Introduction

Driver IEC 61850 for Elipse E3 and Elipse Power

Version 2.0

Elipse Software ® 2013

Technical Facts

Technical Facts	
File	IEC61850.DLL
Manufacturer	IEC 61850 Standard
Supported Protocols	IEC 61850 / MMS
Platform	Win32
Driver Version	V2.00
Dependencies	IOKit v2.00 - Elipse E3 and Elipse Power (Elipse SCADA is not supported)

The Elipse Software's IEC61850 driver communicates with protection relays using IEC61850 protocol over Ethernet TCP/IP. Actually the driver has been tested with devices from manufacturers like:
- ABB
- Areva/Alstom
- SEL
- Vamp
- Siemens
- GE
- EFACEC
- Ingeteam

The system has also been tested with switches from manufacturers MOXA and RUGGEDCOM, between others.

The driver allows:
- Communication with several devices at the same driver;
- Tag import directly from devices or from configuration files SCL;
- Support to report unconfirmed messages (Buffered or Unbuffered);
- Polling of variables that are not defined in Reports;
- Support to timestamp and quality information (with 1ms precision);
- Download of disturbance files in Comtrade format, etc..

Additional information about this product can be obtained at the following documents:

- Elipse IEC 61850 Client Driver PIXIT
- Elipse IEC 61850 Client Driver TICS
- Elipse IEC 61850 Client Driver PICS
- Elipse IEC 61850 Client Driver MICS

Configuration

The configuration parameters 'P' are not used. All configurations are made at the driver configuration dialog, with the following configuration pages:
- IEC 61850 Device Config
- IEC 61850 General
- Reports

- Commands
- Files

IEC61850 Device Config

IEC 61850 Device Config: Handles the definition of the devices (servers) we will communicate with.

Browse SCL Files : The relay tags can be created **online** (i.e. when you can communicate with the relay) or **offline**, through import of SCL files. Through this button the driver opens a dialog to choose the .ICD or .SCD files that will be imported. After file selection, for each server description found, a new entry is created at the server list. Right after, a file with .LD extension is created (at the directory specified at **LD File Path** property) with the description of Logical Devices (LD ´s) and Logical Nodes (LN´s). These tags can be imported using the Tag Browsing window.

In case you don´t have the SCL files, it´s possible to configure each relay directly at the list, using the Add, Update and Delete buttons.

Server *:* Defines the device name. It will be used only set an alias for the device at the system, mapping it´s IP address to a name.

IP: Informs the device IP address here. Optionally it´s possible to define the connection port if this is not the standard MMS protocol port 102, using the syntax: X.X.X.X: Port (ex: 192.168.0.10:102).

Backup IP : Inform the Backup IP address of the device if exists. Use the same syntax as the Main IP Address.

PSel *- Presentation Selector* : Device selection value that will be used by OSI Presentation Layer (ISO/IEC 8823). Should be consulted (if fixed) or informed at device configuration, which default value is 1.

SSel *– Session Selector* : Device selection value that will be used by OSI Session Layer (ISO/IEC 8327). Should be consulted (if fixed) or informed at device configuration, which default value is 1.

TSel *– Transport Selector* : Device selection value that will be used by OSI Transport Layer (ISO/IEC 8073). Should be consulted (if fixed) or informed at device configuration, which default value is 1.

Rem AP ID *– Remote Application Process Identificator* : Identifier used by OSI Association Layer (ISO/OSI 8650), using a ASN.1 (Abstract Syntax Notation 1) format. Indicates the data format adopted by AARQ function (Association Request), which default value is 1,1,999,1,1 (iso.1.999.1.1).

Rem AE Qual *– Remote Application Entity Qualifier* : Identifier used by OSI Association Layer as a format. pela camada de Associação como formatador. Should be consulted (if fixed) or informed at device configuration, which default value is 12.

Disable : Disables this device, so when starting the driver, there will be no communication with this particular device.

Use Backup IP : Informs if the Backup IP address will be used.

NOTE: For maximum number of IED´s supported, please consult the section "Driver Limitations", at the end of this document.

IEC61850 General

Allows definition of other driver parameters.

Full Log Details : Enables detailed logging information about event notifications to any tag. (Note: Main log is enabled at Setup Page).

Check Nameplate Mismatches : Enables IED identification parameters (nameplate) on communication startup, in order to check if there was an IED change. If positive, a Cache update (.LD files rebuild) is performed.

Apply Local Offset to Timestamps : The timestamps adopted by IEC 61850 always refer to UTC (Universal Time Coordinate) standard. Checking this option will instruct the driver to apply local offset (TIME ZONE and DAYLIGHT SAVINGS) to the timestamp sent by the device.

No LD Database Scan : Informs that no request for Logical Devices or Logical Nodes will be performed. Shall be used when the driver has only file transfer functions.

Application Category : Whenever sending commands, the driver uses the information selected at this field to set the *OrCat (Origin Category)* property, that specifies the type of application who is sending the command for safety purposes or conflict resolution. The options are: *Bay*, *Station* , *Remote* , *Maintenance* .

Local P Selector : Driver Selection value, used by OSI Presentation Layer (ISO/IEC 8823).

Local S Selector : Driver Selection value, used by OSI Session Layer (ISO/IEC 8327).

Local T Selector : Driver Selection value, used by OSI Transport Layer (ISO/IEC 8073).

Local App ID *– Local Application Process Identifier* : Identifier used by OSI Association Layer (ISO/OSI 8650), at ASN.1 (Abstract Syntax Notation 1) format. Indicates the data format adopted by AARQ (Association Request) function, with default value of 1,1,999,1,1 (iso.1.999.1.1).

LD File Path *:* Standard folder where the driver will create the descriptor files of each Logical Device found (Cache Files), with the objective of speeding up the startup process. At initialization, if a file corresponding to each Device LD is found, the LD will be described from the file, which name is defined by the format SERVER_LDNAME.LD (where SERVER is the IED name and LDNAME is the Logical Device name). The driver offers several ways to detect changes at the IED database in order to update the cache files.

Status Check (ms) : Interval to send a Status message, which must be answered by IED. In a response timeout, the connection health can be checked, forcing an IED disconnection and reconnection. The Status Check value shall be greater than Msg Timeout parameter (see below) and smaller than IOKIT "Disconnect if non responsive" timeout (At Setup Page).

Msg Timeout(ms) : Wait time for a complete message or answer, that can be formed by several intermediate messages. The timeout of each intermediary message (or byte timeout) is defined at IOKIT *Setup* page.

Local AE Qual *– Local Application Entity Qualifier* : Identifier used by Association Layer as format, with default value of 12.

RFC 1006 Source TSAP : This driver uses the RFC 1006 specification as the transport layer of ISSO packets over TCP. For this is necessary to inform the *Transport Service Access Point (TSAP)* used by the driver to establish the connection using this protocol. The default value is 1.

Reports

Prefer Buffered Report Control Blocks (Uses Unbuffered if not available) : The IEC61850 protocol presents the ***Report*** objects, that allows client applications to be notified about data changes. The data to be reported are defined by the user in a ***DataSet*** . Each report can have a single DataSet associated, but it´s possible for a device to have several Reports and DataSets defined.

The Reports can be one of 2 types: Buffered and Unbuffered. Buffered means that all changes that happen at dataset elements during a disconnection are stored in queues (or buffers) in such a way that all changes (since the device have enough memory space, etc..) are stored and sent to client after reconnection. This type of report is used mainly for SOE data (Sequence of Events). On the other hand, the Unbuffered reports only keep the last value.

Both reports can be configured to send events spontaneously, at cyclical intervals or wait for an explicit request from client application (Using a GI - General Interrogation or GetDataSetValues operation).

Anyway, some points must be observed:

- If two client applications (example: 2 different Elipse E3 applications) are connected to a device, only one of them can be connected to each Buffered report. This is because when data are sent, they are removed from the Report internal queue.

- Two or more client applications can connect to the same Unbuffered Report, if not using an "exclusive" report connection.

In this way, the option "Prefer Buffered Report Control Blocks" instructs the driver to search, whenever at tag enters at communication process (what we call advise or scan), among all device´s buffered reports, checking if the tag belongs to the respective dataset.

If tag is found, then the report is enabled (if not already enabled), thus starting to receive data changes. If tag is not found, the driver repeats the same process searching now at the Unbuffered reports, and if consequently found the report is enabled (if not already done).

If tag is still not found, the tag will have no value, unless if the option "Poll Tags not found in any Report" is enabled.

Is the option "Prefer Buffered Report Control Blocks" is disabled, the driver will perform the same procedure, but searching directly at the Unbuffered reports (discarding the buffered search).

Poll Tags not Found in any Report : When a tag is not found in any report, this option allows cyclical reading (polling) of each tag according to its scan rate. **IMPORTANT**: This method is not the most recommended or efficient, causing low update rates and loss of fast events.

Check Report Revision Mismatch **:** Instructs the driver to check the report versions at communication startup. In case a mismatch between the actual version and the cache file version declared, the cache files are updated.

User Defined Report List : If you don´t want the driver to decide automatically which reports to use , it´s possible to define a fixed Report List that should be enabled. This is done through write operations to a specific tag. Below there is an example of a Report List configuration. You can run this scripts at the driver AfterStart event.

TagName: UserReportList
Device: *ServerName*
Item: *UserDefinedReportList*

Example:

```
Sub DRV_61850_AfterStart()
'This script mounts a vector containing the Logical Device configuration and
it's reports
'which will be passed to the driver at the 'UserReportList' tag write
'Please note that it has defined a 2-position vector, in case you want to
add more reports, it will be
'necessary to adjust the vector size

   Dim arr(1)

   arr(0) = Array("LogicalDeviceName","LLN0$BR$brcbEV101")
   arr(1) = Array("LogicalDeviceName","LLN0$BR$brcbEV102")

   Set Cmd = Application.GetObject("DriverName.IEDName.UserReportList")
   Cmd.WriteEx(arr)

End Sub
```

IMPORTANT: **If this option is SET the driver will only complete the startup procedure after receiving the list write**.

RW Report List File (.RPT) : Informs if the fixed report list above shall be read or write to a file, in a way that script execution is not necessary always. The file construction can happen automatically from the script (UserDefinedReportList tag write) or through direct file edit, which must be at standard cache folder (.LD), with the name "IEDName.RPT" (one file per IED).

RPT FILE FORMAT

Number_Of_LogicalDevices
LogicalDeviceN; Number_Of_Reports

ReportName; ReportOption1:ReportOption1Value; ReportOptionN:ReportOptionNValue

Parameter	Description
Number_Of_LogicalDevices	Total Number of Logical Devices which appear at this file
LogicalDeviceN;*Number_Of_Reports*	For each LogicalDevice, insert its name and how many of its reports will be used.
ReportName	Right below each LD name, follows a list of reports and optional fields.
ReportOption;ReportValue	The following optional fields are supported: "DatSet:DataSetName" - Dataset name which shall be associated to the report; "TrgOps:Options" - Report Trigger Options.The value options corresponds to a 6-bit mask, with the following values: Bit 0: Not used Bit 1: DataChange Bit 2: Quality Change Bit 3: Data Update Bit 4: Integrity Bit 5: GI (General Interrogation) "IntgPd:Period" - Interval in ms for Auto-Integrity. "BufTm:Value" - Timer for buffering new events before sending, after first event notification for transmission.(in ms) "Resv:Value" - URCB usage in exclusive mode. Value = 0 or 1 "ResvTms:Value" - Wait time after disconnection to reserve a BRCB for the same client that was connected. "PurgeBuf:Options" - Erases the event queue. Options: 0 = Not Set (do nothing) 1 = Set Always 2 = Set if EntryID Write Fails "RptID:RptName" - Report Identifier "OptFlds:Options" - Optional fields for report message. "Options" corresponds to a 10-bit mask, with the following options: Bit 0 = Reserved Bit 1 = SequenceNumber* Bit 2 = ReportTimeStamp* Bit 3 = ReasonForInclusion* Bit 4 = DataSetName* ** Bit 5 = DataReference Bit 6 = BufferOverflow* Bit 7 = EntryID* Bit 8 = ConfRevision* Bit 9 = Segmentation Obs: fields marked with * are set by default when OptFlds is not specified. ** DataSetName field is obrigatory. Other parameters are fixed and cannot be changed.

RPT FILE EXAMPLE

1
Device;1

LLN0BRBRCB1;ResvTms:1000

User Defined Datasets : If using a pre-defined report list, it´s possible to inform if datasets shall be created dinamically by the client (Elipse IEC 61850 Driver). You can define it by writing to 2 (two) specific tags.

TagName: DeclareDSList
Device: *ServerName*
Item: *DeclareClientDSList*

Shall be informed a descriptor array, where each member is another array of 3 elements, containing:
- Logical Device Name
- Dataset Name (add a @ at the beginning if volatile)
- Index: Unique index for each dynamic dataset

Example:

```
Dim arr
arr = Array("LogicalDeviceName","DataSetName", 1 )
Set Cmd = Application.GetObject("DriverName.IEDName.DeclareDSList")
Cmd.WriteEx(arr)
```

TagName: PopulateDSList
Device: *ServerName*
Item: *PopulateClientDSList*

Shall be informed an array, where each member is another array with 2 elements, containing:
- DataSet Index
- LN/DO/DA Name (Logical Node/Data Object/Data Attribute) at format like "LogicalDevice\LN$DO$DA"

Example:

```
Dim arr(4)
arr(0) = Array(1,"LDName\GGIO1$ST$stval")
arr(1) = Array(1,"LDName\GGIO2$ST$stval")
arr(2) = Array(1,"LDName\GGIO3$ST$stval")
arr(3) = Array(1,"LDName\GGIO4$ST$stval")
arr(4) = Array(1,"LDName\GGIO5$ST$stval")
Set Cmd = Application.GetObject("DriverName.IEDName.PopulateDSList")
Cmd.WriteEx(arr)
```

RW DataSet List File (.CDS) : Indicates if the Dataset list explained previously shall be read/write to a file, in a way that script execution is not necessary always. The file construction can be done automatically from the tag write operations (like explained) or through direct file edition, which shall be at standard cache folder (.LD) with the name "IEDName.CDS" (one file per IED).

CDS FILE FORMAT

Number_Of_LogicalDevices
LogicalDeviceN; Number_Of_DataSets
DataSetName; NumberofMembers
MemberNames1...N

CDS FILE EXAMPLE

1
Device;1

MyDataset1;3
Device/LLN0DCNamPlt
Device/LLN0STMod
Device/LPHD1DCPhyNam

Check BRCB Entry ID : Through this option the user informs that when enabling a Buffered Report (BRCB), the **EntryID** parameter shall be set, containing an unique identifier corresponding to the last report message processed. In this way, after an application start-up or on a redundant server pair switch-over, the report will only send the messages not yet processed.

If this option is set, the application shall keep synchronized the "EntryID" parameters of all Buffered reports in use, and beyond that perform tag writes of the last value at driver startup, informing the last processed value.

More information can be obtained at the section *Using Entry ID* .

RW EntryID List File (.EID) : Indicates if the "EntryID" values explained previously shall be saved into a file, in a way that it is not necessary to execute scripts or tag writes at application startup. The files are saved at the standard cache files (.LD) with the name "IEDName.EID" (one file per IED).

More information can be obtained at the section *Using EID File (Entry ID)* .

Check BRCB TimeOfEntry : Through this option the user informs the driver that is should compare the timestamp of each report message (parameter TimeOfEntry) with the last event received at the redundant Server, when a swicth-over has happened. This option makes the new "hot" server to discard possible repeated events, due to the use of Buffered Reports.

So checking this option, it´s necessary that your application keeps synchronized the "TimeofEntry" values of each Buffered Report in use, writing the last received time at driver startup.

IMPORTANT: This option shall be preferably not used, being replaced by EntryID control.

More information can be obtained at the section *Using TimeOfEntry* .

Polled Integrity Rpt (s) *:* If this value is different from zero (0), the driver will ask a General Interrogation (GI) for all enabled reports, according to the informed interval. This interrogation purpose is to update the tag values, verifying if any data change was lost. This is a safety procedure concept used in other eletric and Power protocols.

Auto Integrity Rpt (s) *:):* If this value is different from zero (0), the driver will configure each **enabled** report (i.e. only the reports that are being used) to send an unsolicited general interrogation (GI) cyclically at the informed rate.

Use Exclusive URCB´s : Defines if Unbuffered Reports (URCB) will be enabled in exclusive mode by default (Property Reserved=1) or not (Reserved=0). An URCB enabled in exclusive mode cannot be used by other clients.

Use Quality Change Trigger : Indicates report Quality Change Trigger option shall be used by default (when not using a fixed report list OR when TrgOps is not defined).

Commands

Asynchronous Write : Allow command requests to return immediately without waiting for a response, since the message has been sent successfully (i.e. there is an active connection). This feature can be used to increase speed of applications that uses a great number of commands. However, this command doesn´t check if there has been a failure at command processing at the device.

Use Single Tag Command Alias : Instructs the driver to create during Tag Browsing (see below) a single tag to send commands. If not checked, a Block with 5 elements will be created (See Section *Communication Tags*).

NOTE: If you are using Elipse E3 as an OPC Server, the command tag can be read (property AllowRead=TRUE). This will make the tag to receive the initial value of the CtlVal property of the referenced command object, so the OPC Client can know the datatype in advance. After sucessfully sending a command, the driver will return an automatic read for the tag,

with the same value written. In this way, the driver parameter WriteFeedBackMode, that defines how tags will receive the feedback value of the operation can be left at 0 – WaitNextRead, preferably.

Use Cmd Tag N1 as Check Condition : In case you are using a single command tag, use this option to inform at Tag N1 parameter the command check condition. N1, in this case, should be a number between 0 and 3, as result of an OR between two bits:
Bit 0: INTERLOCKING
Bit 1: SYNCHROCHECK

Files

Save Comtrade Files : Indicates if the driver needs to search for COMTRADE files (disturbances), saving them at the folder defined at *Comtrade Path* .

Delete Files After Upload : Instructs the driver to erase files after successful transfer.

Browse Root Folder Only : Some devices save the COMTRADE files (correctly) into a folder "\COMTRADE" from device root, and others place the files at the root folder indicating the folder as a part of the name. Ex: "\COMTRADE\Arquivo.cfg". Check this option if the device has this second behavior.

Comtrade Directory Check Interval : Interval in seconds, used to request the list of files to the device. The driver compares the date/time of the last transferred file with the files at the list, transferring the newest files.

Comtrade Upload Interval (s) : In case that more than one file needs to be transferred, the driver will wait this time to perform each transfer, defining a specific time window for this activity. However, event reception and command requests are not stopped while a file is being transferred.

Comtrade Path : Standard directory where COMTRADE files will be dropped.

Fixed Comtrade Path on Device : Ignores the search for a "COMTRADE" path inside device and uses a fixed Path informed at this property.

Tag Browsing

When clicking at Elipse E3 Driver's "Tag Browsing" icon, a Tag Browsing Window will appear, allowing drag and drop of tags found at the devices.

3 Top-folders are shown:

IOKIT: Contains standard Elipse IOKIT Tags, allowing read/write of general parameters, status, etc...

ONLINE: Shows the enabled servers declared at the "IEC16850 Device Config". Upon clicking at each server, the driver will try to communicate with the device, creating right below a new folder for each Logical Device (LD) found at the device. By clicking at the LD, the driver will search and show all tags that may have been found.

OFFLINE: Shows in the same way the servers, but shows LD´s and Tags described at the .LD files, if found. The LD files can be created trough the SCL import or application execution.

To use the tags in your application just drag the desired tag(s) or folder(s) at the right (Tags available at the driver) to the left (Tags of current project).

NOTE: Tag Browsing inserts at each Logical Device, a folder called "DataSets" where you can find all DataSets of that Logical Device that are being referenced by a report.

About IEC61850 Standard

The IEC 61850 standard was developed primarily for substation automation and communication with digital protection relays. Each relay/device has internally the following structure:

Logical Devices (LD) *:* These are the logical devices that are mapped, corresponding to a real device (ex: a Bay) that is mapped inside the relay. A Logical Device is always mapped into a single IED (Intelligent Electronic Device) – Logical Devices cannot be distributed.

Logical Nodes (LN) : These are the real world device functions that are mapped inside a Logical Device. For example, the visual representation of a circuit breaker into a Bay is a Logical Node, with the standard name of XCBR. Generally a Logical Device is formed by several Logical Nodes..

Data Objects (DO) e Data Attributes (DA) *:* information inside a Logical Node are organized into sets of specific data (Data Objects) with each property as Data Attributes. The set of all Data Attributes of a Data are called CDC (Common Data Class).

Functional Constraint (FC) : These are the specific services that can be used on each Data Attribute. The FC's can also make part of the variable "path", depending on device configuration.

The format used by this driver (for all tags) is:

Device: Server:LD
Item: LN$Data$DataAttribute ou LNFCData$DataAttribute

Exemple:

Device: ArevaP139:UPC12AL1Control
Item: XCBR1STMod$stVal

Tag Reference

Communication Tags

The N tag parameters are not used. Tags are addressed using the parameters Device and Item.

Parameter Device: Server:LD
Parameter Item: LN$Data$DataAttribute or LNFCData$DataAttribute

Device: IEDName:LogicalDevice (IEDName is configured at Device Config page)
Item: See below

Item	Oper	Significado
LN$Data$DataAttribute ou LNFCData$DataAttribute	R/W	Any device Tag, will be read according to the Report or Polling configuration. If it is a write parameter, this will be accepted normally.
LNRPurcbXXXX ou LNRPbcrbXXXX	--	The driver doesn't show the report tree, for simplicity reasons and tag economy. All report programming is performed internally. BRCBs shows 2 tags only: TimeOfEntry and EntryID, which can be used to avoid reception of duplicated events.
LNCOxxxx ou LNSPxxxxx	W	Command Block (Control). To simplify the command usage, it shows in place of the command block tree a Tag Block with 5 elements: Operation: Should receive the operation to be performed in plain text: OPERATE, SELECT, SELECTWITHVALUE, CANCEL Value: Value to bet set (depends of object type)

		Time: Time instant (in VB standard, days since 1900) for use in a Time Select operation. Test: Parameter that indicates if the command is a test procedure (value =1) or not (value=0). Check: Verification type made before the command effectuation. Should be a number between 0 and 3, as a result of an OR operation between 2 bits: Bit 0: INTERLOCKING Bit 1: SYNCHROCHECK The command immediate result (i.e. the command acceptance or not) can be obtained in 2 ways: - Through script, by the Block WriteEx method, at the wWriteStatus parameter. See at the next item the description for this value parameter (block status – first element). - Through status block (item below).
LNCOxxxx$Oper, LN$CO$xxxx$TimeActOper LNSPxxxxxSBO, LNCO$xxxx$SBOw, LNCOxxxx$Cancel		If the option *Use Single Tag Command Alias* is set, the driver will create from the tag browser a single tag in place of block command. In this case the command will be like this: - Tag Value will be used at CtlVal property; - Tag Timestamp will be used at T property; - Property Test=0 (fixed); - Property Check = 3 (INTERLOCKING and SYNCHROCHECK(fixed), except if the option "Use Cmd Tag N1 as Check Condition" is set (inform the value at tag N1 in this case). The operation will be defined by the Item property, respecting the suffixes: Oper = Operate SBO = Select Before Operate SBOw = Select Before Operate With Value Cancel = Cancel The TimeActivatedOperate (suffix TimeActOper) will be available when the LN contains the property OpertTm. The operation is similar to the Operate, with the following differences: - Tag timestamp will be used as Operation schedule time; - A scheduled operation can be cancelled through Cancel operation.
LNCOxxxx ou LNSPxxxxx	R	Command Status Block. For each command block (previous item) the driver shows a second block, with 2 elements containing the operation status. The status can be altered as a result of a command deny (ex: command not supported), as an acceptance result (ex: command accepted/not accepted) or as the action effectiveness (breaker open/closed), etc... The first block element – Status – contains a numeric code. The second – StatusText, contains de textual explanation about the numerical code.

		The following codes are used: 0 = Terminated OK 2 = Select Accepted 3 = Select With Value Accepted 4= Cancel Accepted 5 = Operate Accepted 50 = Terminate Fail 100 = AppError:Unknown 101 = AppError:NotSupported 102 = AppError:BlockedBySwitchingHierarchy 103 = AppError:SelectFailed 104 = AppError:InvalidPosition 105 = AppError:PositionReached 106 = AppError:ParameterChangeInExecution 107 = AppError:StepLimit 108 = AppError:BlockedByMode 109 = AppError:BlockedByProcess 110 = AppError:BlockedByInterlocking 111 = AppError:BlockedBySynchrocheck 112 = AppError:CommandAlreadyInExecution 113 = AppError:BlockedByHealth 114 = AppError:1_Of_N_Control 115 = AppError:AbortionByCancel 116 = AppError:TimeLimitOver 117 = AppError:AbortionByTrip 118 = AppError:ObjectNotSelected 200 = WriteError:Object_invalidated 201 = WriteError:Hardware_fault 202 = WriteError:Temporarily_unavailable 203 = WriteError:Object_access_denied 204 = WriteError:Object_undefined 205 = WriteError:Invalid_address 206 = WriteError:Type_unsupported 207 = WriteError:Type_inconsistent 208 = WriteError:Object_attribute_inconsistent 209 = WriteError:Object_access_unsupported 210 = WriteError:Object_non_existent 211 = WriteError:Object_value_invalid 212 = WriteError:Error_unknown 213 = WriteError:Timeout 214 = WriteError:OutOfMemory 215 = WriteError:DecodeError 216 = WriteError:WrongParameters 217 = WriteError:CmdnotSupported

Internal or Action Tags

Device	Item	Oper	Significado
ServerName	*ServerStatus*	R	Returns driver internal status. If the device parameter is the ServerName only, so the status will be related to the whole device: 0 = Initializing 1 = Reading .LD Files 2 = Obtaining the directory (GetServerDirectory) 3 = Processing a single LD (Tag Browsing) 4 = Obtaining file directory

			5 = Processing LDs 6 = Concluding 7 = Finished OK 8 = Concluding with Error 9 = Finished with Error 10 = Reconnecting 11 = Concluded (Idle)
ServerName:LDName	*ServerStatus*	R	If the device parameter is ServerName:LD so the status will be related to the Logical Device: 0 = Idle 1 = Reading .LD file 2 = Obtaining LD directory 3 = Obtaining LN directory 4 = Obtaining Data definition 5 = Creating Tag cache 6 = Discovering Reports 7 = Obtaining DataSets 8 = Obtaining DataSet contents 9 = Obtaining Nameplates 10 = Obtaining Reports versions 11 = Checking versions 12 = Programming Reports 13 = Operation (Runtime) 14 = Concluding 15 = Concluding with Error
ServerName	*ComtradeTransferStatus*	R	Returns the Comtrade file transfer status: 0 = Not connected 1 = Connected 4 = Waiting to List Files 5 = Listing Files 6 = Files transferred (Synchronized) 7 = Waiting to transfer file 8 = Transferring file 101 = Empty file list at IED 102 = Read command formatting errors 103 = Error saving Comtrade file 104 = File transfer error
ServerName	*LastComtradeFileName*	R	Informs the name of the last Comtrade file transferred.
ServerName	*ServerRebuild*	W	Removes the cache files (.LD) and restarts the device communication.
ServerName	*UserDefinedReportList*	W	Allows definition of a fixed report list (User Defined report list)
ServerName	*LastComtradeFileTime*	R/W	Informs the date of last Comtrade file transferred. At application startup, in order to avoid transferring files already collected, this tag shall be written with the date of the last file transferred. We recommend configuring the application to save this value before finishing and recover and set it back at startup.
ServerName	*ComtradeInfo*	R	Returns a 3-element block with the following Comtrade properties: Element0 = *ComtradeTransferStatus* Element1 = *LastComtradeFileTime* Element2 = *LastComtradeFileName*
ServerName	*BRCBActualList*	R	Returns a 4-element block with the list of actually used Buffered Reports:

			Element1: LDName Element2: ReportName Element3: TimeOfEntry Element4: EntryID
ServerName	*URCBActualList*	R	Returns a block with 2 elements with the list of Unbuffered Reports actually in use: Element1: LDName Element2: ReportName
ElipseClientInfo	*InternalClock*	R	Returns a 2-element block with current time and time quality of client internal clock: Element 0: Current Time (TIME) Element 1: TimeQuality Bit 7: Leap Seconds Known Bit 6: not used Bit 5: Sync Error Bit 4-0: Precision (in power of 2^{-y})

-	*ServerInitialBuild*	W	Allows changes at IED parameters before communication startup.
			
			At example below, based on an existing tag "IniBuild" with parameter Item = "ServerInitialBuild", we are enabling an IED of index 0 and changing its TSel parameter:
			
			`Dim CommArr(1)`
			`CommArr(0) = Array("IEC61850.Device[0].Disable",0)`
			`CommArr(1) = Array("IEC61850.Device[0].TSel",1)`
			`Write -1,0,0,3,CommArr`
			`Item("IniBuild").WriteEx(1)`

ServerName	*IPSelect*	R	Informs which device IP is at operation, in case of redundant IP's are informed.
ServerName	*IOKitEvent*	R	Returns IOKIT events for the specified IED, as informed at IOKIT manual - Read Driver Events (ex: connection, disconnection, etc...)
ServerName	*GetFileDirectory*	R	Informs a list of files at IED, after a *GetFileDirectory* request (write operation). If there is no previous request or there is no data available, an empty list will be returned.
ServerName	*FinishedWriteTimeOfEntry*	W	See section "Using TimeOfEntry"
ServerName	*FinishedWriteEntryID*	W	See section "Using EntryID"
ServerName	*IPSwitch*	W	Requests the switch-over from Main IP to Backup IP or vice-versa.
ServerName	*DeclareClientDSList*	W	Declares the list of dynamic datasets. See *Reports* Item.
ServerName	*PopulateClientDSList*	W	Populates the dataset members defined at the item above. See *Reports* item for more information.
ServerName	*GetFileDirectory*	W	Requests a list of IED files, based on a folder name informed at tag value, or at the root folder if no value is informed.
ServerName	*GetFile*		Requests a file transfer, based at the name informed at tag value. The file will be saved with the same name at the Comtrade standard folder.

ServerName	*DeleteFile*		Requests the deletion of a file informed at tag value, directly at IED.
ServerName	*CompareLogicalDeviceDirectory*		Compares the contents of a Logical Device read from a cache file (.LD) with the current IED contents. In a mismatch, a new cache file is generated. The LD Name shall be informed at tag value.
ServerName:LDName	*CompareDataDefinition*		Compares the contents of a Logical Node (LN) read from a cache file (.LD) with the current IED contents. In a mismatch, a new cache file is generated. The LN name shall be informed at tag value.
ServerName:LDName	*GetDataValues*	W	Requests an item read, which may be a LN/DO/DA informed at the tag value. Item values will be returned at its respective tags.
ServerName:LDName	*SetDataValues*	W	Requests a write operation of current values of the item informed at tag value. The item can be a LN/DO/DA.
ServerName:LDName	*GetDataSetValues*	W	Requests a read operation of all items belonging to a dataset informed at tag value. Values will be returned at its respective tags.
ServerName:LDName	*GetAllDataValues*	W	Requests a read operation of all items belonging to a Functional Constraint.Tag value shall be at format "LN$FC". Values will be returned at its respective tags.
ServerName:LDName	*DeleteDataSet*	W	Requests the deletion of a dataset informed at tag value.

Redundancy

There are two methods available at this driver to avoid reception of duplicated events at startup:

- Checking the last TimeOfEntry processed (shall be avoided);
- Setting the last EntryID processed before enabling the Report. **The easiest way to perform this is to enable the generation of EID Files,** containing the last EntryID values.

Using TimeOfEntry

When using the option ***Check BRCB TimeOfEntry*** the application shall follow some steps for correct working:

1) Create at Application (or import by Tag Browsing) the tags "TimeOfEntry" which are available at each Buffered Report used by the driver.

2) If you don´t know if a Buffered Report is being used or not, there is no problem on creating these tags for all Buffered Reports. However through a read in a special block, it´s possible to get this list:

   BlockName: GetBRCBList
   Number of Elements: 4
   Device: *ServerName*
   Item: *BRCBActualList*

   The reading of this block will bring, at any moment, a list with all Buffered Reports in use by the driver, with the 1st element containing the name of the Logical Device (LD); the 2nd the report name, 3rd the last TimeOfEntry and 4th the last EntryID.

   It´s also possible to obtain the list of Unbuffered Reports in use, Just defining in a 2-Element Block Item the text *URCBActualList* .

3) Create in your application an internal tag for each TimeOfEntry that you need to synchronize. The internal tag should have the option Retentive set to TRUE. (OBS: E3 Version 3.2 or earlier can do the same thing with XObject properties, enabled with the retentive option).

4) Each Internal tag or XObject property should receive the value changes of each corresponding TimeOfEntry, using scripts or via associations (links).

5) At application startup, there should be a script that writes the value of each TimeOfEntry according to its corresponding interval tag value, that should be updated by the redundant station. At the end of the process, a special tag should be written indicating that the TimeOfEntry write process has finished. This tag should be configured like this:

   TagName: FinishedWriteTimeOfEntry
   Device: ServerName
   Item: FinishedWriteTimeOfEntry

   You can perform any write operation at this tag to be accepted.

   Warning: If TimeOfEntry Check option is set, driver will only proceed the startup after the write operation of this tag.

Using EntryID

When using the option ***Check BRCB EntryID*** the application shall follow some steps for correct working:

1) Create in your application (or import via TagBrowsing) the tags "EntryID" that are available at each Buffered Report used by the driver. This tag is String type.

2) If you don´t know if a Buffered Report is being used or not, you can get the list of used BRCB´s as explained on item "Using TimeofEntry".

3) Create in your application an internal tag for each EntryID that you need to synchronize. The internal tag should have the option Retentive set to TRUE. (OBS: E3 Version 3.2 or earlier can do the same thing with XObject properties, enabled with the retentive option).

4) Each Internal tag or XObject property should receive the value changes of each corresponding EntryID, using scripts or via associations (links).

5) At application startup, there should be a script that writes the value of each EntryID according to its corresponding interval tag value, that should be updated by the redundant station. At the end of the process, a special tag should be written indicating that the EntryID write process has finished. This tag should be configured like this:

   TagName: FinishedWriteEntryID
   Device: ServerName
   Item: FinishedWriteEntryID

   You can perform any write operation at this tag to be accepted.

   Warning: If EntryID Check option is set, driver will only proceed the startup after the write operation of this tag.

Using EID File (EntryID)

Using the option "R/W EntryID File (.EID)" the driver will generate a file for each IED with EID extension (IEDName.EID), containing the last EntryID´s of each report processed. This file is generated up to 2 seconds after a report reception, and also before a driver stop (or application shutdown).

The file will be generated at the cache folder (.LD).

On driver startup, the system will wait until this file is available to proceed next steps, in order to obtain EntryID values processed until the moment the application was stopped.

In a redundant system, your application is responsible to synchronize this file between the redundant servers.

This example (which must be executed cyclically) copies EntryID files between redundant servers. As the 61850 Driver runs at IOServer.EXE which runs at SYSTEM account, the code uses the program psExec.exe (Sysinternals) to add elevation privileges to execute a .BAT file with an user and password.

More information about the psExec program can be obtained at http://technet.microsoft.com/en-us/sysinternals/bb897553.aspx

```
Sub MyTimer_OnPreset()

Set objWScript = CreateObject("WScript.Network")
strServer = objWScript.ComputerName
Set sw = CreateObject("WScript.shell")
Set fso = CreateObject("Scripting.FileSystemObject")
Set folder = fso.GetFolder("C:\MyAppDir\MyEntryIdDir")

'get the .EID files
  for each file in folder.files
            GetAnExtension = fso.GetExtensionName(file.path)
                                    if GetAnExtension = "EID" then
                                              'copy files
                                              FSO.CopyFile file.path, "C:\ MyAppDir\MyEntryIdDir\EID\"
                                    End if
  next


If strServer = "Server2" then
            sw.run "C:\ MyAppDir\Bat\psExec.exe -i 0 -u UserAccount -p Domain@password cmd /c C:
\MyAppDir\Bat\toServer1.bat",0
End IF
If strServer = "Server1" then
            sw.run "C:\ MyAppDir\Bat\psExec.exe -i 0 -u UserAccount -p Domain@password cmd /c C:\
MyAppDir\Bat\toServer2.bat",0
End IF

End Sub
```

File toServer1.bat

```
copy C:\MyAppDir\EntryId\EID, \\Server1\EntryID
```

File toServer2.bat

```
copy C:\MyAppDir\EntryId\EID, \\Server2\EntryID
```

Dynamic Configuration

This driver allows some configuration parameters to be informed during runtime through script configuration, using the IOKIT write parameters. They are:

Propriety	Parameter	Data Type
Total Device Number	IEC61850.DeviceCount	DWORD
Server	IEC61850.Device[%u].Name	STRING
IP	IEC61850.Device[%u].IP	STRING
Backup IP	IEC61850.Device[%u].IPBackup	STRING
PSEL	IEC61850.Device[%u].PSel	DWORD
SSEL	IEC61850.Device[%u].SSel	DWORD
TSEL	IEC61850.Device[%u].TSel	DWORD
Rem AE Qual	IEC61850.Device[%u].AEQualifier	DWORD
Rem AP ID	IEC61850.Device[%u].AppID	STRING
Disable	IEC61850.Device[%u].Disable	BYTE

Use Backup IP	IEC61850.Device[%u].UseIPBackup	BYTE
Prefer Buffered Report Control Blocks	IEC61850.UseReports	BYTE
User-Defined Report List	IEC61850.UserReportList	BYTE
Poll Tags not found in any report	IEC61850.PollTags	BYTE
Local P Selector	IEC61850.LocalPSel	DWORD
Local S Selector	IEC61850.LocalSSel	DWORD
Local T Selector	IEC61850.LocalTSel	DWORD
Local AE Qualifier	IEC61850.LocalAEQualifier	DWORD
Local App ID	IEC61850.LocalAppID	STRING
RFC 1006 Source TSAP	IEC61850.SourceTSAP	DWORD
App Category	IEC61850.AppCategory	STRING
Polled Intg Rpt	IEC61850.RptGI	DWORD
Auto Intg Prt	IEC61850.IntgPd	DWORD
Use Quality change trigger	IEC61850.UseQChgTrgOps	
Conformance Blocks (internal use)	IEC61850.CBB	STRING
Services (internal use)	IEC61850.Services	STRING
LD File Path	IEC61850.LDPath	STRING
Comtrade Path	IEC61850.ComtradePath	STRING
Diretorio Comtrade no Device (internal use)	IEC61850.ComtradeDir	STRING
Msg Timeout	IEC61850.Timeout	DWORD
Full Log Details	IEC61850.DetailedLog	BYTE
Save Comtrade Files	IEC61850.SaveComtrade	BYTE
Delete Files after upload	IEC61850.DeleteComtrade	BYTE
Comtrade Directory Check Interval	IEC61850.CheckComtrade	BYTE
Comtrade Upload Interval	IEC61850.UploadComtrade	DWORD
Check Report Time of Entry	IEC61850.RedundantBRCB	BYTE
Status Check	IEC61850.StatusCheck	DWORD
Asynchronous Write	IEC61850.AsyncWrite	BYTE
Use Single Tag Command Alias	IEC61850.SingleTagCmdAlias	BYTE
Apply Local Time Offset to Timestamps	IEC61850.ApplyLocalTime	BYTE
Use Cmd Tag N1 as check condition	IEC61850.SingleTagCmdN1	BYTE
No LD Database Scan	IEC61850.NoLDScan	BYTE
Browse Root Folder Only	IEC61850.ComtradeBrowseRootFolderOnly	BYTE
Use Exclusive URCB	IEC61850.ReserveURCB	BYTE

Driver Limitations

Driver IEC61850 is released in two different license types by Elipse Software:

IEC61850.dll: Allows communication with until 25 IED´s, to guarantee communication performance. For more relays, a new driver license shall be used.

Power_IEC61850.dll: Allows communication with a single IED, using a standard license which allows replacement by other power protocol. For a complete list of power drivers that share this license, please consult Elipse Software.

Version History

Version	Date	Author	Comments
V 2.00	20/Mar/2013	M. Salvador	- Migration to IOKIT 2.0 - Generation of Entry ID Files (EID) - Support to dynamic datasets
V 1.01 until Build 27	17/Mar/2012	M. Salvador	Includes the following changes since version 1.01 Build 1: - Build 2: Fixed Connection/Disconnection Process - Build 3: Report treatment with BitInclusion incoherent with Dataset size - Build 4: Fixed Connection/Disconnection process (created new verification thread) - Build 6: Changed treatment of read exceptions - Build 7: Reject message was being treated as Confirmed Error in some cases - Build 8: Fixed lists processing bug generated by Build 6 - Build 9: Treatment of invalid real numbers (QNAN) - Build 10: Fixed Connection Status check - Build 11: Fixed command execution status - Build 12: Additional options for comtrade file search - Build 16: Fixed possible deadlock between reconnection semaphores - Build 17: Comtrade Block Status - Build 18: Support to EntryID at Report startup - Build 19: Support to add devices in runtime - Build 20: Propagation of IOKIT events - Build 21: Fixed oscillography collection for relays AREVA/ALSTOM - Build 22: Support to Double type to mms_float - Build 23: DNSNames with port specification at IED IP Address - Build 24: Fixed import of SEL SCL, additional protection to avoid writings and pollings during a reconnection - Build 25: Support to user session requirements at Presentation CPA PPDU - Build 26: Fixed memory leak when disconnected - Build 27: COTP changes, new semaphore for reconnection process
V 1.01	13/Jan/2010	M. Salvador	Includes the changes between version 1.0.0.31 and 1.0.0.45: -Build 31: Logs TimeZone at driver startup -Build 32: Considers TimeZone Unknown as valid for local offset calculation -Build 33: Changed the way report re-enable works after reconnection -Build 34: Logical Device creation is not allowed except during .LD file reading or after a GetServerDirectory (Case 11378); Removal of left zeros at IP addresses (Case 11379)

Version	Date	Author	Comments
			-Build 35: Removal of left zeros was not considering the presence of port indication after IP address Ex: 127.0.0.1:102 -Build 36: Change at Build 33 removed, now reprogramming reports completely after a reconnection -Build 37: Creation of tags IPSelect, IPSwitch, removed property TimeofEntry of tag pollings (if enabled) -Build 38: Implemented a thread to check status of communication TCP/IP Hosts -Build 39: 20 Tag Limit for polling and update of RptEna property, used when there are tags with no report associated (Limbo) -Build 40: Fixed problem of wrong random reception at SESSION and PRESENTATION layers -Build 41: Changed parameter "Nesting Level Requested" from 5 to 6 (INGETEAM) - Build 42: GetVariableAccessAtributes requests one folder below if current folder fails (INGETEAM) ; EntryID and OptFlds now are not requested at OptFlds when report is Unbuffered - Build 43: Fixed potential failure inside MMS::SendRequest which may happen when there is a connection failure; Improved read performance for status tags; TCP/IP disconnection upon failure of MMS::Status request (or response) was not performed in some situations - Build 44: Rebuild request is now asynchronous - Build 45: Increased system starup performance, mainly at .LD file reading
V 1.00 Build 30	20/Jan/2010	M. Salvador	- case 11097: Correction of encoding of integer numbers for commands – ctlNum property - Case 11104: Parameter WriteStatus of Elipse E3 WriteEx operations were not informed in some cases.
V1.00 Build 28	30/Nov/2009	M. Salvador	- Added comtrade support; - Async commands; - Individual tags for commands
V 1.00 Build 14		M. Salvador M. Bihre	- Fixed ICD/SCD support.
V1.00	10/Set/2008	M. Salvador M. Bihre	- original driver version.

Headquarters
Rua 24 de Outubro, 353 - 10º andar
90510-002 Porto Alegre
Phone: (+55 51) 3346-4699
Fax: (+55 51) 3222-6226
E-mail: elipse-rs@elipse.com.br

Taiwan
9F., No.12, Beiping 2nd St., Sanmin Dist.
807 Kaohsiung City - Taiwan
Phone: (+886 7) 323-8468
Fax: (+886 7) 323-9656
E-mail: evan@elipse.com.br

Check our website for information about a representative in your country.
www.elipse.com.br
kb.elipse.com.br
forum.elipse.com.br
www.youtube.com/elipsesoftware
elipse@elipse.com.br

Gartner 2014 Cool Vendor

OPC FOUNDATION MEMBER

Gartner, Cool Vendors in Brazil 2014, April 2014.
Gartner does not endorse any vendor, product or service depicted in its research publications, and does not advise technology users to select only those vendors with the highest ratings. Gartner research publications consist of the opinions of Gartner's research organization and should not be construed as statements of fact. Gartner disclaims all warranties, expressed or implied, with respect to this research, including any warranties of merchantability of fitness for a particular purpose.

Microsoft Partner
Gold Independent Software Vendor (ISV)