



Product:	UDP2SQLLogger
Product version:	v0.2
Document ID:	UM-UDP2SQLLogger
Doc revision:	PA2
Written/Apr.:	KD /
Date:	20. Feb. 2009

Industrial Control Design AS



UDP2SQLLogger v0.2

User Manual

The content of this document is confidential information not to be published without the consent of Industrial Control Design AS.

Industrial Control Design AS, www.icd.no, support@icd.no, forum.icd.no

Contents

1. INTRODUCTION.....	3
2. APPLICATION CONFIGURATION.....	4
2.1. About.....	4
2.2. How to add the UDP2SQLLogger component to a CDP Application.....	4
3. COMPONENT CONFIGURATION.....	5
3.1. About.....	5
3.2. UDPIOserver component configuration.....	5
3.2.1. IOConfig element.....	5
3.2.2. Packet element.....	5
3.2.3. NetworkInterface attribute.....	5
3.2.4. Signal element.....	6
3.2.5. Bit element.....	6
3.3. SQLLogger component configuration.....	6
3.3.1. SignalGroup element.....	6
3.3.2. SignalSource.....	6
3.3.3. Attributes.....	6
3.4. UDP2SQLLogger component configuration.....	6
4. DEMO APPLICATION.....	7
4.1. Running a demo.....	7
4.2. Sending UDP packets for testing.....	8
5. APPENDIX.....	9
5.1. Sample Configuration.....	9
5.1.1. UDPIOserver.xml.....	9
5.1.2. SQLLogger.xml.....	10
5.2. References.....	11

1. Introduction

This document describes how the UDP2SQLLogger CDP application works and how to configure and use it.

The UDP2SQLLogger is an example CDP application and component used for receiving UDP (User Datagram Protocol) packets containing data to be logged to an SQL database.

The UDP2SQLLogger application has the following features:

- Input UDP packets are defined by packet descriptions in the component XML file. Signals are created for each item in the packet and routing is done like for other CDP signals.
- The data in the packets consists of either standard CDP Signals, or registers where each bit may be mapped to either a bool Signal or a component state.
- Uses the UDPIOserver component for receiving UDP packets and extracting the Signal data.
- Uses the SQLLogger component from CDP2SQL to log the Signal data to an SQLite3 database.

This kind of configuration means that the source of the data we are interested in logging does not need to know about CDP. It only needs to feed out the signal data to log in a well-defined UDP packet sent to the UDP2SQLLogger application.

Component Diagram: Logging UDP to SQL

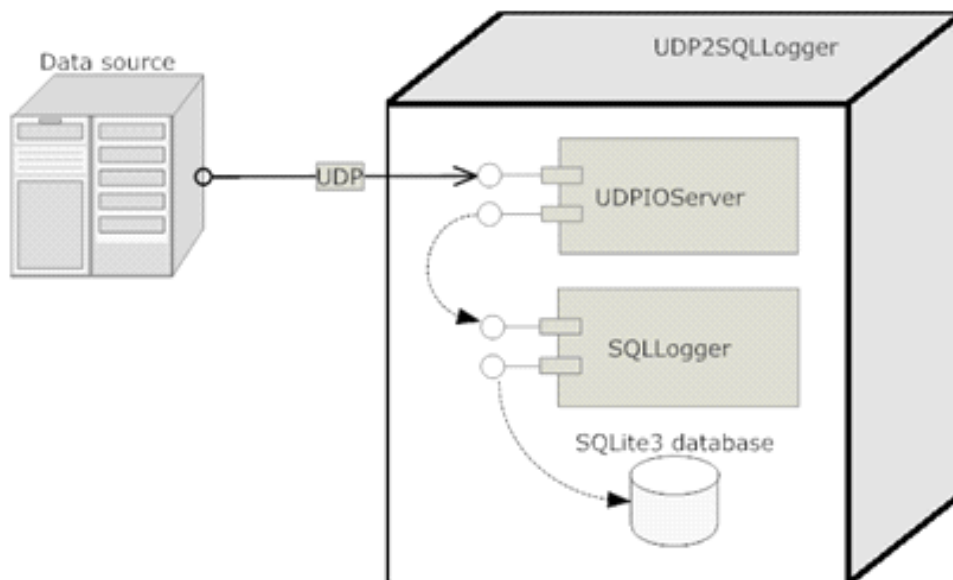


Illustration 1: Component Diagram for UDP2SQLLogger application

2. Application Configuration

2.1. About

This chapter describes how to use the UDP2SQLLogger component within a CDP application.

2.2. How to add the UDP2SQLLogger component to a CDP Application

Add the following to your project's Application.xml:

Inside the **<Components>** element, add an instance of an UDPIOserver component, for instance:

```
<Component Name="UDP2SQLLogger" Model="UDP2SQLLogger"></Component>  
<Component Name="UDPIOserver" Model="UDPIOserver"></Component>  
<Component Name="SQLLogger" Model="SQLLogger"></Component>
```

This will tell CDP to initialize three components named from component XML files located in “Components” folder.

3. Component Configuration

3.1. About

Configuration is done by modifying the component XML files. It should not be necessary to modify the model XML file(s). For this application, the relevant component XML files are:

- UDPIOServer.xml
- SQLLogger.xml
- UDP2SQLLogger.xml

These should all be found in `Application/Components` of the sample application.

3.2. UDPIOServer component configuration

Please refer to the UDPIOServer component User Manual for details. This document will only go into those settings that are needed for setting up logging.

3.2.1. IOConfig element

The IOConfig XML element goes inside the Component element and contains all the Packet configurations, both for out-going and incoming UDP packets. For logging purposes, we are only interested in incoming packets.

3.2.2. Packet element

The Packet XML element defines a UDP packet to be sent or received. The following attributes are relevant for our use of the **Packet** element:

Attribute	Description
LocalPort	May be specified for input packets to override the default port 2000.
NetworkInterface	Local ethernet interface to use. Must match an interface listed in <code>Application.xml</code> . (See note below)

Example XML:

```
<Packet Name="Incoming" LocalPort="2020" NetworkInterface="ETH0">
  <Signal Type="double" Name="Incoming_Double" NetworkConvert="1"></Signal>
</Packet>
```

3.2.3. NetworkInterface attribute

Note that the ETH0 interface is *handled specially* in Windows, meaning that it will use the default network interface selected by `CDPNetworkSetup`.

Also note that listening on 127.0.0.1 is not the same as listening on the computer's IP-address. To be able to listen to 127.0.0.1, create an additional network interface in `Application.xml` with this IP-address, like this:

```
<!-- NetworkInterfaces -->
<NetworkInterface Name="ETH0" IPAddress="127.0.0.1" SubnetMask="255.255.255.0"></NetworkInterface>
<NetworkInterface Name="ETH1" IPAddress="127.0.0.1" SubnetMask="255.255.255.0"></NetworkInterface>
```

This is necessary since under windows, ETH0 will be changed from 127.0.0.1 to the IP-address chosen by `CDPNetworkSetup`.

3.2.4. Signal element

One or more Signal elements go inside the Packet element to make up the data that is received using UDP.

The following attributes are special for the **Signal** element in this context:

Attribute	Description
NetworkConvert	May be set to indicate that the data should be converted to/from network format (big-endian format). The default is to not convert.
Type	The number type of the signal. Valid values include: double, float, int, short, unsigned int, unsigned short, unsigned char.

Example XML:

```
<Signal Type="double" Name="Generator" Routing="Sinus.Output" NetworkConvert="1"></Signal>
```

3.2.5. Bit element

The **Bit** element is used to specify *registers*, or collection of bits. It goes inside the **Signal** element, where Bit elements define bool signals for each bit. (A bit behaves like a Signal of type bool).

Example XML:

```
<Signal Type="unsigned short" Name="Status" NetworkConvert="1">
  <Bit Nr="0" Name="HPU Running" Component="HPU" State="Running"></Bit>
  <Bit Nr="1" Name="Pump1 Running mode" Component="HPU.Pump1" State="Running"></Bit>
  <Bit Nr="2" Name="Pump2 Running mode" Component="HPU.Pump2" State="Running"></Bit>
  <Bit Nr="3" Name="Pump3 Running mode" Component="HPU.Pump3" State="Running"></Bit>
</Signal>
```

3.3. SQLLogger component configuration

Please refer to the CDP2SQL User Manual for details. This document will only go into those settings that are needed for setting up logging with UDP.

3.3.1. SignalGroup element

This is where we add all the signals that we wish to log. Example XML:

```
<SignalGroup>
  <SignalSource Name="DoubleSig" Type="double" Routing="UDPIOserver.Incoming_Double"></SignalSource>
  <SignalSource Name="IntSig" Type="int" Routing="UDPIOserver.Incoming_Int"></SignalSource>
</SignalGroup>
```

3.3.2. SignalSource

A signal to be logged as part of a SignalGroup. Example XML:

```
<SignalSource Name="DoubleSig" Type="double" Routing="UDPIOserver.Incoming_Double"></SignalSource>
```

3.3.3. Attributes

Attribute Name	Description
Name	The signal name, which will also be used as column name in the signal logging table.
Type	Signal data type that will get converted to valid SQL before creating the table. Valid options include 'double', 'int', and 'bool'.
Routing	The actual signal that we want to log. We'll want to use this to route the signal data from the UDPIOserver component.

3.4. UDP2SQLLogger component configuration

The UDP2SQLLogger component currently has no modifiable configuration. Defaults should suffice.

4. Demo Application

4.1. Running a demo

To run the demo application, enter the 'Application' directory and run the 'CDP.exe' binary. This will start the application with the default example configuration.

```

D:\Work\Applications\Loggers\UDP2SQLLogger\Application\CDP.exe
Binary image build date: Feb 12 2009 11:51:09

CDP Version 2.3.1.6.

Libraries included in build:
UDP2SQLib build date: Thu Feb 12 11:50:02 2009
UDPIOserverLib build date: Wed Oct 29 10:47:36 2008 Version 1.6
CDP build date: Thu Jan 22 16:06:57 2009 Version 2.3.1.6
UDP2SQLLib build date: Mon Jan 26 12:55:22 2009 Version 0.2
* Thu Feb 12 2009 *
11:51:43.741 WARNING: This is a DEMO ONLY! Functional limitations apply!
11:51:43.762 TestUDP2SQLLogger: WorkerTask started.
11:51:43.789 UDPPacket::ReceiveThread(): Listening on 10.0.2.218:2020 for UDP p
acket 'PacketIn'.
11:51:43.790 WebServer: Starting WebServer on 0.0.0.0:80
11:51:43.806 Messenger is using ip-address 10.0.2.218:2040 (subnetmask 255.255.
255.0).
    
```

Goto the web interface of the CDP application in a web browser (IE or Firefox) or at <http://localhost:80/> or use CDPBrowser. Click your way into the SQLLogger component. Set the 'LogControl' signal to '1' to start logging. Notice that the row count 'SignalTableRows' increases.

Signal name	Value	Unit	Routing	Description
Inputs				
LogControl	1		No routing	Start/stop logging (also sets log frequency if enabled in.xml).
Outputs				
Process Timer	3.631746e-3	s/s	No routing	Process run time each s.
Process Period	0.100733	s	No routing	Process interval [s].
BufferedEvents	0	Count	No routing	The number of samples in the event buffer.
BufferedSignals	0	Count	No routing	The number of samples in the signal buffer.
EventTableRows	0	Count	No routing	The number of samples in the event table.
SignalTableRows	17	Count	No routing	The number of samples in the signal table.
DoubleSig	0		UDP2SQLLogger.Incoming_Double	
IntSig	0		UDP2SQLLogger.Incoming_Int	
CharSig	0		UDP2SQLLogger.Incoming_UChar	
BinSig	0		UDP2SQLLogger.Incoming_Binary	
Bit0	0		UDP2SQLLogger.Inc_Bin_0	
Bit1	0		UDP2SQLLogger.Inc_Bin_1	
Bit2	0		UDP2SQLLogger.Inc_Bin_2	
Bit3	0		UDP2SQLLogger.Inc_Bin_3	

However, since the component is not yet receiving any UDP packets, all the logged signals have zero values. We'll need some way of sending UDP packets to it.

4.2. Sending UDP packets for testing

The benefit of using UDPIOserver is that there is no dependency on CDP for the system producing the data we want to log. As such, there is an example Ruby script for sending a handful of UDP packets to the UDPIOserver.

Open a console, go into the demo application directory and run:
`ruby test_send_udp.rb [your IP address]`

Note that you cannot use localhost IP address 127.0.0.1, as CDP will by default bind to the configured network adaptor when specifying localhost. (See NetworkInterface attribute above for details.)

```

C:\WINDOWS\system32\cmd.exe
D:\Work\Applications\Loggers\UDP2SQLLogger\Application>ruby send_test_udp.rb 10.0.2.218
0. Sending ...
1. Sending ...
2. Sending ...
3. Sending ...
4. Sending ...
5. Sending ...
6. Sending ...
7. Sending ...
8. Sending ...
9. Sending ...
Done.
D:\Work\Applications\Loggers\UDP2SQLLogger\Application>
    
```

You should then see the Signal values change in both UDPIOserver component and SQLLogger. The DoubleSig signal should approach PI as ten packets are sent. Later you can also inspect the `cdp2sql.db` file using an SQLite3 client to see that the data values were logged.

Signal name	Value	Unit	Routing	Description
Inputs				
LogControl	1		No routing	Start/stop logging (also sets log frequency if enabled in xml).
Outputs				
Process Timer	3.073016e-7	s/s	No routing	Process run time each s.
Process Period	0.100780	s	No routing	Process interval [s].
BufferedEvents	0	Count	No routing	The number of samples in the event buffer.
BufferedSignals	0	Count	No routing	The number of samples in the signal buffer.
EventTableRows	0	Count	No routing	The number of samples in the event table.
SignalTableRows	231	Count	No routing	The number of samples in the signal table.
DoubleSig	3.141592		UDPIOserver.Incoming_Double	The input signals
IntSig	1111104		UDPIOserver.Incoming_Int	
CharSig	82		UDPIOserver.Incoming_UChar	
BinSig	3		UDPIOserver.Incoming_Binary	
Bit0	1	<input checked="" type="checkbox"/>	UDPIOserver.Inc_Bin_0	
Bit1	0	<input type="checkbox"/>	UDPIOserver.Inc_Bin_1	
Bit2	0	<input type="checkbox"/>	UDPIOserver.Inc_Bin_2	
Bit3	1	<input checked="" type="checkbox"/>	UDPIOserver.Inc_Bin_3	

Alternatively you can also set up a separate CDP application with a UDPIOserver configured for sending UDP packets. Refer to the UDPIOserver User Manual for details.

5. Appendix

5.1. Sample Configuration

5.1.1. UDPIOServer.xml

```
<?xml version="1.0" encoding="iso-8859-1"?>

<Component Name="UDPIOServer" Model="UDPIOServer">
  <OverrideState>Online</OverrideState>
  <Debug>1</Debug>
  <Activate>1</Activate>
  <fs>1</fs>

  <Description>
    <![CDATA[Receive preconfigured UDP packets.]]>
  </Description>

  <IOConfig>
    <!--
    * Listen on port 2020 for UDP packets with data to log.
    * The data is, in order:
    * - double,
    * - unsigned int,
    * - unsigned char and
    * - four bits wrapped in an unsigned char.
    -->
    <Packet Name="PacketIn" LocalPort="2020" NetworkInterface="ETH0" >
      <Signal Type="double" Name="Incoming_Double" Description="Incoming" NetworkConvert="1"></Signal>
      <Signal Type="unsigned int" Name="Incoming_Int" Description="Incoming" NetworkConvert="1"></Signal>
      <Signal Type="unsigned char" Name="Incoming_UChar" Description="Incoming" NetworkConvert="1"></Signal>
      <Signal Type="unsigned char" Name="Incoming_Binary" Description="Bits" NetworkConvert="1">
        <Bit Nr="0" Name="Inc_Bin_0" Description="Binary digit."></Bit>
        <Bit Nr="1" Name="Inc_Bin_1" Description="Binary digit."></Bit>
        <Bit Nr="2" Name="Inc_Bin_2" Description="Binary digit."></Bit>
        <Bit Nr="3" Name="Inc_Bin_3" Description="Binary digit."></Bit>
      </Signal>
    </Packet>
  </IOConfig>

  <Signals>
    <Signal Name="Send-Receive Roundtrip time" Unit="s" Type="double" Description="The time needed to perform one send and receive."></Signal>
    <Signal Name="Time" Unit="s" Type="double" Description="The relative time."></Signal>
  </Signals>

  <Alarms>
    <Alarm Name="Transmission Error" Level="Warning" Enabled="1" Text="Didn't receive all packets in time.." Unacknowledged="0"></Alarm>
  </Alarms>

  <Parameters>
    <Parma Name="SignalTimeout" Value="1.1" DefaultValue="1" PreviousValue="0.1" TimeLastChanged="Sun Nov 21 15:40:06 2004" Description="Timeout before setting Transmission Error alarm."></Parma>
  </Parameters>

  <Subcomponents></Subcomponents>

  <RemoteComponents></RemoteComponents>

</Component>
```

5.1.2. SQLLogger.xml

```

<?xml version="1.0" encoding="iso-8859-1"?>
<Component Name="SQLLogger" Model="SQLLogger">
  <Activate>1</Activate>
  <fs>3</fs>
  <InitialState></InitialState>

  <Description><![CDATA[
    Component for logging CDP signals/events to a SQL database.
  ]]></Description>

  <Database DBName="cdp2sql.db" Type="sqlite3"></Database>

  <LogControl TimeFormat="0" TransactionRetry="5">
    <SignalLogging LogFs="10" LogControlFs="1">
      <BufferOptions BufferSize="10" BufferKeep="10" BufferFlush="10"></BufferOptions>
      <TableOptions MaxRows="999" OverflowMethod="0" OverwriteTable="1"></TableOptions>
      <BurstOptions BurstFs="10" BurstPeriodMs="2000"></BurstOptions>
    </SignalLogging>
    <EventLogging>
      <BufferOptions BufferSize="10" BufferKeep="10" BufferFlush="10"></BufferOptions>
      <TableOptions MaxRows="999" OverflowMethod="2" OverwriteTable="1"></TableOptions>
    </EventLogging>
    <QueryRequest>
      <BufferOptions BufferSize="1"></BufferOptions>
    </QueryRequest>
  </LogControl>

  <!-- We are routing in the values that UDPIOserver receives in UDP packets and preparing it for logging to
  SQL. -->
  <SignalGroup>
    <SignalSource Name="DoubleSig" Type="double" Routing="UDPIOserver.Incoming_Double"></SignalSource>
    <SignalSource Name="IntSig" Type="int" Routing="UDPIOserver.Incoming_Int"></SignalSource>
    <SignalSource Name="CharSig" Type="int" Routing="UDPIOserver.Incoming_UChar"></SignalSource>
    <SignalSource Name="BinSig" Type="int" Routing="UDPIOserver.Incoming_Binary"></SignalSource>
    <SignalSource Name="Bit0" Type="bool" Routing="UDPIOserver.Inc_Bin_0"></SignalSource>
    <SignalSource Name="Bit1" Type="bool" Routing="UDPIOserver.Inc_Bin_1"></SignalSource>
    <SignalSource Name="Bit2" Type="bool" Routing="UDPIOserver.Inc_Bin_2"></SignalSource>
    <SignalSource Name="Bit3" Type="bool" Routing="UDPIOserver.Inc_Bin_3"></SignalSource>
  </SignalGroup>

  <Signals>
    <Signal Name="LogControl" Input="1" Type="double" Unit="" Value="" Routing="No routing"
    Description="Start/stop logging (also sets log frequency if enabled in xml)."></Signal>
    <Signal Name="BufferedEvents" Input="0" Type="int" Unit="Count" Description="The number of samples in the
    event buffer."></Signal>
    <Signal Name="BufferedSignals" Input="0" Type="int" Unit="Count" Description="The number of samples in the
    signal buffer."></Signal>
    <Signal Name="EventTableRows" Input="0" Type="int" Unit="Count" Description="The number of samples in the
    event table."></Signal>
    <Signal Name="SignalTableRows" Input="0" Type="int" Unit="Count" Description="The number of samples in the
    signal table."></Signal>
  </Signals>

  <Alarms>
    <Alarm Name="EventTableOverflow" Group="" Level="Warning" Trig="0" Enabled="1" EnabledState="" Signal=""
    Inverted="0" SignalOutSet="" Text="The event table reached the maximum number of rows." Description="The event
    table reached the maximum number of rows."></Alarm>
    <Alarm Name="SignalTableOverflow" Group="" Level="Warning" Trig="0" Enabled="1" EnabledState="" Signal=""
    Inverted="0" SignalOutSet="" Text="The signal table reached the maximum number of rows." Description="The
    signal table reached the maximum number of rows."></Alarm>
  </Alarms>

  <Parameters>
    <Parma Name="EnableLogControl" Unit="" Value="1" Min="0" Max="1" DefaultValue="1" PreviousValue="0"
    TimeLastChanged="Thu Dec 04 12:29:26 2008" Description="When set, this parameter enables the LogControl signal.
    Thus, disabling control by message."></Parma>
    <Parma Name="EnableDeltaLogging" Unit="" Value="0" Min="0" Max="1" DefaultValue="0" PreviousValue="1"
    TimeLastChanged="Mon Jan 19 12:07:23 2009" Description="Enable logging based on change of value."></Parma>
  </Parameters>

  <Subcomponents></Subcomponents>

  <RemoteComponents></RemoteComponents>
</Component>

```

5.2. References

- Ruby scripting language
 - <http://www.ruby-lang.org/en/> - Ruby home page
 - <http://rubyinstaller.rubyforge.org/> – One-click installer for Ruby on Windows.
- Database GUI browsers
 - <http://sqlitebrowser.sourceforge.net/> - SQLite Database Browser.
 - <http://code.google.com/p/sqlite-manager/> - Firefox add-on 'SQLite Manager'