

**Citect for Windows  
Driver Specification  
Fidelix**

## Driver version history

Version	Date	Modified By	Details
1.0.0.B1	2013-03-05	Tomas Rook	Main development – no writes, no TimeTables
1.0.0.B2	2013-03-26	Tomas Rook	Writes implemented (still no TimeTables)
1.0.0.0	2013-04-22	Tomas Rook	TimeTables and Calendar implemented
1.0.0.1	2013-10-22	Tomas Rook	Temporary WriteToLogFile -> not released
1.0.0.2	2013-10-25	Tomas Rook	Increased max include 64 -> 256, changed memset in init, increased MaxChannel 64 -> 256 -> not released
1.0.3.0	2015-03-30	Tomas Rook	OID lookup changed to prevent variable mixup between include projects, new BuildVariableMap to work better in redundancy, PortFrom defaults to port 1235 and TCP. XTA/XVA range fixup (did not compile correct)
1.0.4.0	2015-03-31	Tomas Rook	Removed IOserver compare stuff, added debug for defined variables missing in the Fidelix device
1.0.4.1	2017-05-10	Tomas Rook	Softwareprotection implemented

## Driver documentation version history

Date	Modified By	Details
2013-03-05	Tomas Rook	Preliminary
2013-03-26	Tomas Rook	Statistics and parameters updated
2013-04-22	Tomas Rook	Parameters and Fidelix.dbf updated
2015-03-30	Tomas Rook	Parameters and Fidelix.dbf updated

## Contents

<b>1. QA</b>	<b>5</b>
1.1 Introduction	5
1.2 Procedure for generating a new driver	5
<b>2. TARGET DEVICE(S) AND PROTOCOL</b>	<b>6</b>
2.1 Introduction	6
2.2 Device Manufacturer	6
2.3 Device Definition	6
2.4 Communications Method	6
2.5 Communications/Hardware Configuration	6
2.6 Contacts	6
<b>3. PROTOCOL REQUIREMENTS</b>	<b>7</b>
3.1 Introduction	7
3.2 Pointlist	7
3.3 OID vs Index based addressing	7
3.4 FIDELIX.DBF	8
3.5 Supported variables/parameters	9
3.6 Address examples (OID and Index lookup)	11
3.7 Time channel addressing	13
<b>4. USER INTERFACE</b>	<b>14</b>
4.1 Introduction	14
4.2 Driver Name	14
4.3 Ports Form (TCPIP setup)	14
4.3.1 Baud Rate	14
4.3.2 Data Bits	14
4.3.3 Stop Bits	14
4.3.4 Parity	14
4.3.5 Special Opt	14
4.4 IO Devices Form	14
4.4.1 Protocol	14
4.4.2 Address	14
4.5 Pulldown lists Help	15
4.6 PROTDIR.DBF	15
4.7 Parameters and INI options	15
4.7.1 Standard Parameters	15

---

4.7.2	Driver Specific Parameters	15
<b>4.8</b>	<b>Debug Messages</b>	<b>16</b>
<b>4.9</b>	<b>Stats Special Counters</b>	<b>16</b>
<b>4.10</b>	<b>Hints and Tips</b>	<b>17</b>
<b>5.</b>	<b>BASIC TESTING</b>	<b>18</b>
5.1	Introduction	18
5.2	Procedure	18
<b>6.</b>	<b>PERFORMANCE TESTING</b>	<b>20</b>
6.1	Introduction	20
6.2	Calculating the Blocking Constant – Not applicable	20
<b>7.</b>	<b>REFERENCES</b>	<b>21</b>
7.1	References	21

# 1. QA

## 1.1 Introduction

This document follows the development of the new driver. It serves as a functional specification, design specification and test specification.

## 1.2 Procedure for generating a new driver

The following check list defines the QA steps for generating a new driver. This procedure must be followed for drivers to be integrated into Citect.

	Description	Person	Date
1	This specification document is written.	TR	13.03.05
2	Specification reviewed and accepted by R&D department.		n/a
3	Driver coded.	TR	13.03.05
4	Code and specification reviewed and accepted by R&D department.		n/a
5	Testing with connection project, and performance test.	TR	13.03.05
6	Driver integrated into Citect source and built.		n/a
7	Documentation is written (HLP or MVB files)		n/a
	<b>At this checkpoint coding is done and the driver is available as a beta.</b>		
8a	Full testing is carried out.	TR	13.04.23
8b	Performance testing is carried out.	TR	13.04.23
8c	Specification and documentation updated from testing/performance tests	TR	13.04.23
	<b>At this checkpoint the testing is complete.</b>		
9a	Review for completeness by developer, tester, documentor and R&D staff		n/a
9b	Add driver to install disks		n/a
9c	Add driver to protocols database		n/a
9d	Support notified of new driver for training purposes		n/a
10	Sales notified of new driver		n/a
	<b>The driver is now finished.</b>		

The hand over of a driver requires that all the above steps are completed and checked off.

## 2. Target Device(s) and Protocol

### 2.1 Introduction

This section defines the types of I/O Devices that are targeted by this driver.

### 2.2 Device Manufacturer

Fidelix Sverige AB  
 Box 62  
 647 05 Åkers Styckebruk

### 2.3 Device Definition

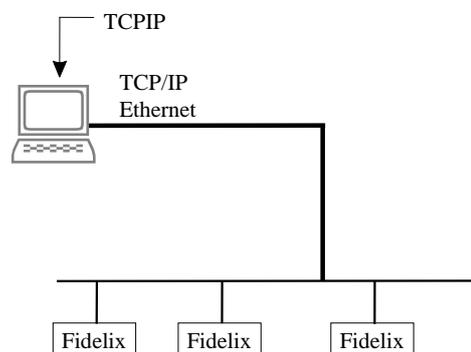
Different model of devices, ex FX-2025

### 2.4 Communications Method

TCP/IP only.

### 2.5 Communications/Hardware Configuration

The driver connects to the Fidelix devices using TCP/IP and subscribe for tag updates.



### 2.6 Contacts

Driver Development	TroSoft AB, Tomas Rook	+46 8 532 57262
Project Management	Fidelix Svenska AB, Antero Engberg	+46 8 556 558 33

## 3. Protocol Requirements

### 3.1 Introduction

The driver is a classic front-end/back-end cached driver using OID for variable lookup. It is also capable of index based variable lookup which will allow use in 'blackbox' projects (2013-03-05 not yet tested)

Driver was developed using DDK 5.5 and Citect 6.10 SpA, and has been tested with Citect 7.30. Later OS's will report driver is old and maybe not compatible. The driver will be recompiled for VS2010 when Citect release their new DDK.

### 3.2 Pointlist

When the driver DLL has loaded and communication takes place it will first of all read the Fidelix version and compare with the stored pointlist version. If it differs (or the pointlist file does not exist) it will be read. The pointlist is stored in Citect's data folder, where a new folder 'Fidelix' is created. The pointlist will be named as the I/O device in Citect communications folder, ex *C:\Documents and Settings\All Users\Application Data\Citect\CitectSCADA 7.10\Data\fidelix2.dat*. An alternative path could be used, see *Parameters* section.

### 3.3 OID vs Index based addressing

Fidelix devices only support text based addressing. Citect compiler is however not capable of handling text based addressing methods, hence the driver make use of direct variable database (*Variable.DBF*) address lookups. When the driver DLL is loaded it will scan through all variable tag databases in both the start project and all include projects. The unique OID key for all Fidelix variables will be stored in memory together with the variable address field. When Citect wants to read some variable, it sends the OID to the driver which look in its variable lists and hopefully finds the variable address.

To force the driver reading the pointlist again (if device configuration has been changed), the virtual digital tag *DEVICE.GETPOINTLIST* could be set to *TRUE*. The pointlist will always be read at startup if it does not exist or the parameter *ReadPointListAtStartup* is set to 1. The driver will automatically try to read the version of the Fidelix program and reload the variable list if it is changed.

If the pointlist file is not updated when configuration changes some variables could be missing causing address range error.

In some cases this is a method that will not function properly, ex if the variable tag databases does not exist locally (BlackBox project). To allow for these configurations the driver also support index based lookups. To enable index based addressing, the parameter *AllowBlackBox* should be set to 1. When enabled, the driver DLL will also create a text based cross-reference index file and put in the same folder as the pointlist file. Here are some variables in index file *C:\Documents and Settings\All Users\Application Data\Citect\CitectSCADA 7.10\Data\fidelix2.txt*:

```
...
1343=S03_LB01_GQ4X-MIN_MV.Value
1344=S03_LB01_GQ4X-MIN_MV.ValueAuto
1345=S03_LB01_GQ4X-MIN_MV.LockState
1346=S03_LB01_GQ4X-MIN_MV.Text
1347=S03_LB01_GQ4X-MIN_MV.Page
1348=S03_LB01_GQ4X-MIN_MV.Unit
...
```

the file will be quite big because all variables are expanded with their parameters.

To access variable *S03\_LB01\_GQ4X-MIN\_MV.Text* the index 1346 should be used.

If pointlist is reread, the driver will try to update the index file. It will never delete any variables, even if they are deleted from the Fidelix. New variables will be added using previously never used index numbers.

If index based lookup has been used, the index file(s) should never be deleted because the driver will create and enumerate a new file, and the index numbers may change causing severe work for the user.

It is important distributing also the index files if the project is moved.

### 3.4 FIDELIX.DBF

<i>Template</i>	<i>Unit type</i>	<i>Raw type</i>	<i>Bit width</i>	<i>Read/Write</i>	<i>Comment</i>
DEVICE.GETPOINTLIST	0x50000000	0	1	R/W	Force pointlist reread
OTA:%K%#.TimeArray	0x70000000	1	16	R/W	Time array (72 int's)
OVA:%K%#.ValueArray	0x71000000	1	16	R/W	Time value array (72 int's)
XTA:%U%*256	0x20000000	1	16	R/W	Time array (72 int's)
XVA:%U%*256	0x21000000	1	16	R/W	Time value array (72 int's)
XD:%U%*256	0x30000000	0	1	R/W	Digital value from Fidelix
XI:%U%*256	0x31000000	1	16	R/W	Integer value from Fidelix
XR:%U%*256	0x32000000	2	32	R/W	Real value from Fidelix
XL:%U%*256	0x34000000	4	32	R/W	Long value from Fidelix
XS:%U%*256	0x37000000	7	1024	R/W	String value from Fidelix
XB:%U%*256	0x38000000	8	8	R/W	Byte value from Fidelix
OD:%J%K%!	0x60000000	0	1	R/W	Digital value from Fidelix
OI:%J%K%!	0x61000000	1	16	R/W	Integer value from Fidelix
OR:%J%K%!	0x62000000	2	32	R/W	Real value from Fidelix
OL:%J%K%!	0x64000000	4	32	R/W	Long value from Fidelix
OS:%J%K%!	0x67000000	7	1024	R/W	String value from Fidelix
OB:%J%K%!	0x68000000	8	8	R/W	Byte value from Fidelix
%U%*256	0x30000000	0	1	R/W	Digital value from Fidelix
%U%*256	0x31000000	1	16	R/W	Integer value from Fidelix
%U%*256	0x32000000	2	32	R/W	Real value from Fidelix
%U%*256	0x34000000	4	32	R/W	Long value from Fidelix
%U%*256	0x37000000	7	1024	R/W	String value from Fidelix
%U%*256	0x38000000	8	8	R/W	Byte value from Fidelix
%J%K%!	0x60000000	0	1	R/W	Digital value from Fidelix
%J%K%!	0x61000000	1	16	R/W	Integer value from Fidelix
%J%K%!	0x62000000	2	32	R/W	Real value from Fidelix
%J%K%!	0x64000000	4	32	R/W	Long value from Fidelix
%J%K%!	0x67000000	7	1024	R/W	String value from Fidelix
%J%K%!	0x68000000	8	8	R/W	Byte value from Fidelix

The *DEVICE.GETPOINTLIST* template could be used to force the driver re-reading the pointlist. If set to 1 it will begin reading, when finished the driver will automatically reset it to 0.

Point description texts are not writable. If changed in device, a new pointlist must be retrieved for the texts to be updated.

### 3.5 Supported variables/parameters

#### IN

Value  
ValueAuto  
LockState  
Text  
Page  
StatusText

#### DO

Value  
ValueAuto  
LockState  
Text  
Page  
StatusText

#### AI

Value  
ValueAuto  
LockState  
Text  
Page  
Unit  
Limit[1-8].Value  
Limit[1-8].Manual  
Limit[1-8].Text

#### AO

Value  
ValueAuto  
LockState  
Text  
Page  
Unit

#### CT

Offset  
Text  
Lookup[1-15]in  
Lookup[1-15]out

#### TT

Value  
LockState  
Text  
TimeArray (array of 72 int's)  
ValueArray (array of 72 int's)

**AL**

Value  
ValueAuto  
LockState  
Acknowledged  
Priority  
Status  
StatusText  
AlarmGroup  
AlarmNumber  
AlarmStatus  
Text

**ST**

Value[0-19]  
Text[0-19]

**CO**

Text  
Page  
DaySetLockState  
RunState  
RunLockState  
NightSetValue  
NightPBand  
ReturnLimitStart  
ReturnLimitPBand  
StatusText  
DaySetValue  
StagesUsed  
Stage[0-5]Name  
Stage[0-5]Value  
Stage[0-5]PBand  
Stage[0-5]LimitStart  
Stage[0-5]LimitPBand  
Stage[0-5]OutMin  
Stage[0-5]OutMax  
Stage[0-5]OffValue  
PBand  
ITime  
DeadZone  
SampleTime  
SlowSpeedFactor

**CALENDAR**

Calendar[0-9]  
Holiday[0-9]

### 3.6 Address examples (OID and Index lookup)

Variable Tag Name: S03\_EM01\_WL2\_MV\_Limit1\_Value      Data Type: REAL

I/O Device Name: fidelix2      Address: S03\_EM01\_WL2\_MV.Limit1.Value

Raw Zero Scale:      Raw Full Scale:      Eng Zero Scale:      Eng Full Scale:      Eng Units:      Format:

Comment:

Buttons: Add, Replace, Delete, Help

Record: 21      End of file      Deleted      Linked: No

Typical tag of *Data Type Real* using OID lookup. Address format here is same as in the Fidelix OPC server.

Variable Tag Name: H3D1\_AS2\_LB01\_F001\_L.Text      Data Type: STRING

I/O Device Name: Fdx1      Address: OS:H3D1\_AS2\_LB01\_F001\_L.Text

Raw Zero Scale:      Raw Full Scale:      Eng Zero Scale:      Eng Full Scale:      Eng Units:      Format:

Comment:

Buttons: Add, Replace, Delete, Help

Record: 19      Deleted      Linked: No

Typical tag of *Data Type String* using OID lookup, forced to data type *String*. Sometimes the Citect compiler is not able to determine the size of a variable (mostly strings) and will return bad data to the user. Prefixing the address with a specific data type solves this problem. The prefixes *OD, OI, OR, OL, OS, OB* are used with OID lookups.

Variable Tag Name: calendar\_calendar\_0      Data Type: STRING

I/O Device Name: Fdx1      Address: OS:calendar.Calendar0

Raw Zero Scale:      Raw Full Scale:      Eng Zero Scale:      Eng Full Scale:      Eng Units:      Format:

Comment:

Buttons: Add, Replace, Delete, Help

Record: 18      Linked: No

String tag from Calendar, it contains semicolon separated dates, ex:

*100913Fr03;101013Fr03;0000000000; 0000000000;....*

Prefixing the address with a specific data type (OS) ensures variable is handled as a string internally in Citect.

Variable Tag Name: H3D1\_AS2\_LB01\_F001\_L\_StatusText      Data Type: STRING

I/O Device Name: Fdx1      Address: XS:3101

Raw Zero Scale:      Raw Full Scale:      Eng Zero Scale:      Eng Full Scale:      Eng Units:      Format:

Comment:

Buttons: Add, Replace, Delete, Help

Record: 18      Deleted      Linked: No

Typical tag of *Data Type String* using index lookup, forced to data type *String*. Sometimes the Citect compiler is not able to determine the size of a variable (mostly strings) and will return bad data to the user. Prefixing the address with a specific data type solves this problem. The prefixes *XD, XI, XR, XL, XS, XB* are used with index lookups.

Variable Tag Name:  Data Type: REAL

I/O Device Name:  Address:

Raw Zero Scale:  Raw Full Scale:

Eng Zero Scale:  Eng Full Scale:

Eng Units:  Format:

Comment:

Buttons:

Record: 17  Deleted Linked: No

Typical tag of *Data Type Real* using index lookup.

### 3.7 Time channel addressing

Time values are stored in two arrays, *TimeArray* and *ValueArray* containing timestamps (stored as integers 0000..2359 or 9999 for 'not in use') and its desired value.

The array(s) have 72 elements (3 start/stop values *Monday..Friday*, followed by 3 start/stop values for *E1, E2, E3, HD, HA*).

Variable Tag Name:  Data Type: INT

I/O Device Name:  Address:

Raw Zero Scale:  Raw Full Scale:

Eng Zero Scale:  Eng Full Scale:

Eng Units:  Format:

Comment:

Buttons:

Record: 13 End of file  Deleted Linked: No

Time array (timestamps) example.

Ex from above *TEST\_LB01\_TimeArray[24]* would contain first start time for *Friday*.

*TEST\_LB01\_ValueArray[24]* would contain the first start value for *Friday*.

*TEST\_LB01\_TimeArray[25]* would contain the first stop value for *Friday*.

## 4. User Interface

### 4.1 Introduction

This section defines how the user will see the driver. This relates directly to how the Citect forms need to be filled out and any special INI options. For the kernel, the debug trace messages and the *Stats.Special* counters are documented.

### 4.2 Driver Name

Fidelix

### 4.3 Ports Form (TCPIP setup)

#### 4.3.1 Baud Rate

n/a

#### 4.3.2 Data Bits

n/a

#### 4.3.3 Stop Bits

n/a

#### 4.3.4 Parity

n/a

#### 4.3.5 Special Opt

- i The destination server IP-address, ex -i192.168.123.10.
- p The destination server IP-port, ex -1235.
- t Use TCP/IP (default)

Neither RS-232 (COMX) nor UDP can be used.

### 4.4 IO Devices Form

#### 4.4.1 Protocol

Fidelix

#### 4.4.2 Address

n/a

## 4.5 Pulldown lists Help

The following entries should be included in the Citect Help.DBF spec file.

TYPE	DATA	FILTER
PROTOCOL	FIDELIX	

## 4.6 PROTDIR.DBF

The following entries should be included in the Citect Protdir.DBF spec file.

TAG	FILE	BIT_BLOCK	MAX_LENGTH	OPTIONS
FIDELIX	FIDELIX	16	2048	0x2000cf

## 4.7 Parameters and INI options

### 4.7.1 Standard Parameters

Block	16	(Citect blocking is prohibited)
Delay	1 ms	
MaxPending	75	
Polltime	0 ms	
Timeout	10000 ms	
Retry	0	(Sets the complete packet retries, should be 0)
WatchTime	30 s	

### 4.7.2 Driver Specific Parameters

All Fidelix specific parameters are located in the section 'Fidelix'

PARAMETER	DEFAULT	DESCRIPTION
TimeOut	10000 ms	Response timeout for the device (even partly received packets will reset the timeout timer)
MaxPending	75	Number of DCBs the driver may handle simultaneously. Actually the default is quite low for this driver, but Citect is not capable of handling an unlimited number of outstanding DCBs. This number must be multiplied by number of channels to get the total DCBs for Citect.
WatchTime	30 s	When Citect send DCB requests to the driver, it have to respond within this time. This is also the cycle time for init and status requests for offline devices.
ConnectTimeout	15000 ms	msecs before we consider the IP connection/reconnect has failed. The IP layer may have an unreasonable timeout up to 120 seconds.
TraceToLogFile	0	If set to 1, it will cause the driver to write debug data to the file <i>Fidelix.log</i> stored in Citect's data folder. The file will only be created if debug in Kernel is used. No check is made to ensure disk is not

		filled by the log file.
TraceOptions	0	Options for trace (may be bit-weighted together): 1 – Trace found I/O devices in DBF's 2 – Trace variables found in DBF's (will also trace I/O devices)
TraceLimit	40	Limits the number of bytes presented in Kernel when using debug-command.
IgnoreStartupErrors	0	If set to 1, irregularities in the variable database will not be reported. However, the parameters will be set as bad values if requested by Citect.  Default is that the driver reports any potential problem at startup time using a messagebox. The messagebox will however block the startup sequence.
AllowBlackbox	0	If set to 1 the driver will create and also update an text based index file used for index variable lookup.
RequestInterval	50 ms	Each request cycle, the driver will scan through internal queues for work to do (create packets, update statistics, cleanup memory, return new values from cache etc). Performance is linear to this time. If cache is fully updated the driver will return an average of <i>MaxPending / RequestTime</i> values per second (default $75 / .050 = 1500$ values).  Min value is 10 msec. Timer granularity is depending of OS.
ReadPoint-ListAtStartup	0	If set to 1, it will force the driver DLL to reread the pointlist when Citect is starting, even if the pointlist file already exists.
RestartInterval	5000	When device times out, it will wait this small amount of time before trying to initiate communications again. The Fidelix device close sockets very fast sometimes if no communication takes place, hence this time should not be very long.
UpdateInterval	2000	Number of msec between each request for updated tags
AltRunPath	[CTEDIT] RUN	Alternative Run path, is used to find the top project.
AltUserPath	[CTEDIT] USER	Alternative User path. Is used by OID lookup mechanisms to find the <i>Master.DBF</i> file.
AltDataPath	[CTEDIT] DATA	Alternative Data path. Is used to determine where to store pointlist and index files. If set to the comms project path also index files will be backed up.
TimeTableWriteDelay	500	Delay time before TimeTables are written. This prevents multiple writes to device when same tables is updated.

## 4.8 Debug Messages

Shows the outgoing/incoming data packets. Header in verbose format and data/value in hex.

## 4.9 Stats Special Counters

Number	Label	Purpose/Meaning of this counter
--------	-------	---------------------------------

0	DCB Requests	Total number of driver requests
1	Tx bytes	Transmitted bytes
2	Rx bytes	Received bytes
3	Tx packets	Transmitted packets
4	Rx packets	Received packets
5	GetVersions	Number of GetVersion commands (issued at startup/reconnect)
6	GetPointlists	GetPointList commands (number of point list rereads)
7	GetChgPoints	GetChangedPoint commands (number of variable updates)
8	SetPoints	SetPoint commands (number of variable writes)
9	Updated vars	Number of updated variables (read)
10	n/a	
11	n/a	
12	n/a	
13	n/a	
14	n/a	
15	n/a	
16	n/a	
17	n/a	
18	n/a	
19	n/a	

## 4.10 Hints and Tips

None for now

## 5. Basic Testing

### 5.1 Introduction

The programmer will perform a minimum level of testing which is outlined here.

A sample Project is available which can be used as a starting point for the programmers test Project. When the programmer has completed basic testing and debugging this Project should be backed up and supplied to the Citect Testing department.

### 5.2 Procedure

The following are points should be covered by basic testing.

- On startup the IO Device comes online without errors.
  - Ok
- The driver supports IO Devices of addresses as documented in the specification.
  - Ok
- The driver reports the IO Device offline when the IO Device is a) powered down, b) disconnected.
  - Ok
- The driver will re-establish communication with the IO Device after a) power cycle, b) disconnection/ reconnection.
  - Ok
- Confirm that retries (if supported) and error reporting operate correctly.
  - Ok (Statistics report)
- The driver reads all the device data types documented as readable in this specification.
  - Ok
- The driver writes to all the device data types documented as write-able in this specification.
  - Ok
- The driver reads and writes all data formats supported by the protocol, ie DIGITAL, INT, LONG, REAL, STRING.
  - Ok
- Test the limit of the IO Devices request size, this should be done for at least DIGITAL and an INT data formats.
  - Not applicable to this driver.

- Let the driver run over night and check that no retries or other errors have occurred.
  - Ok
  
- If a multidrop or network protocol and if the hardware is available then the protocol should be tested with more than one IO Device connected.
  - Ok

## 6. Performance Testing

### 6.1 Introduction

Tests which give some indication of the drivers performance. The programmer needs to perform these tests since the results feed back into the Constants structure and the PROTDIR.DBF.

### 6.2 Calculating the Blocking Constant – Not applicable

Because this driver is a hybrid of the 'Front-End-Back-End' type, and due to the nature of the Fidelix protocol, it does not support blocking of any kind

## 7. References

### 7.1 References

n/a.