RLD Automation

Van Sonsbeeckstraat 11
5344 JB Oss
Telefoon 0412 – 655 990
Fax 0412 – 655 991
info@rldautomation.eu
www.rldautomation.eu

Handleiding: Handleiding Datataker applicatie

Aluchemie
Locatie: Rotterdam

Opdracht PO-000127-1

Auteur(s)
Marcel Jordaan

Datum opgesteld
19 april 2012
Datum gewijzigd
19 april 2012
Project
Vervangen DataTaker appplicatie
Opdracht: PO-000127-1
Locatie: Rotterdam
Versie
1.0
Status
Definitief

Inhoud

1 Inleiding

Dit is de handleiding behorende bij de Datakater applicatie voor Aluchemie, te Rotterdam. Deze handleiding beschrijft de opbouw van de datataker applicatie zoals deze binnen Aluchemie gebruikte wordt, de installatie en configuratie van de datataker applicatie zelf en de opbouw van de direct bijbehorende MSSQL 2005 databasa configuratie.

1.1 Documenten

Document ID	Omschrijving	Revisie #	Datum	Status
Template.doc	Titel of andersoortige omschrijving	0.1	1 jan 03	Concept

1.2 Document historie

Versie #	Datum	Status	Auteur	Omschrijvingen
0.1	12 sep '11	Concept	MJO	Eerste uitgave, voorconcept goedkeuring

1.3 Toegepaste documenten

Bij het opstellen van dit technisch ontwerp is gebruik gemaakt van de volgende documenten:

Titel	Versie	Datum uitgifte

1.4 Definities en begrippen

Om duidelijkheid te creëren in de gebruikte terminologie en afkortingen worden de verschillende begrippen en definities evenals afkortingen hier verklaard.

Afkorting / begrip	Omschrijving

2 DataTaker applicatie

2.1 Algemeen

De DataTaker applicatie is een integrale applicatie bestaande uit een C++ programma, dat serieel communiceert met een fysieke DataTaker, en verschillende MSSQL-server instanties, verdeeld over verschillende servers en T-SQL scripts.
De opzet van de DataTaker applicatie zoals deze door Aluchemie is opgesteld is opgenomen in bijlage A: BijlageA-SysteemConfiguratie.

Het DataTaker C++ programma communiceert op seriële basis met de DataTaker hardware, een overzicht van de communicatie is te vinden in bijlage B: BijlageB-Seriele Communicatie.

De interactie tussen het DataTaker C++ programma en MSSQL2005 databases is schematisch weergegeven in bijlage C: BijlageC-DataTakerDatabase.

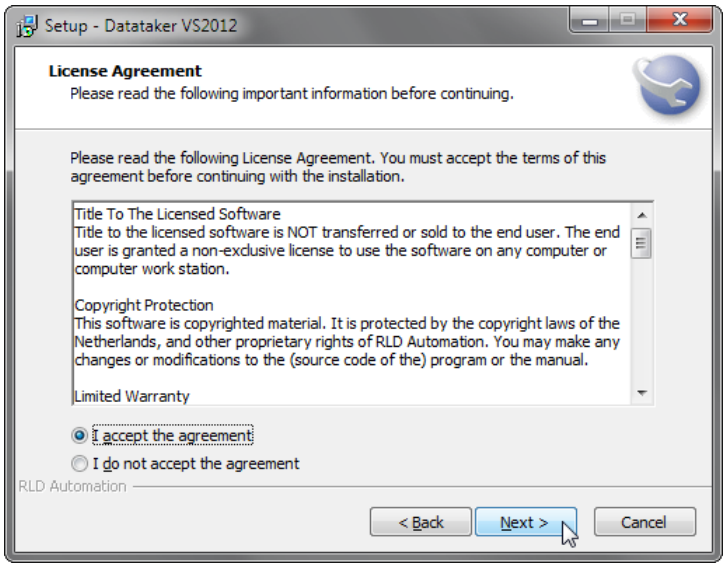
2.2 Installatie DataTaker C++ programma

Het DataTaker C++ programma is voorzien van een installatie wizard dit is een uitvoerbaar bestand met de naam: *DatatakerInstall1.1.0.1.exe*. Dit programma kan direct vanuit de verkenner gestart worden, indien voor de installatie meer rechten benodigd zijn dan de huidige gebruiker heeft, wordt automatisch toestemming gevraagd voor opwaardering van het rechten niveau.
In de naam van het installatie bestand is de versie opgenomen, bij latere versies kan de naam enigszins afwijken van de hier gegeven naam.

Na het starten van de installatie kan de wizard doorlopen worden, zoals hieronder is weergegeven.

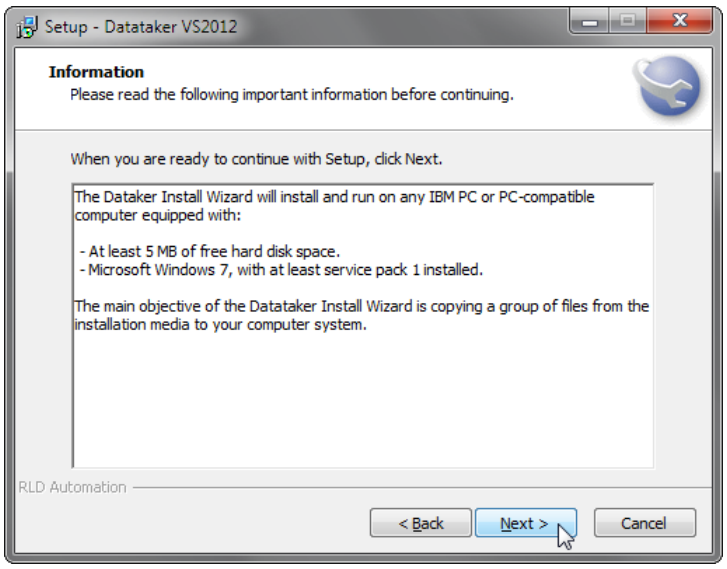
Het welkomst scherm van de installatie wizard wordt getoond. In dit scherm is het versie nummer van het C++ programma opgenomen, dit kan afwijken van het hier getoonde nummer.

Ga verder met “Next >”.

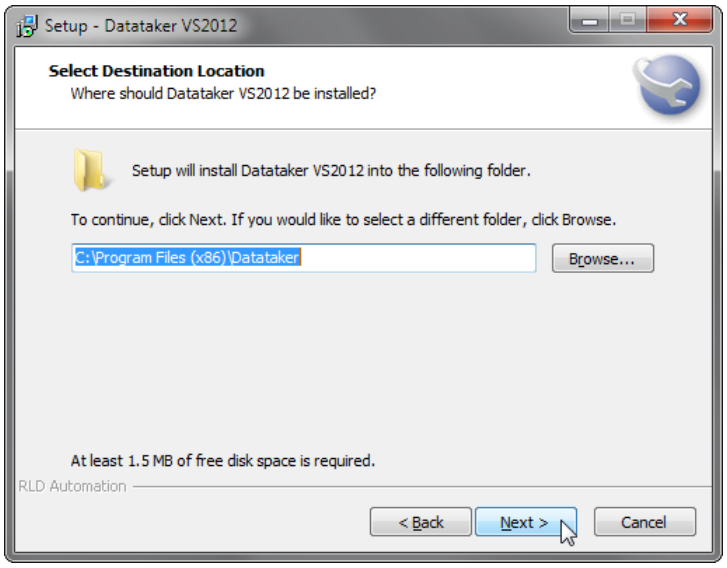


Accepteer de licentie overeenkomst en ga verder met “Next >”.

Wordt de licentie overeenkomst niet geaccepteerd, dan wordt het programma niet geïnstalleerd.

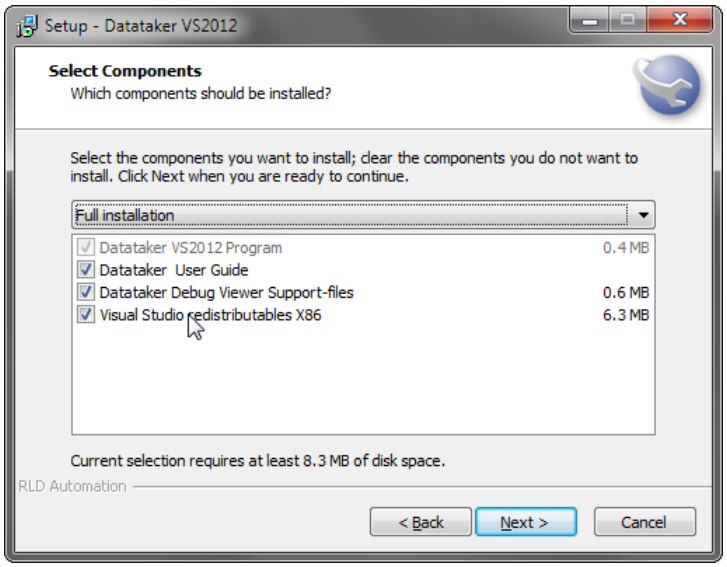


De minimaal benodigde schijfruimte en Windows versie worden weergegeven, ga verder met “Next >”.



De map waar het programma wordt geïnstalleerd kan naar wens gewijzigd worden. Normaal kan de standaard instelling aangehouden worden.

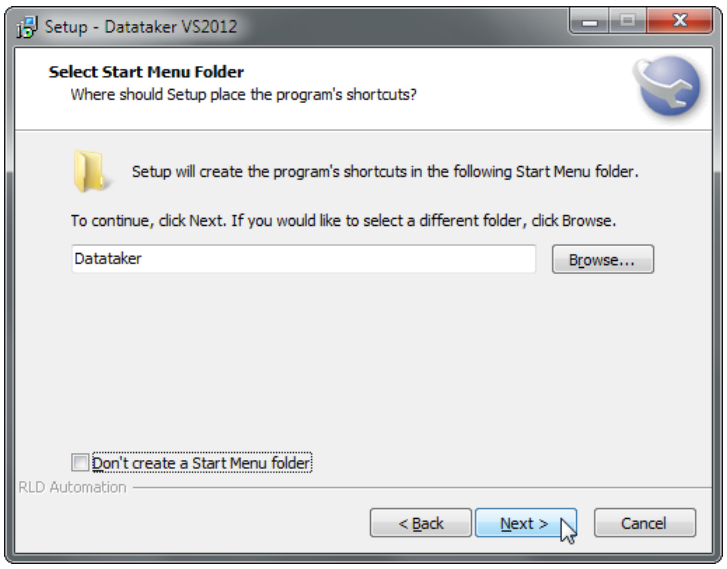
Ga verder met “Next >”.



Verschillende onderdelen kunnen voor installatie gekozen worden, het C++ programma wordt altijd geïnstalleerd. Indien een upgrade wordt uitgevoerd, wordt altijd het hoogste versie nummer geïnstalleerd. Indien het versie nummer van de installatie lager is dan een reeds geïnstalleerde versie, dan wordt deze NIET overschreven.

Behalve het C++ programma kan het programma dbgview (Debug Viewer Support) geïnstalleerd worden, deze is ook beschikbaar via Microsoft: http://technet.microsoft.com/en-us/sysinternals/bb896647.aspx

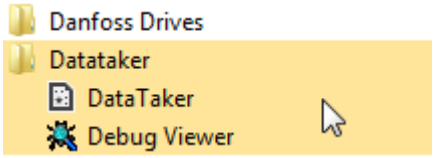
Nadat een keuze is gemaakt, kan verder gegaan worden met “Next >”.



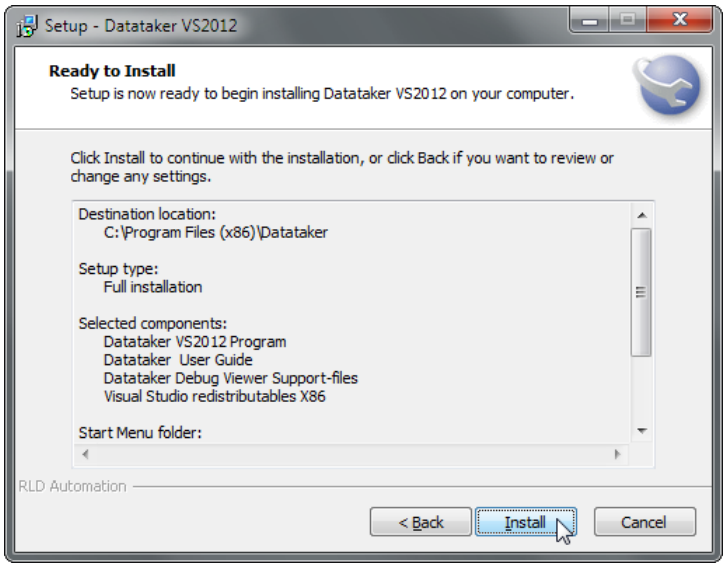
In het start menu wordt programma groep aangemaakt, de naam kan naar wens gewijzigd worden. Standaard wordt de naam ‘DataTaker’ gebruikt.

Indien gewenst wordt er geen programma groep gemaakt, hiervoor dient de optie ‘*Don’t create a Start Menu folder*’ aangevinkt te worden.

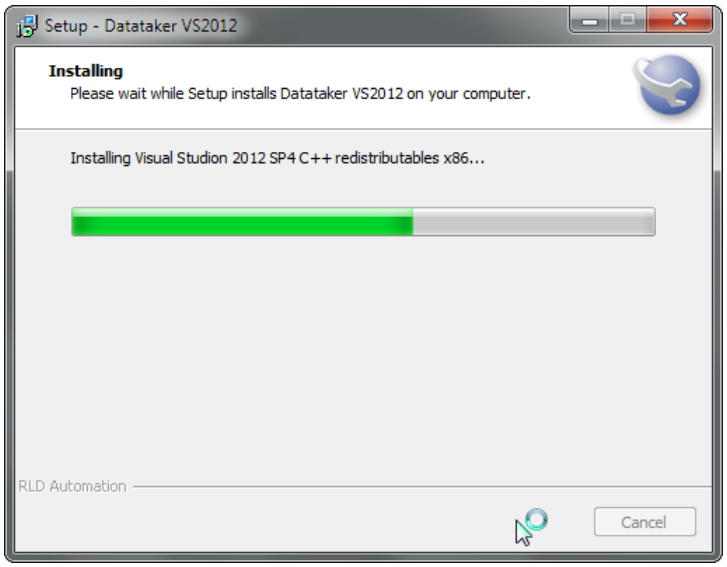
Wordt voor een programma groep gekozen, dan ziet deze er als volgt uit:



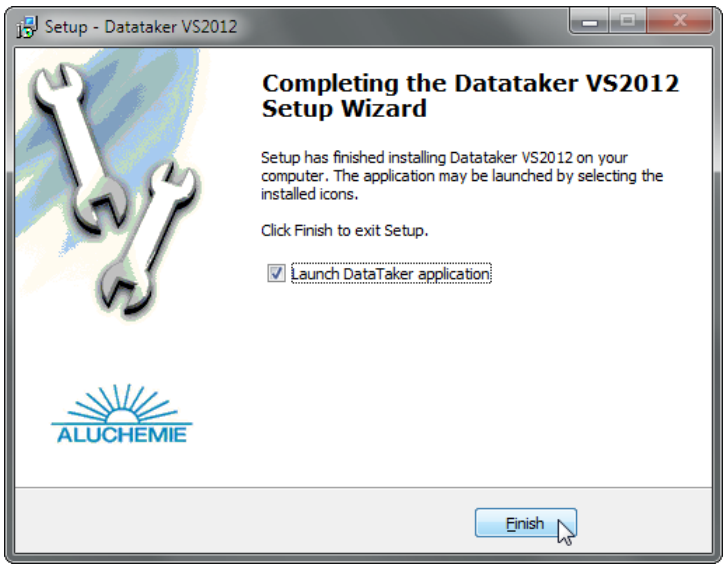
Ga verder met “Next >”.



Een overzicht van de gekozen installatie opties wordt gegeven, start de installatie met “Install”.



De voortgang van de installatie wordt weergegeven, wacht tot de installatie is voltooid.

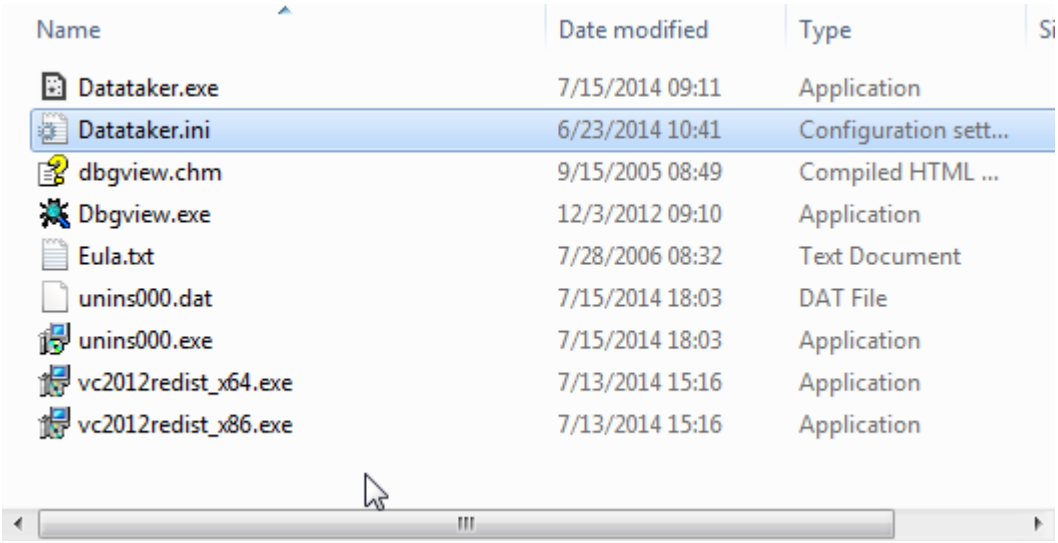


Nadat de installatie voltooid is wordt de optie geboden om de applicatie direct te starten.

Is dit gewenst dan dient de optie '*Launch Datataker application*' aangevinkt te worden.

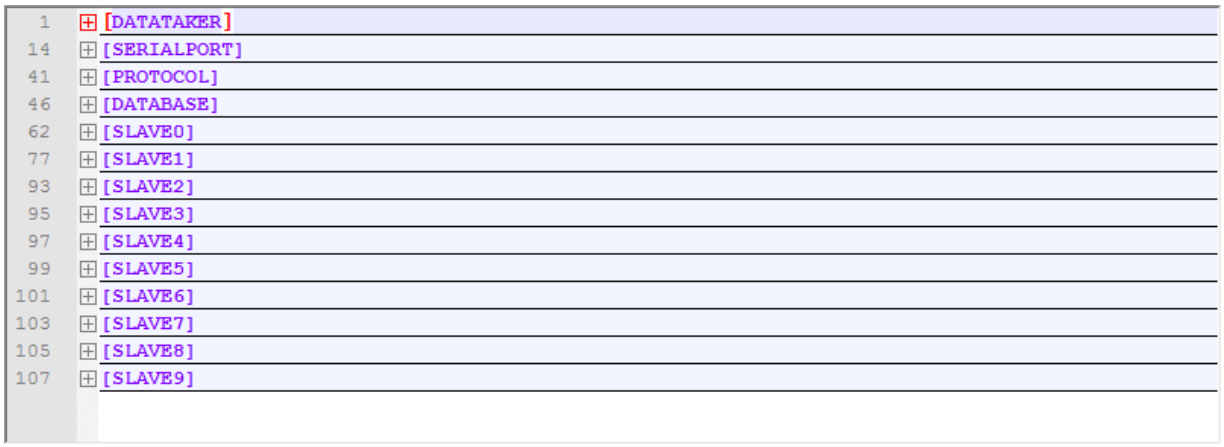
2.3 Configuratie DataTaker C++ programma

De DataTaker applicatie wordt met behulp van een zogeheten INI-bestand geconfigureerd, de bestandsnaam is datataker.ini, en het bestand staat in dezelfde map waarin de applicatie is geïnstalleerd. Standaard is dit de map: C:\Program Files\Datataker (voor een x86 systeem).



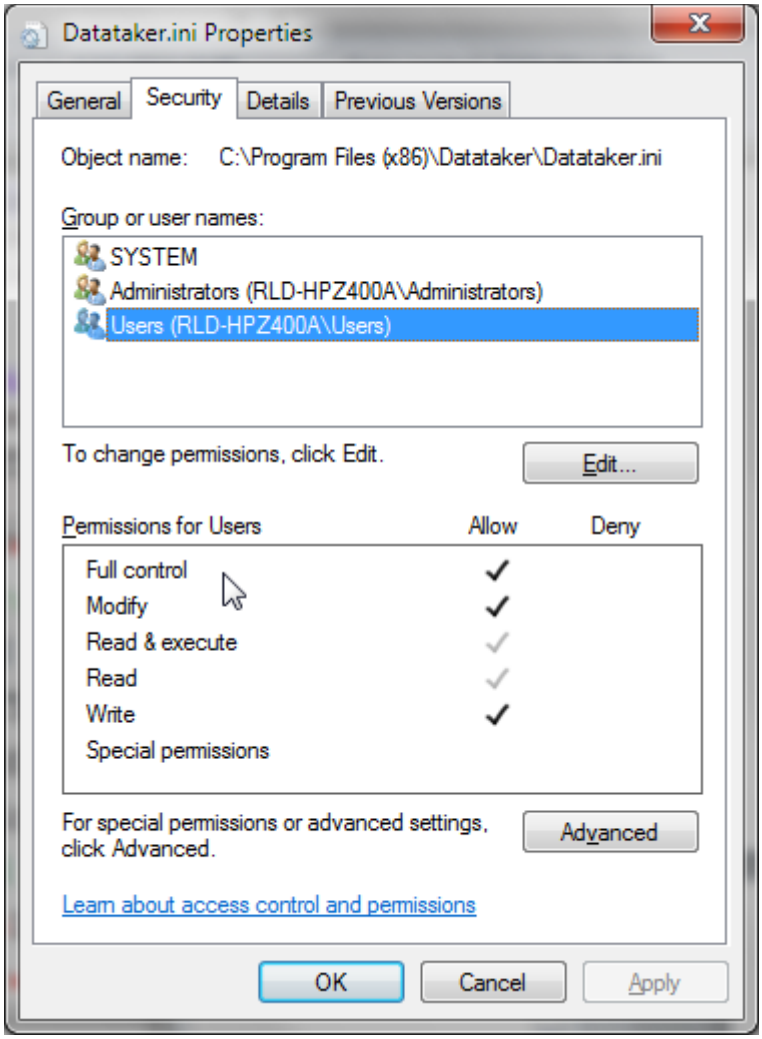
De inhoud van de installatie map is bovenstaande figuur weergegeven, het ini-bestand is met blauw gemarkeerd. Een ini-bestand kan met een editor geopend worden, standaard wordt hiervoor notepad (kladblok) gebruikt. In deze handleiding is de tekst-editor Notepad++ gebruikt, deze maakt standaard gebruik van syntax kleuring, en maakt de indeling van het bestand overzichtelijker.

Een ini-bestand bestand bestaat uit 1 of meer secties, in onderstaande figuur zijn alle secties van het datataker.ini bestand 'dicht' gevouwen.



Standaard staat het datataker.ini bestand in de map c:\program files\datataker, en hebben normale gebruikers geen recht om dit bestand te wijzigen. Dienen wijzigingen uitgevoerd te worden in dit bestand dan dient de beveiliging van dit bestand aangepast te worden: gebruikers dienen alle rechten te krijgen, zie onderstaande figuur.

Het datataker.ini bestand wordt door de applicatie slechts eenmalig bij starten ingelezen, wijzigingen in het bestand worden dan ook pas overgenomen na een herstart van de DataTaker applicatie.

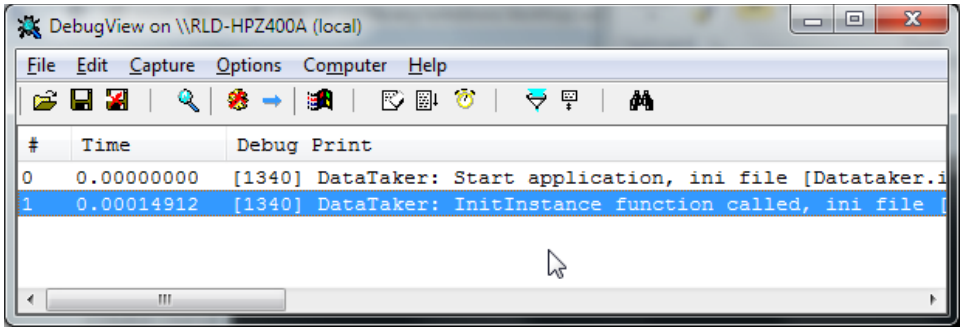


2.3.1 Datataker.ini: sectie DATATAKER

```
[DATATAKER]
#debug settings for application
Log2DbgViewer=1
Log2Screen=0
Log2File=0
Tracelevel=9
#maximaal aantal slaves voor de datataker
MaxSlave=9
#windows to open
CMD-Window=0
DLG-Window=1
#dialog initialisation
StartString=R120S
```

In de sectie [DATATAKER] van het bestand datataker.ini zijn algemene instellingen van de applicatie opgenomen.

Log2DbgViewer: Log niveau voor log info naar debug viewer, des hoger het getal des te meer info getoond wordt. Het niveau kan echter nooit hoger zijn dan de instelling TraceLevel. Waarde = 0 .. 9, voor waarde 0 wordt geen informatie getoond.



Log2Screen: Log niveau voor log info naar commando scherm (DOS-box), des hoger het getal des te meer info getoond wordt. Het niveau kan echter nooit hoger zijn dan de instelling TraceLevel. Waarde = 0 .. 9, voor waarde 0 wordt geen informatie getoond.

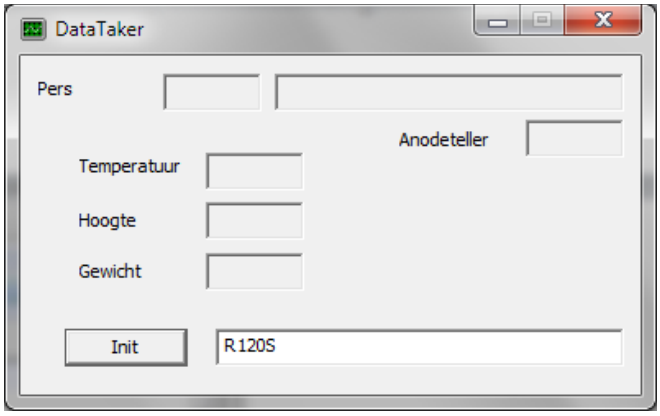
Log2File: Log niveau voor log info naar bestand datataker.txt in de programma map, des hoger het getal des te meer info getoond wordt. Het niveau kan echter nooit hoger zijn dan de instelling TraceLevel. Waarde = 0 .. 9, voor waarde 0 wordt geen informatie getoond.

TraceLevel: Absolute maximale niveau voor debug informatie, des hoger het getal des te meer info getoond wordt. Deze instelling begrensd de hiervoor genoemde instellingen. Waarde = 0 .. 9, voor waarde 0 wordt geen informatie getoond.

MaxSlave: Het maximale aantal slaves dat aan de datataker gekoppeld kan zijn, de standaard waarde is 9.

CMD-Window: Instelling om debug informatie zichtbaar te maken in een commando vernster (Dos-box), bij een waarde 0 wordt geen commando venster geopend, voor een waarde ongelijk aan nul wordt een venster geopend. Dit commando venster is alleen benodigd indien de waarde van Log2Screen ongelijk aan nul is.

DLG-Window: Instelling om het dialoog venster met blokinfo bij opstart van de applicatie te openen. Wordt dit venster gesloten dan wordt tevens de applicatie afgesloten. Zie ook onderstaande figuur.



StartString: Instelling van de waarde in de tekstbox rechts naast de button Init, de hier getoonde waarde (bovenstaande figuur) is 'R120S'. Deze waarde wordt gebruikt om iedere slave te initialiseren, nadat op de knop 'Init' gedrukt wordt. Voor de commando's wordt verwezen naar bijlage E: BijlageE-Datataker Pocket Reference.

2.3.2 Datataker.ini: sectie SERIALPORT

```
[SERIALPORT]
#normal port settings
COMport=1
BaudRate=1200
Databits=8
#stopbits: ONE=1 (default), ONE5=1.2, TWO=2
Stopbits=ONE
#parrity: N=None (default), E=Even, M=Mark, O=Odd, S=Space
Parity=N
UseRTS_CTS=0
UseDTR_DSR=0
UseXon_Xoff=0
#xon and xoff character are decimal values
XonChar=17
XoffChar=19
#timeout parameters
ReadIntervalTimeout=5
ReadTotalTimeoutMultiplier=10
ReadTotalTimeoutConstant=10
#event mask, leave at 0
EventMask=0
#debug settings for serial port
Log2DbgViewer=1
Log2Screen=0
Log2File=0
Tracelevel=9
```

In de sectie [SERIALPORT] zijn de instellingen van de seriële poort opgenomen, de meeste instellingen die betrekking hebben op de configuratie van de seriële poort zijn standaard juist ingesteld en zullen bij wijziging veelal tot gevolg hebben dat de seriële verbinding niet meer tot stand komt. Uitzondering is de parameter COMPort dit is het nummer van de seriële (COM) poort waaraan de DataTaker fysiek gekoppeld is.

COMport: Nummer van de COM poort waarmee met een DataTaker gecommuniceerd wordt. De waarde dient te liggen in de range 1 .. 32.

Baudrate: Baudrate van de seriële verbinding, integer waarde, voor de DataTaker dient dit 1200 te zijn.

DataBits: Aantal databits voor de seriële verbinding, integer waarde 5..8, voor de DataTaker dient dit 8 te zijn.

StopBits: Aantal stopbits voor de seriële verbinding, tekst veld, voor de DataTaker dient dit ‘ONE’ te zijn.

Parity: Pariteit voor de seriële verbinding, tekst veld, voor de DataTaker dient dit ‘N’ (None) te zijn.

De volgende drie parameters hebben betrekking op de timeouts die gehanteerd worden voor het ontvangen van karakters, deze komen direct overeen met instellingen in Windows, en hier wordt dan ook de verklaring van Microsoft weergegeven.

ReadIntervalTimeout: The maximum time allowed to elapse before the arrival of the next byte on the communications line, in milliseconds. If the interval between the arrival of any two bytes exceeds this amount, the ReadFile operation is completed and any buffered data is returned. A value of zero indicates that interval time-outs are not used.

ReadTotalTimeoutMultiplier: The multiplier used to calculate the total time-out period for read operations, in milliseconds. For each read operation, this value is multiplied by the requested number of bytes to be read.

ReadTotalTimeoutConstant: A constant used to calculate the total time-out period for read operations, in milliseconds. For each read operation, this value is added to the product of the ReadTotalTimeoutMultiplier member and the requested number of bytes.
A value of zero for both the ReadTotalTimeoutMultiplier and ReadTotalTimeoutConstant members indicates that total time-outs are not used for read operations.

EventMask: Event masker van toepassing op de seriële verbinding, integer waarde, voor de DataTaker dient dit 0 te zijn. Deze waarde wordt gebruikt om het event masker op Ce COM poort op Windows niveau n te stellen, dit wordt gedaan met de Windows functie SetCommMask.

Log2DbgViewer: Log niveau voor log info over de seriële verbinding naar debug viewer, des hoger het getal des te meer info getoond wordt. Het niveau kan echter nooit hoger zijn dan de instelling TraceLevel. Waarde = 0 .. 9, voor waarde 0 wordt geen informatie getoond.

Log2Screen: Log niveau voor log info over de seriele verbinding naar commando scherm (DOS-box), des hoger het getal des te meer info getoond wordt. Het niveau kan echter nooit hoger zijn dan de instelling TraceLevel. Waarde = 0 .. 9, voor waarde 0 wordt geen informatie getoond.

Log2File: Log niveau voor log info over de seriële verbinding naar bestand datataker.txt in de programma map, des hoger het getal des te meer info getoond wordt. Het niveau kan echter nooit hoger zijn dan de instelling TraceLevel. Waarde = 0 .. 9, voor waarde 0 wordt geen informatie getoond.

TraceLevel: Absolute maximale niveau voor debug informatie, des hoger het getal des te meer info getoond wordt. Deze instelling begrenst de hiervoor genoemde instellingen. Waarde = 0 .. 9, voor waarde 0 wordt geen informatie getoond.

2.3.3 Datataker.ini: sectie PROTOCOL

```
[PROTOCOL]
Delimiter=,
Decimal=.
Terminator=4
```

In de sectie [PROTOCOL] zijn de instellingen van het DataTaker protocol voor de seriële verbinding opgenomen, deze instellingen zijn voor de DataTaker vaste instellingen.

Delimiter: Het scheidingsteken tussen waarden in het DataTaker protocol, voor de DataTaker is dit ‘,’.

Decimal: Het decimale scheidingsteken, voor de DataTaker is dit ‘.’.

Terminator: Einde markering van een volledig bericht van de DataTaker, alle blokinfo is ontvangen. Voor de DataTaker is dit een karakter met decimale waarde 4.

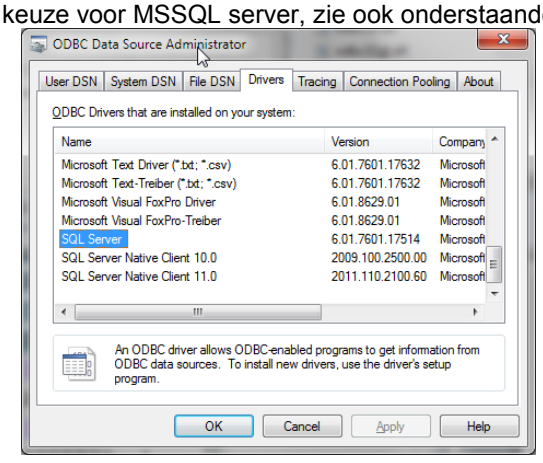
2.3.4 Datataker.ini: sectie DATABASE

In de sectie [DATABASE] zijn de instellingen van MS-SQL 2005 database en koppeling tussen deze database en de applicatie opgenomen. De hier ingegeven waarden komen overeen met de database layout zoals deze is weergegeven in bijlage C: BijlageC-DataTakerDatabase.

```
[DATABASE]
DRIVER={SQL Server}
SERVER=DATATAKER-5\SQLEXPRESS
DBNAME=DATATAKER
UID=sa
PWD=Qw1234
#table Tellerstand definitie
TblTellerstand=[datataker].[dbo].[tellerstand]
TellerStandInsert=INSERT INTO %s VALUES (0,0)
#table BlokInfo definitie
TblBlokInfo=[datataker].[dbo].[blokinfo]
BlokInfoColumns=([pers],[datum_tijd],[hoogte],[gewicht],[temp],[anodeteller]
)
BlokInfoParameters=(?,CONVERT(datetime, ?, 105),?,?,?,?)
```

De parameters DRIVER, SERVER, DBNAME, UID en PWD leggen samen de ODBC connectie string voor de DataTaker applicatie vast.

Driver: De tekenreeks van deze parameter bepaald de te gebruiken ODBC driver voor koppeling van de applicatie met een database. De tekenreeks “{SQL Server}” is de keuze voor MSSQL server, zie ook onderstaande figuur.



Server: De tekenreeks van deze parameter legt de server vast waarop de SQL database zich bevindt.

DBname: De tekenreeks van deze parameter legt de naam van de database vast waarmee een connectie opgebouwd wordt.

UID: De tekenreeks van deze parameter is de User ID, dit is de gebruikers waarmee bij SQL server wordt ingelogd.

PWD: De tekenreeks van deze parameter is het wachtwoord behorende bij de parameter UID.

TblTellerStand: De tekenreeks van deze parameter is de naam van de tabel waar voor iedere pers (of te wel iedere slave) een blokkenteller wordt bijgehouden. De kolomnamen in deze tabel dienen overeen te komen met de slave namen, zie volgende paragraaf.

TellerStandInsert: De tabel voor gebruik van tellerstanden wordt normaal alleen bewerkt met een UPDATE, er is slechts 1 record in deze tabel aanwezig. Is er nog geen record in deze tabel aanwezig dan wordt het hier gegeven insert statement gebruikt: `INSERT INTO %s VALUES (0,0).`

TblBlokInfo: De tekenreeks van deze parameter is de naam van de tabel waar voor iedere pers (of te wel iedere slave) de blok informatie wordt bijgehouden. Onderdeel van de blok informatie is de naam van de pers, deze komt overeen met de naam van de slave, zie ook de volgende paragraaf. De blok informatie is ook zichtbaar in het dialoog venster van de applicatie (onder voorwaarde dat deze geopend is).

BlokInfoColumns: De parameter legt de namen van de 6 kolomnamen in de BlokInfo tabel vast, deze dienen in de database van deze naam voorzien te worden. De standaard namen zijn `[pers],[datum_tijd],[hoogte],[gewicht],[temp],[anodeteller].`

BlokInfoParameters: Voor het toevoegen van blok informatie aan de tabel BlokInfo wordt een insert statement met parameters gebruikt, iedere parameter wordt als ? voorgesteld de volgorde van de parameters komt overeen met de kolom namen in de vorige parameter. Standaard wordt de volgende tekenreeks gebruikt: `(?,CONVERT(datetime, ?, 105),?,?,?,?).` In combinatie met de standaard reeks voor de kolom namen wordt duidelijk dat de kolom datum_tijd de tweede parameter is, en geconverteerd wordt, in dit geval naar Nederlandse notatie.

2.3.5 Datataker.ini: sectie SLAVE1

In de sectie [SLAVE1] zijn de instellingen voor slave 1 van de DataTaker opgenomen, in dit geval is dit de enige slave die fysiek aan de DataTaker gekoppeld is, voor de andere slave secties geldt dat er slechts 1 parameter per slave sectie is: Address=-1. De waarde -1 voor de Address parameter geeft aan dat de slave met id x, niet aan de DataTaker gekoppeld is.

```
[SLAVE1]
Address=3
Name=335
InitString=/c/e/k/L/m/n/O/r/t/u P22=44 P24=4 P39=0
#number of days in this year is added by driver
DateString=D=%d
#current time is added by driver
TimeString=T=%s
StartString=R1-E
ReadString=D T 1+..17+V
TermString=\r\n
ScanString=U
AbortString=/Q
ClearString=CLAST
SlaveString=#%d
```

Address: Adres nummer van een slave, voor de configuratie hier is alleen slave met adres #3 aanwezig. De parameter kan een waarde hebben van 1 .. 99.

Name: Naam van de slave, in dit geval wordt de slave aangeduid met pers ‘335’.

De prameters welke hieronder beschreven worden hebben betrekking op de protocol afhandeling. Reden om deze parameters hier op te nemen is dat per DataTaker meerdere slaves aanwezig kunnen zijn, en de commando’s per slave kunnen verschillen. Voor de opbouw van commando’s wordt verwezen naar bijlage E: BijlageE-Datataker Pocket Reference.

InitString: Initialisatie commando voor een slave.

DateString: Datum string voor een slave, deze dient ondermeer te bestaan uit het (actuele) dag nummer, 1 januari van een jaar is altijd dag 1. De ‘%d’ is een plaats houder voor een integer, dit is volgens de C printf formatering.

TimeString: Tijd string voor een slave, deze dient ondermeer te bestaan uit de (actuele) tijd in de vorm uu:mm:ss. De ‘%s’ is een plaats houder voor een string, dit is volgens de C printf formatering.

StartString: Commando string om vast te leggen wat de basis actie van de DatTaker voor deze slave is, deze string komt overeen met het ‘Init’ commando in het dialoog venster van de applicatie.

ReadString: Commando string om aan te geven wat van de DataTaker geezen dient te worden.

TermString: String waarmee ieder bericht van of naar een slave mee beindigd wordt.

ScanString: Commando string waarmee aan de DataTaker gevragd of voor deze slave gegevens beschikbaar zijn.

AbortString: Command string waarmee een actieve ScanString wordt afgebroken, zonder dat nieuwe blok informatie is ontvangen.

ClearString: Is na een ScanString data ingelezen en verwerkt, dan wordt de data met de ClearString de laatst gestuurde data gewist in de DataTaker.

SlaveString: String om een slave te adresseren, deze dient ondermeer te bestaan uit hetslave nummer, 3 in ons voorbeeld. De ‘%d’ is een plaats houder voor een integer, dit is volgens de C printf formatering.

3 Debug View handleiding

Sysinternals DebugView

DebugView is an application that lets you monitor debug output on your local system, or any computer on the network that you can reach via TCP/IP. It is capable of displaying both kernel-mode and Win32 debug output generated by standard debug print APIs, so you don't need a debugger to catch the debug output your applications or device drivers generate, and you don't need to modify your applications or drivers to use non-Windows debug functions in order to view its debug output.

License

Capabilities

DebugView works on Windows NT 4.0, Windows 2000, Windows XP, Windows Server 2003, Windows 95, Windows 98 and Windows Me. Note: if you want to run *DebugView* on Windows 95 you must install the WinSock2 update, available for free download from Microsoft's Web site.

Under Windows 9x/Me *DebugView* can capture output from the following sources:

- Win32 **OutputDebugString**
- Win16 **OutputDebugString**
- Kernel-mode **Out_Debug_String**
- Kernel-mode **_Debug_Printf_Service**

Under Windows NT and Win2k *DebugView* can capture:

- Win32 **OutputDebugString**
- Kernel-mode **DbgPrint**
- All kernel-mode variants of **DbgPrint** implemented in Windows XP and .NET Server

DebugView also extracts kernel-mode debug output generated at the time of a crash from crash dump files if DebugView was capturing output at the time of the crash.

Starting DebugView

Simply execute the *DebugView* program file (dbgview.exe) and *DebugView* will immediately start capturing debug output. Note that if you wish to capture kernel-mode debug output under Windows NT/2K, you must have the "load driver" privilege.

Menus, hot-keys, or toolbar buttons can be used to clear the window, save the monitored data to a file, search output, and change the window font. In addition, you can toggle on and off capture of kernel or Win32 debug output.

As events are printed to the output, they are tagged with a sequence number. If *DebugView*'s internal buffers are overflowed during extremely heavy activity, this will be reflected with gaps in the sequence number.

Each time you exit *DebugView* it remembers the position of the window, the widths of the output columns, the font selection, configured filters, and the time-stamp mode.

Command-line Options

DebugView supports several command-line options that let you modify its behavior when it starts. Several are relevant when starting *DebugView* as a client on a system that will send debug output across the network to

a *DebugView* instance that displays the output on another computer, and are described in the Remote Monitoring section. However, others modify the behavior of *DebugView* when you run it to display output, and are useful if you want to execute DebugView from a batch file or logon script and want it to capture debug output as soon as it starts. You can have DebugView display all of its command-line options by using the /? option.

Here are the command-line options supported when you run *DebugView* in non-client mode:

debugview [/f] [/t] [/l Logfile [/a] [[/m nnn [/w]] | [/n [/x]]] [/h nn]] [Logfile]

The /f option has *DebugView* skip the filter confirmation dialog when filters were active the previous execution.

The /t option has *DebugView* launch into the system tray, rather than as a window. This has DebugView capture debug output as soon as it starts while not taking up screen real-estate. *DebugView*'s tray behavior is further described in the Running in the Tray section.

The /l option directs *DebugView* to begin writing output to the indicated logfile as soon as *DebugView* executes. The /m option allows you to specify a size limit (in MB) for the log file, and the /a option has *DebugView* append to the logfile if it already exists, rather than overwrite it and /w has the log file wrap when it reaches the maximum size you specify. The /n switch has DebugView create a new log file, named with the date, each day. If you include /x with /n the display clears when a new log file is created.

Finally, the /h switch controls the history depth, which is the count of most recent output lines shown in the *DebugView* display. These options correspond to the logfile commands available through menu items when *DebugView* is running, which are described in Saving and Print.

Capturing Debug Output

Global Capture

You control *DebugView*'s global capture mode by toggling capture-on and capture-off with the  toolbar button, the **Capture|Capture Events** menu entry, or the Ctrl+E hot-key sequence. *DebugView* does not capture any data when its capture-mode is off. The state of Win32 capture and kernel capture determine what kind of debug output (if any) is captured when the global capture mode is on.

Capturing Win32 Debug Output

If you specify, *DebugView* will register to receive and print debug output generated by Win32 programs that call OutputDebugString. The  toolbar button, the **Capture|Capture Win32** menu item, and the Ctrl+W hot-key sequence can toggle this capture on and off. If the Win32 PID options is set (under **Options|Win32 PIDs**) then information identifying processes that generate Win32 debug output is prefixed to each line of Win32 debug output. If you are running *DebugView* on Windows NT/2K, then the process ID of the processes are prefixed in brackets to each line of Win32 debug output. If the option is set and you are running on a Win9x system, then the process name is prefixed in brackets to the output.

If you run DebugView in a remote logon session of Windows 2000 Terminal Services, DebugView adds a Capture Global Win32 menu item to the Capture menu. Whereas the Capture Win32 menu item and associated toolbar button enable and disable capture of debug output in DebugView's local logon session, the Capture Global Win32 menu item lets you enable and disable the capture of debug output that is generated in the console (global) session. Win32 services run in the console session, so this feature lets you capture the output that services generate even when you are running DebugView in another logon session.

Capturing Kernel-Mode Debug Output

You can configure DebugView to capture kernel-mode debug output generated by device drivers and/or the Windows kernel by using the **Capture|Capture Kernel** menu selection,  toolbar button, or the Ctrl+K hot-key. Process IDs are not reported for kernel-mode output since such output is typically not process-context related.

On Windows NT/2K, kernel-mode capture is only possible if the user account in which you run DebugView has the "load driver" privilege. If the account does not have this privilege then DebugView disables the kernel-mode capture and pass-through mode toolbar buttons and menu items.

Under Windows NT/2K, Win32 debug output and kernel-mode debug output originate with two different

sources. Therefore, *DebugView* captures the different outputs into two separate buffer pools, and merges the outputs in the display window according to their relative sequence numbers. While this means that the order of kernel-mode and Win32 output is correctly represented, the update of such information may not be sequential i.e. *DebugView* may display a number of Win32 debug messages, and shortly after merge in kernel-mode debug messages that have interleaved sequence numbers. This is the reason that sequence numbers are represented as 8-digit numbers: the display's listview auto-sorting feature is used by *DebugView* to order merged kernel-mode/Win32 output.

Pass-Through Mode

DebugView can be configured to pass kernel-mode debug output to a kernel-mode debugger or to swallow the output. The pass-through mode is toggled with the **Capture|Pass-Through** menu selection or  toolbar button. The pass- through mode allows you to see kernel-mode debug output in the output buffers of a conventional kernel-mode debugger while at the same time viewing it in *DebugView*.

Inserting Comments

You can insert comments into the output log by selecting Edit|Append Comment. Comments insert into the currently viewed output. Type comments into the dialog followed by the return key and dismiss the dialog when you are done entering comments.

Clearing the Display

An application that generates debug output can cause the DebugView display to clear by printing the string "DBGVIEWCLEAR".

Searching, Filtering, and Limiting Output

DebugView has several features that can help you zoom-in on the debug output you are interested in. These capabilities include searching, filtering, and limiting the number of debug output lines saved in the display.

Clearing the Display

To reset the output window, use the **Edit|Clear Display** menu item,  toolbar button, or Ctrl+X hot-key sequence. This also causes the sequence number to be reset to 0.

Searching

If you want to search for a line containing text of interest you use the find dialog. The find dialog is activated with the Ctrl+F hot-key sequence, the **Edit|Find** menu entry, or the  toolbar button. If the search you specify matches text in the output window, *DebugView* will highlight the matching line and turn off the display's auto-scroll in order to keep the line in the window. To repeat a successful search, use the F3 hot-key.

Filtering

Another way to isolate output that you are interested in is to use *DebugView*'s filtering capability. Use the **Edit|Filter/Highlight** menu item,  toolbar button, or Ctrl-L hot-key to activate the filter dialog. The dialog contains two edit fields: include and exclude. The include field is where you enter substring expressions that match debug output lines that you want *DebugView* to display, and the exclude field is where you enter text for debug output lines that you do not want *DebugView* to display. You can enter multiple expressions, separating each with a semicolon (';'). Do not include spaces in the filter expression unless you want the spaces to be part of the filter. Note that the filters are interpreted in a case-insensitive manner, and that you should use '*' as a wildcard.

As an example, say you want *DebugView* to display debug output that contains either "error" or "abort", but want to exclude lines that contain either of those strings and the word "gui". To configure *DebugView*'s filters for this you enter "error;abort" for the include filter and "gui" for the exclude filter. If you wanted to have DebugView show only output

that has "MyApp:" at the start of the output line and "severe" at the end, you could use a wildcard in the include filter: "myapp:*severe".

Highlighting

DebugView also has another type of filtering: highlighting. If you want output lines that contain certain text to be highlighted in the *DebugView* output window, enter a highlight filter. DebugView implements support for up to five different highlight filters, each with its own foreground and background color settings. Use the filter drop-down in the highlight filter area of the filter dialog to select which highlight filter you want to edit. Use the same syntax just described for include and exclude filters when defining a highlight filter.

To change the colors used for the foreground and background of highlighted lines, click on the Colors button when you have selected the highlight filter you wish to modify. You will be asked to pick colors from a palette, and your choices will be remembered by *DebugView* from run to run.

Saving and Restoring Filters

Use the Load and Save buttons on the filter dialog to save and restore filter settings, including the include, exclude and highlighting filters, as well as the highlighting colors settings.

History-Depth

A final way to control *DebugView* output is to limit the number of lines that are retained in the display. You use the **Edit|History-Depth** menu item,  toolbar button, or the Ctrl+H hot-key sequence to activate the history-depth editor. Enter the number of output lines you want *DebugView* to retain and it will keep only that number of the most recent debug output lines, discarding older ones. A history-depth of 0 represents no limit on output lines retained.

You do not need to use the history-depth feature to prevent all of a system's virtual memory from being consumed in long-running captures. *DebugView* monitors system memory usage and will go into a low-memory state when it detects that memory is running low. *DebugView*'s low-memory state consists of it not capturing further debug output until the low-memory condition is no longer in effect.

Saving, Printing and Logging

DebugView lets you both save and print captured debug output.

Saving Output

You can save the contents of the *DebugView* output window as a text file (.log extension) using the **File|Save** or **File|Save As** menu items, or the Ctrl+S hot-key sequence.

Using **Edit|Copy** or the Ctrl+C hot-key sequence you can copy the debug output contained within selected output lines to the clipboard.

Logging to a File

To have *DebugView* log output to a file as it displays it, use the **File|Log to File** or **File |Log to File As** menu items, the  toolbar button, or the Ctrl+O hot-key sequence. Log file settings you specify include the name of the log file, the maximum size it should be allowed to grow, and whether or not *DebugView* should restart the log or append to it if the file specified already contains output. If you select the wrap option then DebugView will wrap around to the beginning of the file when the file's maximum specified size is reached.

If you select the Create New Log Every Day option then DebugView will not limit the size of the log file, but will create a new log file every day that has the current date appended to the base log file name you enter.

When logging is active the log file toolbar button will look like  . To stop logging simply select the toolbar button or the **File |Log to File** menu item. If the log file's maximum size is reached logging to the file stops and the logging toolbar button changes to  .

If you are monitoring debug output from multiple remote computers and enable logging to a file, all output is logged to the file you specify. Ranges of output from different computers are separated with a header that indicates the name of the computer from which the subsequent records were recorded.

Printing Output

You can use **File|Print** or **File|Print Range** to print the contents of the display to a printer. Choose **Print Range** if you only want to print a subset of the sequence numbers displayed, or **Print** if you want to print all the output records.

The Ctrl+P hot-key sequence corresponds to **File|Print**.

Using the **Print Range** dialog you can also specify whether or not sequence numbers and timestamps will be printed along with the debug output. Omitting these fields can save page space if they are not necessary. The settings you choose are used in all subsequent print operations.

In order to prevent wrap-around when output lines are wider than a page, consider using landscape mode instead of portrait when printing.

Loading Output

Use the File|Open menu item to load a previously saved DebugView log file into the output window.

Options

There are a number of options that let you adjust several characteristics of *DebugView*, including the way that it behaves and looks.

Timing Format

DebugView displays time stamps of captured debug output in one of two formats: as clock-time (the time of day), or as relative time. When displaying relative time *DebugView* represents the time of a debug output record as the difference between its timestamp and the timestamp of the first record in the display. This mode is helpful when you debug timing-related problems. Use the **Options|Clock** Time menu item,  toolbar button, or Ctrl+T hot-key sequence to toggle between clock time and relative time modes.

When DebugView is in clock-time mode you can select the **Options|Show Milliseconds** menu item to have DebugView show timestamps that include millisecond resolution.

Force Carriage Returns

The default behavior of *DebugView* is to buffer output strings in an internal buffer *DebugView* maintains until a carriage- return is encountered or the buffer overflows. This allows applications and drivers to build output lines with multiple invocations of debug output functions.

Select **Options|Force Carriage Returns** to cause *DebugView* to display every string passed to a debug output function on a separate output line, regardless of whether the string is terminated with a carriage return.

Auto Scroll

Use the **Options|Auto Scroll** menu item,  toolbar button, or Ctrl+A hot-key sequence to toggle *DebugView* between auto-scroll and non-auto scroll modes. When in auto-scroll mode *DebugView* will always keep the most recent debug output visible in the display window.

Hiding the Toolbar

You can gain more display space by hiding the *DebugView* toolbar. Use the **Options|Hide Toolbar** menu item or Ctrl+B hot-key sequence to toggle the toolbar's presence. *DebugView* will remember the toolbar state when you exit it and restore the same state the next time you start it.

Win32 PIDS

Setting this option using the **Options|Win32 PIDs** menu item will cause DebugView to prefix Win32 debug output with either the process ID (Windows NT/2K) or the process name (Windows 9x) of the process that generated the output.
Deselecting this option can save screen space if you are not interested in what process generates Win32 output.

Changing the Font

Use the **Edit|Font** menu entry to open a font-selection dialog where you can choose a font that *DebugView*

will use in its output window.

Always on Top

To keep *DebugView* as the top-most window on the desktop, use the **Options|Always On Top** menu item. Selecting the menu item a second time will toggle off the always-on-top mode.

Running in the Tray

Running *DebugView* in the system tray is useful if you want it to capture debug output but do not it to take up space on the desktop or task bar. You minimize *DebugView* to the system try by selecting the Edit|Minimize to Tray menu item, which both changes the menu item to Edit|Minimize to Task Bar and has DebugView appear as an icon on the tray. To reactivate *DebugView* from the tray you double-click on its tray icon. Subsequently minimizing *DebugView* by clicking on its window minimize button will minimize it to the tray. To minimize *DebugView* to the task bar, select Edit|Minimize To Task Bar, after which the minimize button will function as normal.

When *DebugView* is in the tray its icon is colored if global capture is enabled and black-and-white if global capture is disabled. The right-click context menu for the *DebugView* tray icon is a copy of the *DebugView* Capture menu, which allows you to enable and disable global capture, Win32 capture, and kernel-mode capture.

Logging at Boot Time (WinNT/2K/XP Only)

Under Windows NT/2K/XP DebugView can capture kernel-mode debug output generated during the boot process. To have it do so, select the *Capture|Log Boot* menu item, which is enabled when the local computer is Windows NT/2K and the DebugView window is connected to the local computer. When boot logging is enabled, DebugView buffers up to 1MB of debug output beginning at the earliest point in the system's next boot process. You can view the buffered debug output by connecting DebugView to the system, at which time the buffering ceases.

Crash Dumps (WinNT/2K/XP Only)

Under Windows NT/2K/XP you can configure the system to save a dump of physical memory when the operating system crashes. Using this crash dump facility *DebugView* allows you to view any debug output your kernel-mode driver made up to the time of a crash. If your driver is sufficiently instrumented with debug output, then this feature permits users that experience a crash using your driver to send you a debug output file instead of an entire memory dump. You must be capturing kernel-mode debug output with *DebugView* at the time of crash for this option to work.

Use the **Edit|Process Crash** menu item to select a crash dump file for *DebugView* to analyze. *DebugView* will process the file, looking for its debug output buffers. If it finds debug output in the crash dump *DebugView* will prompt you for the name of the log file where it should save the output. You can load saved output files into *DebugView* for viewing.

Remote Monitoring Startup

DebugView has advanced remote monitoring capabilities that allow you to view debug output generated on remote systems from a central location. The remote systems must be accessible via TCP/IP. *DebugView* lets you monitor multiple remote systems simultaneously, using a hot-key or a menu selection to switch between them. If both the computer you are running the *DebugView* GUI on (the server) and the system you want to monitor (the client) are running Windows NT/2K, and they are in the same Network Neighborhood, then *DebugView* will automatically install its client software on the client. For all other combinations you must manually install and start *DebugView*'s client software on the client.

Manual Client Startup

If either the server or the client is running Windows 9x, or the server and client are not mutually accessible via the Windows Network Neighborhood, then you must manually start the *DebugView* client on the client computer. To do this, run the *DebugView* program on the client and specify "/c" as a command-line argument:

dbgview /c [/t] [/s] [/e] [/g]

The *DebugView* client window will appear and indicate that it is waiting for a connection from the *DebugView*

server. After you have started the *DebugView* client use the **Computer|Connect** menu item or Ctrl+R hot-

key sequence of the
DebugView server to open a computer connection dialog. In the dialog enter the name or IP address of the client computer. If the client computer is in the server's Network Neighborhood you can also use the browse button in the dialog to open a view of the Network Neighborhood and visually select the client computer.

If you want to run the client in a "headless" mode, specify "/s" (silent) in addition to the "/c" command-line argument when you start the *DebugView* client. This will cause the *DebugView* client to not display a window, and to remain active until the current user logs out, silently connecting with and disconnecting from *DebugView* servers.

Use the "/e" option when starting the client if you want it to notify you when server connections break. When a server connection is broken and this switch is specified you must close the notification window before the client will accept further connections.

The "/t" option has the *DebugView* client run in the tray. The client presents a gray tray icon when there's no connection to a server and a colored icon when a server is connected. You can open the client window by double clicking the tray icon and store it back in the tray by minimizing the client window.

If you are running *DebugView* from a non-console login on a system with Terminal Services you can direct the *DebugView* client to capture global (console) debug output with the /g

switch. If you specify "/?" *DebugView* will tell you its supported

command-line options.

Automatic Client Startup

Automatic startup is not supported on the Alpha.

If both the client and server are running Windows NT/2K and are in the same Network Neighborhood, there is no need for you to install the *DebugView* client on the client computer. Instead, specify the client computer name or address in the connection dialog as you would if you were connecting to a manually started *DebugView* client, and *DebugView* will automatically install and start the *DebugView* client on the client computer. When you disconnect from the client *DebugView* uninstalls its client software for you. In case you want to clean up client files yourself after a non-graceful exit of the server, the files *DebugView* installs on the client are placed in <winnt>\system32 and include dbgvsvc.exe and dbgv.sys.

The *DebugView* server will always attempt an automatic install, and if that fails it falls back on trying to connect to a manually installed client.

Managing Connection Views

When a remote capture session is established *DebugView* creates a new computer view for the session. The active computer view is the one that has captured output displayed in the *DebugView* GUI, and is identified on the *DebugView* title bar. To switch from one computer view to another select the desired view, which is listed by computer name, in the **Computer** menu. Alternatively, you can use Ctrl+Tab to cycle through the computer views.

The state of global capture, Win32 debug capture, kernel capture, and pass-through for a newly established remote session are all adopted from the current settings of the local view (the view of the computer on which the *DebugView* is executing). Changes you make to these settings only apply to the active computer view.

Disconnecting a Remote Session

When you are through capturing debug output from a remote system, make the view for the computer from which you want to disconnect the active view and then use the **Computer|Disconnect** menu entry to close the session.

When you exit *DebugView* it saves the state of the local view, including the width of the display columns, and *DebugView* applies those settings the next time you start it.

Managing Multiple Windows

DebugView allows you to open multiple DebugView windows on the same computer, allowing you to capture

debug output from different computers into different windows. This is an alternative to connecting to multiple computers from the same DebugView window, and is desirable when you wish to simultaneously view different output sources.

By default, when you start the first DebugView window on a computer it connects with the local computer. This means that it captures and displays any debug output generated on the computer. You can open a second instance of DebugView either by starting it again, or by selecting the File|New Window menu entry.

You can use the Computer|Connect Local menu entry to connect DebugView to the local computer, and choose the Computer|Disconnect menu entry to disconnect from the local computer when it is selected as the active computer view within a DebugView window. Note that only one DebugView instance can be connected to any computer at a given time.

If you start a new *DebugView* window by executing the program again the configuration settings *DebugView* uses reflect those of the last *DebugView* window that was closed. If you start a new *DebugView* window using the **File|New Window** menu entry, the configuration settings are adopted from the window in which you select the menu item.

Reporting Problems

If you encounter a problem while running *DebugView*, please visit the **Sysinternals** web site (www.sysinternals.com) to see if an update has been released that might correct the bug. If the problem has not been fixed, please submit a thorough report of the problem, including information on your system configuration and details on how to reproduce the problem, to: mark@sysinternals.com

4 MSSQL 2005 Database configuration

4.1 Algemeen

De interactie tussen het DataTaker C++ programma en MSSQL2005 databases is schematisch weergegeven in bijlage C: BijlageC-DataTakerDatabase.

Alle SQL servers in gebruik zijn voorzien van een versie van MSSQL Server 2005, er worden twee versie gebruikt: SQL Express op de PC met de C++ datataker applicatie en Advanced server op de overige systemen. Het start punt van de SQL datastroom is de C++ DataTaker applicatie, de datastroom is in onderstaande tabel opgenomen.

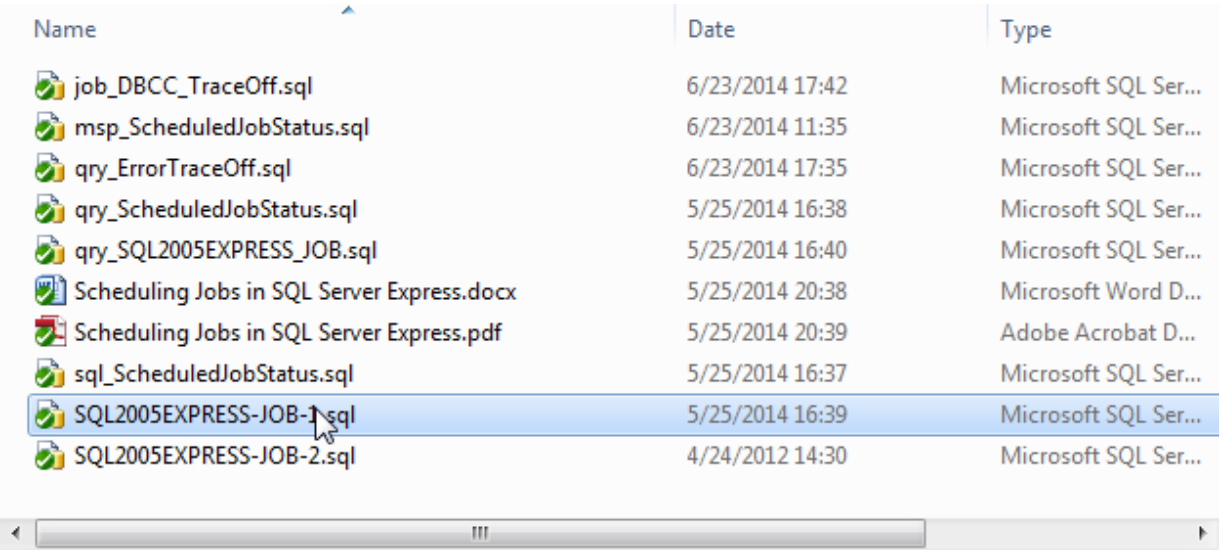
Server	Database	SQL versie
Datataker-4	DataTaker	SQL Express 2005
Datataker-4	ScheduledJobsExpress	SQL Express 2005
Sql-server-1/sql-server-2	Pers2	SQL Advanced 2005
Sql-server-1/sql-server-2	Flmassa	SQL Advanced 2005
Sql-server-1/sql-server-2	Mesal	SQL Advanced 2005

In de volgende paragrafen wordt beschreven in welke volgorde de database, tabellen, gebruikers, procedures etc. geconfigureerd dienen te worden. Alle benodigde configuraties zijn als bron bestand beschikbaar, het bron bestand is een sql-bestand met T-SQL instructies. Deze bron bestanden kunnen direct met door SQL Management Studio uitgevoerd worden.

4.2 Datataker-4 systeem

4.2.1 Database ScheduledJobsExpress

De scheduledJobsExpress database is benodigd om binnen de SQL Express omgeving ‘jobs’ te kunnen uitvoeren. Standaard zijn binnen SQL Express geen jobs aanwezig, met behulp van deze database en Management Objects kan een alternatief voor jobs gemaakt worden, en kunnen taken eenmalig of op interval basis uitgevoerd worden.

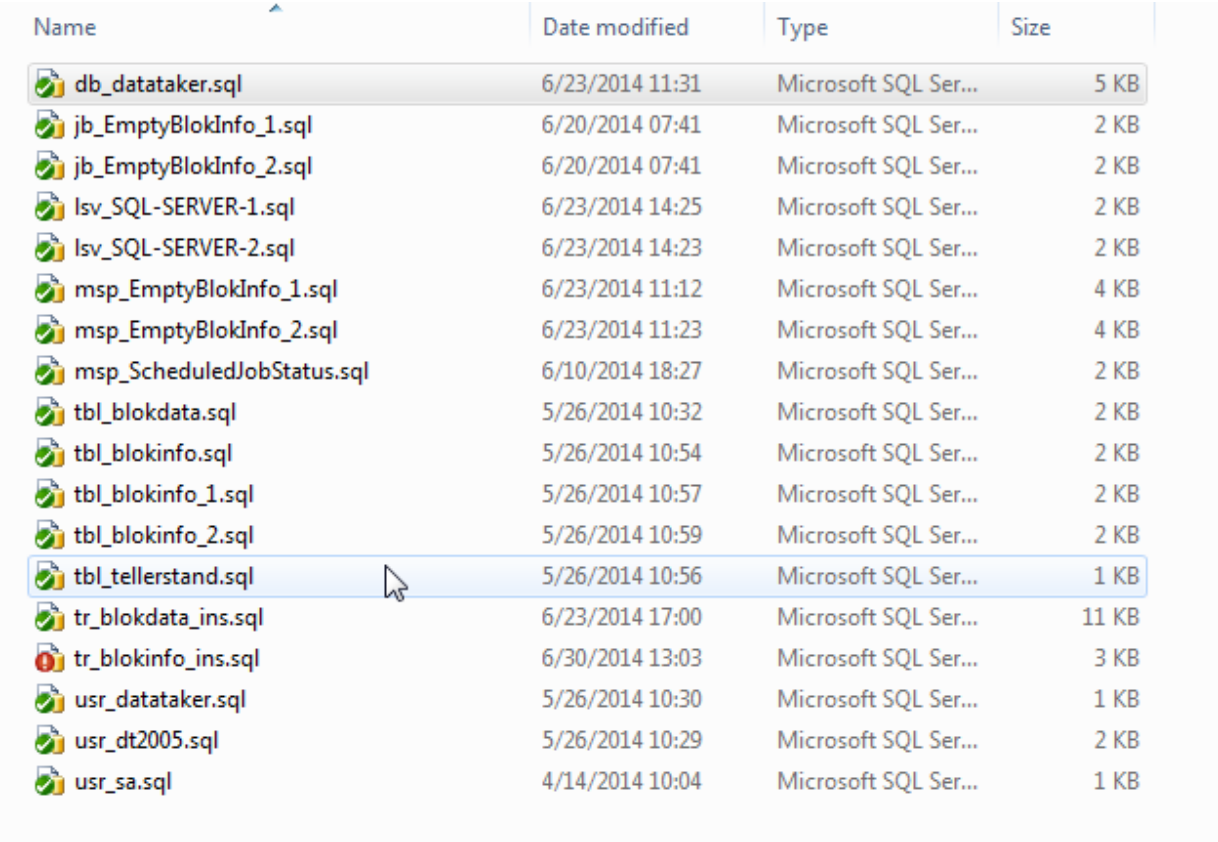


De ScheduledJobsExpress database op het Datataker-4 systeem kan geconfigureerd worden met de volgende T-SQL bestanden, deze dienen in de hier gegeven volgorde uitgevoerd te worden.

- SQL2005EXPRESS-JOB-1.sql — Aanmaken ScheduledJobExpress database en andere basis instellingen voor scheduled jobs.
- msp_ScheduledJobStatus.sql — Stored procedure voor status bewaking van jobs.
- job_DBCC_TraceOff.sql — Job voor uitzetten van tracing in log file van SQL, deze job eenmalig bij opstart van SQL Express (systeem start) uitgevoerd. Doel is kort houden van log bestand.
- qry_ErrorTraceOff.sql — Query ter controle van trace instelling.

- qry_ScheduledJobStatus Query voor status controle van jobs.

4.2.2 Database datataker

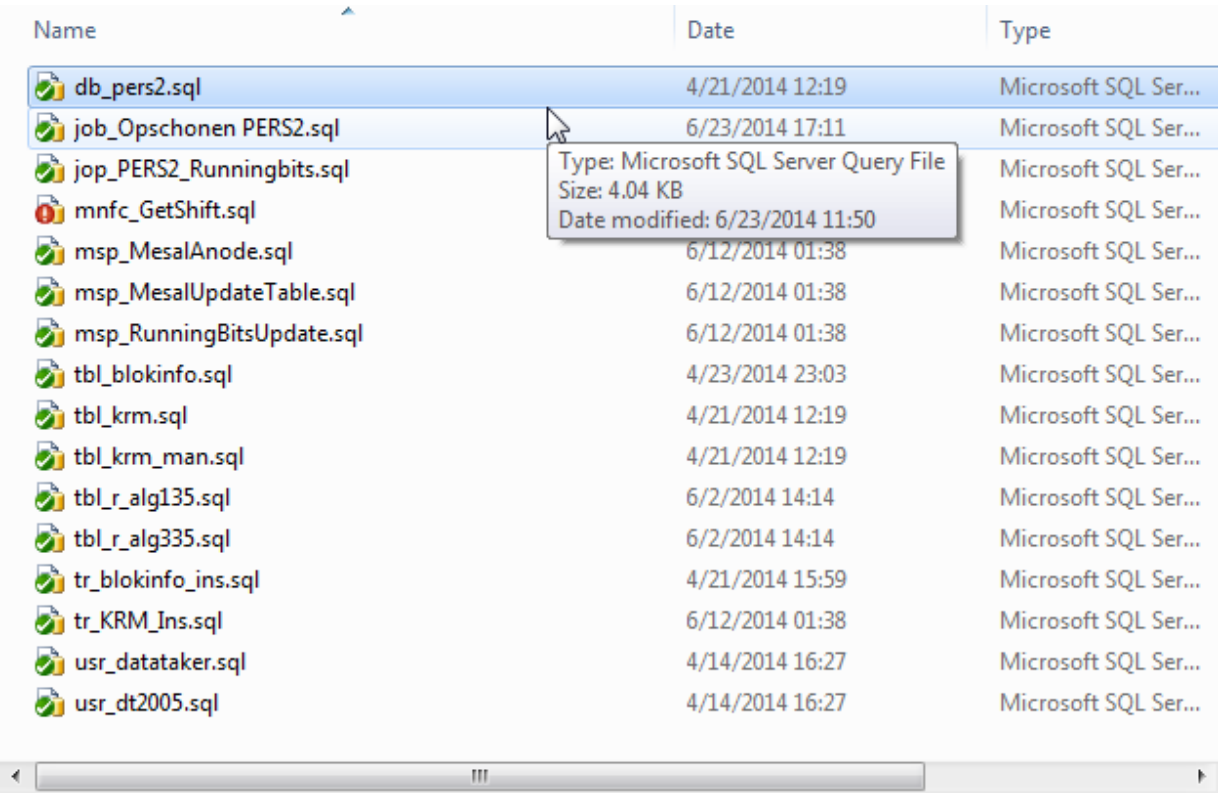


Name	Date modified	Type	Size
db_datataker.sql	6/23/2014 11:31	Microsoft SQL Ser...	5 KB
jb_EmptyBlokInfo_1.sql	6/20/2014 07:41	Microsoft SQL Ser...	2 KB
jb_EmptyBlokInfo_2.sql	6/20/2014 07:41	Microsoft SQL Ser...	2 KB
lsv_SQL-SERVER-1.sql	6/23/2014 14:25	Microsoft SQL Ser...	2 KB
lsv_SQL-SERVER-2.sql	6/23/2014 14:23	Microsoft SQL Ser...	2 KB
msp_EmptyBlokInfo_1.sql	6/23/2014 11:12	Microsoft SQL Ser...	4 KB
msp_EmptyBlokInfo_2.sql	6/23/2014 11:23	Microsoft SQL Ser...	4 KB
msp_ScheduledJobStatus.sql	6/10/2014 18:27	Microsoft SQL Ser...	2 KB
tbl_blokdata.sql	5/26/2014 10:32	Microsoft SQL Ser...	2 KB
tbl_blokinfo.sql	5/26/2014 10:54	Microsoft SQL Ser...	2 KB
tbl_blokinfo_1.sql	5/26/2014 10:57	Microsoft SQL Ser...	2 KB
tbl_blokinfo_2.sql	5/26/2014 10:59	Microsoft SQL Ser...	2 KB
tbl_tellerstand.sql	5/26/2014 10:56	Microsoft SQL Ser...	1 KB
tr_blokdata_ins.sql	6/23/2014 17:00	Microsoft SQL Ser...	11 KB
tr_blokinfo_ins.sql	6/30/2014 13:03	Microsoft SQL Ser...	3 KB
usr_datataker.sql	5/26/2014 10:30	Microsoft SQL Ser...	1 KB
usr_dt2005.sql	5/26/2014 10:29	Microsoft SQL Ser...	2 KB
usr_sa.sql	4/14/2014 10:04	Microsoft SQL Ser...	1 KB

De datataker database op het Datataker-4 systeem kan geconfigureerd worden met de volgende T-SQL bestanden, deze dienen in de hier gegeven volgorde uitgevoerd te worden.

- db_datataker.sql Database definitie.
- usr_datataker.sql Gebruiker ‘datataker’ aanmaken
- usr_dt2005.sql Gebruiker ‘dt2005’ aanmaken
- tbl_blokdata.sql Tabel BlokData aanmaken.
- tbl_blokinfo.sql Tabel BlokInfo aanmaken.
- tbl_blokinfo_1.sql Tabel BlokInfo_1 aanmaken.
- tbl_blokinfo_2.sql Tabel BlokInfo_2 aanmaken.
- tbl_tellerstand.sql Tabel TellerStand aanmaken.
- lsv_SQL-SERVER-1.sql Linked server definitie voor SQL-SERVER-1, zie ook naar bijlage D: BijlageD-DataTaker Linked SQL Server.
- lsv_SQL-SERVER-2.sql Linked server definitie voor SQL-SERVER-1, zie ook naar bijlage D: BijlageD-DataTaker Linked SQL Server.
- tr_blokdata_ins.sql Trigger voor tabel BlokData.
- tr_blokinfo_ins.sql Trigger voor tabel BlokInfo.
- msp_EmptyBlokInfo_1.sql Stored procedure voor legen van tabel BlokInfo_1.
- msp_EmptyBlokInfo_2.sql Stored procedure voor legen van tabel BlokInfo_2.
- jb_EmptyBlokInfo_1.sql Query voor aanmaken van een scheduled job om iedere 5 minuten een poging te ondernemen om records in BlokInfo_1 te kopieren naar SQL-SERVER-1.
- jb_EmptyBlokInfo_2.sql Query voor aanmaken van een scheduled job om iedere 5 minuten een poging te ondernemen om records in BlokInfo_2 te kopieren naar SQL-SERVER-2.

4.2.3 Database Pers2



De pers2 database op het SQL-SERVER-1/SQL-SERVER-2 systeem kan geconfigureerd worden met de volgende T-SQL bestanden, deze dienen in de hier gegeven volgorde uitgevoerd te worden.

- db_pers2r.sql — Database definitie.
- usr_datataker.sql — Gebruiker ‘datataker’ aanmaken
- usr_dt2005.sql — Gebruiker ‘dt2005’ aanmaken
- tbl_blokinfo.sql — Tabel BlokInfo aanmaken.
- tbl_krm.sql — Tabel KRM aanmaken.
- tbl_krm_man.sql — Tabel KRM_MAN aanmaken.
- tbl_R_ALG135.sql — Tabel R_ALG135 aanmaken, running bit.
- tbl_R_ALG335.sql — Tabel R_ALG335 aanmaken, running bit.
- tr_blokinfo_ins.sql — Trigger (insert) voor tabel BlokInfo.
- tr_KRM_ins.sql — Trigger (insert) voor tabel KRM.
- mnfc_GetShift.sql — Value functie voor bepaling shift uit datum en tijd.
- msp_MesalAnode.sql — Stored procedure voor anode gegevensverzameling in Mesal.
- msp_MesalUpdateTable.sql — Stored procedure voor update gegevens in Mesal.
- msp_RunningBitsUpdate.sql — Stored procedure voor actualiseren runningbits van pers.
- jop_PERS2_Runningbits.sql — Job om de runningbits te actualiseren, maakt gebruik van msp_RunningBitsUpdate.
- job_OpschonenPERS2.sql — Job voor opschonen tabellen in database PERS2..

5 DataTaker applicatie als Windows service

5.1 Algemeen

De DataTaker C++ applicatie kan als service gebruikt worden, hiervoor dient een helper programma gebruikt te worden: XYNTService. Hiermee kan ieder programma als service opgestart worden.
Uitvoer van een programma als service heeft als voordeel dat programma automatisch start bij opstart systeem zonder dat een icoon in de menugroep Opstarten geplaats hoeft te worden. Verder kan een service onder Windows 7 niet gestopt worden door een normale gebruiker, onder voorwaarde dat AUC (Advanced user Control) is ingeschakeld

5.2 XYNTService

5.2.1 Introduction

Typically, a Windows service -- used to be called NT service -- is a console application that does not have a message pump. A Windows service can be started without the user having to log into the computer and it won't die after the user logs off. However, it is hard -- sometimes impossible -- to use many existing ActiveX controls within a console application.

On the other hand, MFC and VB applications are Windows applications, so using ActiveX controls in MFC or VB programs is extremely easy. It would be nice to make your MFC and VB programs run like a Windows service, so that:

- They will be started before the user logs onto the computer.
- They will keep running after the user has logged off.

It is possible to write a Windows service as a Windows program, but here I am proposing a much easier solution. I have included with this article the source code for a simple Windows service program that can start and shut down other programs. All you need to do is install this service and modify the INI file. Here are the advantages of using this simple Windows service:

- It can start up to 127 programs of your choice. The started programs behave like Windows services in that they will be running in the background without the user having to log into the machine.
- A user cannot kill the programs started by this service without proper privilege unless, of course, the machine is shutdown.
- You can test and debug your programs outside of the Windows service. For example, you can run your programs in the Visual Studio debugger and step into the source code to find bugs, etc. When it is "bug free," you can deploy it in production, starting it from the Windows service.
- You can run a broad range of programs using this service, including .NET programs, VB scripts and batch files. This service has been used by the open source world to run numerous Java programs.

5.2.2 XYNTService

XYNTService.exe is the name of the executable for this Windows service program. You may get this Windows service from other websites, but Code Project is the official place to get the most recent version. You can freely use and modify the source code included with this article. I am aware that there are other utility programs providing almost the same functionality as XYNTService. However, as you will see, XYNTService has more features and is a lot easier to use. For example, no editing of the registry is required. Here is how to use the program:

- To install the service, run the following at the command prompt: XYNTService -i.
- To un-install the service, run the following at the command prompt: XYNTService -u.

By default, the installed service will be started automatically when you reboot the computer. You can also start and shut down the service from the Control Panel or the Administrative Tools using the Services option. When the service is started, it will create all of the processes you defined in the XYNTService.ini file one by one. When the service is shut down, it will terminate each of the processes it created in reverse order. The XYNTService.ini file should be placed in the same directory as the executable. Here is a sample of the file:

```
[Settings]
ServiceName = XYNTService
CheckProcessSeconds = 30
[Process0]
CommandLine = c:\program files\datataker\datataker.exe
WorkingDir= c:\ program files\datataker
PauseStart= 1000
PauseEnd= 1000
UserInterface = Yes
Restart = Yes
[Process1]
CommandLine = java.exe MyPackage.MyClass
Restart = No
UserName =
```

```
Domain =
Password =
```

The ServiceName property specifies the name you want to use for this NT service. The default name is XYNTService. If you copy the executable and the INI file into a different directory and then modify the ServiceName property in the INI file, you can install and configure a different service!

The sections [Process0], [Process1], ..., etc. define the properties related to each of these processes. As you can see, there are two processes to create in this example. datataker.exe and java.exe are the names of these programs. You can specify the parameters for each of these processes in the CommandLine property. You must specify the full path of the executable file for the corresponding process in the CommandLine property unless the executable is already in the system path.

The CheckProcessSeconds property specifies if and how often you want to check the processes started by XYNTService. If the property has value 0, then no checking is done. If, for example, the property value is 30 then every 30 seconds XYNTService will query the operating system to see if the processes it started are still running. The dead ones will be restarted if the Restart property value (explained later) is defined as Yes for that process. The default value of this property, if you don't specify it, is 600 (10 minutes).

Note: In the previous versions of XYNTService, the ProcCount value specified how many processes were started by XYNTService. This is no longer required. The CheckProcess value is the number of minutes instead of seconds. The current version will work as before if you use CheckProcess instead of CheckProcessSeconds; you should not use both.

The WorkingDir property is the working directory of the current process. If you don't specify this property, then the working directory of the current process will be c:\winnt\system32. The PauseStart property is the number of milliseconds the service will wait after starting the current process and before starting the next process. This is useful where the next process depends on the previous process. For example, the second process has to "connect" to the first process in such a way that it does not run until the first process is finished with initialization. If you don't specify the PauseStart property, the default value will be 100 milliseconds.

When XYNTService is shut down, it will post WM_QUIT messages to the processes it created first and then call the Win32 function TerminateProcess. The PauseEnd property is the number of milliseconds the service will wait before TerminateProcess is called. This property can be used to give a process (started by XYNTService) a chance to clean up and shutdown itself. If you don't specify the PauseEnd property, the default value will be 100 milliseconds.

The UserInterface property controls whether a logged-in user can see the processes created by XYNTService. However, this only works when XYNTService is running under the local system account, which is the default. In this case, processes created by XYNTService will not be able to access a specific user's settings, such as mapped network drives, etc. You can configure XYNTService to run under a user account, which is done easily from the Control Panel or Administrative Tools. Just select Services and then double click XYNTService in the installed services list to bring up a dialog box.

The Restart property is used to decide whether you want XYNTService to restart a dead process. If this property is No, which is the default if you don't specify any value, then the corresponding process will not be restarted. If this property is Yes, then the dead process will be restarted by XYNTService. See the CheckProcess property above on how often dead processes are restarted. You can bounce (stop and restart) any process defined in the INI file from the command line. For example, the following command...

```
XYNTService -b 2
```

...will stop and restart the process defined in the [Process2] section of the INI file. XYNTService can also be used to start and stop other services from the command line. Here are the commands to start (run) and stop (kill) other services:

```
XYNTService -r NameOfServiceToRun
XYNTService -k NameOfServiceToKill
```

In particular, you can use the above commands to start and stop XYNTService from the command line! Important note: You cannot start XYNTService by running it from the command prompt without an argument.

All errors while running XYNTService are written into a log file in the same directory as the executable. The error code in the log file is a decimal number returned by the GetLastError API. You can look for it on MSDN.

5.2.3 Impersonating a user (new feature)

The UserName, Domain, and Password properties are used to impersonate a user. If UserName and Password are not empty, then the corresponding process will be started using the specified user account. If Domain is empty, then the specified user account must be an account on the local machine instead of a network account. In order to impersonate a user, the service must be run by a local system. Also, the corresponding user account must have the "logon as service" privilege. If you configure a service to run under a user account, that account will acquire the "logon as service" privilege. Please note that you won't be able to see the user interface of an impersonated process.

Since we may need to put a user password in the INI file, it is important to make sure that the INI file is only accessible by administrators and the local system. The following command will protect the current folder, making it accessible only by administrators and the local system.

```
cacls . /g system:F administrators:F
```

6 Bijlagen

6.1 Overzicht bijlagen

Bijlage	Titel	Versie	Versiedatum	Status
A	Viso: Systeem configuratie DataTaker applicatie.	1.0	18-02-2014	definitief
B	Viso: Seriele communicatie.	1.0	18-02-2014	definitief
C	Visio: DataTaker Database	1.0	18-02-2014	definitief
D	Visio: DataTaker Linked SQL Server	1.0	18-02-2014	definitief
E	Datataker Pocket Reference			

Bestek = onderdeel bestek en niet aan wijziging onderhevig
SPA = Standaard Proces Automatisering en binnen dit document niet aan wijziging onderhevig
Volgt = wordt opgezet en volgt
Vervallen = Bijlage is vervallen
Toegevoegd = is bijgevoegd (kan nog aan wijzigingen onderhevig zijn).
As Built = status na inbedrijf name.