Citect
Industrial Strength Solutions

CitectSCADA

# DRIVER DEVELOPMENT TOOLKIT

# VERSION 6.0

# INTRODUCTION

This paper is intended for any CitectSCADA users interested in developing a driver for CitectSCADA using the driver development toolkit (DDK).

# Introduction

The Citect Driver Development Kit (DDK) provides all the information you will need to write a device driver for a communication protocol. It provides a common base of code that is used by all CitectSCADA drivers, and also includes examples of working drivers. To build a driver you will also need a suitable C++ compiler and linker.

Device drivers are used so that a single generic version of CitectSCADA can use any type of communication protocol. Because device drivers communicate with the physical device, they provide CitectSCADA with the flexibility of communicating with any type of I/O Device.

This DDK is aimed at developers developing drivers for CitectSCADA. Citect Ltd also recommends that you develop drivers on the latest released CitectSCADA version. By definition, CitectSCADA drivers are 32-bit drivers. This DDK should work with versions of CitectSCADA from version v5.50 onwards.

Once you have written a CitectSCADA driver, you can add it to the CitectSCADA system and it will then be used in the same way as the drivers supplied with CitectSCADA. Drivers are implemented as Windows DLLs (Dynamic Link Libraries).

This manual assumes that you are familiar with basic programming techniques, and the 'C' or 'C++' programming languages.

## *System requirements*

The system requirements to allow you to develop CitectSCADA drivers include:

- Windows XP / 2000 / Windows NT4.0
- CitectSCADA V5.5 on
- A suitable compiler. Citect Pty Ltd recommends and uses Microsoft Visual C/C++ Version 6 to build CitectSCADA drivers. If you use a different product then you may have unforeseen compatibility problems, and we are unable to support you with the specifics of that compiler.
- Editor for dBase III files (*.DBF). Applications which can edit DBF files include Microsoft Excel, Microsoft Access, FoxPro, Dbu.exe (utility included with DDK), dBase (III and above), etc.

## *Before You Start*

It should take from three to six weeks to design and write a simple driver, depending on the complexity of the protocol, and how familiar you are with CitectSCADA.

Before attempting to write a driver, you should:

- Print the source code for the "skeleton" driver and example drivers supplied on the DDK CD-ROM.
- Read all documentation (the tutorial, source files and this manual) thoroughly. You should understand how CitectSCADA and the driver interact, the start-up sequence, running requests, the shutdown sequence, redundancy and error systems.
- Try building one of the example drivers.
- Experiment with the CitectSCADA Kernel "DriverTrace", (see Appendix H) to get a feel for the commands sent to drivers.

## *What's Included With This DDK?*

The Driver Development Kit includes the following:

- This manual.
- Header and Library files required to build a CitectSCADA Driver.
- A shell driver specification document.
- A framework "skeleton" source code for a new driver.
- A simple CitectSCADA Project for initial debugging.
- Source code of example drivers.
- A spreadsheet for calculating the blocking constant.
- A full-release of the latest version of CitectSCADA.
- A CitectSCADA protection key. This key will allow 8 hours run-time, which is sufficient for most testing purposes. This is only valid when a Professional Edition of the CitectSCADA DDK is purchased.
- Several utilities to assist in the development, testing, and distribution of the drivers you develop.
- 12 months free Driver Development Support from date of purchase. Refer to the following section for more information. This is only valid when a Professional Edition of the CitectSCADA DDK is purchased.

## *Manual Outline*

Following is a brief description about the content of each chapter in this DDK:

| | |
|---|---|
| 1. Introduction | This chapter explains the things you need to know before embarking on CitectSCADA driver development. |
| 2. Overview | Discusses what a CitectSCADA driver is, and how it interacts with the CitectSCADA system. |
| 3. The CitectSCADA Compiler | Discusses how the CitectSCADA compiler is used to check user configuration. |
| 4.Processing CitectSCADA requests | Explains how a driver interacts with CitectSCADA, and how it manages requests. |
| 5. Data Formats | Explains the common data formats used by CitectSCADA. |
| 6. Driver Parameters | Explains the standard parameters that every CitectSCADA driver supports, plus how to define and use driver specific parameters. |
| 7. Error Handling | Explains how to handle errors, including those returned by the device. |
| 8. Debug information & Statistics | Details how to take advantage of CitectSCADA's standard debugging and tracing facilities, and use it in a run-time system. |
| 9. Request blocking | Explains how CitectSCADA's performance improving mechanisms work. |
| 10. Building drivers | Shows how to build a CitectSCADA driver using Microsoft VC++. |
| 11. Testing drivers | Gives some tips and methods for performing testing of the driver. |
| 12. Data Structures | Explains the common data structures employed by CitectSCADA. |
| 13. Utility functions | Describes the function and use of the utility functions provided with the DDK. |
| 14. Serial communications functions | Describes the function and use of the serial communications functions provided with the DDK. |
| 15. Queue functions | Describes the function and use of the queue functions provided with the DDK. |
| 16. Timer functions | Describes the function and use of the timer functions provided with the DDK. |
| 17. Database functions | Describes the function and use of the database functions provided with the DDK. |
| 18. V Database functions | Describes the function and use of the virtual database functions provided with the DDK. |
| 19. Hardware access functions | Describes the function and use of the hardware access functions provided with the DDK. |
| 20. Software protection for drivers | Highlights how to protect your drivers against illegal use. |
| 21. Driver development utilities | Describes the function and use of the utilities provided with the DDK. |
| Appendix A – Driver Commands | Describes the commands sent to the driver by CitectSCADA for the driver to act on. |
| Appendix B – Generic Errors | Describes the Generic error code available to the driver. |
| Appendix C – Driver Errors | Describes the specific driver error codes available to the driver. |
| Appendix D – FAQs | Details a number of frequently asked questions and their answers. |
| Appendix E – Example driver | Provides tutorials on some example drivers. |
| Appendix F – Superceded template specifiers | Describes the template specifiers that are superceded by this version of the DDK. |
| Appendix G – Creating a Stand Alone Driver Pack | Provides step by step instructions for creating a setup program to install a single driver. |

## *Directory structure of installed DDK*

The DDK setup program installs a large directory tree of files and documents to help you build your CitectSCADA drivers. Please refer to the document – DDK Structure.pdf on the DDK CD-ROM for a complete description of the file contents.

Note: Private build is the equivalent of the debug build.
Public build is the equivalent of the release build.

**\Citect\DDK**

- **\CtBin** - Contains utilities required for building a driver.
- **\CtLib** - Contains libraries required by drivers.
  - **\Private** - Contains library files for Private builds.
  - **\Public** - Contains library files for Public builds.
- **\Doc** - Contains documents and a basic test project.
- **\Drivers** - Contains driver source code.
  - **\Protection** - Contains driver protection source code.
    - **\res** - Contains driver protection resources.
  - **\Skeleton** - Contains Skeleton driver source code.
  - **\Modnet** - Contains Modnet driver source code.
  - **\Tiway** - Contains Tiway driver source code.
  - **\Toshiba** - Contains Toshiba driver source code.
- **\Private** - This is the Private (Debug) build area. This directory, and any of it's sub-directories and files are created during a Private build, and they may be safely deleted.
  - **\Drivers** - This directory contains all the outputs of Private Driver builds.
    - **\Skeleton** - Outputs of the Skeleton Driver Private builds.
    - **\Protection** - Outputs of the Protection Driver Private builds.
  - **\obj** - This directory contains the intermediate files of Private Driver builds.
    - **\Drivers** - This directory contains the intermediate files of Private Driver builds.
      - **\Skeleton** - Contains the intermediate files of the Skeleton Driver Private builds.
      - **\Protection** - Contains the intermediate files of the Protection Driver Private builds.
    - **\Utilities** - This directory contains the intermediate files of Utilities builds.
      - **\RegCentre** - This directory contains the intermediate files of Registration Centre utility Private builds.
  - **\setup** - Used by internal driver developers. Third party developers may ignore this directory.
    - **\specfile**
  - **\Utilities** - This directory contains all the outputs of Utilities Private builds.
    - **\Regcentre** - Contains outputs of Registration Centre Private builds.
- **\Public** - This is the Public (Release) build area. This directory, and any of it's sub-directories and files are created during a Public build, and they may be safely deleted. The directory structure is same as Private build area except it contains output of Public builds.
  - **\Drivers**
    - **\Skeleton**
    - **\Protection**
  - **\obj**
    - **\Drivers**
      - **\Skeleton**
      - **\Protection**
    - **\Utilities**
      - **\RegCentre**
  - **\setup**
    - **\specfile**
  - **\Utilities**
    - **\Regcentre**
- **\Source** - Contains common files required for building drivers.
  - **\Include** - Contains common header files required for building drivers.
- **\Utilities** - Contains source code for Utilities program.
  - **\Regcentre** - Contains Registration Centre utility source code.
    - **\res** - Contains Registration Centre utility resources.

## *Generating a CitectSCADA Driver*

When writing a driver, you should:

- Read the protocol manuals provided with the device, so you know exactly how the protocol works.
- Write a specification for the driver. A blank specification document (in Word 97 format) is provided with the DDK.
- Define the driver and its data format(s) in the compiler specification databases. These databases define how CitectSCADA works with the new driver.
- Use the skeleton (provided on the DDK CD-ROM) or an existing driver as a basis for the new driver. Write the code according to your driver specification.
- Debug the driver. Start with a simple test project—to get the first communications. Extend the project to test and debug data formatting, operation with larger block sizes, robust operation, etc.
- Update your documentation, and generate user documentation.
- Test the driver against your user documentation. Test all the types, data formats, address ranges etc.

## *Glossary*

The following are terms and abbreviations that are used in this manual or in the Automation industry in general:

| | |
|---|---|
| API | Application Programming Interface |
| DCS | Distributed Control System. |
| DDK | Driver Development Kit |
| Driver | A software component that abstracts communications with a specific device. |
| IED | Intelligent Electronic Device |
| IEEE | Institute of Electrical and Electronic Engineers |
| ISO | International Organisation for Standardisation |
| LAN | Local Area Networks |
| OLE | Object Linking and Embedding |
| OPC | OLE for Process Control |
| OSI | Open Systems Interconnection |
| PLC | Programmable Logic Controller |
| Protocol | A well-defined method of communication between CitectSCADA and a device. It does not always define a physical communications media. |
| PSTN | Public Switch Telephone Network |
| RBE | Report By Exception. A report generated by the device on the occurrence of some event. The report usually includes data. |
| RS232 | International standard for serial communications protocols. Two unbalanced data lines, with only one receiver per line. Allows full duplex serial communication between two points. |
| RS422 | International standard for serial communications protocols. Two balanced data lines, with only one transmitter and up to ten receivers per line. Either allows simplex serial communications between one central point and up to ten other points, or full duplex serial communication between two points. |
| RS485 | International standard for serial communications protocols. Two balanced data lines, with up to 32 transmitters and up to 32 receivers per line. |
| RTU | Remote Telemetry Unit |
| SCADA | Supervisory Control And Data Acquisition |
| SOE | Sequence of Events. A log of chronologically ordered list of time-stamped events that are normally held by the device for interrogation, storage and display by a SCADA system. |
| TCP/IP | Transmission Control Protocol / Internet Protocol |
| UDP/IP | User Datagram Protocol / Internet Protocol |
| Unsolicited messages. | Messages sent by the Slave (or Server) that are not triggered by a request from the Master (or Client). |
| WAN | Wide Area Networks |