 5.2.2) the local time stamp is transformed into the universal time stamp.

5.3.3 Connecting with the database and reading data:

Please proceed as in section 5.2.

5.3.4 Providing data for DataGrid and/or Crystal Report:

Please proceed as in section 5.2.4.

The following program code shows how the status of a message can be displayed is character chain instead of a number value (as in WinCC Alarm Control). The number value of the status of a message is evaluated in a switch instruction and the respective character chain assigned in the various case branches.

Note

Information on the possible number values, which the “status” of a message can take on, is available in entry 24842903 or in the WinCC Information System at:
 “Working with WinCC
> ANSI-C for creating procedures and actions
> ANSI-C function description > Appendix > Structure definitions
> Structure definition MSG_RTDATA_STRUCT ”

```
//szState = String.Format("0x{0:X}", row["State"]).PadLeft(10);
iState = (short)(row["State"]);
switch (iState){
  case 1:
    szState = row["TxtCame"].ToString();
    break;
case 2:
    szState = row["TxtWent"].ToString();
    break;
  case 3:
    szState = row["TxtAck"].ToString();
    break;
  case 16://0x10 (Quit System)
    szState = row["TxtAck"].ToString();
    break;
  default:
    szState = String.Format("0x{0:X}", row["State"]).PadLeft(10);
    break;
}//switch row["State"]
newRow["State"] = szState;
```

5.3.5 DataGrid data connection:

The name of the DataTable to be displayed is assigned to the property **“.DataSource”** of the DataGridView control element.

```
myGrid.DataSource = myTableAlarmsModify;
```

5.3.6 Crystal Report data connection:

Please proceed as in section 5.2.5.

```
//   Activating Crystal-Report
CRAlarmLogging myDataReportAlarms = new CRAlarmLogging();
myDataReportAlarms.SetDataSource(myTableAlarmsModify);
crystalReportViewer1.ReportSource = myDataReportAlarms;
```

5.3.7 Closing the connection with archive

```
myConnection.Close();
```

5.4 C#-Code to evaluate the user archive

5.4.1 Definition for the connection process

The “string” tab “myConnectionString” is initialized with the required information for the connection established with the archive database. The principle setup of the string for the connection process is displayed below:

```
   string myConnectionString =
    "Provider =SQLOLEDB; //Microsoft OleDBProvider
     Data Source = <Rechnername>\WINCC>;
     uid = <User Name>
     pwd = <Password>
     Initial Catalog = <Data Source Name>“;
```

In the program the objects of the “Connection” group (“txtSource”, “txtCatalog”, “txtProvider”, “txtUid” and “txtPwd”) are used to initialize the string for the connection process.

5.4.2 Definition for the data selection

The “string” tab “**mySelectQuery**” is initialized with the required information for the actual SQL data query. The setup of the string for the data selection is displayed below:

```
mySelectQuery = "SELECT iID,szName, iCount, fWeight FROM UA#Products" ;
```

5.4.3 Connecting with the database and reading data:

Please proceed as in section 5.2.

5.4.4 Providing data for DataGrid and/or Crystal Report:

Please proceed as in section 5.2.4.

```
//  Providing data for data grid
myTableProducts.TableName = "myTableProducts";
myAdapter.Fill(myTableProducts);
```

5.4.5 DataGrid data connection:

The name of the DataTable to be displayed is assigned to the property **".DataSource"** of the DataGridView control element.

```
myGrid.DataSource = myTableProducts;
```

5.4.6 Crystal Report data connection:

Please proceed as in section 5.3.6.

```
//   Activating Crystal-Report
CRProducts myDataProducts = new CRProducts ();
myDataProducts.SetDataSource(myTableProducts);
crystalReportViewer1.ReportSource = myDataProducts;
```

5.4.7 Closing the connection with archive

```
myConnection.Close();
```

6 Bibliography

Internet Links

The following list is by no means complete and only provides a selection of appropriate sources.

Table 6-1 Internet links

	Topic	Title
\1\	Reference to this entry	http://support.automation.siemens.com/WW/view/en/35840700
\2\	Siemens I IA/DT Customer Support	http://support.automation.siemens.com
\3\	Determining the Data Source Name.	http://support.automation.siemens.com/WW/view/en/9061684
\4\	Creating a user for the WinCC Runtime database	http://support.automation.siemens.com/WW/view/en/27147643
\5\	“Status“ of a message	http://support.automation.siemens.com/WW/view/en/24842903
\6\	Access to WinCC archive	http://support.automation.siemens.com/WW/view/en/22578952

7 History

Table 7-1 History

Version	Date	Changes
V1.0	06.12.2007	First issue
V1.1	08.10.2009	• User guidance by activating/deactivating input groups • Export of messages and user archives realized • Application can be used without installed Crystal Report