

service & SUPPORT

How Can Archived WinCC Data Be Accessed
with a C# Windows Application?

FAQ

SIEMENS

Table of Contents

Table of Contents	2
Question	3
How can archived WinCC data be accessed with a C# Windows application?	3
1 Introduction	4
1.1 Necessity	4
1.2 Overview	4
2 Visual C# Sample Program Installation	6
2.1 Requirements	6
2.2 Downloading, extracting and calling the sample project	7
3 Functional Description of the Visual C# Sample Program	8
3.1 Detailed description of the screen objects	8
3.1.1 Menu bar	9
3.1.2 "Connection" group	11
3.1.3 "Data selection" group	14
3.1.4 "Export data" group	16
3.1.5 "Time Interval" group	17
3.1.6 Tab for the Runtime data display	18
3.2 Reading out, displaying and exporting WinCC process value archives	19
3.3 Reading out and displaying WinCC message archive	22
3.4 Reading out and displaying WinCC User Archive	25
4 C# Program Code Description	28
4.1 C# code for evaluating the process value archives	28
4.1.1 Definition for the connection setup	28
4.1.2 Definition for the data selection	28
4.1.3 Setting up the connection to the database and reading data	30
4.1.4 Providing data for DataGrid and/or Crystal Report:	30
4.1.5 Data connection of the DataGrid:	32
4.1.6 Data connection of the Crystal Report:	33
4.1.7 Closing the connection to the archive	33
4.1.8 Exporting the archive values to a CSV file	34
4.2 C# code for evaluating the alarms and messages	34
4.2.1 Definition for the connection setup	34
4.2.2 Definition for the data selection	34
4.2.3 Setting up the connection to the database and reading data:	35
4.2.4 Providing data for DataGrid and/or Crystal Report:	35
4.2.5 Data connection of the DataGrid:	35
4.2.6 Data connection of the Crystal Report:	36
4.2.7 Closing the connection to the archive	36
4.3 C# code for evaluating the User Archives	36

4.3.1	Definition for the connection setup	36
4.3.2	Definition of the data selection.....	36
4.3.3	Setting up the connection to the database and reading data:	37
4.3.4	Providing data for DataGrid and/or Crystal Report:	37
4.3.5	Data connection of the DataGrid:	37
4.3.6	Data connection of the Crystal Report:.....	37
4.3.7	Closing the connection to the archive.....	37
5	Creating a Report in Crystal Reports.....	38

This entry is from the Internet offer of Siemens AG, Automation and Drives, Service & Support. The link below takes you directly to the download page of this document.

<http://support.automation.siemens.com/WW/view/en/26697936>

Question

How can archived WinCC data be accessed with a C# Windows application?

1 Introduction

1.1 Necessity

The WinCC Runtime database of **WinCC V6.0 or higher** is segmented; this means that the data is stored in several archive segments (several databases). The storing of the data is partly compressed in a binary form. The “**WinCC Connectivity Pack**” WinCC option provides the **WinCC OleDBProvider** with which the Tag and Alarm Logging Runtime data can be read directly. The WinCC OleDBProvider provides the data from the corresponding archive segments in decompressed (decoded) form. When accessing the Tag and Alarm Logging data, the WinCC Connectivity Pack user does not have to worry about the segmentation of the archives and their encoding.

1.2 Overview

This entry describes how the archived WinCC Runtime data of the Tag Logging (process data archiving), of the Alarm Logging (archived messages and alarms) and of the User Archives are accessed with a separate C Sharp Windows application and with the aid of the WinCC Connectivity Pack.

It is described how the Runtime data of the Tag Logging, of the Alarm Logging and of the User Archives can be read, displayed and output via Crystal Reports or to a CSV file.

This document does not focus on creating and describing a C# Windows application but on the necessary mechanisms to access the WinCC archive data. These mechanisms are:

- Setting up the database connection
- Preparing the data (adjusting the time-of-day format, local time and universal time)
- Using the MS OleDB interface to read the WinCC archive configuration and the WinCC User Archives

Note:

This example uses only read accesses to the data. When using the MS OleDB interface, write accesses to the data are technically possible by means of corresponding SQL commands (for example, update, insert, delete ...). **Attention!** However, write accesses have only been tested and released for the data of the User Archives.

- Using the WinCC OleDBProvider to read out the archived process values (WinCC Tag Logging) and alarms and messages (WinCC Alarm Logging).
- Tabular display of the data with the “DataGrid” control element
- Output of the data to a CSV file

- Output of the data via Crystal Reports

This entry includes a complete Visual C# sample program that illustrates the access mechanisms listed above by means of a runnable Windows application.

Note:

The entry <http://support.automation.siemens.com/WW/view/en/22578952> provides an overview of further options for accessing WinCC archives.

2 Visual C# Sample Program Installation

2.1 Requirements

Two separate computers (**WinCC server** and **Connectivity Pack client**) were used for this example. The WinCC server performs the 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	Intel Pentium 4 CPU 2.4 GHz, 2GB RAM
Operating system	MS Windows XP Professional SP2
WinCC software	SIMATIC WinCC V6.2 (includes SQL Server 2005 SP1)

Connectivity Pack client:

Table 2-2

Hardware	Intel Pentium 4 CPU 2.4 GHz, 1GB RAM
Operating system	MS Windows XP Professional SP2 Note: The "Microsoft Message Queuing" Windows component must be installed. You can install this component in "Control Panel > Add or Remove Programs > Add/Remove Windows Components > Message Queuing".
Development environment	MS Visual Studio 2005 Professional; (Visual C#)
WinCC software	SIMATIC WinCC/ConnectivityPack V6.2 (client) for using the WinCC OleDbProvider

2.2 Downloading, extracting and calling the sample project

Download the sample program available as a download and extract the received zip archive. The "WinCCcopack" folder is generated during this process. The "WinCCcopack > appCopack" subfolder contains the C# project created with Visual Studio 2005.

Notes:

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

- Visual Studio development environment is installed

If Visual Studio is installed on your computer, you can open the project by double-clicking the "**appCopack.sln**" file. After the project has been opened in Visual Studio, you can edit the sources, compile the program and run it using the "Debug > Start Without Debugging" menu command.

- Visual Studio development environment is not installed

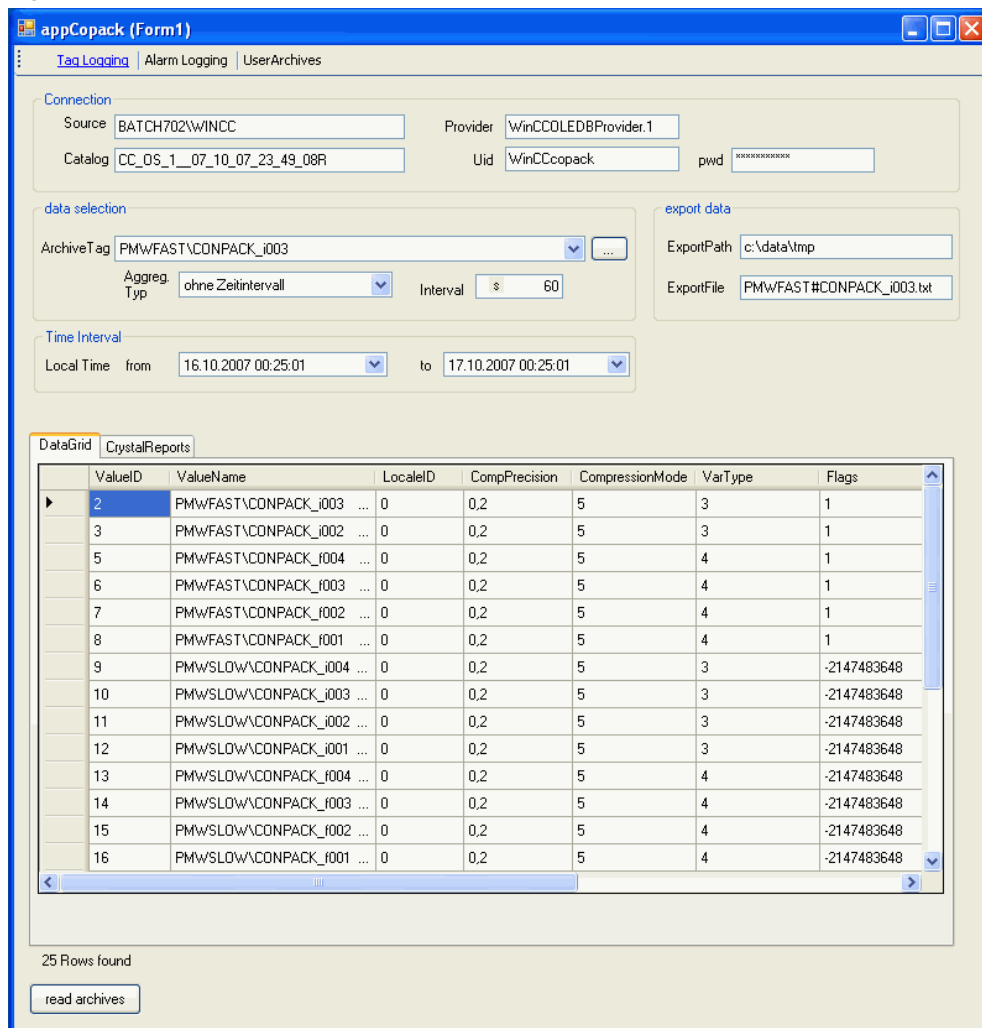
If Visual Studio is not installed on your computer, you can run the program by double-clicking the "**.WinCCcopack > appCopack\obj\Debug\appCopack.exe**" file.

3 Functional Description of the Visual C# Sample Program

3.1 Detailed description of the screen objects

After the Windows application has been started, the following “appCopack (Form1)” program window appears.

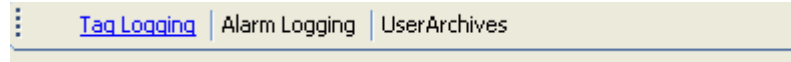
Figure 3-1



The program consists of a program window. The following table describes its components.

3.1.1 Menu bar

Figure 3-2



The menu bar contains menu commands for changing the data sources. The following menu items (data sources) are available:

“Tag Logging” ,

“Alarm Logging” or

“User Archives”

By selecting a data source, the user can decide which type of data is to be read.

When a menu item is selected, the blue font color indicates the selection, another menu item is deselected. Only one menu item can be selected at a specific time.

When starting the program, the “Tag Logging” menu item is selected by default.

Table 3-1

Screen objects (object name)	Description
<p>“Tag Logging” menu item Type: ToolStripLabel Name: toolStripLbITagLogging</p>	<p>By selecting the “Tag Logging” menu item, “WinCC Tag Logging” is used as a data source for the following data query. When this item is selected, the Tag Logging Runtime data is displayed in the “DataGrid” control element or in the CrystalReportViewer.</p> <p>Note:</p> <ul style="list-style-type: none"> • The screen objects of the “Data selection” and “Export Data” groups can only be operated or activated when the “Tag Logging” menu item is selected. • The Tag Logging data is automatically (when displaying the data) exported to the configured CSV file <p>After starting the program or after selecting the “...” button of the “Data selection” group, the available (configured) archive tags are displayed in the “DataGrid” control element. The archive data of the selected archive tags is only displayed in the “DataGrid” control element after selecting the “read archives” button.</p>
<p>“Alarm Logging” menu item (toolStripLbIAlarmLogging)</p>	<p>By selecting the “Alarm Logging” menu item, “WinCC Alarm Logging” is used as a data source for the following data query. When this item is selected, the Alarm Logging Runtime data is displayed in the “DataGrid” control element or in the CrystalReportViewer.</p> <p>Notes:</p> <ul style="list-style-type: none"> • The screen objects of the “Data selection” and “Export Data” groups cannot be operated. • The Alarm Logging data is not exported to a CSV file. • On the one hand, the CrystalReportViewer provides the functions for exporting the data. On the other hand, you can expand this sample program so that the Alarm Logging Runtime data is automatically exported to a CSV file.
<p>“User Archives” menu item (toolStripLbIUserArchives)</p>	<p>By selecting the “User Archives” menu item, a “WinCC User Archive” is used as a data source for the following data query. When this item is selected, the Runtime data of the “Products” User Archive is displayed in the</p>

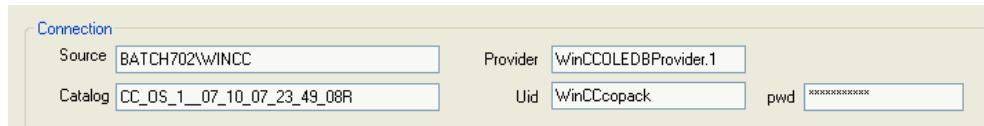
Screen objects (object name)	Description
	<p>“DataGrid” control element or in the CrystalReportViewer.</p> <p>Notes:</p> <ul style="list-style-type: none"> • The screen objects of the “Data selection” and “Export Data” groups cannot be operated. • The User Archive data is not exported to a CSV file. • On the one hand, the CrystalReportViewer provides the functions for exporting the data. On the other hand, you can expand this sample program so that the User Archive data is automatically exported to a CSV file.

3.1.2 “Connection” group

Type: GroupBox

Name: grpConnection

Figure 3-2



The screenshot shows a 'Connection' group with the following fields:

- Source: BATCH702\WINCC
- Provider: WinCCOLEDBProvider.1
- Catalog: CC_OS_1_07_10_07_23_49_08R
- Uid: WinCCcopack
- pwd: [masked]

The input boxes of this group are used to parameterize the connection setup to the data source. When starting the program, the boxes are initialized with default values. The user can change the connection parameters in Runtime and perform the data query.

Screen objects (object name)	Description
<p>“Source” input box Type: TextBox Name: txtSource</p>	<p>This input box includes the WinCC server name, followed by the instance name of the SQL server whose Runtime data is to be accessed.</p> <p><computer name>\WINCC</p> <p>When starting the program, the default input for this box is “ESJPGWINCC”.</p>
<p>“Catalog” input box Type: TextBox Name: lblCatalog</p>	<p>In this input box, the user has to specify the Data Source Name (DSN) of the Runtime database whose data is to be accessed.</p> <p>Notes:</p> <ul style="list-style-type: none"> • In WinCC Runtime, the internal “@DataSourceNameRT” WinCC tag contains the “Data Source Name” of the WinCC Runtime database. You can read out this tag to determine the desired Data Source Name. The entry http://support.automation.siemens.com/W/view/en/9061684 provides detailed information on this topic. • When starting the program, the default input for this box is the value “CC_OS_1__0710_10_13_38_39R”.

Screen objects (object name)	Description
<p>“Provider” input box Type: TextBox Name: txtProvider</p>	<p>In this input box, the user has to specify the name of the WinCC OleDbProvider.</p> <p>Notes:</p> <ul style="list-style-type: none"> • The WinCC OleDbProvider is provided by the Connectivity Pack and used for reading the Runtime data of Tag and Alarm Logging. • In the program, the “SQLOLEDB” provider is permanently used for accesses to WinCC Runtime data (e.g., archive configuration or User Archives) with the MS OleDb interface. • When starting the program, the default input for this box is the value “WinCCOLEDBProvider.1”
<p>“ Uid ” input box Type: TextBox Name: txtUid</p>	<p>In this input box, the user must enter the name of the database user for the Runtime database access.</p> <p>Notes:</p> <ul style="list-style-type: none"> • This user name is only used for database accesses with the aid of the MS OleDb interface. This box is not relevant for database accesses with the aid of the WinCC OleDbProvider. • Create a separate user for the MS OleDb accesses in the WinCC Runtime database and assign a password and user rights. The entry http://support.automation.siemens.com/W/view/en/27147643 provides detailed information on how to create a user. • When starting the program, the default input for this box is the value “WinCCcopack”
<p>“ Pwd ” input box Type: TextBox Name: txtPwd</p>	<p>In this input box, the user has to enter the password of the database user for the Runtime database access. The input of the password is hidden, i.e. stars “****” are displayed.</p> <p>Notes:</p> <ul style="list-style-type: none"> • The password is only used for database accesses with the aid of the MS OleDb interface. This box is not relevant for database accesses with the aid of the WinCC OleDbProvider.

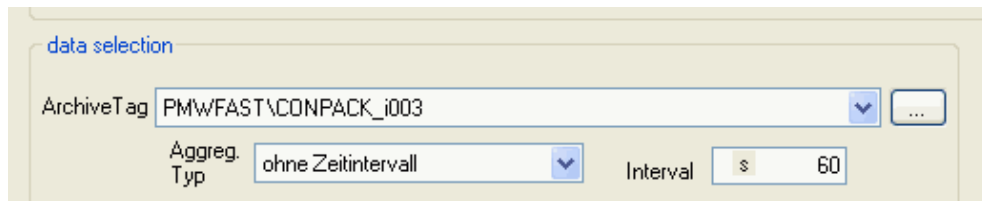
	<ul style="list-style-type: none"> When starting the program, the default input for this box is the value "WinCCcopack"
--	--

3.1.3 "Data selection" group

Type: GroupBox

Name: grpDataSelection

Figure 3-4



The objects of this group are only used to parameterize the accesses to the **Tag Logging** Runtime data. In this group, an available archive tag of the WinCC Tag Logging can be selected and special parameters for data reduction can be specified.

Screen objects (object name)	Description
<p>"Archive Tag" drop-down list Type: ComboBox Name: cmbTags</p>	<p>The "Archive Tag" drop-down list includes the archive tags configured in the Runtime database. By clicking, you can open the drop-down list and select an archive tag whose values are to be read from the Runtime database.</p> <p>Notes:</p> <ul style="list-style-type: none"> Before the actual data query the archive tag ID is transferred to the WinCC OleDbProvider instead of the archive tag name for performance reasons. When starting the program, the first available archive tag is selected in the drop-down list.

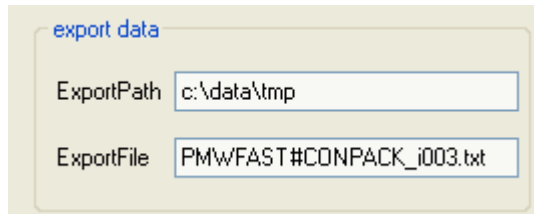
Screen objects (object name)	Description
<p>“Aggreg. Typ” button Type: ComboBox Name: cmbInterpol</p>	<p>The “Aggreg. Typ” drop-down list includes the aggregation types that are supported by the WinCC Connectivity Pack. You can select an aggregation type to combine (compress) several successive archive values of the Tag Logging Runtime in the specified time interval during the data query.</p> <p>When starting the program, the value “Without Aggreg.” is entered in the drop-down list. When this value is selected, the values are not combined with the WinCC OleDbProvider during the query.</p>
<p>“Interval” input box Type: TextBox Name: txtStep</p>	<p>In this input box, you enter the time interval in seconds in which the values are combined (compressed). The value entered in this box is only of importance if a value not equal to “Without Aggreg.” is selected in the “Aggreg. Typ” drop-down list.</p> <p>When starting the program, the default input for this box is the value 60 (seconds).</p>

3.1.4 “Export data” group

Type: GroupBox

Name: grpExport

Figure 3-5



The objects of this group are used to configure the export of the read **Tag Logging** Runtime data to a **CSV file**.

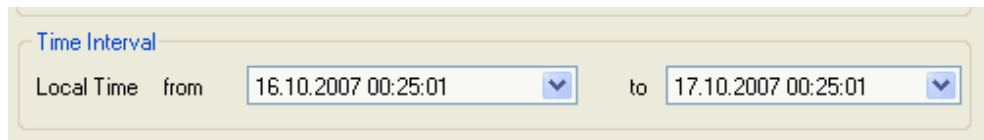
Screen objects (object name)	Description
<p>“ExportPath” input box Type: TextBox Name: txtExportPath</p>	<p>In this input box, enter the directory in which the CSV file with the Tag Logging Runtime data is to be created.</p> <p>Note: Please note that the path entered here must exist, since otherwise no CSV file is created. Missing directories are not automatically generated by the program.</p> <p>When starting the program, the default input for this box is the following path: “C:\data\tmp”.</p>
<p>“ExportFile” input box Type: TextBox Name: txtExportFile</p>	<p>Enter the file name of the CSV file.</p> <p>When selecting an archive tag or when starting the program, the default input value assigned to this box is the name of the selected archive tag followed by the “txt” string.</p> <p>“<ARCHIV_TAGNAME>.txt”</p>

3.1.5 “Time Interval” group

Type: GroupBox

Name: grpTimeInterval

Figure 3-6



The objects of this group are used to specify a time interval that is used as a filter criterion for the query of the **Tag and Alarm Logging** Runtime data.

Notes:

- The time is specified in local time-of-day format. When querying the data, the time interval specified here is converted to UTC time and then transferred to the WinCC OleDbProvider as a filter criterion.
- When querying the user archives, the time interval is not used.
- When starting the program or when changing the data source by selecting the menu items in the menu bar, the time interval is set to the last hour.

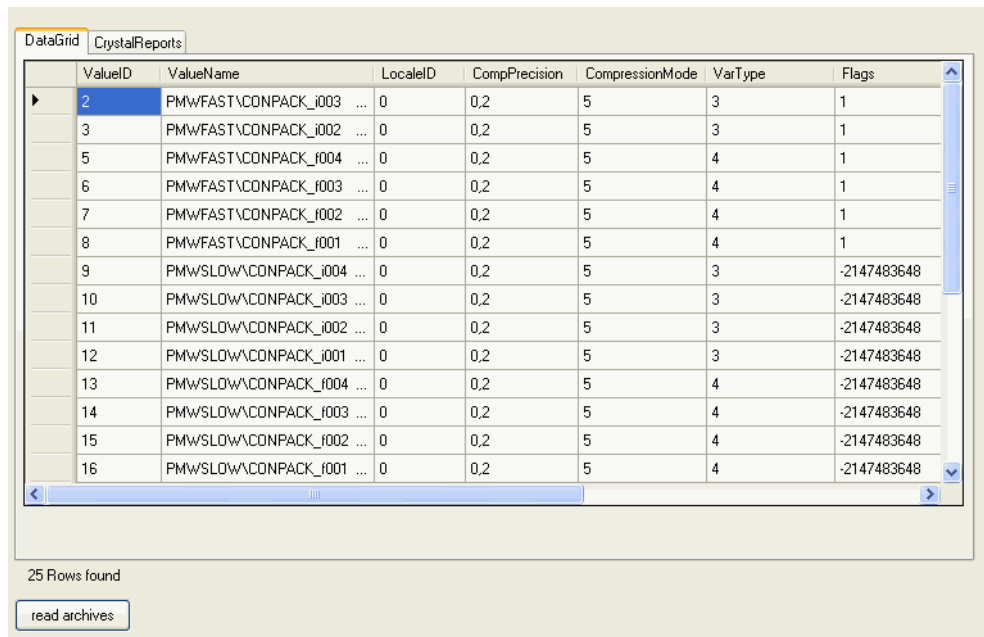
Screen objects (object name)	Description
<p>“Local Time from” DateTime list box Type: DateTimePicker Name: dtpFrom</p>	<p>With this list box you specify the start time for the time interval of the query.</p>
<p>“Local Time to” DateTime list box Type: DateTimePicker Name: dtpTo</p>	<p>With this list box you specify the end time for the time interval of the query.</p>

3.1.6 Tab for the Runtime data display

Type: TabControl

Name: tabView

Figure 3-7



The result of the data query is displayed in a tab. The **“DataGrid”** tab is available for a tabular display and the **“CrystalReports”** tab is available for a formatted display (e.g., printable version).

Screen objects (object name)	Description
“DataGrid” tab Type: tabPage Name: tabPageDataGrid <i>includes</i> DataGrid control element Type: DataGridView Name: myGrid	The “DataGrid” control element is used for tabular display of the data.

Screen objects (object name)	Description
<p>“CrystalReports” tab Type: TabPage Name: tabPageCrystalReports</p> <p><i>includes</i></p> <p>Report control element Type: CrystalReportViewer Name: crystalReportViewer1</p>	<p>The CrystalReportViewer is used for formatted display of the data.</p>
<p>“Rows found” display text Type: Label Name: lblAnz</p>	<p>This display text indicates the number of result data records supplied by the database request.</p>
<p>“read archives” button Type: Button Name: btnRead</p>	<p>By selecting the “read archives” button, the database query is performed. The supplied data is shown in tabular or formatted form.</p> <p>When displaying Tag Logging Runtime data, the CSV file is written. If the CSV file already exists, it is overwritten.</p>

3.2 Reading out, displaying and exporting WinCC process value archives

The following figures show how the Tag Logging Runtime data is displayed in the “DataGrid” control element or in the Crystal Reports Viewer and output to a CSV file.

To display the archived values of a process tag, proceed as follows:

- Select the “Tag Logging” menu item.
- In the “data selection” group, use the “...” button to select an archive tag.
- Select the time interval.
- Select the “read archives” button.

The figure below shows the tabular display of the Tag Logging archive data.

Figure 3-3

appCpack (Form1)

Tag Logging | Alarm Logging | UserArchives

Connection

Source: BATCH702\WINCC Provider: WinCCOLEDBProvider.1

Catalog: CC_OS_1_07_10_07_23_49_08R Uid: WinCCcopack pwd: *****

data selection

ArchiveTag: PMWFAST#CONPACK_I001

Aggreg. Typ: ohne Zeitintervall Interval: 60

export data

ExportPath: c:\data\tmp

ExportFile: PMWFAST#CONPACK_I001.txt

Time Interval

Local Time from: 13.10.2007 22:52:39 to: 17.10.2007 22:52:39

DataGrid | CrystalReports

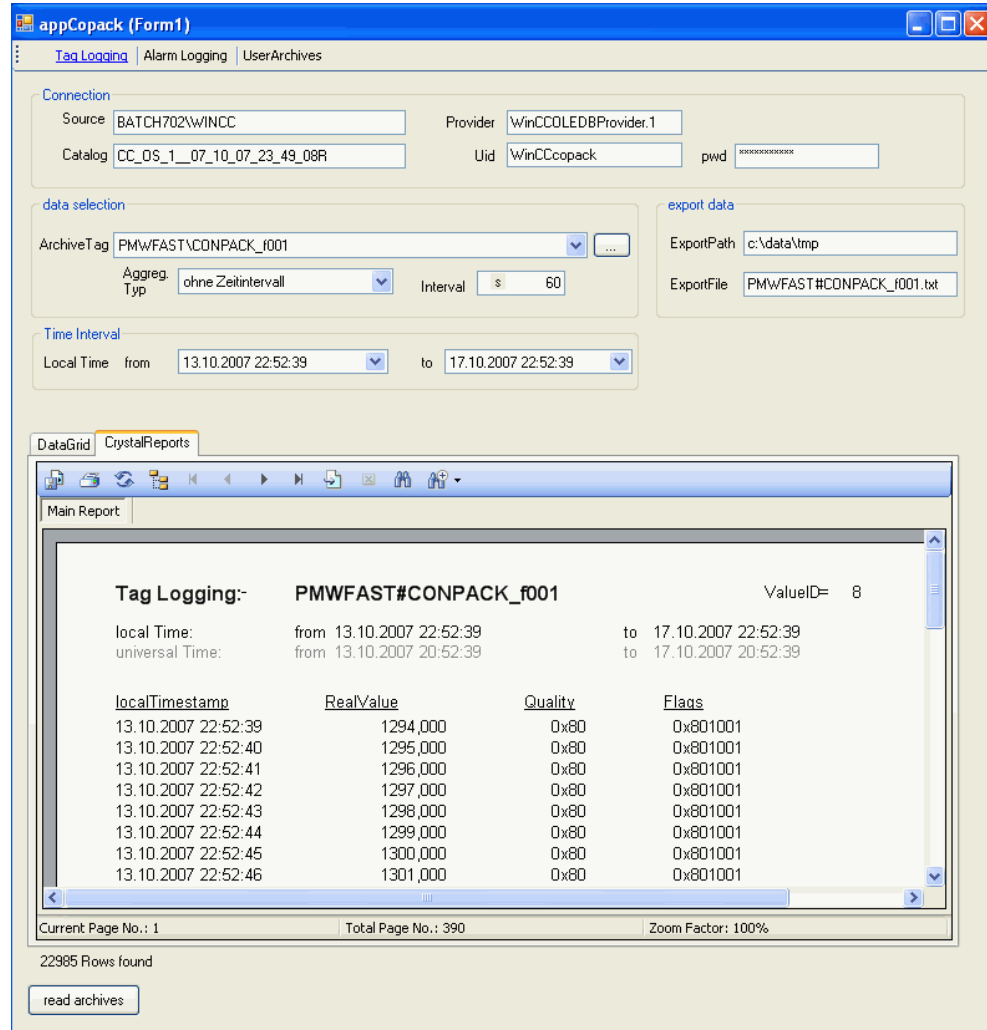
ValueID	ValueName	localTimestamp	RealValue	Quality	Flags
8	PMWFAST#CONPACK_I001	13.10.2007 22:52:39	1294,000	0x80	0x801001
8	PMWFAST#CONPACK_I001	13.10.2007 22:52:40	1295,000	0x80	0x801001
8	PMWFAST#CONPACK_I001	13.10.2007 22:52:41	1296,000	0x80	0x801001
8	PMWFAST#CONPACK_I001	13.10.2007 22:52:42	1297,000	0x80	0x801001
8	PMWFAST#CONPACK_I001	13.10.2007 22:52:43	1298,000	0x80	0x801001
8	PMWFAST#CONPACK_I001	13.10.2007 22:52:44	1299,000	0x80	0x801001
8	PMWFAST#CONPACK_I001	13.10.2007 22:52:45	1300,000	0x80	0x801001
8	PMWFAST#CONPACK_I001	13.10.2007 22:52:46	1301,000	0x80	0x801001
8	PMWFAST#CONPACK_I001	13.10.2007 22:52:47	1302,000	0x80	0x801001
8	PMWFAST#CONPACK_I001	13.10.2007 22:52:48	1303,000	0x80	0x801001
8	PMWFAST#CONPACK_I001	13.10.2007 22:52:49	1304,000	0x80	0x801001
8	PMWFAST#CONPACK_I001	13.10.2007 22:52:50	1305,000	0x80	0x801001
8	PMWFAST#CONPACK_I001	13.10.2007 22:52:51	1306,000	0x80	0x801001
8	PMWFAST#CONPACK_I001	13.10.2007 22:52:52	1307,000	0x80	0x801001

22985 Rows found

read archives

The following figure shows the formatted display of the Tag Logging archive data in the Crystal Reports Viewer.

Figure 3-4



The figure below shows an excerpt of the CSV file with the exported Tag Logging archive data.

Figure 3-5

```

strExportFile=PMWFAST#CONPACK_f001.txt
Provider=winCCOLEDBProvider.1; Data Source =BATCH702\WINCC; Catalog = CC_OS_1__07_10_07_23_49_08R;
mySelectQuery="TAG:R,8,'2007-10-13 20:52:39','2007-10-17 20:52:39'"
LocalTimestamp; RealValue; Quality; Flags
13.10.2007 22:52:39; 1294,000; 0x80; 0x801001
13.10.2007 22:52:40; 1295,000; 0x80; 0x801001
13.10.2007 22:52:41; 1296,000; 0x80; 0x801001
13.10.2007 22:52:42; 1297,000; 0x80; 0x801001
13.10.2007 22:52:43; 1298,000; 0x80; 0x801001
13.10.2007 22:52:44; 1299,000; 0x80; 0x801001
13.10.2007 22:52:45; 1300,000; 0x80; 0x801001
13.10.2007 22:52:46; 1301,000; 0x80; 0x801001
13.10.2007 22:52:47; 1302,000; 0x80; 0x801001
13.10.2007 22:52:48; 1303,000; 0x80; 0x801001
13.10.2007 22:52:49; 1304,000; 0x80; 0x801001
13.10.2007 22:52:50; 1305,000; 0x80; 0x801001
13.10.2007 22:52:51; 1306,000; 0x80; 0x801001
13.10.2007 22:52:52; 1307,000; 0x80; 0x801001
13.10.2007 22:52:53; 1308,000; 0x80; 0x801001
13.10.2007 22:52:54; 1309,000; 0x80; 0x801001
13.10.2007 22:52:55; 1310,000; 0x80; 0x801001
13.10.2007 22:52:56; 1311,000; 0x80; 0x801001
13.10.2007 22:52:57; 1312,000; 0x80; 0x801001
13.10.2007 22:52:58; 1313,000; 0x80; 0x801001
13.10.2007 22:52:59; 1314,000; 0x80; 0x801001
13.10.2007 22:53:00; 1315,000; 0x80; 0x801001
13.10.2007 22:53:01; 1316,000; 0x80; 0x801001
13.10.2007 22:53:02; 1317,000; 0x80; 0x801001
13.10.2007 22:53:03; 1318,000; 0x80; 0x801001
13.10.2007 22:53:04; 1319,000; 0x80; 0x801001
13.10.2007 22:53:05; 1320,000; 0x80; 0x801001
13.10.2007 22:53:06; 1321,000; 0x80; 0x801001
13.10.2007 22:53:07; 1322,000; 0x80; 0x801001
13.10.2007 22:53:08; 1323,000; 0x80; 0x801001
13.10.2007 22:53:09; 1324,000; 0x80; 0x801001
13.10.2007 22:53:10; 1325,000; 0x80; 0x801001
13.10.2007 22:53:11; 1326,000; 0x80; 0x801001
13.10.2007 22:53:12; 1327,000; 0x80; 0x801001
13.10.2007 22:53:13; 1328,000; 0x80; 0x801001
13.10.2007 22:53:14; 1329,000; 0x80; 0x801001
13.10.2007 22:53:15; 1330,000; 0x80; 0x801001
13.10.2007 22:53:16; 1331,000; 0x80; 0x801001
13.10.2007 22:53:17; 1332,000; 0x80; 0x801001
13.10.2007 22:53:18; 1333,000; 0x80; 0x801001
13.10.2007 22:53:19; 1334,000; 0x80; 0x801001
13.10.2007 22:53:20; 1335,000; 0x80; 0x801001
13.10.2007 22:53:21; 1336,000; 0x80; 0x801001
13.10.2007 22:53:22; 1337,000; 0x80; 0x801001
13.10.2007 22:53:23; 1338,000; 0x80; 0x801001
    
```

Copyright © Siemens AG 2007 All rights reserved
WinCC_CopackCsharp_en.doc

Notes:

- By default, the WinCC OleDbProvider supplies the data with a time stamp in UTC format. To display the data, this time stamp is converted to local time.
- The values “Quality” and “Flags” are displayed as hexadecimal values.

3.3 Reading out and displaying WinCC message archive

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

To display the archived alarms and messages, proceed as follows:

- Select the “Alarm Logging” menu item.

- Select the time interval.
- Select the “read archives” button.

The figure below shows the tabular display of the Alarm Logging archive data.

Figure 3-6

The screenshot shows the 'appCopack (Form1)' application window. It has three tabs: 'Tag Logging', 'Alarm Logging', and 'UserArchives'. The 'Alarm Logging' tab is active.

Connection Section:

- Source: BATCH702\WINCC
- Provider: WinCCOLEDBProvider.1
- Catalog: CC_DS_1_07_10_07_23_49_08R
- Uid: WinCCcopack
- pwd: [redacted]

data selection Section:

- ArchiveTag: PMWFAST\CONPACK_f001
- Aggreg. Typ: ohne Zeitintervall
- Interval: s 60

export data Section:

- ExportPath: c:\data\lmp
- ExportFile: PMWFAST#CONPACK_f001.txt

Time Interval Section:

- Local Time from: 13.10.2007 22:57:06
- to: 14.10.2007 00:57:06

DataGrid Section:

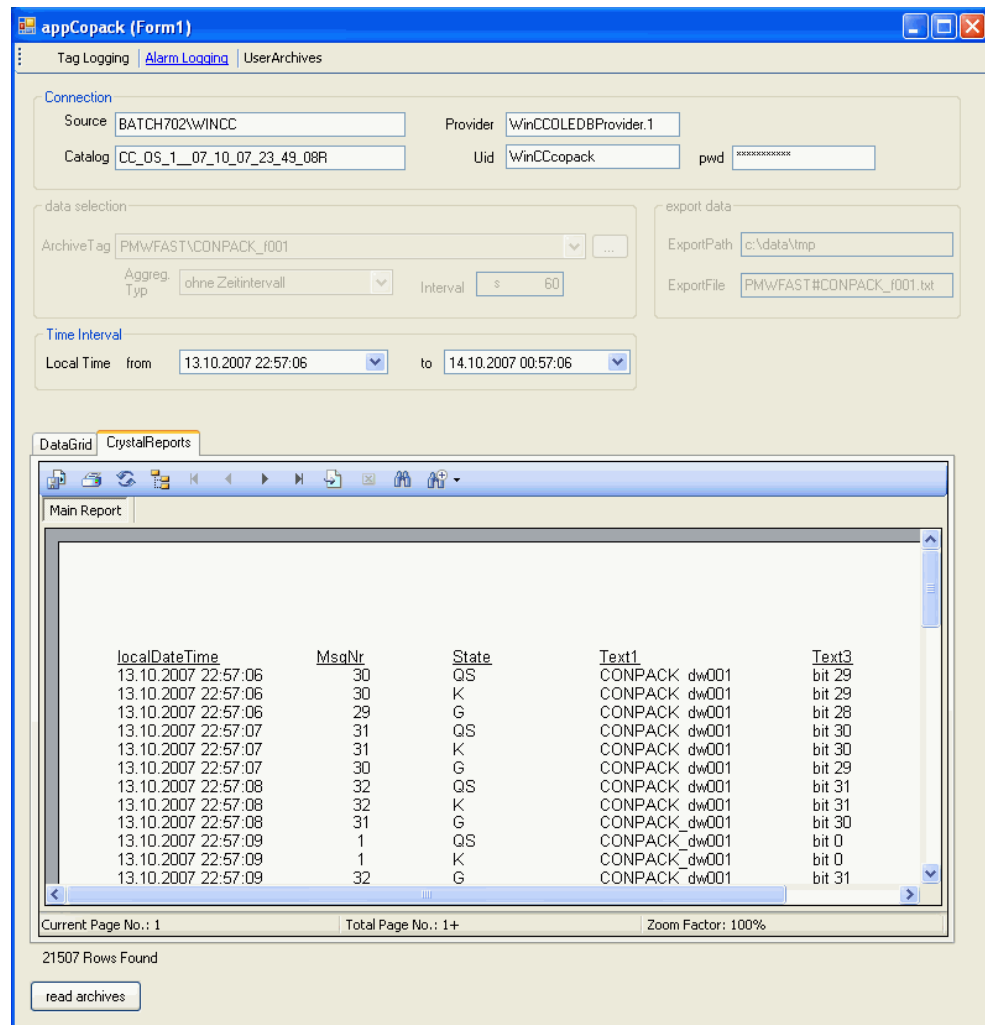
localDateTime	MsgNr	State	Text1	Text3	localDateTimeFrom	localDateTime
13.10.2007 22:5...	30	QS	CONPACK_dw001	bit 29	13.10.2007 22:5...	14.10.2007 00
13.10.2007 22:5...	30	K	CONPACK_dw001	bit 29	13.10.2007 22:5...	14.10.2007 00
13.10.2007 22:5...	29	G	CONPACK_dw001	bit 28	13.10.2007 22:5...	14.10.2007 00
13.10.2007 22:5...	31	QS	CONPACK_dw001	bit 30	13.10.2007 22:5...	14.10.2007 00
13.10.2007 22:5...	31	K	CONPACK_dw001	bit 30	13.10.2007 22:5...	14.10.2007 00
13.10.2007 22:5...	30	G	CONPACK_dw001	bit 29	13.10.2007 22:5...	14.10.2007 00
13.10.2007 22:5...	32	QS	CONPACK_dw001	bit 31	13.10.2007 22:5...	14.10.2007 00
13.10.2007 22:5...	32	K	CONPACK_dw001	bit 31	13.10.2007 22:5...	14.10.2007 00
13.10.2007 22:5...	31	G	CONPACK_dw001	bit 30	13.10.2007 22:5...	14.10.2007 00
13.10.2007 22:5...	1	QS	CONPACK_dw001	bit 0	13.10.2007 22:5...	14.10.2007 00
13.10.2007 22:5...	1	K	CONPACK_dw001	bit 0	13.10.2007 22:5...	14.10.2007 00
13.10.2007 22:5...	32	G	CONPACK_dw001	bit 31	13.10.2007 22:5...	14.10.2007 00
13.10.2007 22:5...	2	QS	CONPACK_dw001	bit 1	13.10.2007 22:5...	14.10.2007 00
13.10.2007 22:5...	2	K	CONPACK_dw001	bit 1	13.10.2007 22:5...	14.10.2007 00

21507 Rows Found

read archives

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

Figure 3-7



Notes:

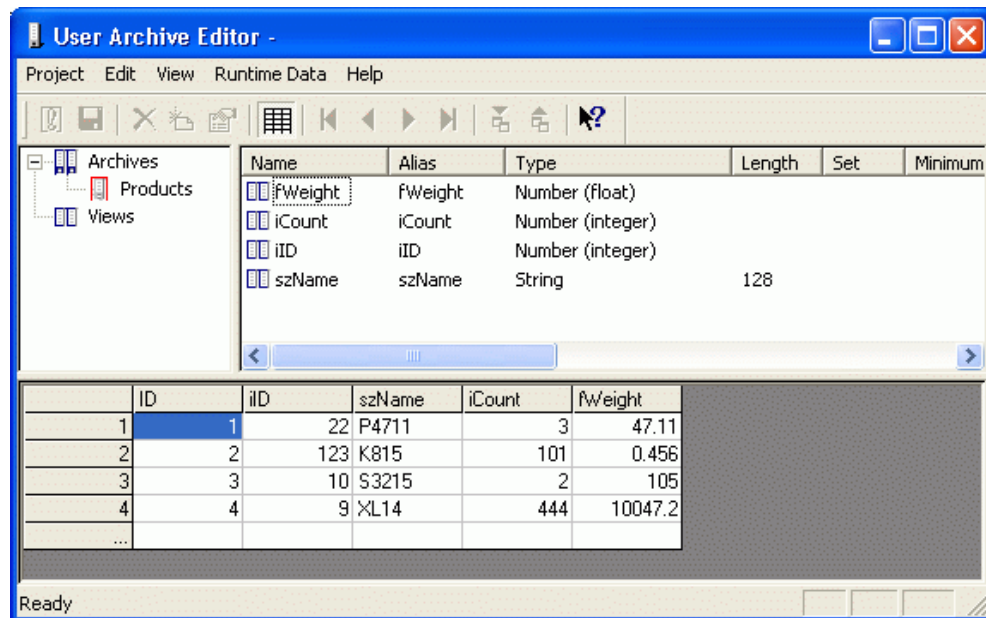
- By default, the WinCC OleDbProvider supplies the data with a time stamp in UTC format. To display the data, this time stamp is converted to local time.
- By default, the “State” message status is supplied as a decimal value by the WinCC OleDbProvider. The message status is converted to a string for display. The characters are used that are configured in the Alarm Logging Editor to form a message class.

3.4 Reading out and displaying WinCC User Archive

The following figures show how the Runtime data of a User Archive is displayed in the “DataGrid” control element or in the Crystal Reports Viewer.

This application displays the data of the “Products” User Archive. This requires that the User Archive is configured in the WinCC project as follows:

Figure 3-8



To display the data of the “Products” User Archive, proceed as follows:

- Select the “UserArchives” menu item.
- Use the “read archives” button.

The figure below shows the content of the “Products” User Archive in tabular form.

Figure 3-9

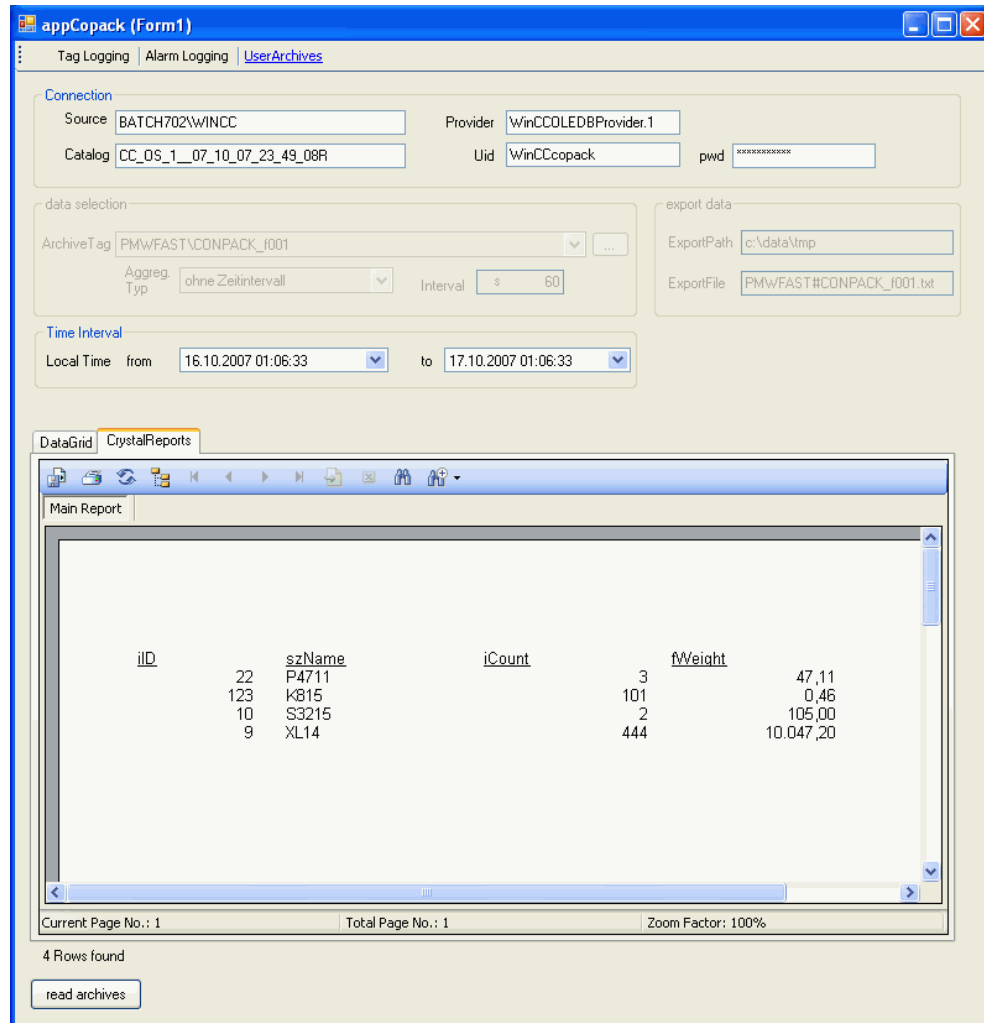
The screenshot shows the 'appCopack (Form1)' application window. It has a menu bar with 'Tag Logging', 'Alarm Logging', and 'UserArchives'. The 'UserArchives' tab is active. The interface is divided into several sections:

- Connection:** Source: BATCH702\WINCC, Provider: WinCCOLEDBProvider.1, Catalog: CC_05_1_07_10_07_23_49_08R, Uid: WinCCcopack, pwd: [redacted]
- data selection:** ArchiveTag: PMWFAST\CONPACK_f001, Aggreg. Typ: ohne Zeitintervall, Interval: s 60
- export data:** ExportPath: c:\data\tmp, ExportFile: PMWFAST#CONPACK_f001.txt
- Time Interval:** Local Time from: 16.10.2007 01:06:33 to: 17.10.2007 01:06:33
- DataGrid:** A table with 5 columns: iID, szName, iCount, fWeight, and an empty column. It contains 4 rows of data.
- CrystalReports:** A tab that is currently inactive.
- Footer:** 4 Rows found and a 'read archives' button.

iID	szName	iCount	fWeight	
22	P4711	3	47,11	
123	K815	101	0,456	
10	S3215	2	105	
9	XL14	444	10047,2	

The following figure shows the content of the "Products" User Archive in the Crystal Reports Viewer.

Figure 3-10



4 C# Program Code Description

This sample program was created with the “New Project” wizard in the Microsoft Visual Studio development environment. The “Windows Application” template available in Visual Studio was used. The runnable program was generated by the wizard. The generated program was then expanded with regard to the graphical screen objects (user interface) and the associated program code.

This section describes the program code to access the WinCC archive data.

4.1 C# code for evaluating the process value archives

4.1.1 Definition for the connection setup

The “myConnectionString” “string” variable is initialized with the necessary information for the connection setup to the archive database. The basic string structure for the connection setup is shown below:

```
string myConnectionString =  
    "Provider = WinCCOLEDBProvider.1; ///WinCC OleDbProvider  
    Data Source = <computer name>\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 setup.

4.1.2 Definition for the data selection

The “mySelectQuery” “string” variable is initialized with the necessary information for the actual SQL data query. The string structure for the data selection is shown 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=step size in seconds  
//Typ=compression type(e.g., AVG  
//for the average value)
```

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

Notes:

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

This sample program reads the data of only one archive tag. The values of the archive tag selected in the "cmbTags" drop-down list box are read.

- The archive tags are stored in the Runtime database with the universal time stamp (UTC). The query period has to be specified to the WinCCOLEDBProvider in universal time format (UTC) to ensure that the supplied data does not have a time offset to the local time. For this reason, the time stamps specified in the "dtpFrom" and "dtpTo" objects are converted from local time to universal time before they are used for the data query. The following program code shows the conversion of the time stamp to universal time format and the preparation of the time stamp for the data selection. The "tfrom" and "tto" "string" variables are used to set 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());
```

4.1.3 Setting up the connection to the database and reading data

The following program code shows the connection setup to 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, the data is read in via [OleDbDataAdapter](#). Alternatively, [OleDbDataReader](#) can also be used. This does not influence the actual transfer of the SQL query but the further data processing. By means of [OleDbDataAdapter](#) the information can be directly provided in the [DataGridView](#) without having to be concerned with the rows and columns.

4.1.4 Providing data for DataGrid and/or Crystal Report:

The sample program uses the [DataGridView](#) control element for tabular display of the data and the [CrystalReportViewer](#) for formatted output of the data. An object of the [DataTable](#) type is used as a data source for both displays. The “**myTableTags**” [DataTable](#) object is supplied with the read data of the SQL query by the [OleDbDataAdapter](#) by means of the “**Fill()**” method.

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

The data of the “**myTableTags**” DataTable are read row-by-row (data record-by-data record), prepared for display and written into another “**myTableTagsModify**” DataTable. The data of the modified “myTableTagsModify” DataTable are used for display.

The following program code shows the data preparation:

In this case, the structure (columns) of the “myTableTagsModify” DataTable has to be “manually” created. The following code shows the creation of the first three columns of the “myTableTagsModify” DataTable.

```
//=====
//
//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 the “filling” of the “myTableTagsModify” DataTable. Compared to the “myTableTags” DataTable, the DataTable is adapted as follows:

The time stamp is converted from universal time format (UTC) to local time format for display.

The value of the archive tag is displayed with 3 places after the decimal point.

The display of quality code and tag status is hexadecimal.

In this section, the “ValueName”, “localDateTimeFrom”, “localDateTimeTo”, “univDateTimeFrom” and “univDateTimeTo” columns are additionally created and supplied with values. These columns are 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;
```

4.1.5 Data connection of the DataGrid:

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

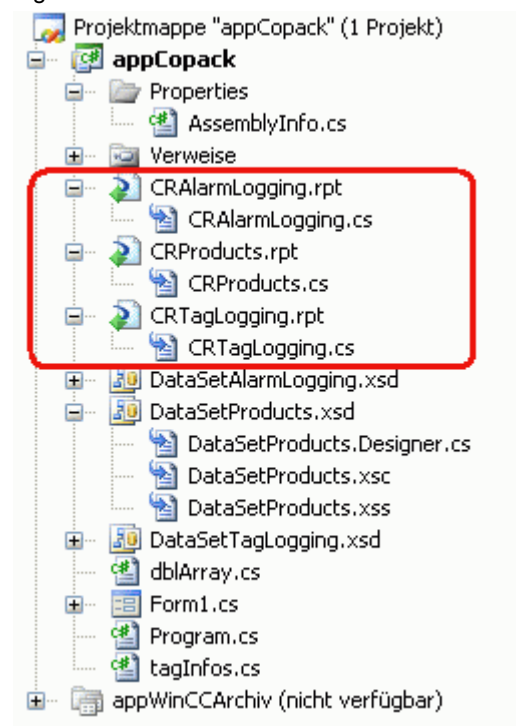
```
myGrid.DataSource = myTableTagsModify;
```


4.1.6 Data connection of the Crystal Report:

In this example, a separate report was generated for each report (Tag Logging, Alarm Logging and User Archives). For each report, Visual Studio creates a separate report class of the same name.

The figure below shows the reports created in this project and the existing report classes.

Figure 4-1



The data pool of the generated instance is connected to DataTable via [SetDataSource](#).

In our example, there is only one Crystal Reports Viewer. Which report it is to display is determined by the [.ReportSource](#) connection and the desired report instance, here [myDataReportAlarms](#).

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

4.1.7 Closing the connection to the archive

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

4.1.8 Exporting the archive values to a CSV file

The “myTableTagsModify“ DataTable is read row-by-row (data record-by-data record) and the content is written into 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();
```

4.2 C# code for evaluating the alarms and messages

4.2.1 Definition for the connection setup

Proceed as described in section 4.1.

4.2.2 Definition for the data selection

The “mySelectQuery” “string” variable is initialized with the necessary information for the actual SQL data query. The basic string structure for the data selection is shown 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 are stored in the Runtime database with the **universal time stamp (UTC)**. As in the data selection for the process archive values (see 4.1.2), the local time stamp is converted to the universal time stamp.

4.2.3 Setting up the connection to the database and reading data:

Proceed as described in section 4.1.1.

4.2.4 Providing data for DataGrid and/or Crystal Report:

Proceed as described in section 4.1.4.

The following program code shows how the status of a message can be displayed as a string (as in WinCC Alarm Control) instead of a numerical value. The numerical value of the status of a message is evaluated in a switch statement and the corresponding string is assigned in the different case branches. For information on the possible numerical values the “status” of a message can take, please refer to the entry [24842903](#) or to the WinCC Information System in “Working with WinCC > ANSI-C for Creating Functions and Actions > ANSI-C function descriptions > 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;
```

4.2.5 Data connection of the DataGrid:

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

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

4.2.6 Data connection of the Crystal Report:

Proceed as described in section 4.1.6.

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

4.2.7 Closing the connection to the archive

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

4.3 C# code for evaluating the User Archives

4.3.1 Definition for the connection setup

The “myConnectionString” “string” variable is initialized with the necessary information for the connection setup to the archive database. The basic string structure for the connection setup is shown below:

```
string myConnectionString =
    "Provider =SQLOLEDB; //Microsoft OleDBProvider
    Data Source = <Rechnername>\WINCC";
    uid = <username>
    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 setup.

4.3.2 Definition of the data selection

The “mySelectQuery” “string” variable is initialized with the necessary information for the actual SQL data query. The string structure for the data selection is shown below:

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

4.3.3 Setting up the connection to the database and reading data:

Proceed as described in section 4.1.1.

4.3.4 Providing data for DataGrid and/or Crystal Report:

Proceed as described in section 4.1.4.

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

4.3.5 Data connection of the DataGrid:

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

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

4.3.6 Data connection of the Crystal Report:

Proceed as described in section 4.1.6.

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

4.3.7 Closing the connection to the archive

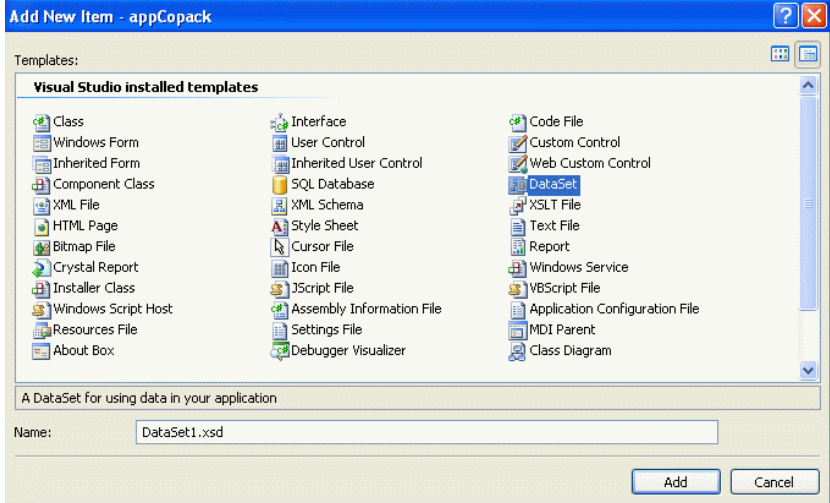
```
myConnection.Close();
```

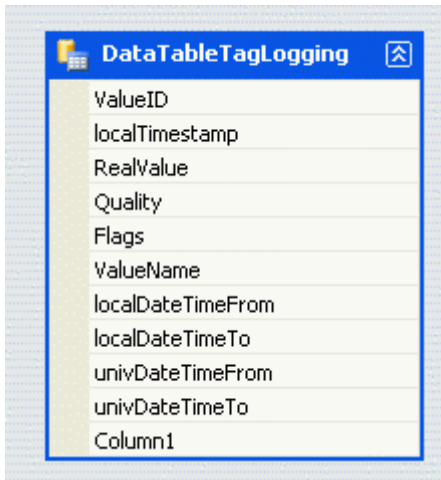
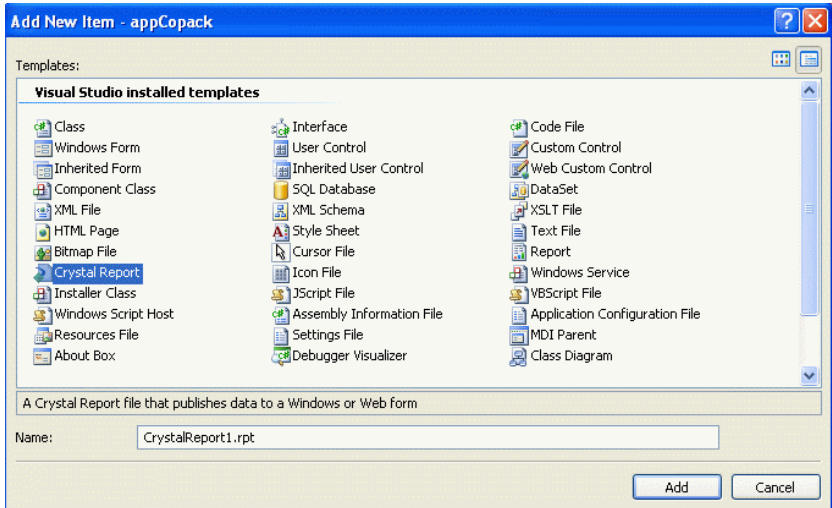
-

5 Creating a Report in Crystal Reports

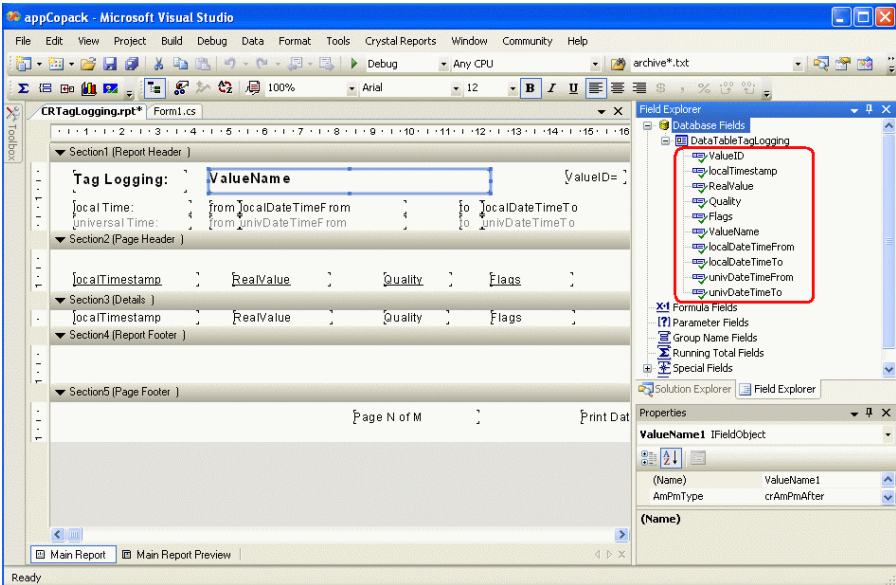
The following table describes how to create a Crystal Report in a form in MS Visual Studio 2005.

Table 5-1

No.	Procedure
1	<p>Adding DataSet</p> <p>In this step, you create the DataSet via which the read-in data is provided to the report:</p> <ul style="list-style-type: none"> In Solution Explorer, right-click the context menu of the project and select the “Add > New Item” menu command. A window with a list of templates opens. Select the “DataSet” item.  <p>Assign the final name.</p> <ul style="list-style-type: none"> Select the “Add” button to create the DataSet in the project.
2	<p>DataSet > Adding DataTable</p> <ul style="list-style-type: none"> In Solution Explorer, double-click the previously created DataSet. The Dataset Designer opens. Right-click in a “free” field within the Dataset Designer. The context menu opens. In this window, select the “Add > DataTable” menu item. <p>An empty DataTable is provided.</p> <ul style="list-style-type: none"> Adapt the name of the just created DataTable.

No.	Procedure
3	<p>DataSet > DataTable > Adding Columns</p> <ul style="list-style-type: none"> Use the right mouse button to select the DataTable header within the DataSet. A context menu opens. Select the “Add > Column” menu item. A new column is added. Adapt the column name according to the archive data to be read in later. Create an associated DataTable column of the DataSet for each column of a database table to be read in. 
4	<p>Inserting Crystal Report</p> <ul style="list-style-type: none"> In Solution Explorer, right-click the context menu of the project and select the “Add > New Item” menu command. A window with a list of templates opens. Select the “Crystal Report” item.  <ul style="list-style-type: none"> Assign the final name for the report. (This name is used for the report class generation.)

No.	Procedure
	<ul style="list-style-type: none"> Select the "Add" button to create the report in the project.
5	<p>Crystal Report with DataSet > Connecting DataTable</p> <ul style="list-style-type: none"> Opening report template Open a report in the Report Designer. To do this, you can double-click a "<Reportname>.rpt" report in Solution Explorer. Opening Database Expert Right-click in a free section of the report. The context menu opens. Select the "Database > Database Expert..." item. To open the Database Expert, you can also use the "Crystal Reports > Database > Database Expert..." menu item. The Database Expert opens. DataSet > Adding DataTable In the "Available Data Sources" list, the previously set up DataSet can be found in the "ADO.NET (XML)" item. Click the desired DataSet to show the included DataTable. Double-click the DataTable or use the ">" button to add the DataTable to the "Selected Tables" list. <div data-bbox="513 920 1350 1666" data-label="Image"> </div> <p>From this moment on, the table columns are available for use in the Crystal Reports Designer.</p>

No.	Procedure
<p>6</p>	<p>Configuring Crystal Reports with available Database Fields</p> <p>Click in a free section of the report. The context menu opens. Select the “Field Explorer” menu item. You can also use the “Crystal Reports > Field Explorer” menu command to open Field Explorer. In Field Explorer, the previously connected DataTable is listed with the previously set up columns. Using the mouse, you can now “drag” the columns to the report. A column that is used in the report is marked with a green checkmark in “Field Explorer > Database Fields”.</p> 
<p>7</p>	<p>Inserting CrystalReportViewer into a Form</p> <p>To use a Crystal Report in an application, insert the CrystalReportViewer into a form of your application. The CrystalReportViewer is available in the toolbox in “Crystal Reports > CrystalReportViewer”.</p> <p>This ensures that previewing, printing and exporting the report is possible in Runtime.</p> <p>Notes:</p> <ul style="list-style-type: none"> • The class of the report to be displayed is only assigned to the “.ReportSource” property of the inserted CrystalReportViewer object in Runtime. (see, for example, 4.1.6). • In the default setting, toolbar and status bar are not displayed. To display toolbar and status bar, set the “.DisplayStatusbar” and “.DisplayToolbar” properties to the value “true”. <p>The toolbar (menu bar in the top area) provides functions for exporting/printing/paging/zooming and searching. The status bar (display in the footer area) provides information on page and zoom factor.</p>

Copyright © Siemens AG 2007 All rights reserved
WinCC_CopackCsharp_en.doc