EDS Data Interfaces User's Guide

Version 9.2
09.08.2019

Copyright Notice

Since the equipment explained in this document has a variety of uses, the user and those responsible for applying this equipment must satisfy themselves as to the acceptability of each application and use of the equipment. Under no circumstances will Emerson Process Management be responsible or liable for any damage, including indirect or consequential losses resulting from the use, misuse, or application of this equipment.

The text, illustrations, charts, and examples included in this manual are intended solely to explain the use and application of the OvationTM Unit. Due to the many variables associated with specific uses or applications, Emerson Process Management cannot assume responsibility or liability for actual use based upon the data provided in this manual.

No patent liability is assumed by Emerson Process Management with respect to the use of circuits, information, equipment, or software described in this manual.

This manual is printed in the USA and is subject to change without notice.

Emerson Process Management

Power & Water Solutions
200 Beta Drive
Pittsburgh, PA 15238
USA

E-Mail: Technical.Communications@Emerson.com
Web site: https://www.ovationusers.com

Contents

CHAPTER 1

Ovation Interface for Windows OS

IN THIS SECTION

1.1 Ovation Interface

The **Ovation Interface** is a set of tools designed for:

- importing process points from Ovation systems,
- updating process point dynamic data (values, statuses),
- reading alarms from Ovation OPC Server (SOE/OE).

The functions mentioned above are realized by different executable, installed by one of the following packages:

EDSW3 PACKAGE	OVATION SOFTWARE LEVEL
EDSW3_2.1-rel.x.x.x.x.msi	2.1
EDSW3_2.2-rel.x.x.x.x.msi	2.2
EDSW3_2.3-rel.x.x.x.x.msi	2.3
EDSW3_2.4-rel.x.x.x.x.msi	2.4
EDSW3_3.1-rel.x.x.x.x.msi	3.1
EDSW3_3.2-rel.x.x.x.x.msi	3.2
EDSW3_3.3-rel.x.x.x.x.msi	3.3
EDSW3_3.3.1-rel.x.x.x.x.msi	3.3.1
EDSW3_3.5-rel.x.x.x.x.msi	3.5

where **x.x.x.x** signifies the EDS software version. The installation is done by running the appropriate EDSW3 package.

1.2 Installing the EDS Ovation Interface

To install EDS Ovation Interface perform the following steps:

1. Login to the MS Windows environment with administrative privileges.
2. Run setup program:

EDSW3_y.y-rel.x.x.x.x.msi

where **x.x.x.x** signifies the_EDS software version and **y.y** the Ovation software level. The software installs by default in the following directory:

C:\Program Files (x86)\EDS92\W3_y.y

> ***Note:*** *The path segment "Program Files (x86)" is a characteristic of 64-bit systems. On a 32-bit system it will be replaced by "Program Files". Ovation operates on 32-bit, hence the "...(x86)" path.*

This path can be modified during the process of installation.

Figures below display typical steps during EDS Ovation Interface installation. Follow the installation wizard's instructions to install the EDS Ovation Interface. Use default installation options.

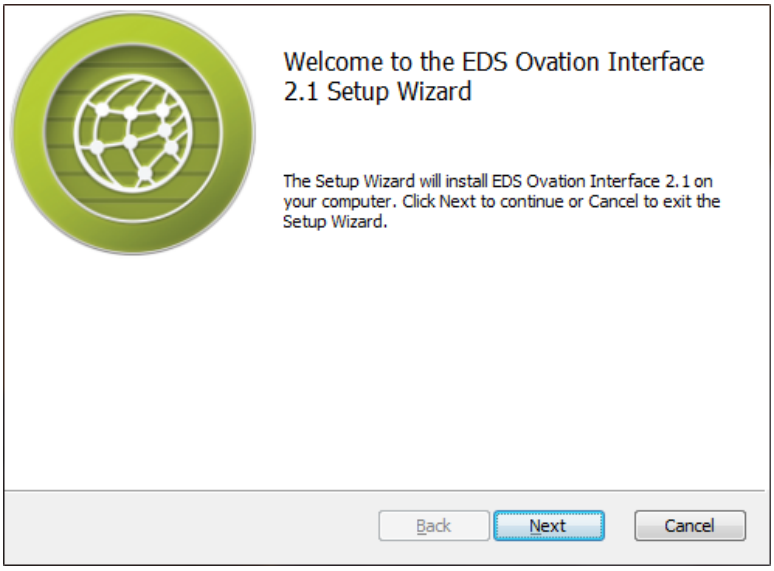


Figure 1.1 EDS Ovation Interface installation screen

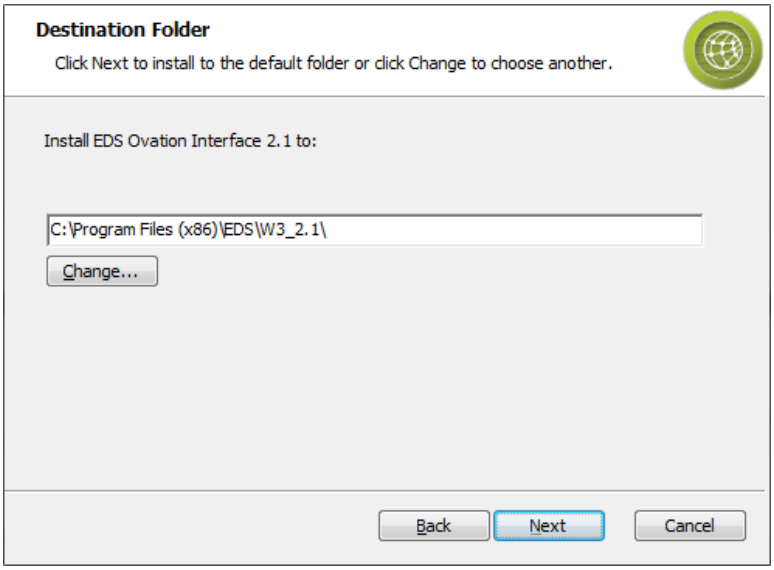


Figure 1.2 EDS Ovation Interface installation screen - destination directory

After the **EDS Ovation Interface** installation, the configuration utility starts automatically. Figures presented in next sections display default EDS Ovation Interface programs configurations.

1.3 Adding new feeder instance

To add a new feeder instance:

1. Click the add new  button on the right of the identifier name.

2. **New configuration** window opens:

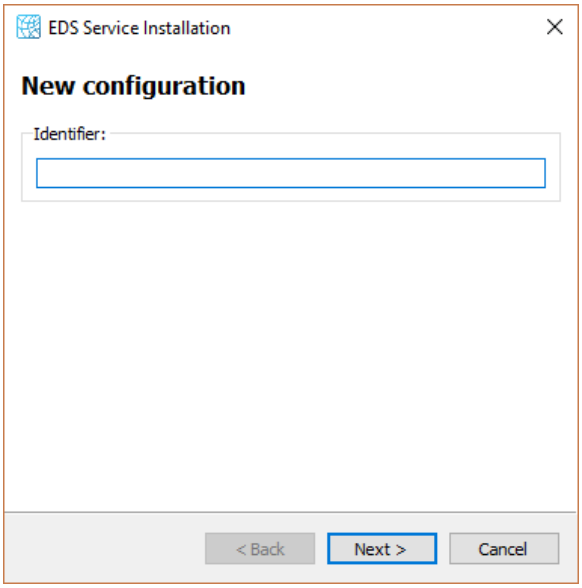
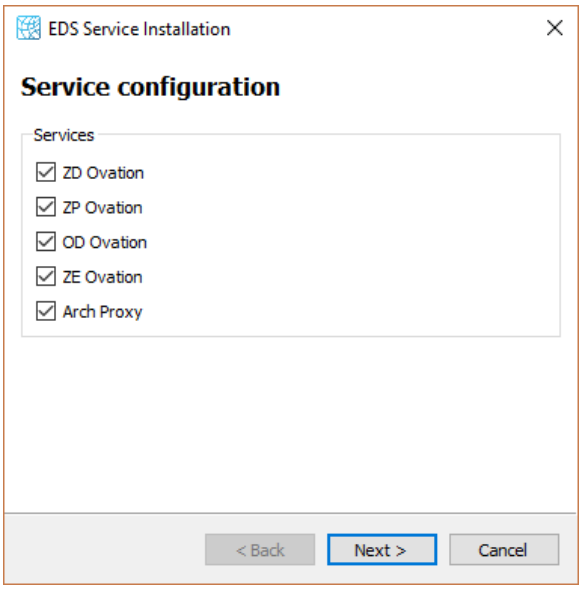


Figure 1.3 New configuration window

3. Enter the name (identifier) for the configuration and click next.
4. Select the check boxes to create the desired services and click next:



Note: *ZD Ovation and ZP Ovation services are added by default.*

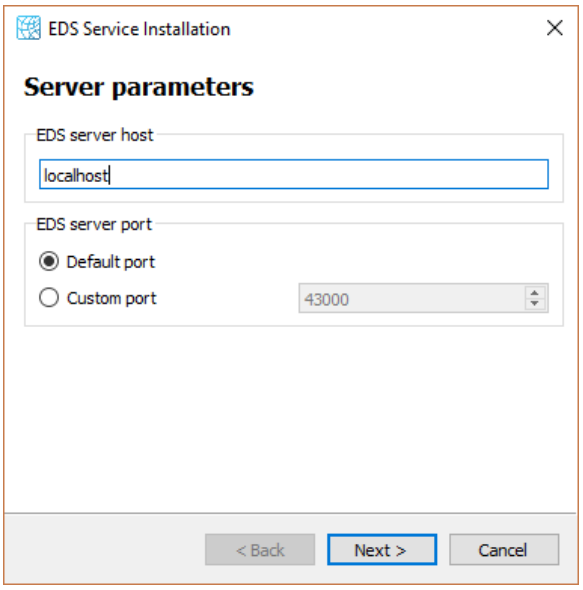


Figure 1.4 Server parameters window

5. Enter server parameters and click next.

- EDS server host name
- EDS server port: default of custom

6. Enter the Point source name (ZD) and click next:

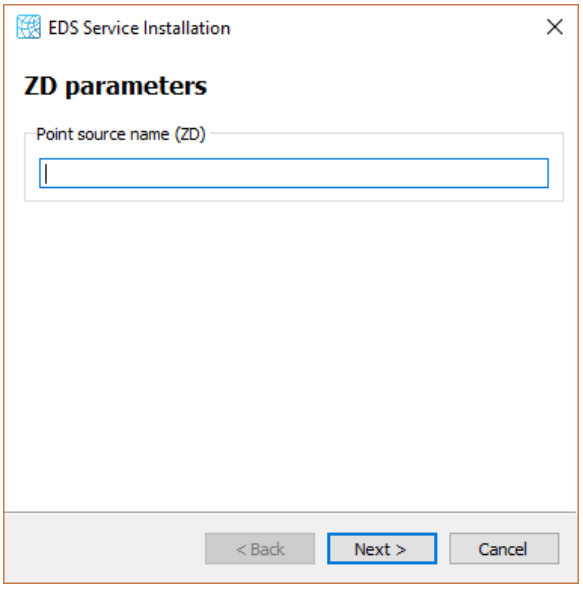


Figure 1.5 ZD parameters window

7. Choose ZP listening port (default or custom) and click next:

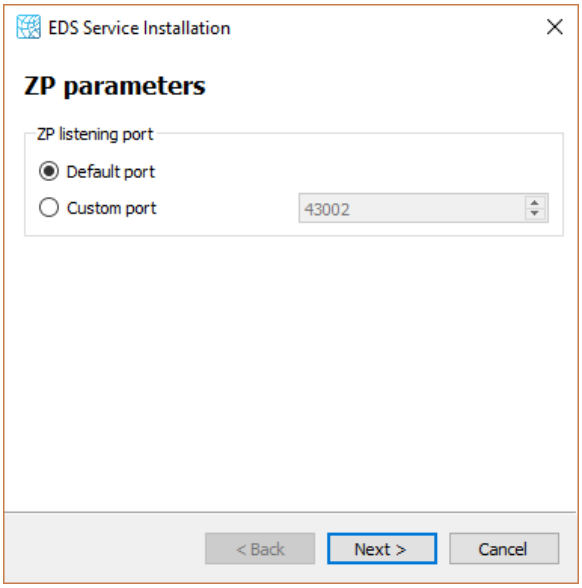


Figure 1.6 ZP parameters window

8. Choose whether or not to buffer data and click next.

- buffer data
- fill in gaps automatically (active only with buffering turned on)

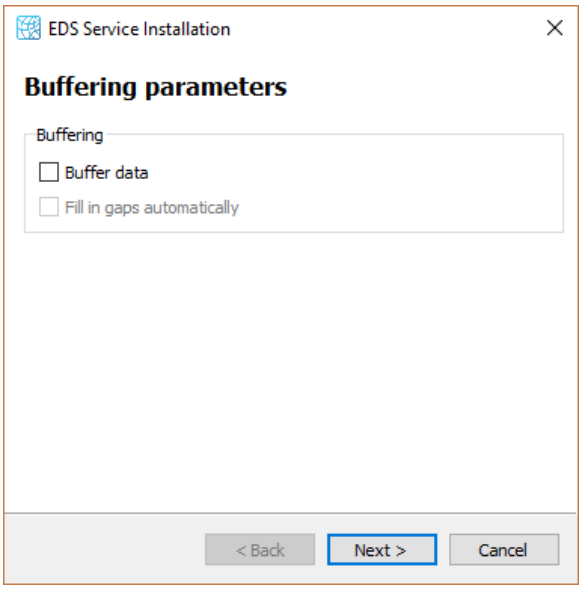


Figure 1.7 Buffering parameters window

9. Select desired logger parameters (or leave default), and click next:

- debug level (0-7)
- output
  - file (with the possibility to set file size limit and file count limit)
  - console

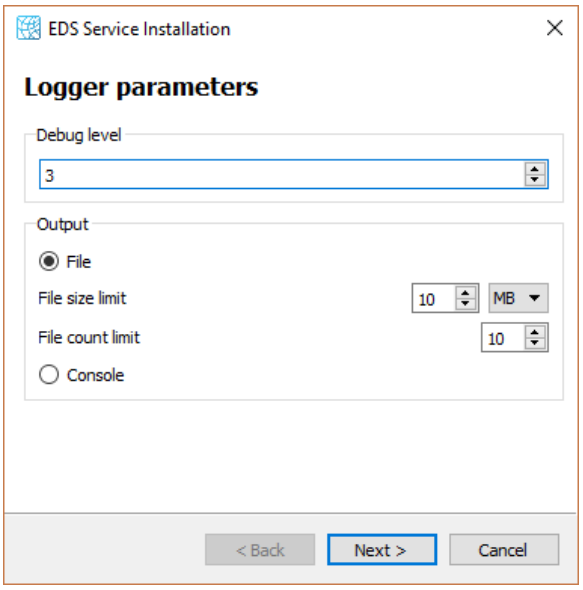


Figure 1.8 Logger parameters window

10. The configuration summary window will be displayed with a summary of all selected options:

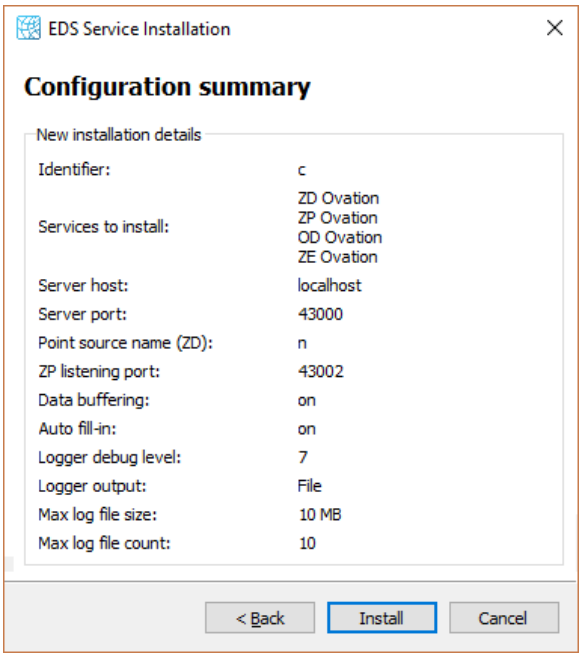


Figure 1.9 Configuration summary window

11. Click "Install" to configure and install selected services.

12. Installation summary window will confirm the success of the installation:

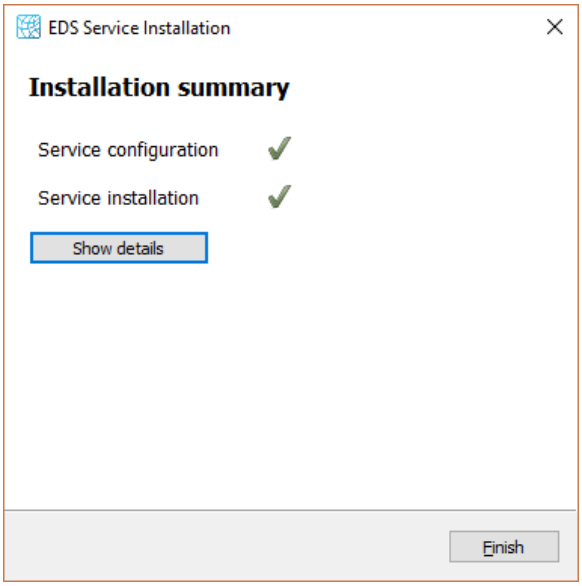


Figure 1.10 Installation summary window

13. Click "Finish" to complete the process of adding a new feeder instance. New instance will be available under the selected name (identifier) from a drop-down list of identifiers.

1.4 Configuring the Ovation Interface

The Ovation Interface programs are configured during the installation. If some configuration parameters modifications are required, the **w3config.exe** program should be used in the following way:

1. Open the **w3config.exe** program, by default found in `C:\Program Files\EDS92\W3_y.y\` with administrative privileges. This opens the configurator.

The **W3config** consists of the following functions:

BUTTON	DESCRIPTION
	Reloads options from registers
	Saves options to registers
	Import configuration settings from a file
	Export configuration settings to a file
	Start the system interface for management of Services
	Opens a logger window displaying application messages.
	Opens '**Help**' window

The top of the window contains parameters shared by all Ovation Interface applications:

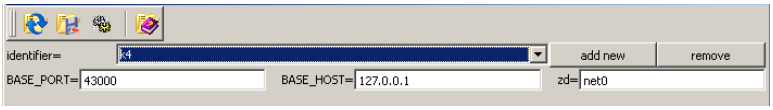


Figure 1.11 Parameters shared by all Ovation Interface applications

2. Set-up the EDS Server IP address (**BASE_HOST=** parameter), the name of data source (**zd=** parameter) and leave the other parameters unchanged.

PARAMETER	DESCRIPTION
identifier=	**<string>** A drop down list containing all configured scanners. The default name is **enterdb**. This name identifies the scanner in MS Windows Service names. The name specified in the **add new** add new dialog box is used to create a new instance of a scanner. To delete a scanner (remove all the services related to a specified identifier), press the **remove** remove button.
BASE_PORT=	**<number>** The port number on which EDS Server communicates.
BASE_HOST=	**<name | IP>** The host name or IP address of EDS Server
zd=	**<string>** This parameter identifies which process points should be updated by the scanner

***Note:** The name specified in **identifier=** field cannot contain special characters.*

Parameters common for all tabs

debug	Level of significance of messages to be logged; numbers have the following meaning: ■ **0** - Emergency, system is unusable ■ **1** - Alert, action must be taken immediately ■ **2** - Critical, critical conditions ■ **3** - Errors, error conditions ■ **4** - Warning, warning condition ■ **5** - Notice, normal but significant condition ■ **6** - Informational ■ **7** - Debugging EDS effectively uses levels from **3** to **7**. When **debug:** is set to **5**, then it results in a printout of messages of levels **5**,**4** and **3**. The default level is **3** (log errors only).
logger	Logger initialization string. For more information see *1.4.7 Logger Initialization String on page 34*.
subsystems	List of names of the EDS subsystems that need to be logged. A special keyword **ALL** selects all subsystems, which is the default and recommended setting for normal operation.

1.4.1 ZD Ovation tab

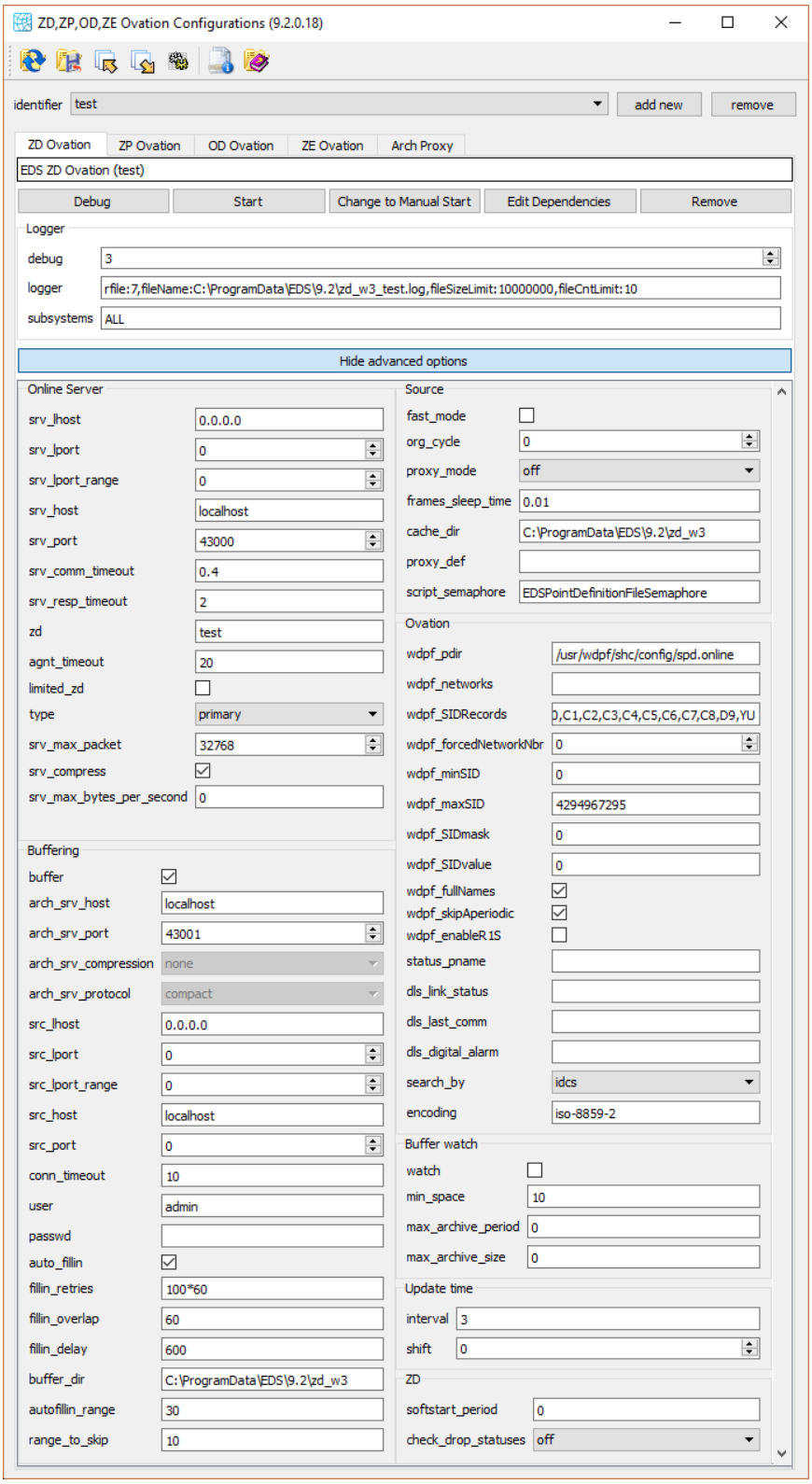


Figure 1.12 ZD Ovation tab

PARAMETER	DESCRIPTION
	Online Server section
srv_lhost	**<name | IP>** - local host for connection with **Online Server**.
srv_lport	**<number>** - local port for connection with **Online Server**.
srv_lport_range	**<number>** - local port range for connection with **Online Server**.
srv_host	**<name | IP>** - host of **Online Server**.
srv_port	**<number>** - port of **Online Server**.
srv_comm_ timeout	**<seconds>** - communication timeout time in seconds when the application waits to receive acknowledge from the destination (by default it is set to **0.4** seconds).
srv_resp_timeout	**<seconds>** - response timeout time in seconds after which the application retransmits data packet. If it did not receive acknowledges from the destination. It may be increased if network delays are big (by default it is set to **2.0**).
zd	**<name>** - name of the source; specifies which process points should be originated by this application.
agnt_timeout	**<number>** - agent timeout, describes the number of seconds after which Server will change status of this scanner from on-line to timedout when it stops updating process point values. Default value is **20** seconds
type	**<primary | backup | offline>** - specifies source of output type. You can choose : **primary** - updates the process points normally;, **backup** - checks if a primary in Online and in case of emergency switches to **backup**; offline - **synchronize**, but doesn't update process points.
srv_max_packet	**<bytes>** - max packet size for connection with **Online Server** .
srv_compress	**<on | off>** - enables compression for connection with **Online Server**.
srv_max_bytes_ per_second	**<number>** - limit on the number of bytes sent on the connection with **Online Server** (setting to 0 disables the check).
	Buffering section
buffer	**<on | off>** - saves scanned values to an external file; if enabled (set to **on**), the scanner archives the values of process points that have the **AR**= parameter set to **L** or **F**. If disabled (set to **off**), process points are not archived.
arch_srv_host	**<name | ip>** - host of **Archive Server**.
arch_srv_port	**<name>** - port of **Archive Server**.
arch_srv_ compression	**<on | off>** - **Archive Server** compression protocol.
arch_srv_protocol	**<UDP | TCP>** - type of **Archive Server** protocol.
src_lhost	**<name |IP>** - local host for connection with **Archive Server** (wrapping source historian).
src_lport	**<number>** - local port for connection with **Archive Server** (wrapping source historian).
src_lport_range	**<number>** - local port range for connection with **Archive Server** (wrapping source historian).
src_host	**<number>** - port of **Archive Server** (wrapping source historian).
src_port	**<N1,N2,...>** - list of network names to import points from.

PARAMETER	DESCRIPTION
comm_timeout	**<seconds>** - communication timeout time in seconds when the application waits to receive acknowledge from the destination (by default it is set to **0.4** seconds).
user	**<user>** - user name used for connections to **Archive Server**.
passwd	**<password>** - password used for connections to **Archive Server**.
auto_fillin	**<on | off>** - automatically fill in missing data on Server.
fillin_retries	**<spec>** - specifies how many times a fill-in operation for a given range should be performed.
fillin_overlap	**<number>** - number of seconds extending periods without connectivity.
fillin_delay	**<number>** - number of seconds that must pass before starting an automatic fill-in.
buffer_dir	**<path>** - path of the directory where buffer file will be stored.
autofillin_range	**<number>** - auto filling only specified number of days.
range_to_skip	**<number>** - insignificant gap range to skip data fill-in.
Source section	
fast_mode	**<on | off>** - sends data without waiting for ACK acknowledgment message.
org_cycle:	**<0-255>** -sets data encoding, values **0** an **1** prompt encoding of all data, setting values from **2** to **255** prompt encoding data for every **n**-th point, where **n** is the value set.
proxy_mode	**<0 | 1 | 2>** - specifies proxy mode of the source: **0/off** - normal mode, **1/legacy** - proxy mode (SID), **2/using_idcs** - proxy mode (IDCS). Only sends values, no repeated frames, ZP exports file, ZD use this file.
frames_sleep_ time	**<seconds>** - sleep time after frame is sent.
cache_dir	**<path>** - path of the directory cache files (including point database) will be stored.
proxy_def	**<name>** - file name with definition of points to manage.
script_semaphore	**<name>** - system semaphore name - separates access to files created by processes: **send_file_udp**, **receive_file_udp**, **zp_w3**.
Ovation section	
wdpf_pdir	**<path>** - path to spd.online or spd.configuration.
wdpf_networks	**<N1,N2,...>** - list of network names to import points from.
wdpf_SIDRecords	**<list>** - comma separated list of records treated as Ovation SIDs (to be altered by option wdpf_forcedNetworkNbr)
wdpf_ forcedNetworkNbr	**<0-8>** - forces any SID encoded in EDS points to fake a specified Ovation network number (0 is to off).
wdpf_minSID	**<number>** - minimum SID to import.
wdpf_maxSID	**<number>** - maximum SID to import.
wdpf_SIDmask	**<number>** - condition for originating: ((SID & SIDmask) == SIDvalue).
wdpf_SIDvalue	**<number>** - condition for originating: ((SID & SIDmask) == SIDvalue).
wdpf_fullNames	**<on | off>** - enabling (setting **on**) keeps the **.unit@net** part of the name, disabling (setting **off**) strips it from the name.

PARAMETER	DESCRIPTION
wdpf_ skipAperiodic	**<on | off>** - imports only Fast, Slow and SheetMon; **off** imports all.
wdpf_enableR1S	**<on | off>** - setting **on** enables sending one-shot requests to update point records.
status_pname	**<name>** - name of IESS indicates status point; Indicating the name of drop on which the feeder is installed, the communication begins only when drop is ready.
dls_link_status	**<name>** - packed point for DLS Link Status.
dls_last_comm	**<name>** - analog point for DLS Last Communication.
dls_digital_alarm	**<name>** - digital point name for alarming.
search_by	**<IDCS | IESS>** - determines which field is used to identify points in Ovation API.
encoding	**<encoding>** - character encoding for external data.
Buffer watch section	
watch	**<on | off>** - enables deleting oldest archives, when less than **min_space** is available on the data file system.
min_space	**<percent>** - minimum free space that must be left on the data file system.
max_archive_ period	**<n>** - keeps archives not older than **<n>** days; If set to **0**, checking the archives will not be performed.
max_archive_size	**<n>** - keeps archives not bigger than **<n>** MB; If set to **0**, checking the archives will not be performed.
Update time section	
interval	**<number>** - number of seconds to sleep between updates. If set number will be negative, it will synchronize with moments when **((time-shift) % interval)= 0** . For example **interval= -5.0** will start update cycles at 00, 05, 10, 15.
shift	**<number>** - number in seconds. Used with interval to select update moments.
ZD section	
softstart_period	**<number>** - slows down the initialization process by a number of seconds. Multiple applications attempting to access Ovation at the same time may cause problems. Setting a value greater then **0** here can prevent that. Default value is **0**.
check_drop_ statuses	**<off | skip_aperiodic | skip_all>** - option to skip from reading points originated by unavailable controllers (in timeout or in failed condition).

1.4.2 ZP Ovation tab

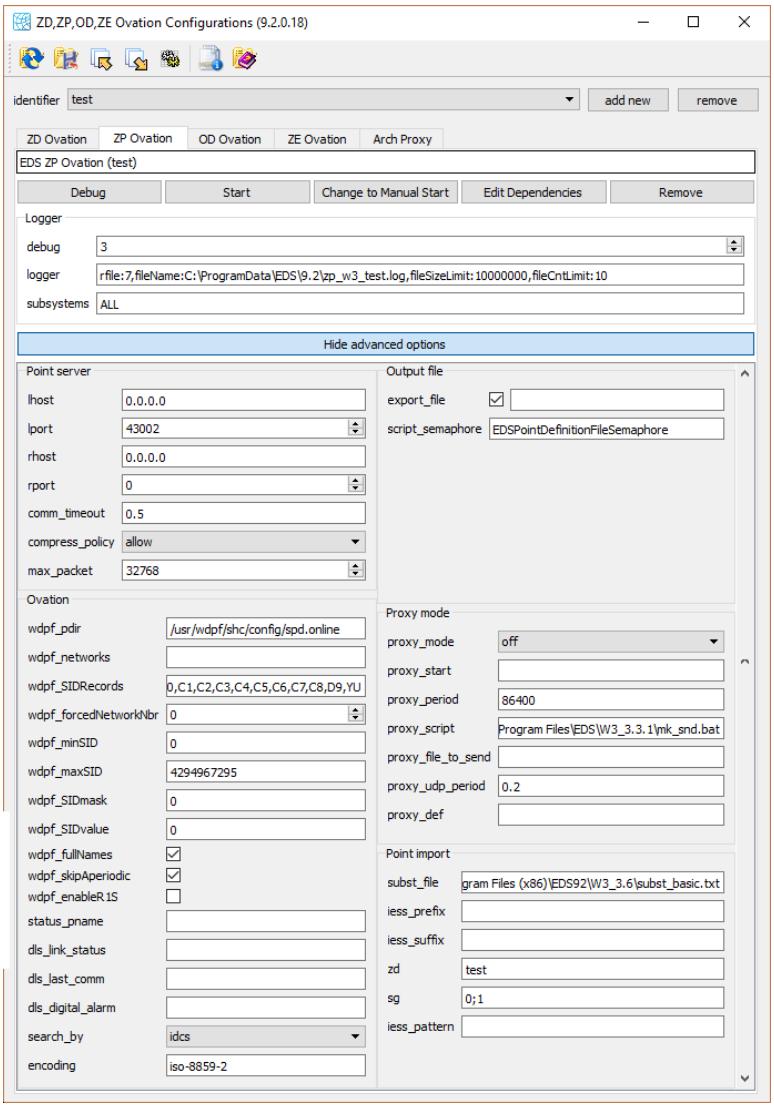


Figure 1.13 ZP Ovation tab

PARAMETER	DESCRIPTION
debug:	Level of significance of messages to be logged; numbers have the following meaning: ■ `0` - Emergency, system is unusable ■ `1` - Alert, action must be taken immediately ■ `2` - Critical, critical conditions ■ `3` - Errors, error conditions ■ `4` - Warning, warning condition ■ `5` - Notice, normal but significant condition ■ `6` - Informational ■ `7` - Debugging EDS effectively uses levels from `3` to `7`. When **debug:** is set to `5`, then it results in a printout of messages of levels `5`,`4` and `3`. The default level is `3` (log errors only).
subsystems:	List of names of the EDS subsystems that need to be logged. A special keyword `ALL` selects all subsystems, which is the default and recommended setting for normal operation.
logger:	Logger initialization string. For more information see *1.4.7 Logger Initialization String on page 34*.
lport:	`<number>` - local UDP port number. Selecting `0` results in automatic assignment of available port number by the operating system, which is the recommended option; it may be necessary to define a port number if there is a firewall between this application and the destination.
sg:	`<0, 1,...>` - security groups to be assigned
proxy_mode:	`<0 | 1>` - `0` - normal mode, `1` - proxy mode.
proxy_def:	`<filename>` - file name with definition of points to manage
proxy_script:	`<filename>` - script preparing the file to be transferred.
proxy_start:	`<YYYYMMDDHHMISS>` - start time to run script for proxy mode.
proxy_period:	`<seconds>` - period to send file beginning from **proxy_start=**
proxy_udp_period:	`<seconds>` - sleep time between sending UDP frames.
export_file:	`<name>` - file to save points to.
wdpf_networks:	`<N1, N2,...>` - list of network names to import points from.
wdpf_fullNames:	`<on | off>` - enabling (setting `on`) keeps the `.unit@net` part of the name, disabling (setting `off`) strips it from the name.
wdpf_ skipAperiodic:	`<on | off>` - on = import only Fast, Slow and SheetMon, off = import all
wdpf_enableR1S:	`<on | off>` - setting `on` enables sending one-shot requests to update point records.
wdpf_Fault_Code:	`<0-255>` - used for setting drop fault, default `190`.
wdpf_Fault_Id:	`<0-255>` - used for setting drop fault (default `0` - do not set drop fault).
wdpf_NetTime:	`<local | gmt>` - alarm timestamp mode

PARAMETER	DESCRIPTION
lhost:	**<name | ip>** - local host name or IP number; the recommended setting is **0.0.0.0**, as this opens local socket on all available network interfaces; it might, however, be necessary to limit communication to a specified network interface, which can be achieved by specifying appropriate value of this parameter.
lport_range:	**<number>** - local UDP port range; allows to set the limit on used port numbers in order to narrow the port ranges opened In firewalls; specifying **lport=43000** and **lport_range=100** will make the application attempt to bind port **43000** - if it is already used and thus not available, it will proceed with **43001** and up to **43100**; first available port in this range will be used; if all ports are unavailable, the application will fail. If no firewalls in the network then this should be set to **0**.
comm_timeout:	**<seconds>** - communication timeout time in seconds when the application waits to receive acknowledge from the destination (by default it is set to **0.4** seconds).
resp_timeout:	**<seconds>** - response timeout time in seconds after which the application retransmits data packet. If it did not receive acknowledges from the destination. It may be increased if network delays are big (by default it is set to **2.0**).
max_packet:	**<bytes>** - a communication parameter (bytes). Specifies the maximum size of a single communication buffer (before compression). In some networks, may need to be decreased (for example to **16348**, **8192**, **4096** or **2048**).
max_bytes_per_second:	**<number>** - limit on the number of bytes sent (setting **0** disables the check)
rhost:	**<name | ip>** - server address
rport:	**<number>** - server UDP port
compress:	**<on | off>** - controls compression of the transmitted data switching it on lowers network load but may cause raise in processor load.
compress_policy:	**<a | f | d>** - setting **a** allows, **f** forces, and **d** denies.
read_fields_sleep_t:	**< a | f | d>** - setting **a** allows, **f** forces, and **d** denies.
scanning_sleep_t:	**<seconds>** - seconds to sleep after each items path scan.
subst_file:	**<name>** - file to read field substitutions from; file structure: **DFLT: <X1=Y1> <X2=Y2>** - defines options for all points **PNFQ: <X1=Y1> <X2=Y2>** - returns full point name (Fully Qualified Point Name): <short_name>.<unit_name>@<network_name> (e.g. ABCD.UNIT1@NET1) **IDCS=<idcs> <X1=Y1> <X2=Y2>** ... - defines for point **<idcs>** only Xn are EDS record type names plus some special keywords: **SKIP=<0 | 1>** - allows to ignore given process point **SCAN_PERIOD=T** - point will not be scanned fasterthen once per T sec. **STATIC_RECORDS='X1=Y1 X2=Y2** ...' - fields added to AUX in the form of Xn=Yn, may be used in process diagrams as IESS:Xn; if Yn is preceded by '@', then the field Yn value is read from Ovation at the import time **DYNAMIC_RECORDS=L** - L is a list of coma-separated field names which will be used to create additional process points (named IESS:Xn) which will bring on-line values of these fields to the EDS server **DYNAMIC_RECORD_TYPES: ANALOG=L1 BINARY=L2 PACKED=L3 DOUBLE=L4 INT64=L5** where Ln is a coma-separated list of Ovation fields (e.g. **X0**,**X1**,**X2**); this allows to define the type of the process point in EDS to be used for a given field; by default this type is the same as the one of the process point to which given field belongs.
iess_prefix:	**<name>** - prefix added to each IESS.

PARAMETER	DESCRIPTION
iess_suffix:	**<name>** - suffix added to each IESS.
wdpf_ forcedNetworkNbr:	**<1-8>** - forces any SID encoded in EDS points to fake a specified Ovation network number.
wdpf_SIDRecords:	comma separated list of records treated as Ovation SIDs (to be altered by option **wdpf_forcedNetworkNbr:**).
iess_pattern:	**<regexp>** - regular expression allowing to import only the points with matching IESS.
operate:	**<on | off>** - accept/send operate commands, controls possibility to write process point values back; values can be set only for process points originated by the workstation on which this program runs.
wdpf_pdir:	**<path>** - path to spd.online or spd.config
wdpf_minSID:	**<number>** - minimum SID to import
wdpf_maxSID:	**<number>** - maximum SID to import
wdpf_SIDmask:	**<number>** - condition for originating: ((SID & SIDmask) == SIDvalue)
wdpf_SIDvalue:	**<number>** - condition for originating: ((SID & SIDmask) == SIDvalue)
wdpf_fullNames	**<on | off>** - enabling (setting **on**) keeps the **.unit@net** part of the name, disabling (setting **off**) strips it from the name.
wdpf_ skipAperiodic	**<on | off>** - imports only Fast, Slow and SheetMon; **off** imports all.
wdpf_enableR1S	**<on | off>** - setting **on** enables sending one-shot requests to update point records.
status_pname	**<name>** - name of IESS indicates status point; Indicating the name of drop on which the feeder is installed, the communication begins only when drop is ready.
hsr_picfile:	**<path>** - path + filename of HSR picfile
dls_link_status:	**<name>** - packed point for DLS Link Status
dls_last_comm:	**<name>** - analog point for DLS Last Communication
dls_digital_alarm:	**<name>** - digital point name for alarming
encoding:	**<encoding>** - character encoding for external data

1.4.3 OD Ovation tab

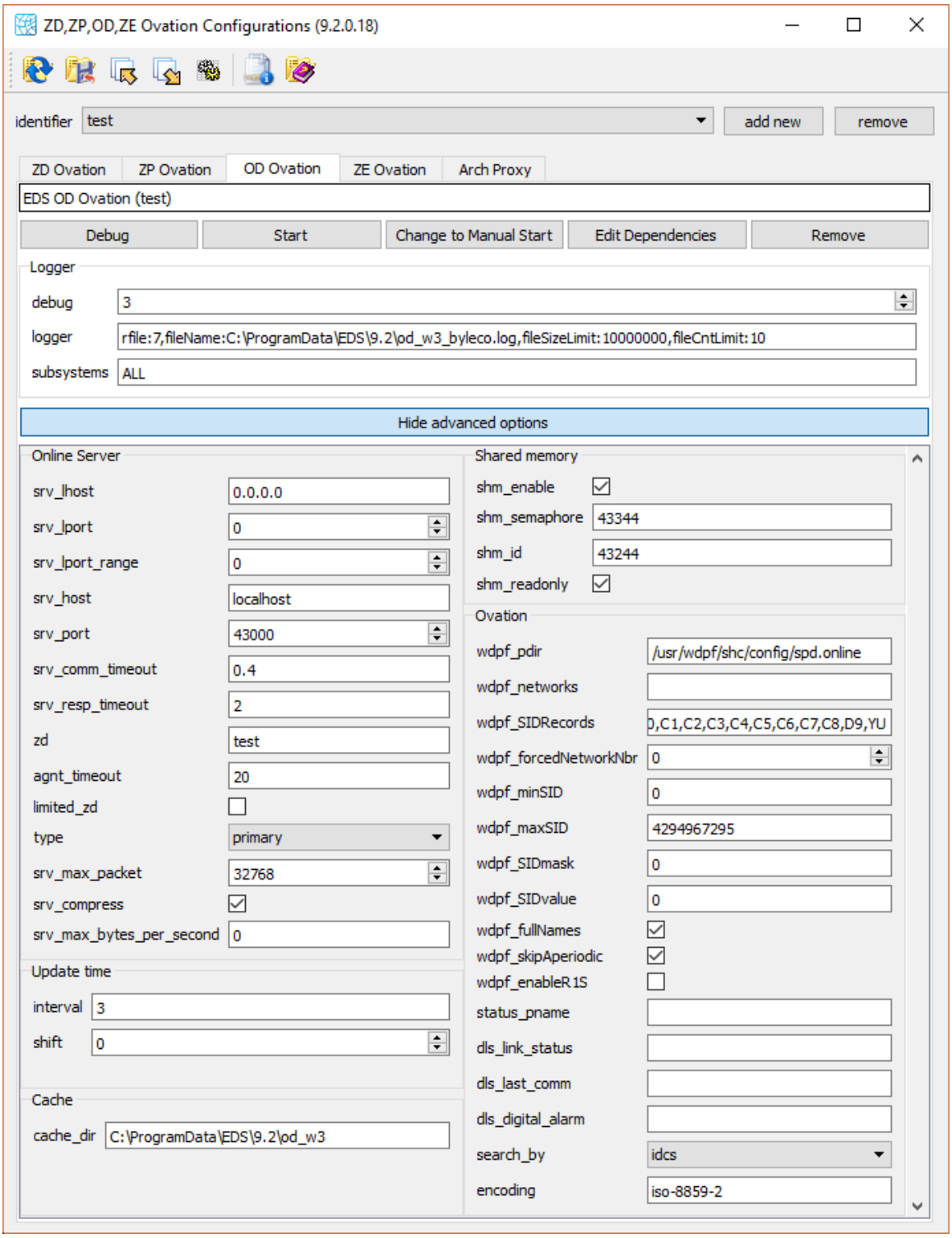


Figure 1.14 OD Ovation tab

PARAMETER	DESCRIPTION
debug:	Level of significance of messages to be logged; numbers have the following meaning: ■ `0` - Emergency, system is unusable ■ `1` - Alert, action must be taken immediately ■ `2` - Critical, critical conditions ■ `3` - Errors, error conditions ■ `4` - Warning, warning condition ■ `5` - Notice, normal but significant condition ■ `6` - Informational ■ `7` - Debugging EDS effectively uses levels from `3` to `7`. When **debug:** is set to `5`, then it results in a printout of messages of levels `5`,`4` and `3`. The default level is `3` (log errors only).
subsystems:	List of names of the EDS subsystems that need to be logged. A special keyword `ALL` selects all subsystems, which is the default and recommended setting for normal operation.
logger:	Logger initialization string. For more information see *1.4.7 Logger Initialization String on page 34*.
dbfile:	`<name>` - name of configuration file to store point database cache.
zd:	`<name>` - source name for points
type:	`<primary | backup | offline>` - source or output type; primary updates the process points, backup checks if a primary in on-line, offline synchronize and does not update the process points (when primary timeouts, backup starts updating process point values).
interval:	`<seconds>` - seconds to sleep between updates; if negative, then it will synchronize with moments when when the time shift is a multiple of the given value, e.g. **interval=**`-5.0` will start update cycles every 5 seconds.
shm_enable:	`<on | off>` - turns usage of shared memory `on` or `off` .
shm_readonly:	`<on | off>` - if set to `on`, the application will not write any values to the shared memory, assuming some other process updates its contents.
shm_id:	`<identifier>` - shared memory identifier.
shm_ semaphore:	`<identifier>` - semaphore identifier associated with shared memory.
wdpf_ networks:	`<N1, N2,...>` - list of network names to import points from.
wdpf_ fullNames:	`<on | off>` - enabling (setting `on`) keeps the `.unit@net` part of the name, disabling (setting `off`) strips it from the name.
wdpf_ skipAperiodic:	`<on | off>` - on = import only Fast, Slow and SheetMon, off = import all
wdpf_ enableR1S:	`<on | off>` - setting `on` enables sending one-shot requests to update point records.
wdpf_Fault_ Code:	`<0-255>` - used for setting drop fault, default `190`.
wdpf_Fault_Id:	`<0-255>` - used for setting drop fault (default `0` - do not set drop fault).
wdpf_NetTime:	`<local | gmt>` - alarm timestamp mode

PARAMETER	DESCRIPTION
lhost:	**<name | ip>** - local host name or IP number; the recommended setting is **0.0.0.0**, as this opens local socket on all available network interfaces; it might, however, be necessary to limit communication to a specified network interface, which can be achieved by specifying appropriate value of this parameter.
lport:	**<number>** - local UDP port number; selecting **0** results in automatic assignment of available port number by the operating system, which is the recommended option; it may be necessary to define a port number if there is a firewall between this application and the destination.
lport_range:	**<number>** - local UDP port range; allows to set the limit on used port numbers in order to narrow the port ranges opened In firewalls; specifying **lport=43000** and **lport_range=100** will make the application attempt to bind port **43000** - if it is already used and thus not available, it will proceed with **43001** and up to **43100**; first available port in this range will be used; if all ports are unavailable, the application will fail. If no firewalls in the network then this should be set to **0**.
comm_timeout:	**<seconds>** - communication timeout time in seconds when the application waits to receive acknowledge from the destination (by default it is set to **0.4** seconds).
resp_timeout:	**<seconds>** - response timeout time in seconds after which the application retransmits data packet. If it did not receive acknowledges from the destination. It may be increased if network delays are big (by default it is set to **2.0**).
max_packet:	**<bytes>** - a communication parameter (bytes). Specifies the maximum size of a single communication buffer (before compression). In some networks, may need to be decreased (for example to **16348**, **8192**, **4096** or **2048**).
max_bytes_per_second:	**<number>** - limit on the number of bytes sent (setting **0** disables the check).
compress:	**<on | off>** - controls compression of the transmitted data switching it on lowers network load but may cause raise in processor load.
protocol:	**<UDP | TCP>** - protocol to use
operate:	**<on | off>** - accepts/sends operate commands; controls possibility to write process point values back; values can be set only for process points originatingfrom the workstation on which this program runs.
qual_map:	**<0 | 1>** quality from the range of the values.
status_pname:	**<name>** - name of IESS indicates status of the point.
idcs_or_iess:	**<1 | 0>** - selecting **1** indicates that IDCS will be used in the device, selecting **0** indicates that IESS will be used.
reinit_cycle:	**<n>** - if the selected number is **0**, no reinitialisation will occur. If the selected number is greater than zero, reinitialisation will occur every **n**-th cycle.
agnt_timeout:	**<seconds>** - agent timeout, number of seconds after which EDS Server will change the status of this scanner from on-line to timedout when it stops updating process point values.
iess_pattern:	**<regexp>** - regular expression allowing to import only the points with matching IESS.
dbdriver:	**<mysql | postgresql | sqlite | sqlserver>** - choice of the SQL database driver.
dbserver:	**<name | ip>** - name or IP address of the host where SQL engine runs.
dbport:	**<number>** - port for connection to SQL database.
dbuser:	**<name>** - name of user.

PARAMETER	DESCRIPTION
dbpasswd:	**<password | @filename>** - user's password or name of file containing the password.
dmname:	**<name>** - name of database.
dbfilesize:	**<size TB|GB|MB|KB>** - specified only for Sqlserver; size of database data file,expressed by an integer with suffixes **[KB,MB,GB,TB]**.
wdpf_pdir:	**<path>** - path to spd.online or spd.configuration.
wdpf_minSID:	**<number>** - minimum SID to import.
wdpf_maxSID:	**<number>** - maximum SID to import.
wdpf_ SIDmask:	**<number>** - condition for originating: ((SID & SIDmask) == SIDvalue).
wdpf_ SIDvalue:	**<number>** - condition for originating: ((SID & SIDmask) == SIDvalue).
wdpf_ fullNames	**<on | off>** - enabling (setting **on**) keeps the **.unit@net** part of the name, disabling (setting **off**) strips it from the name.
wdpf_ skipAperiodic	**<on | off>** - imports only Fast, Slow and SheetMon; **off** imports all.
wdpf_ enableR1S	**<on | off>** - setting **on** enables sending one-shot requests to update point records.
hsr_picfile:	**<path>** - path + filename of HSR picfile.
dls_link_ status:	**<name>** - packed point for DLS Link Status.
dls_last_ comm:	**<name>** - analog point for DLS Last Communication.
dls_digital_ alarm:	**<name>** - digital point name for alarm.
encoding:	**<encoding>** - character encoding.

1.4.4 ZE Ovation tab

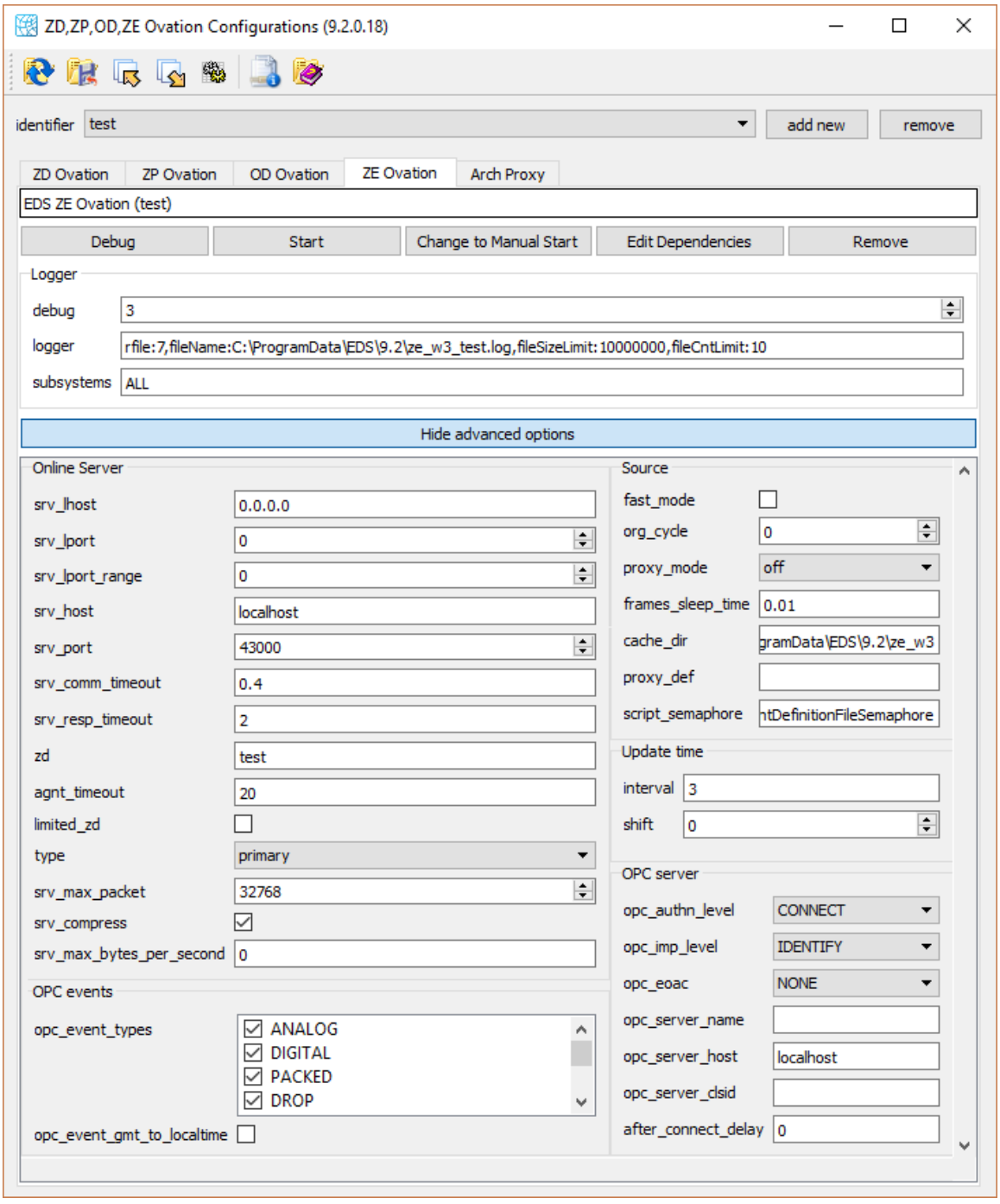


Figure 1.15 ZE Ovation tab

PARAMETER	DESCRIPTION
debug:	Level of significance of messages to be logged; numbers have the following meaning: ■ **0** - Emergency, system is unusable ■ **1** - Alert, action must be taken immediately ■ **2** - Critical, critical conditions ■ **3** - Errors, error conditions ■ **4** - Warning, warning condition ■ **5** - Notice, normal but significant condition ■ **6** - Informational ■ **7** - Debugging EDS effectively uses levels from **3** to **7**. When **debug:** is set to **5**, then it results in a printout of messages of levels **5**,**4** and **3**. The default level is **3** (log errors only).
subsystems:	List of names of the EDS subsystems that need to be logged. A special keyword **ALL** selects all subsystems, which is the default and recommended setting for normal operation.
logger:	Logger initialization string. For more information see *1.4.7 Logger Initialization String on page 34*.
dbfile:	**<name>** - name of configuration file to stores point database cache.
type:	**<primary | backup | offline>** - source or output type; primary updates the process points, backup checks if a primary in on-line, offline synchronize and does not update the process points (when primary timeouts, backup starts updating process point values).
interval:	**<seconds>** - seconds to sleep between updates; if negative, then it will synchronize with moments when when the time shift is a multiple of the given value, e.g. **interval=-5.0** will start update cycles every 5 seconds.
opc_server_clsid:	**<hhhhhhhh-hhhh-hhhh-hhhh-hhhhhhhhhhhh>** - clsid of OPC Server. This parameter is necessary only to connect to remote OPC Server, on which no **opcenum.exe** works.
opc_server_host:	**<name | IP>** - name or IP address of OPC Server Host
opc_server_name:	**<name>** - name of OPC Server Program
opc_authn_level:	**<DEFAULT | NONE | CONNECT | CALL | PKT | PKT_INTEGRITY | PKT_PRIVACY>** - connection authentication level.
opc_imp_level:	**<ANONYMOUS | IDENTIFY | IMPERSONATE | DELEGATE>** - connection imp level.
opc_eoac:	**<NONE | MUTUAL_AUTH | CLOAKING | SECURE_REFS | ACCESS_CONTROL | APPID>** - connection eoac.
opc_event_types:	**ANALOG, BINARY, PACKED, DROP, HARDWARE, MODULE, NODE, DEVICE, COS, SOE, SYSTEM, OEL, MESSAGE** types of events.
opc_event_gmt_to_localtime:	Converts and replaces GMT time to local time in the message event string.
operate:	**<on | off>** - accepts/sends operate commands, controls possibility to write process point values back; values can be set only for process points originated by the workstation on which this program runs.

PARAMETER	DESCRIPTION
softstart_ period:	**<number>** - slows down the initialization process by a number of seconds. Multiple applications attempting to access Ovation at the same time may cause problems. Setting a value greater then **0** here can prevent that. Default value is **0**.
fast_mode:	**<on | off>** - sends data without waiting for ACK acknowlegement message.
org_cycle:	**<0-255>** -sets data encoding, values **0** an **1** prompt encoding of all data, setting values from **2** to **255** prompt encoding data for every **n**-th point, where **n** is the value set.
agnt_timeout:	**<seconds>** - agent timeout, number of seconds after which EDS Server will change the status of this scanner from on-line to timedout when it stops updating process point values.
shift:	**<seconds>** - time shift for the update, can be used to determine exact moments of the updates.
frames_sleep_ time:	**<seconds>** - sleep time after frame is sent.
lhost:	**<name | ip>** - local host name or IP number; the recommended setting is **0.0.0.0**, as this opens local socket on all available network interfaces; it might, however, be necessary to limit communication to a specified network interface, which can be achieved by specifying appropriate value of this parameter.
lport:	**<number>** - local UDP port number. Selecting **0** results in automatic assignment of available port number by the operating system, which is the recommended option; it may be necessary to define a port number if there is a firewall between this application and the destination.
lport_range:	**<number>** - local UDP port range; allows to set the limit on used port numbers in order to narrow the port ranges opened In firewalls; specifying **lport=43000** and **lport_ range=100** will make the application attempt to bind port **43000** - if it is already used and thus not available, it will proceed with **43001** and up to **43100**; first available port in this range will be used; if all ports are unavailable, the application will fail. If no firewalls in the network then this should be set to **0**.
comm_ timeout:	**<seconds>** - communication timeout time in seconds when the application waits to receive acknowledge from the destination (by default it is set to **0.4** seconds).
resp_timeout:	**<seconds>** - response timeout (2.0 s). Time in seconds after which the application retransmits data packet, if it did not receive acknowledge from the destination; this may be increased if network delays are big
max_packet:	**<bytes>** - the maximum size of a single UDP packet (before compression); in some network environments it may be necessary to set this number to 16384, 8192 or less, if there is no transmission
max_bytes_ per_second:	**<number>** - limit on the number of bytes sent default **0** = check disabled
compress:	**<on | off>** - controls compression of the transmitted data switching it on lowers network load but may cause raise processor load
protocol:	**<UDP | TCP>** - protocol to use
after_ connect_ delay:	**<seconds>** - seconds to sleep after initial connect
need_restart_ filename:	**<filename>** - full path file name to write a flag to restart by logwatch

PARAMETER	DESCRIPTION
status_ outfilename:	**<filename>** - full path file name to write device status
time_ between_opc_ renew:	**<seconds>** - time to wait to opc server ready
encoding:	**<encoding>** - character encoding for external data

1.4.5 Arch Proxy tab

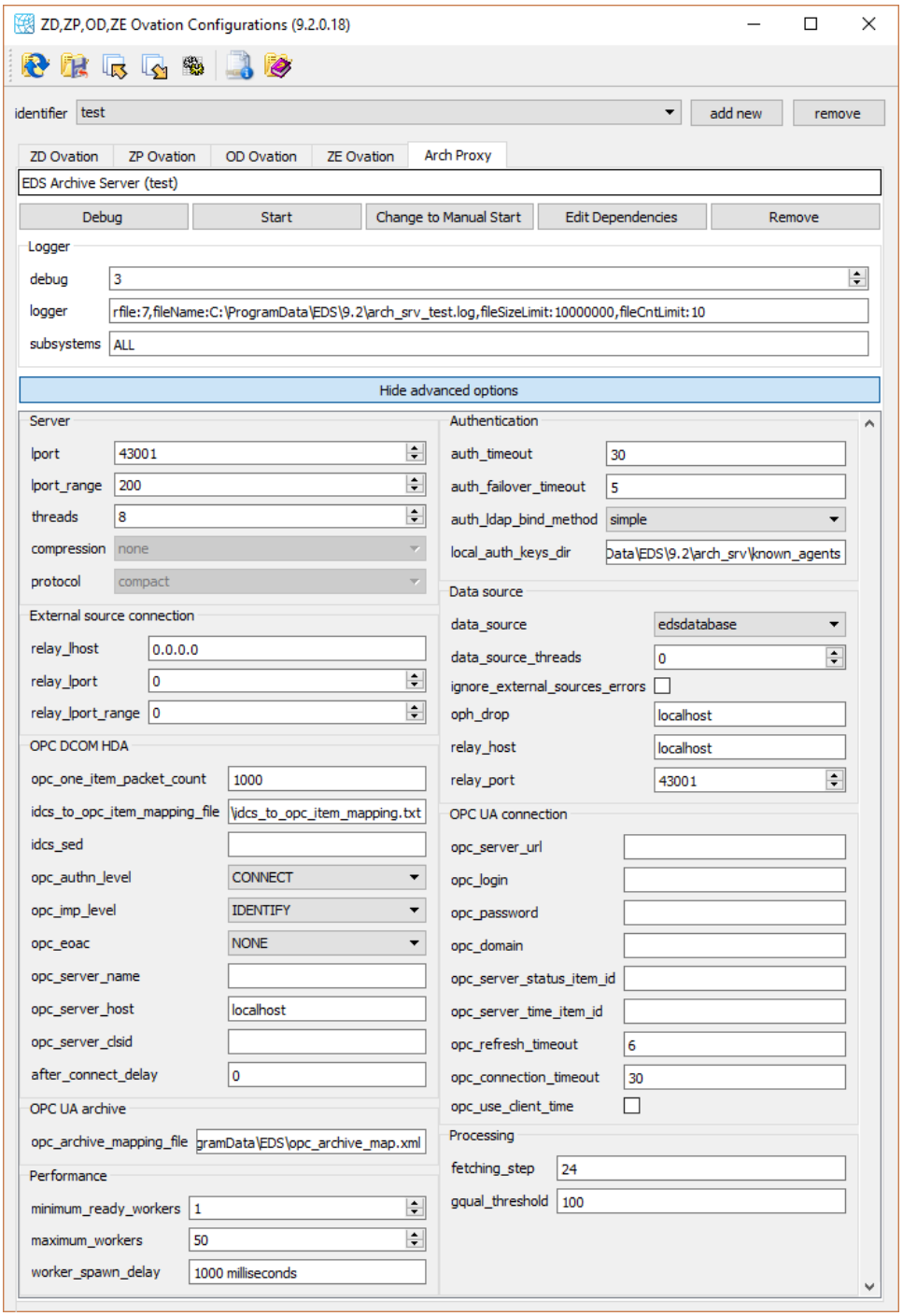


Figure 1.16 Arch Proxy tab

Note: *The parameters in Arch Proxy tab are identical to Archive Server tab of Server Configuration tool (srvconfig).*

PARAMETER	DESCRIPTION
Server section	
lport:	Local port number.
lport_range:	Local port range.
threads:	Number of handler threads.
compression:	Enables compression of the transmitted data switching it on lowers network load but may cause raise processor load.
protocol:	Enables compact or binary protocol.
External source connection	
relay_lhost	Local host for connection with data source.
relay_lport:	Local port for connection with data source.
relay_lport_ range:	Local port range for connection with data source.
OPC DCOM HDA section	
opc_one_ item_packet_ count:	Max samples count of one item in one query.
idcs_to_opc_ item_ mapping_file:	Additional point definitions file for mapping IDCS into OPC items. Expects file in EDS Database Access export format.
idcs_sed:	IDCS substitution command in sed-like format: **<searched_string format>/<replecement>**, e.g. passing **x/y** will result in replacing all occurrences of **x** in IDCS with **y**.
opc_authn_ level:	Authentication level to connect (default, none, connect, call, pkt, pkt_integrity, pkt_ privacy).
opc_imp_ level:	imp level to connect (anonymous, identify, impersonate, delegate).
opc_eoac:	eoac to connect (**none**, **mutual_auth**, **cloaking, secure_refs**, **access_control**, **appid**).
opc_server_ name:	Name of the OPC Server Program.
opc_server_ host:	Name or IP of the OPC Server Host.
opc_server_ clsid:	clsid of the OPC Server, format: **hhhhhhhh-hhhh-hhhh-hhhhhhhh**. This parameter is neccessary to connect only to the remote OPC Server on which no opcenum.exe works.
after_ connect_ delay:	Number of seconds to sleep after initial connection.
OPC UA archive section	
opc_archive_ mapping_file:	Path of XML file with definitions of archive mapping rules.
Performance section	

PARAMETER	DESCRIPTION
minimum_ ready_ workers:	Defines minimum and initial number of workers waiting for requests.
maximum_ workers:	Defines maximum number of workers waiting for requests.
worker_ spawn_delay:	Defines mimium delay between spawning unoccupied (waiting for request) workers
Authentification section	
auth_timeout:	Maximum wait time (in seconds) for user authentication.
auth_ failover_ timeout:	Maximum wait time (in seconds) before authentication is retried with different server Option is supported by some authentication methods when redundant servers are available.
auth_ldap_ bind_method:	LDAP bind method used to authenticate user: **`simple`** - password is sent over encrypted channel; **`digest`** - password's hash is sent over unencrypted channel
local_auth_ keys_dir	Path of folder used to save and/or read authorization keys (write permissions necessary) e.g. **`C:\ProgramData\EDS\9.2\arch_srv\known_agents`**
Data source section	
data_source:	Defines the source of local archives: (**`edsdatabase`** – archive data from EDS database, **`edsophlib_<version number>`** – archive data from Ovation OPH).
data_source_ threads:	Maximum number of threads used by the data source; **`0`** means automatic detection.
oph_drop:	IP or host name of the OPH station.
relay_host:	Host of Relay source.
relay_port:	Port of Relay source.
OPC UA section	
opc_server_ url:	URL of OPC server.
opc_login:	Login to use for connection to OPC server.
opc_passwd:	Password to use for connection to OPC server.
opc_domain:	Domain to use for connection to OPC server.
opc_server_ status_item_ id:	Id of OPC item with server status.
opc_server_ time_item_id:	Id of OPC item with server time. Currently it is useful for calculating time shift between OPC client and server when opc_use_client_time is enabled.
opc_refresh_ timeout:	Number of seconds after which inactive OPC item will be treated as timed out.
opc_ connection_ timeout:	Number of seconds after which inactive OPC server will be treated as disconnected.

PARAMETER	DESCRIPTION
opc_use_ client_time:	Override timestamps of server items with client's local time.
Processing section	
fetching_step	Defines the step's length (in full hours) of the incremental data processing.
gqual_ treshold:	Defines the percentage of samples of a given parameter that have to have a quality value **G** (**good**), so that the quality value of this parameter will also be marked as G.

1.4.6 Finalizing the installation

The newly installed package contains the following services:

APPLICATION	NAME	DESCRIPTION
zd_w3.exe	Data Source	This service updates process point values.
zp_w3.exe	Point Source	This service imports the list of process points.
od_w3.exe	Output Data	This service exports process point values.
ze_w3.exe	Event Source	This service reads alarms from OPC.
w3config.exe	Configuration Utility	This executable opens the GUI configuration utility.
conftool.exe	Support of GCode	This service is a command line utility tool.
esstool.exe	Command line test interface	This executable opens a command line test interface.
srvconfig.exe	Server Configuration	This executable opens the server configuration utility.
arch_srv.exe	ARCH Server	This service provides an archival data server.

At the end of installation process, applications **zp_w3.exe**, **zd_w3.exe** and **ze_w3.exe** (scanner applications) are registered as MS Windows services. This enables their start and stop through the MS Windows **Services** window.

If the process diagrams use **GCode** functions, it is necessary to import color definitions assigned to process point statuses. This can be done by executing the batch file:

dump_macro_alarm_config.bat

which generates the file:

macro_alarm_config.edf

The information in this file is a copy of some Windows Registry contents. The **macro_alarm_config.edf** file should be copied to the directory with all diagram sources (`*.src`) from where it will be imported along with all diagrams.

Note: *For more information on* ***arch_srv.exe*** *see the EDS Server User's Guide.*

1.4.7 Logger Initialization String

- Example of Rotating Files Logger configuration:

```
rfile:5,fileName:/usr/eds/server/obj_
srv.log,fileSizeLimit:1000000,fileCntLimit:3
```

- Example configuration for console logger:

```
console:7
```

- Example configuration for windows events:

```
sysevent:3
```

It is possible to use 2 or more loggers by separating them by semicolon, for example:

```
console:7;sysevent:3
```

Parameters description:

- `rfile:7` – log level 7 overwrites default debug parameter
- `fileSizeLimit=1000000` – maximum size of created log file (in bytes)
- `fileCntLimit=3` – number of log files to create (files: **xxx.log**, **xxx.log.0**, **xxx.log.1** will be created and rotary overwritten; **xxx-** is a server type (for example, **obj_srv** or **arch_srv**); the most current is always the **xxx.log** file).

After finalizing the configuration, click the **Save** button on the configuration tool's toolbar.

1.5 Ovation Interface start-up

After completing Ovation Interface installation and configuration the software may be started. This can be done by rebooting the machine where the software was installed or by starting up the three services created by Ovation Interface setup program. In order to start the services:

1. Open **Control Panel -> Administrative tools -> Services**. The following services should be on the list:
   - **Enterprise ZD Ovation (`<configuration profile name>`)**
   - **Enterprise ZP Ovation(`<configuration profile name>`)**
   - **Enterprise OD Ovation(`<configuration profile name>`)**
   - **Enterprise ZE Ovation(`<configuration profile name>`)**
   - **Enterprise ARCH (`<configuration profile name>`)**
2. Select the service from the list and press the **Start** button on the toolbar.

1.6 Adding custom records for a point

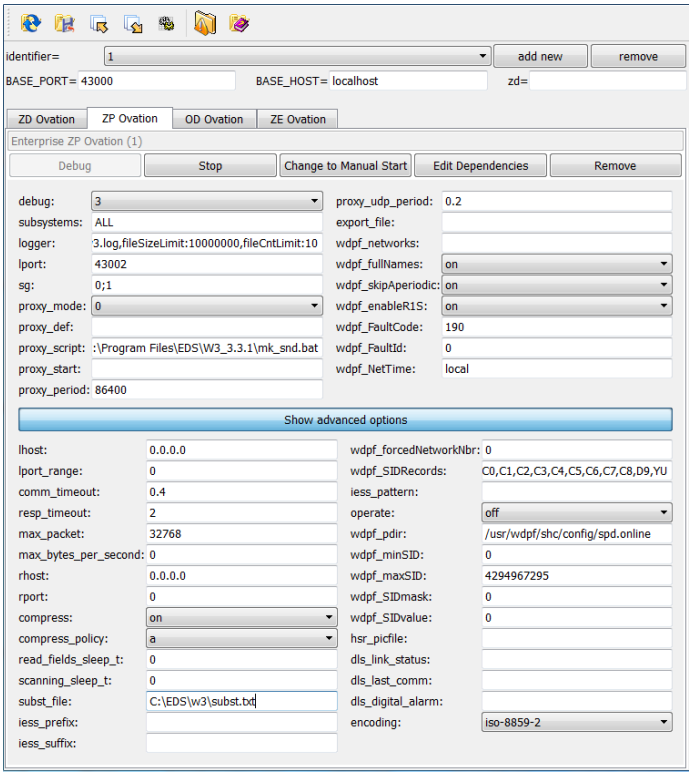
To add custom records for a given point in EDS you can follow the description below. As an example, it is assumed that we want to add records **X1**, **X2**, **X3** to point **NAME1** and **Y1**, **Y2**, **Y3** to point **NAME2**. **NAME1** and **NAME2** are short point names (that is, without the **.UNIT@NETWORK** suffix).

On the Ovation machine running the scanner:

1. Create a text file **C:\subst.txt**. You can adjust the path for your convenience.
2. Edit the file by entering the content:

```
IDCS=NAME1 DYNAMIC_RECORDS=X2,X2,X3
IDCS=NAME2 STATIC_RECORDS=Y2,Y2,Y3
```

3. Start the configurator of the Ovation scanner.
4. Go to the **ZP Ovation** tab
5. Type the path to the created text file in the parameter **subst_file:** `C:\subst.txt`. If the path was adjusted before, adjust it here as well.
6. Click the icon.



7. Import points (using the EDS Database Access application – for more detailed description please refer to Database Access manual. After importation you will have the following points in EDS:
   - IESS=NAME1.UNIT@NETWORK:X1 IDCS=NAME1 ...
   - IESS=NAME1.UNIT@NETWORK:X2 IDCS=NAME1 ...
   - IESS=NAME1.UNIT@NETWORK:X3 IDCS=NAME1 ...
   - IESS=NAME2.UNIT@NETWORK IDCS=NAME2 ... AUX=' Y1=v1 Y2=v2 Y3=v3' ...
   - IESS=NAME1.UNIT@NETWORK IDCS=NAME1 ...

Note:** Points **NAME1** and **NAME2** must exist on **Ovation NETWORK

The example with point **NAME1** demonstrates the method with dynamic custom fields. This is the creation of a separate process point for each custom field, but the values of **X1**, **X2** and **X3** are

dynamically updated in EDS (that is, if they change in Ovation, then this change is transferred to EDS). The field names can also be archived in EDS.

The example with point **NAME2** demonstrates the second method — static custom fields. They are read from Ovation during the import of points and can be used in EDS process diagrams like **\NAME2:Y3** (which returns the value **v3**, that **Y3** had when the points were imported). If value **v3** changes in Ovation, then it will be updated in EDS only after the next import of points.

CHAPTER 2

Ovation Interface for Solaris OS

IN THIS SECTION

2.1 Installation

It may be required to install **libgcc** package libraries before installing this package. It is part of the **ThirdParty** packages. Package **EDSW3-z.z**-rel.**x.x.x.x**-**SunOS-y.y-sparc.pkg**.**Z** (where **z.z** is the Ovation version) installs in the directory **/usr/eds/w3** and contains the following executables:

- **zd_w3** - responsible for reading point values from Ovation data highway
- **zp_w3** - responsible for retrieving list of all points from system point directory in Ovation and sending them to EDS Database Access application upon its request
- **od_w3** - responsible for writing point values to the Ovation data highway
- **w3tool** - a command-line tool for diagnostic and administrative purposes

To install the package perform the following steps:

1. Copy the **EDSW3-z.z-rel.x.x.x.x-SunOS-y.y-sparc.pkg.Z** to **/tmp** of the **WEStation** device where the software is to be loaded.
2. Login to that **WEStation** device as **root**.
3. In order to uncompress the package file:

   type `cd /tmp`

   uncompress **EDS W3*.Z**
4. Install the package:

   `pkgadd -d /tmp/*.pkg`
5. Once installation has been successfully completed, install and modify the startup files:

   cd /usr/eds/w3

   cp S99_XEDSZD /etc/rc3.d

   cp S99_XEDSZP /etc/rc3.d

   ln -s /etc/rc3.d/S99_XEDSZD /etc/rc0.d/K00.EDSZD

   ln -s /etc/rc3.d/S99_XEDSZP /etc/rc0.d/K00.EDSZP

Installation of file **S99_XEDSOD** is optional.

File **/etc/rc3.d/S99_XEDSZD** starts application **zd_w3.**

File **/etc/rc3.d/S99_XEDSZP** starts application **zp_w3.**

File **/etc/rc3.d/S99_XEDSOD** starts the application **od_w3**

To obtain a full list of arguments accepted by each application, start it without arguments, a list of parameters with descriptions will be displayed. Many parameters occur in all the applications. Below all the parameters from the applications **zd_w3**, **zp_w3**, **od_w3** are described:

PARAMETER	DESCRIPTION
lang=	<en | pl> - language (**en** - English, **pl** - Polish)
debug=	Level of significance of messages to be logged; numbers have the following meaning: ■ **0** - Emergency, system is unusable ■ **1** - Alert, action must be taken immediately ■ **2** - Critical, critical conditions ■ **3** - Errors, error conditions ■ **4** - Warning, warning condition ■ **5** - Notice, normal but significant condition ■ **6** - Informational ■ **7** - Debugging EDS effectively uses levels from **3** to **7**. When **debug:** is set to **5**, then it results in a printout of messages of levels **5**,**4** and **3**. The default level is **3** (log errors only).
subsystems=	List of names of the EDS subsystems that need to be logged. A special keyword **ALL** selects all subsystems, which is the default and recommended setting for normal operation.
lhost=	**<name | ip>** - local host name or IP number; the recommended setting is **0.0.0.0**, as this opens local socket on all available network interfaces; it might, however, be necessary to limit communication to a specified network interface, which can be achieved by specifying appropriate value of this parameter.
lport=	**<number>** - local UDP port number. Selecting **0** results in automatic assignment of available port number by the operating system, which is the recommended option; it may be necessary to define a port number if there is a firewall between this application and the destination.
lport_range=	**<number>** - local UDP port range; allows to set the limit on used port numbers in order to narrow the port ranges opened In firewalls; specifying **lport=43000** and **lport_range=100** will make the application attempt to bind port **43000** - if it is already used and thus not available, it will proceed with **43001** and up to **43100**; first available port in this range will be used; if all ports are unavailable, the application will fail. If no firewalls in the network then this should be set to **0**.
comm_timeout=	**<seconds>** - communication timeout time in seconds when the application waits to receive acknowledge from the destination (by default it is set to **0.4** seconds).
resp_timeout=	**<seconds>** - response timeout time in seconds after which the application retransmits data packet. If it did not receive acknowledges from the destination. It may be increased if network delays are big (by default it is set to **2.0**).
max_packet=	**<bytes>** - a communication parameter (bytes). Specifies the maximum size of a single communication buffer (before compression). In some networks, may need to be decreased (for example to **16348**, **8192**, **4096** or **2048**).
cfg_file=	**<name>** name of configuration file to store point database cache
compress_policy=	**<a | f | d>** - setting **a** allows, **f** forces, and **d** denies.
rhost=	**<name | ip>** - server address
rport=	**<number>** - server UDP port

PARAMETER	DESCRIPTION
zd=	**<name>** source name to be assigned for each point
zd=	**<name1, name2,...>** points to be output on the highway; alternatively identifies which process points should be updated by this application
type=	**<primary | backup>** source or output type **primary** updates the process points, **backup** checks if a primary is on-line (when primary timeouts, backup starts updating process point values)
sg=	**<0, 1,...>** - security groups to be assigned
export_file=	**<name>** file to save points to
d=	<name1, name2,...> source name or names; Identifies which process points should be updated by this application
compress=	**<on | off>** - controls compression of the transmitted data switching it on lowers network load but may cause raise in processor load.
operate=	**<on | off>** - accepts/sends operate commands, controls possibility to write process point values back; values can be set only for process points originated by the workstation on which this program runs.
fast_mode=	**<on | off>** - sends data without waiting for ACK acknowlegement message.
org_cycle=	**<0-255>** -sets data encoding, values **0** an **1** prompt encoding of all data, setting values from **2** to **255** prompt encoding data for every **n**-th point, where **n** is the value set.
agnt_timeout=	**<seconds>** - agent timeout, number of seconds after which EDS Server will change the status of this scanner from on-line to timedout when it stops updating process point values.
interval=	**<seconds>** - seconds to sleep between updates; if negative, then it will synchronize with moments when when the time shift is a multiple of the given value, e.g. **interval=-5.0** will start update cycles every 5 seconds.
shift=	**<seconds>** - time shift for the update, can be used to determine exact moments of the updates.
subst_file=	**<name>** - file to read field substitutions from.
iess_prefix=	**<name>** prefix added to each IESS.
iess_suffix=	**<name>** suffix added to each IESS.
console_logger=	**<0 | 1>** enable **1** or disable **0** log on console.
log file_name=	**<name>** name of file for logging messages.
log file_size_limit=	**<number>** maximum size in bytes of the log file, after which it is closed and another file is started. Numbers are added at the end of consecutive file names.
log file_cnt_limit=	**<number>** maximum number of log files kept in the system. When this number is reached, the oldest log file gets overwritten (this prevents the filesystem from overfilling).
wdpf_pdir=	**<path>** path to **spd.online** or **spd.config**
wdpf_networks=	**<N1, N2,...>** list of network names to import points from
wdpf_minSID=	**<number>** minimum SID to import
wdpf_maxSID=	**<number>** maximum SID to import

PARAMETER	DESCRIPTION
wdpf_SIDmask=	**<number>** condition for originating: ((SID & SIDmask) == SIDvalue)
wdpf_SIDvalue=	**<number>** condition for originating: ((SID & SIDmask) == SIDvalue)
wdpf_fullNames=	**<on | off>** - enabling (setting **on**) keeps the **.unit@net** part of the name, disabling (setting **off**) strips it from the name.
hsr_picfile=	**<path>** path and filename of HSR picfile
dls_link_status=	**<name>** packed point for DLS Link Status
dls_last_comm=	**<name>** analog point for DLS Last Communication
dls_digital_alarm=	**<name>** digital point name for alarming
wdpf_FaultCode=	**<0-255>** - used for setting drop fault, default **190**.
wdpf_FaultId=	**<0-255>** - used for setting drop fault (default **0** - do not set drop fault).

CHAPTER 3

EDS OPC Interface

IN THIS SECTION

3.1 OPC DCOM and UA versions

The OPC Distributed Component Object Model (DCOM) technology is being replaced by Unified Architecture (UA). For the time being both the older DCOM version and the newer UA version of both OPC server and client are available in the EDS OPC packages. It is advisable to access servers using clients depending on the same technology (DCOM client for DCOM server and UA client for UA servers), however adjustments can be made to ensure compatibility between DCOM and UA.

3.2 EDS OPC Data Access Client

The EDS OPC Interface is a set of tools designed for:

- importing process points from other OPC Servers,
- updating EDS process points' dynamic data (values, statuses) from other OPC Server
- serve EDS process points' dynamic data by own OPC server.

The functions above are realized by different executables, installed by the following packages:

- EDS92OpcClient-**x**bit-rel.**x.x.x.x**.msi package contains both **DCOM** and **UA** client applications.
- EDS92OPCServer-**x**bit-rel.**x.x.x.x**.msi package contains the **DCOM** server application.
- EDS92OPCSrvUa-**x**bit-rel.**x.x.x.x**.msi package contains the **UA** server application.

where **x** is the Operating System's bit version and **x.x.x.x** signifies the EDS software version.

3.2.1 Installing EDS OPC Client

In order to install the EDS OPC Interface package, it is necessary to login to the MS Windows environment with administrative privileges. The software is installed by default in the directory:

```
C:\Program Files\EDS92\OpcClient\
```

This path can be modified during the process of installation:

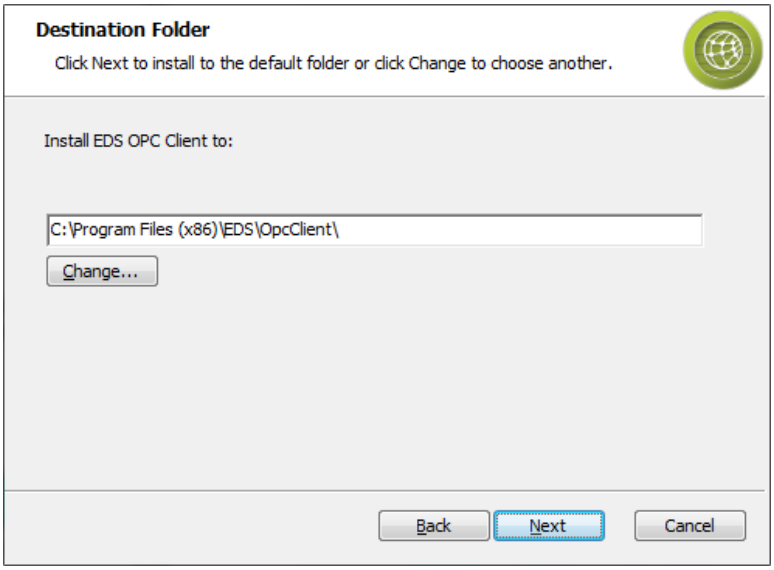


Figure 3.1 EDS OPC Client installation screen - destination directory

The package contains the following executables:

APPLICATION	NAME	DESCRIPTION
zd_opc.exe	Data Source	Updates EDS process point values from OPC values in UA technology
zp_opc.exe	Point Source	Imports the list of process points in UA technology
od_opc.exe	Data Source	Reads process point values and writes to OPC values in UA technology
opcconfig.exe	Configuration Utility	GUI configuration utility for UA
zd_dcom_ opc.exe	Data Source	Updates EDS process point values from OPC values in DCOM technology
zp_dcom_ opc.exe	Point Source	Imports the list of process points in DCOM technology
od__dcom_ opc.exe	Data Source	Reads process point values and writes to OPC values in DCOM technology
dcomconfig.exe	Configuration Utility	GUI configuration utility for DCOM
HDAClient.exe	Historical Data Access Client	Allows reading data history from a server.

At the end of the installation process, applications **zd_opc.exe**, **zp_opc.exe**, **zd_dcom_opc.exe** and **zp_dcom_opc.exe** (the scanner applications) are registered as MS Windows Services, which enables their start and stop through the MS Windows Administration Tools.

zd_opc.exe and **zd_dcom_opc.exe** are programs to send values from any **OPC** (**OLE** for **Process Control**) server to EDS server.

3.2.2 Configuring EDS OPC DCOM Client

The configuration utility **dcomconfig.exe** enables you to modify the communication parameters for the scanner applications for **DCOM** technology. The utility is automatically started after successful installation of all package components. Two tabs allow you to edit parameters of each of the scanner applications.

The EDS OPC Client configuration utility **(C:\Program Files\EDS92\OpcClient\opcconfig.exe**) icons are defined as:

ICON	DESCRIPTION
	Reloads options from registers
	Saves options to registers
	Starts the system interface for management of Services
	Opens **Help** window

Its top part contains parameters shared by all EDS OPC Client applications:

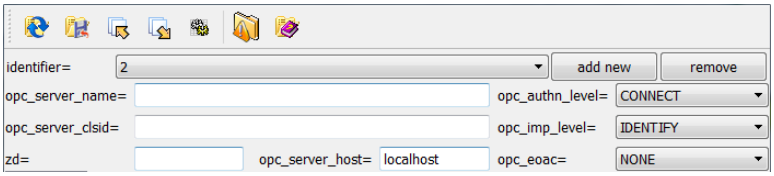


Figure 3.2 Parameters shared by all EDS OPC Client applications

PARAMETER	DESCRIPTION
identifier:	A drop down list containing all configured scanners. Initially it has the default name: **enterdb**. This name identifies scanner in MS Windows Service names. The name specified in the **add new** dialog box is used to create a new instance of scanner. To delete a scanner (remove all services related to specified identifier), press the **remove** remove button.
opc_server_name:	Name of OPC Server Program
opc_server_clsid:	clsid of OPC Server (format: hhhhhhhh-hhhh-hhhh-hhhh-hhhhhhhhhhhh). This parameter is necessary only to connect to remote OPC Server, on which no opcenum.exe works.
zd:	Source name or names identifies which process points should be updated by this application
opc_server_host:	Name or IP of OPC Server Host
opc_authn_level:	Authentication level to connect
opc_imp_level:	imp level to connect
opc_eoac:	eoac to connect

3.2.3 Configuring EDS OPC DCOM Source Data

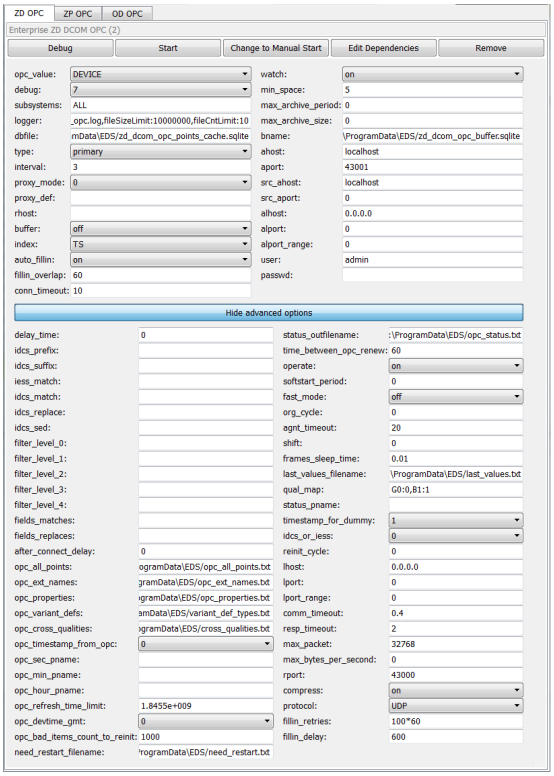


Figure 3.3 Options for ZD_OPC

To connect with the EDS Server typically a set of parameters suffices:

- rhost:
- rport:
- Server address
- Server UDP port (by default **43000**)

The wider set of available parameters is:

PARAMETER	DESCRIPTION
debug:	**<1 .. 7>** - debugging level; level of significance of messages to be logged; numbers have the following meaning: ■ **0** Emergency, system is unusable ■ **1** Alert, action must be taken immediately ■ **2** Critical, critical conditions ■ **3** Errors, error conditions ■ **4** Warning, warning conditions ■ **5** Notice, normal but significant condition ■ **6** Informational ■ **7** Debugging Effectively Levels **3** to **7** are used. Selecting **debug=5** results in printout of messages.
rhost:	Server address

PARAMETER	DESCRIPTION
rport:	Server UDP port (usually default 43000)
subsystems:	**<S1,S2,...>** - subsystems to be logged; list of names of the subsystems that need to be logged; special keyword **ALL** selects all subsystems, which is the default and recommended setting for normal operation
logger:	**<init_string>** - logger initialization string; example of Rotating Files Logger configuration: **rfile:5,fileName:/usr/eds/server/obj_srv.log,fileSizeLimit:1000000,fileCntLimit:3** Example configuration for console logger: **console:7** Example configuration for windows events: **sysevent:3** It is possible to use 2 or more loggers by separating by semicolon: Example use of console logger and windows events: **console:7; sysevent:3** Parameters desctiption: **rfile:7** - log level 7 overwrites default debug= parameter f**ileSizeLimit:1000000** - max size of created log file (in bytes) **fileCntLimit:3** - number of log files to created (files: **obj_srv.log, obj_srv.log.0, obj_srv.log.1** will be created and rotary overwrite. The most current is **obj_srv.log** always)
dbfile:	**<name>** - name of configuration file to stores point database cache.
type:	**<primary | backup | offline>** - source or output type primary updates the process points, backup checks if a primary in on-line, offline synchronize and does not update the process points (when primary timeouts, backup starts updating process point values)
interval:	**<seconds>** - seconds to sleep between writes; if negative, then it will synchronize with moments when ((time-shift) %% interval) = 0 e.g. **interval=-5.0** will start update cycles at 00, 05, 10, 15, etc. seconds
proxy_ mode:	**<0 | 1> - 0** - normal mode, **1** - proxy mode; only send values, no revise frames, **ZP** exports file, **ZD** use this file
proxy_def:	**<filename>** - file name with definition of points to manage
buffer:	**<on | off>** - **on** = save scanned values to an external file. If set to on, date scanner archives process point values (the ones which have **AR='L'|'F'**). If set to off, no archives are created for any points.
index:	**<NONE | T | S | TS>** - forces creation of indices on archive data tables; **T** - timestamps, **S** - SIDs
auto_fillin:	**<on | off>** - **on** = automatically fill in missing data on the Server
fillin_ overlap:	**<number>** - number of seconds extending periods without connectivity
conn_ timeout:	**<number>** - number of seconds that must pass before noticing lack of connection
watch:	**<on | off>** - **on** = delete oldest archives when less than **min_space** is available on the data file system
min_space:	**<0.0 .. 95.0%%>** - minimum free space that must be left on the data file system (default 10%%)
max_ archive_ period:	**<n>** - keep archives not older than **<n>** days; 0 = do not check

PARAMETER	DESCRIPTION
max_ archive_ size:	**<n>** - keep archives not bigger than **<n>** MB; 0 = do not check
bname:	**<name>** - name of the file with buffered data
ahost:	**<name | ip>** - ArchSrv address
aport:	**<number>** - ArchSrv UDP port
alhost:	**<name | ip>** - local host name or IP address for connections to ArchSrv
alport:	**<number>** - local UDP port number for connections to ArchSrv
user:	**<user>** - user name used for connections to ArchSrv
passwd:	**<password>** - password used for connections to ArchSrv
opc_imp_ level:	**<ANONYMOUS | IDENTIFY | IMPERSONATE | DELEGATE>** - imp level to connect
operate:	**<on | off>** - accept/send operate commands, controls possibility to write process point values back; values can be set only for process points originated by the Workstation on which this program runs
fast_mode:	**<on | off>** - send data without waiting for ACK
org_cycle:	**<0-255>** - **0, 1** - encode all data; **2-255** - encode changes, all data for every n-th point
agnt_ timeout:	**<seconds>** - agent timeout (20 s); number of seconds after which Server will change status of this scanner from on-line to timedout when it stops updating process point values
shift:	**<seconds>** - default 0, used with interval to select update moments
frames_ sleep_time:	**<seconds>** - sleep time after send frame, default 0.05 sec
qual_map:	**<0 | 1>** - quality from values range
status_ pname:	**<name>** - name of IESS indicates status point
timestamp_ from_dev:	**<1 | 0>** - **1** = set timestamp from device time, **0** = from local time
timestamp_ for_dummy:	**<1 | 0>** - **1** = set timestamp for not device points, **0** = not set
reinit_cycle:	**<n>** - **n=0**: no reinit, **n>0**: init in every n cycle
lhost:	**<name | ip>** - local host name or IP number; the recommended setting is **0.0.0.0**, as this opens local socket on all available network interfaces; it might, however, be necessary to limit communication to a specified network interface, what can be achieved by specifying appropriate value of this parameter
lport:	**<number>** - local UDP port number; selecting **0** results in automatic assignment of available port number by the operating system, what is the recommended option; it may be necessary to define a port number if there is a firewall between this application and the destination, which has been configured to pass through only specified port numbers for security reasons

PARAMETER	DESCRIPTION
lport_range:	**<number>** - local UDP port range; allows to set the limit on used port numbers in order to narrow the port ranges opened In firewalls; specifying **lport=43000** and **lport_range=100** will make the application attempt to bind port 43000 - if it is already used and thus not available, it will proceed with 43001 and up to 43100; first available port in this range will be used; if all ports are not available, the application will fail. If no firewalls in the network then this should be 0.
comm_timeout:	**<seconds>** - communication timeout (1.0 s); time in seconds when the application waits to receive acknowledge from the destination; this parameter should
resp_timeout:	**<seconds>** - response timeout (2.0 s); time in seconds after which the application retransmits data packet, if it did not receive acknowledge from the destination; this may be increased if network delays are big
max_packet:	**<bytes>** - the maximum size of a single UDP packet (before compression); in some network environments it may be necessary to set this number to 16384, 8192 or less, if there is no transmission
max_bytes_per_second:	**<number>** - limit on the number of bytes sent; default **0** = check disabled
compress:	**<on | off>** - controls compression of the transmitted data; switching it on lowers network load but may cause raise processor load
protocol:	**<UDP | TCP>** - protocol to use
idcs_prefix:	**<name>** - concat IDCS for send to device
idcs_suffix:	**<name>** - append to IDCS for send to device
iess_match:	**<regex>** - regular expression IESS points
idcs_match:	**<regex>** - regular expression IDCS points
idcs_replace:	**<regex>** - regular expression to automatic rename IDCS points
idcs_sed:	**<regex>** - regular expression in sed mode e.g. x/y
filter_level_0:	**<regex>** - simple regex to filter names in level **0**
filter_level_1:	**<regex>** - simple regex to filter names in level **1**
filter_level_2:	**<regex>** - simple regex to filter names in level **2**
filter_level_3:	**<regex>** - simple regex to filter names in level **3**
filter_level_4:	**<regex>** - simple regex to filter names in level **4**

PARAMETER	DESCRIPTION
fields_ matches:	**<filename>** - name of file with field types and regex to match them, e.g. **~IESS~(.*)\.F_CV~** **~DESC~(.*)\.F_CV~** **~ARTD~(.*)\.F_CV~** **~AUX~(.*)\.F_CV~** **~ARD~(.*)\.F_CV~** **~ZD~(.*)\.F_CV~** **~AR~(.*)\.F_CV~** **~AP~(.*)\.F_CV~** **~TG~(.*)\.F_CV~** **~SG~(.*)\.F_CV~** **~UN~(.*)\.F_CV~** **~DP~(.*)\.F_CV~** **~TB~(.*)\.F_CV~** **~BB~(.*)\.F_CV~** **~HL~(.*)\.F_CV~** **~LL~(.*)\.F_CV~** **~SD~(.*)\.F_CV~** **~RD~(.*)\.F_CV~**
fields_ replaces:	**<filename>** - name of file with field types and sub for regex, e.g. **~IESS~\1.IESS~** **~DESC~\1.DESC~** **~ARTD~\1.ARTD~** **~AUX~\1.AUX~** **~ARD~\1.ARD~** **~ZD~\1.ZD~** **~AR~\1.AR~** **~AP~\1.AP~** **~TG~\1.TG~** **~SG~\1.SG~** **~UN~\1.UN~** **~DP~\1.DP~** **~TB~\1.TB~** **~BB~\1.BB~** **~HL~\1.HL~** **~LL~\1.LL~** **~SD~\1.SD~** **~RD~\1.RD~**
after_ connect_ delay:	**<seconds>** - seconds to sleep after initial connect
opc_all_ points:	**<name>** - filename to store OPC points
opc_ext_ names:	**<name>** - filename with non-standard items
opc_ properties:	**<name>** - filename with OPC properties to EDS fields cross
opc_variant_ defs:	**<name>** - filename with VARIANT - ESS_POINT_TYPE cross
opc_cross_ qualities:	**<name>** - filename with OPC - Enterprise qualities cross
opc_value:	**<CACHE | DEVICE>**- **CACHE** = value from server cache, **DEVICE** = from device

PARAMETER	DESCRIPTION
opc_ timestamp_ from_opc:	**<1 | 0>** - **1** = set timestamp from device time, **0** = from local time
opc_sec_ pname:	**<name>** - IDCS for second value
opc_min_ pname:	**<name>** - IDCS for minute value
opc_hour_ pname:	**<name>** - IDCS for hour value
ocp_ refresh_ time_limit:	**<seconds>** - time in seconds of no communication after witch all points get quality Bad
opc_ devtime_ gmt:	**<1 | 0>** - **1** = OPC time is GMT, **0** = local time
opc_bad_ items_ count_to_ reinit:	**<number>** - number of bad items count in IOPCSyncIO->Read() function
opc_show_ items_types:	**<1 | 0>** - **1** = shows item types, **0** = doesn't show item types

3.2.4 Configuring EDS OPC DCOM Points Source

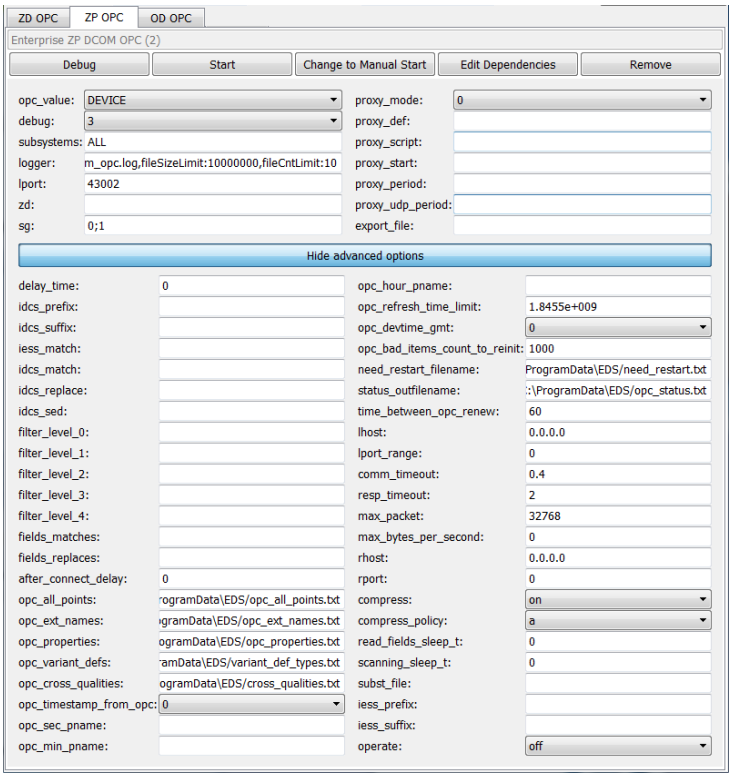


Figure 3.4 Options for ZP_OPC

PARAMETER	DESCRIPTION
debug:	**<1 .. 7>** - debugging level; level of significance of messages to be logged; numbers have the following meaning: **0** Emergency, system is unusable **1** Alert, action must be taken immediately **2** Critical, critical conditions **3** Errors, error conditions **4** Warning, warning conditions **5** Notice, normal but significant condition **6** Informational **7** Debugging Effectively levels **3** to 7 are used. Selecting **debug=5** results in printout of messages of levels **5,4,3**. The default level is **3** (log errors only)
subsystems:	**<S1,S2,...>** - subsystems to be logged; list of names of the subsystems that need to be logged; special keyword **'ALL'** selects all subsystems, which is the default and recommended setting for normal operation
logger:	**<init_string>** - logger initialization string; example of Rotating Files Logger configuration: **rfile:5,fileName:/usr/eds/server/obj_srv.log,fileSizeLimit:1000000,fileCntLimit:3** Example configuration for console logger: **console:7** Example configuration for windows events: **sysevent:3** Is possible use 2 or more loggers by separating by semicolon: Example use of console logger and windows events: **console:7; sysevent:3** Parameters desctiption: **rfile:7** - log level 7 overwrites default debug= parameter **fileSizeLimit:1000000** - max size of created log file (in bytes) **fileCntLimit:3** - number of log files to created (files: **obj_srv.log, obj_srv.log.0, obj_srv.log.1** will be created and rotary overwrite. The most current is **obj_srv.log** always)
lport:	**<number>** - local UDP port number; selecting **0** results in automatic assignment of available port number by the operating system, what is the recommended option; it may be necessary to define a port number if there is a firewall between this application and the destination, which has been configured to pass through only specified port numbers for security reasons
sg:	**<0, 1, ...>** - security groups to be assigned
proxy_ mode:	**<0 | 1>** - **0** - normal mode, **1** - proxy mode, only send values, no revise frames, **ZP** exports file, **ZD** use this file
proxy_def:	**<filename>** - file name with definition of points to manage
proxy_ script:	**<filename>** - script preparing the file to be transferred
proxy_start:	**<YYYYMMDDHHMISS>** - start time to run script for proxy mode
proxy_ period:	**<seconds>** - period to send file beginning from proxy_start=
proxy_udp_ period:	**<seconds>** - sleep time between sending UDP frames
export_file:	**<name>** - file to save points to

PARAMETER	DESCRIPTION
lhost=	**<name | ip>** - local host name or IP number; the recommended setting is 0.0.0.0, as this opens local socket on all available network interfaces; it might, however, be necessary to limit communication to a specified network interface, what can be achieved by specifying appropriate value of this parameter
lport_range=	**<number>** - local UDP port range; allows to set the limit on used port numbers in order to narrow the port ranges opened In firewalls; specifying **lport=43000** and **lport_range=100** will make the application attempt to bind port 43000 - if it is already used and thus not available, it will proceed with 43001 and up to 43100; first available port in this range will be used; if all ports are not available, the application will fail. If no firewalls in the network then this should be **0**.
comm_ timeout:	**<seconds>** - communication timeout (1.0 s); time in seconds when the application waits to receive acknowledge from the destination; this parameter should
resp_ timeout:	**<seconds>** - response timeout (2.0 s); time in seconds after which the application retransmits data packet, if it did not receive acknowledge from the destination; this may be increased if network delays are big
max_ packet=	**<bytes>** - the maximum size of a single UDP packet (before compression); in some network environments it may be necessary to set this number to 16384, 8192 or less, if there is no transmission
max_bytes_ per_second=	**<number>** - limit on the number of bytes sent; default **0** = check disabled
rhost:	**<name | ip>** - server address
rport:	**<number>** - server UDP port
compress:	**<on | off>** - controls compression of the transmitted data; switching it on lowers network load but may cause raise processor load
compress_ policy:	**<a | f | d>** - **a** = allow, **f** = force, **d** = deny
read_fields_ sleep_t:	**<seconds>** - seconds to sleep after each read fields items
scanning_ sleep_t:	**<seconds>** - seconds to sleep after each items path scan
subst_file:	**<name>** - file to read field substitutions from; file structure: **DFLT: <X1=Y1> <X2=Y2>** ... - defines options for all points **PNFQ: <X1=Y1> <X2=Y2>** - returns full point name (Fully Qualified Point Name): <short_name>.<unit_name>@<network_name> (e.g. ABCD.UNIT1@NET1) **IDCS=<idcs> <X1=Y1> <X2=Y2>** ... - defines for point **<idcs>** only Xn are EDS record type names plus some special keywords: **SKIP=<0 | 1>** - allows to ignore given process point
iess_prefix:	**<name>** - prefix added to each IESS
iess_suffix:	**<name>** - suffix added to each IESS
operate:	**<on | off>** - accept/send operate commands, controls possibility to write process point values back; values can be set only for process points originated by the Workstation on which this program runs
idcs_prefix:	**<name>** - concat IDCS for send to device
idcs_suffix:	**<name>** - append to IDCS for send to device
iess_match:	**<regex>** - regular expression IESS points

PARAMETER	DESCRIPTION
idcs_match:	**<regex>** - regular expression IDCS points
idcs_ replace:	**<regex>** - regular expression to automatic rename IDCS points
idcs_sed:	**<regex>** - regular expression in sed mode e.g. x/y
filter_level_ 0:	**<regex>** - simple regex to filter names in level 0
filter_level_ 1:	**<regex>** - simple regex to filter names in level 1
filter_level_ 2:	**<regex>** - simple regex to filter names in level 2
filter_level_ 3:	**<regex>** - simple regex to filter names in level 3
filter_level_ 4:	**<regex>** - simple regex to filter names in level 4
fields_ matches:	**<filename>** - name of file with field types and regex to match them, e.g. **~IESS~(.*)\.F_CV~** **~DESC~(.*)\.F_CV~** **~ARTD~(.*)\.F_CV~** **~AUX~(.*)\.F_CV~** **~ARD~(.*)\.F_CV~** **~ZD~(.*)\.F_CV~** **~AR~(.*)\.F_CV~** **~AP~(.*)\.F_CV~** **~TG~(.*)\.F_CV~** **~SG~(.*)\.F_CV~** **~UN~(.*)\.F_CV~** **~DP~(.*)\.F_CV~** **~TB~(.*)\.F_CV~** **~BB~(.*)\.F_CV~** **~HL~(.*)\.F_CV~** **~LL~(.*)\.F_CV~** **~SD~(.*)\.F_CV~** **~RD~(.*)\.F_CV~**
fields_ replaces:	**<filename>** - name of file with field types and sub for regex, e.g. **~IESS~\1.IESS~** **~DESC~\1.DESC~** **~ARTD~\1.ARTD~** **~AUX~\1.AUX~** **~ARD~\1.ARD~** **~ZD~\1.ZD~** **~AR~\1.AR~** **~AP~\1.AP~** **~TG~\1.TG~** **~SG~\1.SG~** **~UN~\1.UN~** **~DP~\1.DP~** **~TB~\1.TB~** **~BB~\1.BB~** **~HL~\1.HL~** **~LL~\1.LL~** **~SD~\1.SD~** **~RD~\1.RD~**

PARAMETER	DESCRIPTION
after_ connect_ delay:	**<seconds>** - seconds to sleep after initial connect
opc_all_ points:	**<name>** - filename to store OPC points
opc_ext_ names:	**<name>** - filename with non-standard items
opc_ properties:	**<name>** - filename with OPC properties to EDS fields cross
opc_variant_ defs:	**<name>** - filename with VARIANT - ESS_POINT_TYPE cross
opc_cross_ qualities:	**<name>** - filename with OPC - Enterprise qualities cross
opc_value:	**<CACHE | DEVICE>**- CACHE = value from server cache, DEVICE = from device
opc_ timestamp_ from_opc:	**<1 | 0>** - **1** = set timestamp from device time, **0** = from local time
opc_sec_ pname:	**<name>** - IDCS for second value
opc_min_ pname:	**<name>** - IDCS for minute value
opc_hour_ pname:	**<name>** - IDCS for hour value
ocp_ refresh_ time_limit:	**<seconds>** - time in seconds of no communication after witch all points get quality Bad
opc_ devtime_ gmt:	**<1 | 0>** - **1** = OPC time is GMT, **0** = local time
opc_bad_ items_ count_to_ reinit:	**<number>** - number of bad items count in IOPCSyncIO->Read() function
opc_show_ items_types:	**<1 | 0>** - **1** = shows item types, **0** = doesn't show item types

3.2.5 Configuring EDS OPC DCOM Data Output

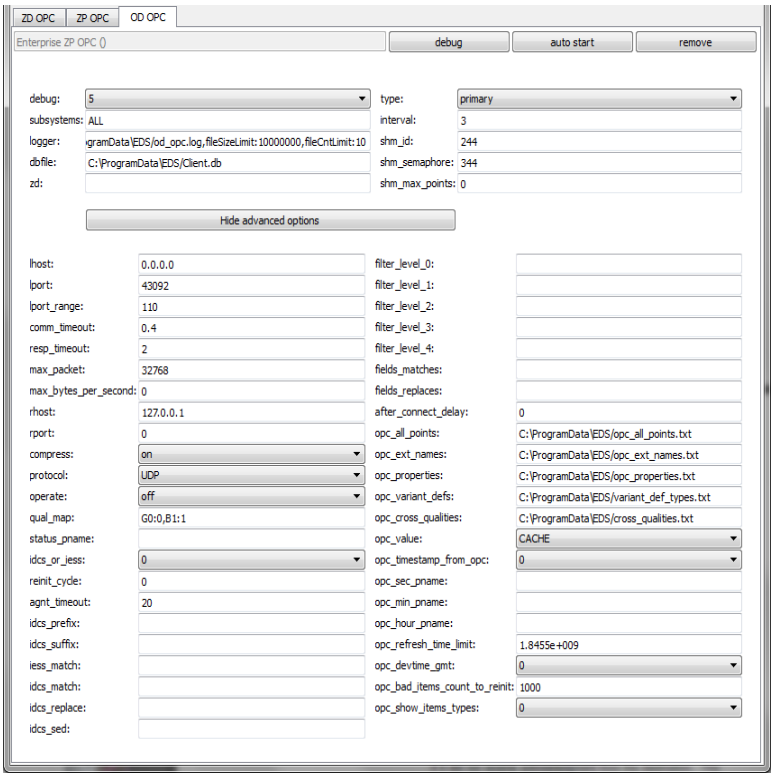


Figure 3.5 Options for OD_OPC PARAMETER DESCRIPTION

PARAMETER	DESCRIPTION
debug:	`<1 .. 7>` - debugging level; level of significance of messages to be logged; numbers have the following meaning: `0` Emergency, system is unusable `1` Alert, action must be taken immediately `2` Critical, critical conditions `3` Errors, error conditions `4` Warning, warning conditions `5` Notice, normal but significant condition `6` Informational `7` Debugging Effectively levels `3` to `7` are used. Selecting `debug=5` results in printout of messages of levels `5,4,3`. The default level is `3` (log errors only)
subsystems:	`<S1,S2,...>` - subsystems to be logged; list of names of the subsystems that need to be logged; special keyword `'ALL'` selects all subsystems, which is the default and recommended setting for normal operation

PARAMETER	DESCRIPTION
logger:	**<init_string>** - logger initialization string; example of Rotating Files Logger configuration: **rfile:5,fileName:/usr/eds/server/obj_srv.log,fileSizeLimit:1000000,fileCntLimit:3** Example configuration for console logger: **console:7** Example configuration for windows events: **sysevent:3** Is possible use 2 or more loggers by separating by semicolon: Example use of console logger and windows events: **console:7; sysevent:3** Parameters desctiption: **rfile:7** - log level 7 overwrites default **debug= parameter** **fileSizeLimit:1000000** - max size of created log file (in bytes) **fileCntLimit:3** - number of log files to created (files: **obj_srv.log**, **obj_srv.log.0**, **obj_srv.log.1** will be created and rotary overwrite. The most current is obj_srv.log always)
dbfile:	**<name>** - name of configuration file to stores point database cache.
zd:	source name or names identifies which process points should be updated by this application
type:	**<primary | backup | offline>** - source or output type; primary updates the process points, backup checks if a primary in on-line, offline synchronize and does not update the process points (when primary timeouts, backup starts updating process point values)
interval:	**<seconds>** - seconds to sleep between writes; if negative, then it will synchronize with moments when ((time-shift) %% interval) = **0** e.g. **interval=-5.0** will start update cycles at 00, 05, 10, 15, etc. seconds
shm_id:	**<identifier>** - shared memory identifier
shm_ semaphore:	**<identifier>** - semaphore identifier associated with shared memory
shm_max_ points:	**<number>** - max number of points in shared memory, default **100000**
lhost:	**<name | ip>** - local host name or IP number; the recommended setting is **0.0.0.0**, as this opens local socket on all available network interfaces; it might, however, be necessary to limit communication to a specified network interface, what can be achieved by specifying appropriate value of this parameter
lport:	**<number>** - local UDP port number; selecting **0** results in automatic assignment of available port number by the operating system, what is the recommended option; it may be necessary to define a port number if there is a firewall between this application and the destination, which has been configured to pass through only specified port numbers for security reasons
lport_range:	**<number>** - local UDP port range; allows to set the limit on used port numbers in order to narrow the port ranges opened In firewalls; specifying **lport=43000** and **lport_ range=100** will make the application attempt to bind port 43000 - if it is already used and thus not available, it will proceed with 43001 and up to 43100; first available port in this range will be used; if all ports are not available, the application will fail. If no firewalls in the network then this should be **0**.
comm_ timeout:	**<seconds>** - communication timeout (1.0 s); time in seconds when the application waits to receive acknowledge from the destination; this parameter should

PARAMETER	DESCRIPTION
resp_ timeout:	**<seconds>** - response timeout (2.0 s); time in seconds after which the application retransmits data packet, if it did not receive acknowledge from the destination; this may be increased if network delays are big
max_packet:	**<bytes>** - the maximum size of a single UDP packet (before compression); in some network environments it may be necessary to set this number to 16384, 8192 or less, if there is no transmission
max_bytes_ per_second:	**<number>** - limit on the number of bytes sent; default **0** = check disabled
rhost:	**<name | ip>** - server address
rport:	**<number>** - server UDP port
compress:	**<on | off>** - controls compression of the transmitted data; switching it on lowers network load but may cause raise processor load
protocol:	**<UDP | TCP>** - protocol to use
operate:	**<on | off>** - accept/send operate commands, controls possibility to write process point values back; values can be set only for process points originated by the Workstation on which this program runs
qual_map:	**<0 | 1>** - quality from values range
status_ pname:	**<name>** - name of IESS indicates status point
reinit_cycle:	**<n>** - **n=0**: no reinit, **n>0**: init in every n cycle
agnt_ timeout:	**<seconds>** - agent timeout (20 s); number of seconds after which Server will change status of this scanner from on-line to timedout when it stops updating process point values
idcs_prefix:	**<name>** - concat IDCS for send to device
idcs_suffix:	**<name>** - append to IDCS for send to device
iess_match:	**<regex>** - regular expression IESS points
idcs_match:	**<regex>** - regular expression IDCS points
idcs_ replace:	**<regex>** - regular expression to automatic rename IDCS points
idcs_sed:	**<regex>** - regular expression in sed mode e.g. x/y
filter_level_ 0:	**<regex>** - simple regex to filter names in level **0**
filter_level_ 1:	**<regex>** - simple regex to filter names in level **1**
filter_level_ 2:	**<regex>** - simple regex to filter names in level **2**
filter_level_ 3:	**<regex>** - simple regex to filter names in level **3**
filter_level_ 4:	**<regex>** - simple regex to filter names in level **4**

PARAMETER	DESCRIPTION
fields_ matches:	**<filename>** - name of file with field types and regex to match them, e.g. **~IESS~(.*)\.F_CV~** **~DESC~(.*)\.F_CV~** **~ARTD~(.*)\.F_CV~** **~AUX~(.*)\.F_CV~** **~ARD~(.*)\.F_CV~** **~ZD~(.*)\.F_CV~** **~AR~(.*)\.F_CV~** **~AP~(.*)\.F_CV~** **~TG~(.*)\.F_CV~** **~SG~(.*)\.F_CV~** **~UN~(.*)\.F_CV~** **~DP~(.*)\.F_CV~** **~TB~(.*)\.F_CV~** **~BB~(.*)\.F_CV~** **~HL~(.*)\.F_CV~** **~LL~(.*)\.F_CV~** **~SD~(.*)\.F_CV~** **~RD~(.*)\.F_CV~**
fields_ replaces:	**<filename>** - name of file with field types and sub for regex, e.g. **~IESS~\1.IESS~** **~DESC~\1.DESC~** **~ARTD~\1.ARTD~** **~AUX~\1.AUX~** **~ARD~\1.ARD~** **~ZD~\1.ZD~** **~AR~\1.AR~** **~AP~\1.AP~** **~TG~\1.TG~** **~SG~\1.SG~** **~UN~\1.UN~** **~DP~\1.DP~** **~TB~\1.TB~** **~BB~\1.BB~** **~HL~\1.HL~** **~LL~\1.LL~** **~SD~\1.SD~** **~RD~\1.RD~**
after_ connect_ delay:	**<seconds>** - seconds to sleep after initial connect
opc_all_ points:	**<name>** - filename to store OPC points
opc_ext_ names:	**<name>** - filename with non-standard items
opc_ properties:	**<name>** - filename with OPC properties to EDS fields cross
opc_variant_ defs:	**<name>** - filename with VARIANT - ESS_POINT_TYPE cross
opc_cross_ qualities:	**<name>** - filename with OPC - Enterprise qualities cross
opc_value:	**<CACHE | DEVICE>** - CACHE = value from server cache, DEVICE = from device

PARAMETER	DESCRIPTION
opc_ timestamp_ from_opc:	<1 | 0> - 1 = set timestamp from device time, 0 = from local time
opc_sec_ pname:	<name> - IDCS for second value
opc_min_ pname:	<name> - IDCS for minute value
opc_hour_ pname:	<name> - IDCS for hour value
ocp_ refresh_ time_limit:	<seconds> - time in seconds of no communication after witch all points get quality Bad
opc_ devtime_ gmt:	<1 | 0> - 1 = OPC time is GMT, 0 = local time
opc_bad_ items_ count_to_ reinit:	<number> - number of bad items count in IOPCSyncIO->Read() function
opc_show_ items_types:	<1 | 0> - 1 = shows item types, 0 = doesn't show item types

3.2.6 Configuring EDS OPC UA Client

The configuration utility **opcconfig.exe** enables you to modify the communication parameters for the scanner applications in **UA** technology. The utility is automatically started after successful installation of all package components. Two tabs allow you to edit parameters of each of the scanner applications.

The EDS OPC Client configuration utility **(C:\Program Files\EDS92\OpcClient\opcconfig.exe)** icons are defined as:

ICON	DESCRIPTION
	Reloads options from registers
	Saves options to registers
	Starts the system interface for management of Services
	Opens **Help** window

Its top part contains parameters shared by all EDS OPC Client applications:

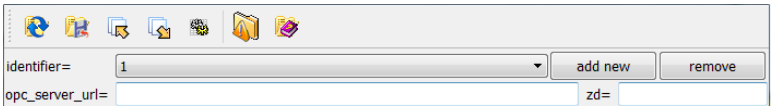


Figure 3.6 Parameters shared by all EDS OPC Client applications

PARAMETER	DESCRIPTION
identifier:	A drop down list containing all configured scanners. Initially it has the default name: **enterdb**. This name identifies scanner in MS Windows Service names. The name specified in the **'add new'** dialog box is used to create a new instance of scanner. To delete a scanner (remove all services related to specified identifier), click **'remove'** button.
opc_server_url:	URL address of OPC Server This parameter should be obtained from the administrator ot OPC server to which EDS will be connected. Example OPC server url: **opc.da://localhost/Matrikon.OPC.Simulation** or **opc.tcp://192.168.1.100:55551/EDSServerUA**
zd:	Source name or names identifying which process points should be updated by this application

3.2.7 Configuring EDS OPC UA Source Data

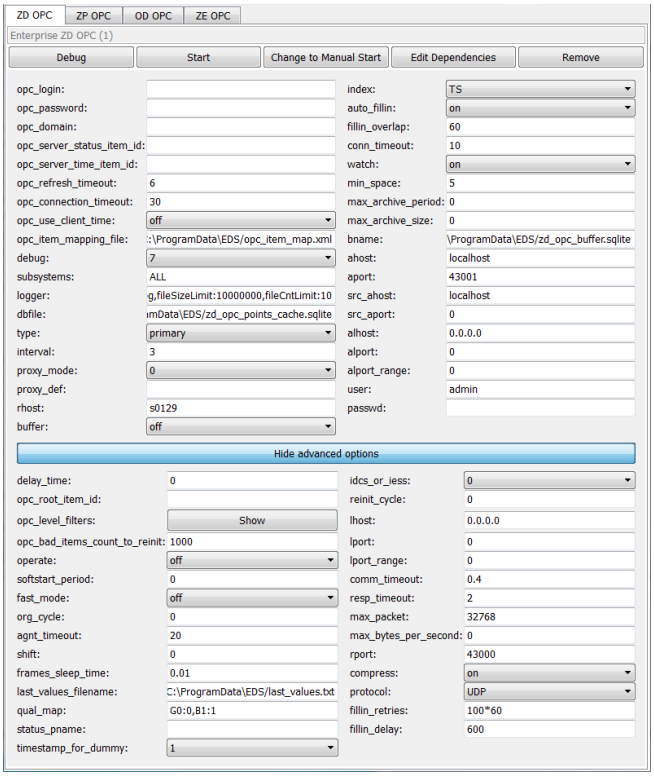


Figure 3.7 Options for ZD_OPC

PARAMETER	DESCRIPTION
opc_login:	**<user>** - login to use for connection to OPC server
opc_password:	**<password>** - password to use for connection to OPC server

PARAMETER	DESCRIPTION
opc_domain:	**<domain>** - domain to use for connection to OPC server
opc_server_status_item_id:	**<item id>** - id of OPC item which contains server status

Note: *These parameters should be obtained from the administrator of the OPC server to which EDS is connected. Often, their default values (empty) are enough for the connection to work properly, as OPC servers can be configured without the authorization requirement.*

PARAMETER	DESCRIPTION
opc_refresh_timeout:	**<seconds>** - number of seconds after which inactive OPC item will be treated as timed out
opc_connection_timeout:	**<seconds>** - number of seconds after which inactive OPC server will be treated as disconnected
opc_use_client_time:	**<off | on>** - determines the source of timestamps for OPC items **off** = use OPC item timestamps as returned by the OPC server (recommended) **on** = use client's time for OPC item timestamps
opc_item_mapping_file:	**<file path>** - path of XML file with definitions of item mapping rules Example file is placed in 'examples' folder in software installation catalogue. Within the XML file are comments describing parameters that can be configured. **zd_item_map.xml** is configured to be launched as default. Within this file, there are parameters for more advanced configuration (e.g. regular expressions for filtering or adding point prefixes) **opc_item_mapping_file** example: **C:\Program Files\EDS92\OpcClient\examples\zd_item_map.xml**
debug:	**<1 .. 7>** - debugging level; level of significance of messages to be logged; numbers have the following meaning: **0** Emergency, system is unusable **1** Alert, action must be taken immediately **2** Critical, critical conditions **3** Errors, error conditions **4** Warning, warning conditions **5** Notice, normal but significant condition **6** Informational **7** Debugging Effectively levels **3** to **7** are used. Selecting **debug=5** results in printout of messages
subsystems:	**<S1,S2,...>** - subsystems to be logged; list of names of the subsystems that need to be logged; special keyword **'ALL'** selects all subsystems, which is the default and recommended setting for normal operation

PARAMETER	DESCRIPTION
logger:	**<init_string>** - logger initialization string; example of Rotating Files Logger configuration: **rfile:5,fileName:/usr/eds/server/obj_srv.log,fileSizeLimit:1000000,fileCntLimit:3** Example configuration for console logger: **console:7** Example configuration for windows events: **sysevent:3** It is possible to use 2 or more loggers by separating by semicolon: Example use of console logger and windows events: **console:7; sysevent:3** Parameters desctiption: **rfile:7** - log level 7 overwrites default debug= parameter f**ileSizeLimit:1000000** - max size of created log file (in bytes) **fileCntLimit:3** - number of log files to created (files: **obj_srv.log, obj_srv.log.0, obj_srv.log.1** will be created and rotary overwrite. The most current is **obj_srv.log** always)
dbfile:	**<name>** - name of configuration file to stores point database cache.
type:	**<primary | backup | offline>** - source or output type primary updates the process points, backup checks if a primary in on-line, offline synchronize and does not update the process points (when primary timeouts, backup starts updating process point values)
interval:	**<seconds>** - seconds to sleep between writes; if negative, then it will synchronize with moments when ((time-shift) %% interval) = 0 e.g. **interval=-5.0** will start update cycles at 00, 05, 10, 15, etc. seconds
proxy_ mode:	**<0 | 1> - 0** - normal mode, **1** - proxy mode; only send values, no revise frames, **ZP** exports file, **ZD** use this file **Normal mode** is mode of bidirectional UDP communication. **Proxy mode** is unidirectional communication mode, where zd_opc only sends UDP frames to the server, not receiving nor waiting for UDP frames from EDS server.
proxy_def:	**<filename>** - file name with definition of points to manage File with definitions of names and point fields (in identical format as point definition export file from Database Access). This file is generated by **zd_opc** and used within it to determine which points will be sent to EDS server. Used only in proxy mode.
rhost:	<name | ip> - Server address
buffer:	**<on | off>** - **on** = save scanned values to an external file. If set to on, date scanner archives process point values (the ones which have **AR='L'|'F'**). If set to off, no archives are created for any points. **buffer=on** activates data buffering Buffering consists of archiving current data to local sqlite file. Buffering is used to increase the reliability of gathering historical data by gathering data in local sqlite file when communication with EDS Server is lost. After the connection is restored, buffered data will be sent to EDS Server. **buffer=off** indicates the state when data will be lost whenever there is no connection with EDS Server

PARAMETER	DESCRIPTION
index:	**<NONE | T | S | TS>** - forces creation of indices on archive data tables; **T** - timestamps, **S** - SIDs This parameter describes a more advanced option of generating indices in sqlite buffer database. **T** parameter value results in adding index to timestamp '**ts**' column in historical data table. **S** parameter value results in adding index to point ID '**ids**' column in historical data table. Generally, adding an index results in faster data reading at the expense of more disk space being used and time period of inserting data into table being longer. It is used when there is a need for quicker performance or limiting required disk space. This parameter is only used with **buffer=on**.
auto_fillin:	**<on | off>** - **on** = automatically fill in missing data on the Server Historical data, for the period of lack of communication with EDS Server, stored in sqlite buffer file, will be automatically sent to EDS Server. This parameter is only used with **buffer=on**.
fillin_ overlap:	**<number>** - number of seconds extending periods without connectivity Used to fill in the period of establishing connection, there is an added overlap of 5 to 10 minutes of no connection period. To avoid values of **zero Bad** during the process of establishing connection. This parameter is only used with **buffer=on**.
conn_ timeout:	**<number>** - number of seconds that must pass before noticing lack of connection Contrary to **fillin_overlap** parameter that is related to the period of reestablishing connection.
watch:	**<on | off>** - **on** = delete oldest archives when less than **min_space** is available on the data file system This parameter controls the amount of disk space occupied by sqlite file - when there is a lack of space, the oldest (chronologically) archives are deleted. This parameter is only used with **buffer=on**.
min_space:	**<0.0 .. 95.0%%>** - minimum free space that must be left on the data file system (default 10%%) Example: setting **min_space=10**, the sqlite file will be reduced or deleted until there is 10% or more free disk space. When there is 11% or more disk space, no action will be taken. When there is 9% or less, all data might be deleted from the buffer (except data from the last 24h period, which is internally protected). This parameter is only used with **watch=on**.
max_ archive_ period:	**<n>** - keep archives not older than **<n>** days; 0 = do not check Limits data in buffer to the last **n** days (days older than **n** are deleted). This parameter is only used with **watch=on** and **min_space**.

PARAMETER	DESCRIPTION
max_ archive_ size:	**<n>** - keep archives not bigger than **<n>** MB; 0 = do not check Limits buffer size to **n** MB (after exceeding this **n** size, the chronologically oldest archives are deleted until the file reaches **n** MB size). This parameter is only used with **watch=on** and **min space**.
bname:	**<name>** - name of the file with buffered data

Note: *The parameters listed below (**ahost**, **aport**, **src_ahost**, **src_aport**, **alhost**, **alport**, **alport_ range**, **user**, **passwd** are related to parameters:* **buffer=on** *and* **auto_fillin=on**.
They allow the software to connect with EDS Archive Server and send historical data to it from buffer for the periods of no communication with EDS Server.
Addressess and passwordsfor the parameters below should be obtained from EDS system administrator.
Ports are set as default, but might be configured differently, depending on network and firewall settings.

PARAMETER	DESCRIPTION
ahost:	**<name | ip>** - Archive Server (ArchSvr) address
aport:	**<number>** - Archive Server (ArchSvr) UDP port
src_ahost:	**<name | ip>** - Archive Server (wrapping source historian) address
src_aport:	**<number>** - Archive Server (wrapping source historian) UDP port
alhost:	**<name | ip>** - local host name or IP address for connections to Archive Server (ArchSvr)
alport:	**<number>** - local UDP port number for connections to Archive Server (ArchSvr)
alport_range:	**<number>** - local UDP/TCP port range for connections to
user:	**<user>** - user name used for connections to Archive Server (ArchSvr)
passwd:	**<password>** - password used for connections to Archive Server (ArchSvr)

PARAMETER	DESCRIPTION
delay_time:	**<seconds>** - start delay time to wait for dependent programs, default **0.0** sec This parameter is used to set waiting time for the completion of launching the OPC server, to which EDS Server is to be connected. It is used within systems where the OPC Server is on the same workstation as EDS Server and only to OPC Servers that malfunction in cases when the connection is executed too early.
opc_root_item_id:	**<item id>** - id of root item for scanned subtree of OPC server items Setting this option will filter out all OPC objects which are not descendants of the specified item id.
opc_level_filters:	**<filter list>** - list of regular expressions which are used to filter OPC items when browsing the server. Every element in the list filters a single level of items in the OPC item hierarchy. Empty elements allow all items from the level.

PARAMETER	DESCRIPTION
opc_bad_items_ count_to_reinit:	**<number>** - number of OPC item subscription errors required to trigger reinitialization of OPC client
operate:	**<on | off>** - accept/send operate commands, controls possibility to write process point values back; values can be set only for process points originated by the Workstation on which this program runs
softstart_period:	**<number>** - slows down the initialization process by a number of seconds. Multiple applications attempting to access Ovation at the same time may cause problems. Setting a value greater then **0** here can prevent that. Default value is **0**. This parameter is identical to **delay_time** parameter.
fast_mode:	**<on | off>** - send data without waiting for ACK [acknowledge]. This parameter is related to UDP communication with the EDS Server for current data. The feeder sends UDP frames with values for all points, in the meantime not waiting for the server confirmation for individual frames, but for the completion of cycle of sending all points which results in the server sending one joint confirmation. This means that there is no genuine unidirectional communication, yet the lack of confirmation for each single frame results in the cycle being faster and points being refreshed more often.
org_cycle:	**<0-255>** - **0, 1** - encode all data; **2-255** - encode changes, all data for every n-th point
agnt_timeout:	**<seconds>** - agent timeout (20 s); number of seconds after which Server will change status of this scanner from on-line to timeout when it stops updating process point values
shift:	**<seconds>** - default 0, used with interval to select update moments
frames_sleep_ time:	**<seconds>** - sleep time after send frame, default 0.05 sec - UDP frames in network communication between software on same or different workstations.
last_values_ filename:	**<filename>** - name of the file to store last valid values File where last known valid point values are stored. It is used during starting the software when there is no connection with data source to read those values from file and to send as last known value with quality set as **bad** (instead of **zero bad**).
qual_map:	**<0 | 1>** - quality from values range **qual_map** parameter default value is **G0:0,B1:1** , meaning that when the value gathered from OPC item value to EDS System is within the 0 to 0 range, will be substituted with quality: **Good**, and item values within the 1 to 1 range will be substituted with quality: **Bad**.
status_pname:	**<name>** - name of IESS indicates status point Value of this point is given by the feeder (in this case **zd_opc**). Value equal to 0 means that there is a valid connection with OPC data source. Value different than 0 means that there is a problem with the connection.

PARAMETER	DESCRIPTION
timestamp_for_ dummy:	**<1 | 0>** - **1** = set timestamp for not device points, **0** = not set With **timestamp_for_dummy=0** , points not found in the OPC data source device (furtherly called: device) will have timeout in EDS. Setting **timestamp_for_dummy=1** means that points not founf in device will be artificially given a timestamp and quality: **zero Bad** (often for the needs of calculations in **cons_calc**).
reinit_cycle:	**<n>** - **n=0**: no reinit, **n>0**: **init** in every **n** cycle, where cycle is defined as reading the values of all feeder points (from first to last point).
lhost:	**<name | ip>** - local host name or IP number; the recommended setting is **0.0.0.0**, as this opens local socket on all available network interfaces; it might, however, be necessary to limit communication to a specified network interface, what can be achieved by specifying appropriate value of this parameter
lport:	**<number>** - local UDP port number; selecting **0** results in automatic assignment of available port number by the operating system, what is the recommended option; it may be necessary to define a port number if there is a firewall between this application and the destination, which has been configured to pass through only specified port numbers for security reasons
lport_range:	**<number>** - local UDP port range; allows to set the limit on used port numbers in order to narrow the port ranges opened In firewalls; specifying **lport=43000** and **lport_range=100** will make the application attempt to bind port 43000 - if it is already used and thus not available, it will proceed with 43001 and up to 43100; first available port in this range will be used; if all ports are not available, the application will fail. If no firewalls in the network then this should be 0.
comm_timeout:	**<seconds>** - communication timeout (1.0 s); time in seconds when the application waits to receive acknowledge from the destination; this parameter should
resp_timeout:	**<seconds>** - response timeout (2.0 s); time in seconds after which the application retransmits data packet, if it did not receive acknowledge from the destination; this may be increased if network delays are big
max_packet:	**<bytes>** - the maximum size of a single UDP packet (before compression); in some network environments it may be necessary to set this number to 16384, 8192 or less, if there is no transmission
max_bytes_per_ second:	**<number>** - limit on the number of bytes sent; default **0** = check disabled
rport:	**<number>** - Server UDP port
compress:	**<on | off>** - controls compression of the transmitted data; switching it **on** lowers network load but may cause raise in processor load
protocol:	**<UDP | TCP>** - protocol to use
fillin_retries:	**<spec>** - specifies how many times a fillin operation for a given range should be performed
fillin_delay:	**<number>** - number of seconds that must pass before starting an automatic fill-in Used only with **buffer=on**.
wdpf_pdir:	**<path>** - path to **spd.online** or **spd.config**

PARAMETER	DESCRIPTION
wdpf_minSID:	**<number>** - minimum SID to import This field is only for software gathering points from Ovation (**zp_w3**). SID is Ovation point ID number. By setting this parameter, the user limits the number of points exported through **zp_w3**.
wdpf_maxSID:	**<number>** - maximum SID to import This field is only for software gathering points from Ovation (zp_w3). SID is Ovation point ID number. By setting this parameter, the user limits the number of points exported through zp_w3.
wdpf_SIDmask:	**<number>** - condition for originating: **((SID & SIDmask) == SIDvalue)**
wdpf_SIDvalue:	**<number>** - condition for originating: **((SID & SIDmask) == SIDvalue)**
hsr_picfile:	**<path>** - path + filename of HSR picfile This parameter is optional (zp_w3 will launch without setting it). If this parameter is set, the picfile must exist under the given path. This file is downloaded from Ovation. The information about HSR - archivization attributes (H as history) of points is stored there.
dls_link_status:	**<name>** packed point for DLS Link Status <name> should be the point name from Ovation (parameter only for zd_w3 and od_w3) for which the current software status will be written. DLS station is an Ovation station dedicated for sharing information (point values) with external systems. When zd_w3 or od_w3 is installed on the DLS station and this parameter is set, the information on the current software status will be corretly shown in Ovation.
dls_last_comm:	**<name>** - analog point for DLS Last Communication Same as above, but on this point the time of last correct communication is written.
dls_digital_alarm:	**<name>** - digital point name for alarming
encoding:	**<encoding>** - character encoding for external data

3.2.8 Configuring EDS OPC UA Points Source

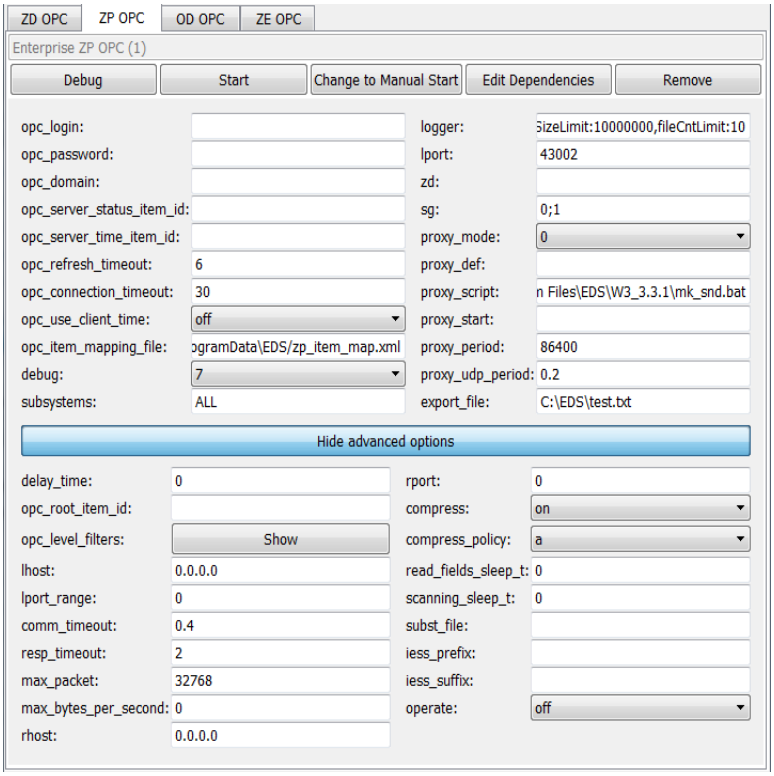


Figure 3.8 Options for ZP_OPC

PARAMETER	DESCRIPTION
opc_login:	**<user>** - login to use for connection to OPC server
opc_password:	**<password>** - password to use for connection to OPC server
opc_domain:	**<domain>** - domain to use for connection to OPC server
opc_server_status_item_id:	**<item id>** - id of OPC item which contains server status

Note: *These parameters should be obtained from the administrator of the OPC server to which EDS is connected. Often, their default values (empty) are enough for the connection to work properly, as OPC servers can be configured without the authorization requirement.*

opc_server_time_item_id:	**<item id>** - id of OPC item which contains current server time. This is useful for calculating time shift between OPC client and server when **opc_use_client_time** is enabled.
opc_refresh_timeout:	**<seconds>** - number of seconds after which inactive OPC item will be treated as timed out
opc_connection_timeout:	**<seconds>** - number of seconds after which inactive OPC server will be treated as disconnected

opc_use_ client_time:	**<off | on>** - determines the source of timestamps for OPC items **off** = use OPC item timestamps as returned by the OPC server (recommended) **on** = use client's time for OPC item timestamps
opc_item_ mapping_ file:	**<file path>** - path of XML file with definitions of item mapping rules Example file is placed in 'examples' folder in software installation catalogue. Within the XML file are comments describing parameters that can be configured. **zd_item_map.xml** is configured to be launched as default. Within this file, there are parameters for more advanced configuration (e.g. regular expressions for filtering or adding point prefixes). **opc_item_mapping_file** example: **C:\Program Files\EDS92\OpcClient\examples\zd_item_map.xml**
debug:	**<1 .. 7>** - debugging level; level of significance of messages to be logged; numbers have the following meaning: **0** Emergency, system is unusable **1** Alert, action must be taken immediately **2** Critical, critical conditions **3** Errors, error conditions **4** Warning, warning conditions **5** Notice, normal but significant condition **6** Informational **7** Debugging Effectively levels **3** to **7** are used. Selecting **debug=5** results in printout of messages of levels **5,4,3**. The default level is **3** (log errors only)
subsystems:	**<S1,S2,...>** - subsystems to be logged; list of names of the subsystems that need to be logged; special keyword **'ALL'** selects all subsystems, which is the default and recommended setting for normal operation
logger:	**<init_string>** - logger initialization string; example of Rotating Files Logger configuration: **rfile:5,fileName:/usr/eds/server/obj_ srv.log,fileSizeLimit:1000000,fileCntLimit:3** Example configuration for console logger: **console:7** Example configuration for windows events: **sysevent:3** Is possible use 2 or more loggers by separating by semicolon: Example use of console logger and windows events: **console:7; sysevent:3** Parameters desctiption: **rfile:7** - log level 7 overwrites default debug= parameter **fileSizeLimit:1000000** - max size of created log file (in bytes) **fileCntLimit:3** - number of log files to created (files: **obj_srv.log, obj_srv.log.0, obj_srv.log.1** will be created and rotary overwrite. The most current is always **obj_srv.log**.)
lport:	**<number>** - local UDP port number; selecting **0** results in automatic assignment of available port number by the operating system, (recommended); it may be necessary to define a port number if there is a firewall between this application and the destination, which has been configured to pass through only specified port numbers for security reasons
zd:	**<name>** - source name to be assigned for each point. Defining that points having ZD on their field, as given in this parameter, will be processed in this software.

sg:	**<0, 1, ...>** - security groups to be assigned. The goal of this parameter is to assign Security Groups (SG) to points exported from this point feeder (**ZP_OPC**). Database Access is connected to this feeder and gathers points from it - such points can be imported to server. After importing, the points will have SG designated, as set in this parameter. Database Access allows user to manually change the security groups for individual points (imported using the method described here or added manually).
proxy_ mode:	**<0 | 1>** - **0** - normal mode, **1** - proxy mode, only send values, no revise frames, **ZP** exports file, **ZD** use this file **Normal mode** is mode of bidirectional UDP communication. **Proxy mode** is unidirectional communication mode, where zd_opc only sends UDP frames to the server, not receiving nor waiting for UDP frames from EDS server.
proxy_def:	**<filename>** - file name with definition of points to manage File with definitions of names and point fields (in identical format as point definition export file from Database Access). This file is generated by **zd_opc** and used within it to determine which points will be sent to EDS server. Used only in proxy mode.
proxy_ script:	**<filename>** - script preparing the file to be transferred
proxy_start:	**<YYYYMMDDHHMISS>** - start time to run script for proxy mode
proxy_ period:	**<seconds>** - period to send file beginning from proxy_start=
proxy_udp_ period:	**<seconds>** - sleep time between sending UDP frames
export_file:	**<name>** - file to save points to. The file will be automatically created. The given catalog must exist and have write access.
lhost=	**<name | ip>** - local host name or IP number; the recommended setting is 0.0.0.0, as this opens local socket on all available network interfaces; it might, however, be necessary to limit communication to a specified network interface, what can be achieved by specifying appropriate value of this parameter
lport_range=	**<number>** - local UDP port range; allows to set the limit on used port numbers in order to narrow the port ranges opened In firewalls; specifying **lport=43000** and **lport_ range=100** will make the application attempt to bind port 43000 - if it is already used and thus not available, it will proceed with 43001 and up to 43100; first available port in this range will be used; if all ports are not available, the application will fail. If no firewalls in the network then this should be **0**.
comm_ timeout:	**<seconds>** - communication timeout (1.0 s); time in seconds when the application waits to receive acknowledge from the destination; this parameter should Contrary to **fillin_overlap** parameter that is related to the period of reestablishing connection.
resp_ timeout:	**<seconds>** - response timeout (2.0 s); time in seconds after which the application retransmits data packet, if it did not receive acknowledge from the destination; this may be increased if network delays are big

max_ packet=	**<bytes>** - the maximum size of a single UDP packet (before compression); in some network environments it may be necessary to set this number to 16384, 8192 or less, if there is no transmission
max_bytes_ per_second=	**<number>** - limit on the number of bytes sent; default **0** = check disabled
rhost:	**<name | ip>** - server address
rport:	**<number>** - server UDP port
compress:	**<on | off>** - controls compression of the transmitted data; switching it **on** lowers network load but may cause raise processor load
compress_ policy:	**<a | f | d>** - **a** = allow, **f** = force, **d** = deny
read_fields_ sleep_t:	**<seconds>** - seconds to sleep after each read fields items
scanning_ sleep_t:	**<seconds>** - seconds to sleep after each items path scan
subst_file:	**<name>** - file to read field substitutions from; file structure: **DFLT: <X1=Y1> <X2=Y2>** ... - defines options for all points **PNFQ: <X1=Y1> <X2=Y2>** - returns full point name (Fully Qualified Point Name): <short_name>.<unit_name>@<network_name> (e.g. ABCD.UNIT1@NET1) **IDCS=<idcs> <X1=Y1> <X2=Y2>** ... - defines for point **<idcs>** only Xn are EDS record type names plus some special keywords: **SKIP=<0 | 1>** - allows to ignore given process point **subst** is a shortened form of "substitution field" for automatic substitution of names and fields of exported points.
iess_prefix:	**<name>** - prefix added to each IESS
iess_suffix:	**<name>** - suffix added to each IESS
operate:	**<on | off>** - accept/send operate commands, controls possibility to write process point values back; values can be set only for process points originated by the Workstation on which this program runs

3.2.9 Configuring EDS OPC UA Data Output

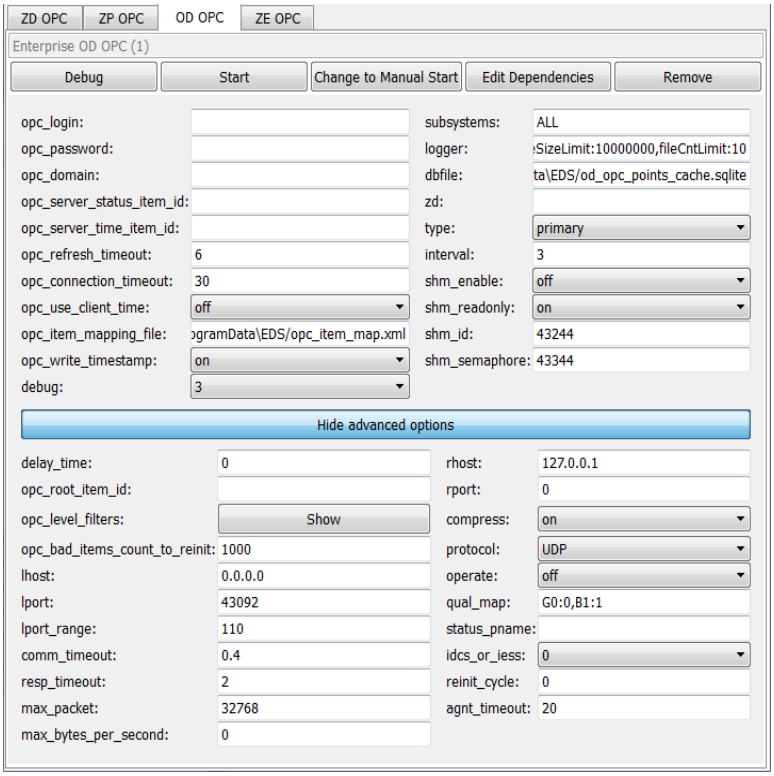


Figure 3.9 Options for OD_OPC parameter description

PARAMETER	DESCRIPTION
opc_login:	**<user>** - login to use for connection to OPC server
opc_ password:	**<password>** - password to use for connection to OPC server
opc_domain:	**<domain>** - domain to use for connection to OPC server
opc_server_ status_ item_id:	**<item id>** - id of OPC item which contains server status
opc_server_ time_item_ id:	**<item id>** - id of OPC item which contains current server time. This is useful for calculating time shift between OPC client and server when **opc_use_client_time** is enabled.

***Note:** These parameters should be obtained from the administrator of the OPC server to which EDS is connected. Often, their default values (empty) are enough for the connection to work properly, as OPC servers can be configured without the authorization requirement.*

PARAMETER	DESCRIPTION
opc_ refresh_ timeout:	**<seconds>** - number of seconds after which inactive OPC item will be treated as timed out
opc_ connection_ timeout:	**<seconds>** - number of seconds after which inactive OPC server will be treated as disconnected
opc_use_ client_time:	**<off | on>** - determines the source of timestamps for OPC items **off** = use OPC item timestamps as returned by the OPC server (recommended) **on** = use client's time for OPC item timestamps
opc_item_ mapping_ file:	**<file path>** - path of XML file with definitions of item mapping rules
opc_write_ timestamp:	**<on | off>** - write sample timestamps Disable this option if OPC DA server reports **BadWriteNotSupported** error
debug:	**<1 .. 7>** - debugging level; level of significance of messages to be logged; numbers have the following meaning: **0** Emergency, system is unusable **1** Alert, action must be taken immediately **2** Critical, critical conditions **3** Errors, error conditions **4** Warning, warning conditions **5** Notice, normal but significant condition **6** Informational **7** Debugging Effectively levels **3** to **7** are used. Selecting **debug=5** results in printout of messages of levels **5,4,3**. The default level is **3** (log errors only)
subsystems:	**<S1,S2,...>** - subsystems to be logged; list of names of the subsystems that need to be logged; special keyword **'ALL'** selects all subsystems, which is the default and recommended setting for normal operation
logger:	**<init_string>** - logger initialization string; example of Rotating Files Logger configuration: **rfile:5,fileName:/usr/eds/server/obj_ srv.log,fileSizeLimit:1000000,fileCntLimit:3** Example configuration for console logger: **console:7** Example configuration for windows events: **sysevent:3** Is possible use 2 or more loggers by separating by semicolon: Example use of console logger and windows events: **console:7; sysevent:3** Parameters desctiption: **rfile:7** - log level 7 overwrites default **debug= parameter** **fileSizeLimit:1000000** - max size of created log file (in bytes) **fileCntLimit:3** - number of log files to created (files: **obj_srv.log**, **obj_srv.log.0**, **obj_srv.log.1** will be created and rotary overwrite. The most current is obj_srv.log always)
dbfile:	**<name>** - name of configuration file to store point database cache.
zd:	source name or names identifying which process points should be updated by this application

PARAMETER	DESCRIPTION
type:	**<primary | backup | offline>** - source or output type; **primary** updates the process points, **backup** checks if a primary in on-line, **offline** synchronizes and does not update the process points (when primary timeouts, backup starts updating process point values)
interval:	**<seconds>** - seconds to sleep between writes; if negative, then it will synchronize with moments when ((time-shift) %% interval) = **0** e.g. **interval=-5.0** will start update cycles at 00, 05, 10, 15, etc. seconds
shm_enable:	**<on | off>** - turns on or off usage of shared memory
shm_ readonly:	**<on | off>** - if set to 'on', the application will not write any values to the shared memory, assuming some other process updates its contents
shm_id:	**<identifier>** - shared memory identifier The identifier of the shared memory can be created bythe on-line data server (SRV Server) to improve communication with other services.The ID number can be found in EDS Server Configuration tool.
shm_ semaphore:	**<identifier>** - semaphore identifier associated with shared memory The identifier of the semaphore used to control accessto the shared memory. The ID number can be found in EDS Server Configuration tool.
delay_time:	**<seconds>** - start delay time to wait for dependent programs, default **0.0** sec
opc_root_ item_id:	**<item id>** - id of root item for scanned subtree of OPC server items Setting this option will filter out all OPC objects which are not descendants of the specified item id. This parameter is used to limit the number of points to be exported from OPC server. Usually the parameter is left blank, and only set according to needs. It can be gathered from OPC item tree in test graphic client connected to the OPC server.
opc_level_ filters:	**<filter list>** - list of regular expressions which are used to filter OPC items when browsing the server. Every element in the list filters a single level of items in the OPC item hierarchy. Empty elements allow all items from the level. This parameter is used to limit the number of points to be exported from OPC server. Usually the parameter is left blank, and only set according to needs. It can be gathered from OPC item tree in test graphic client connected to the OPC server.
opc_bad_ items_ count_to_ reinit:	**<number>** - number of OPC item subscription errors required to trigger reinitialization of OPC client
lhost:	**<name | ip>** - local host name or IP number; the recommended setting is **0.0.0.0**, as this opens local socket on all available network interfaces; it might, however, be necessary to limit communication to a specified network interface, what can be achieved by specifying appropriate value of this parameter
lport:	**<number>** - local UDP port number; selecting **0** results in automatic assignment of available port number by the operating system, what is the recommended option; it may be necessary to define a port number if there is a firewall between this application and the destination, which has been configured to pass through only specified port numbers for security reasons

PARAMETER	DESCRIPTION
lport_range:	**<number>** - local UDP port range; allows to set the limit on used port numbers in order to narrow the port ranges opened In firewalls; specifying **lport=43000** and **lport_range=100** will make the application attempt to bind port 43000 - if it is already used and thus not available, it will proceed with 43001 and up to 43100; first available port in this range will be used; if all ports are not available, the application will fail. If no firewalls in the network then this should be **0**.
comm_ timeout:	**<seconds>** - communication timeout (1.0 s); time in seconds when the application waits to receive acknowledge from the destination; this parameter should
resp_ timeout:	**<seconds>** - response timeout (2.0 s); time in seconds after which the application retransmits data packet, if it did not receive acknowledge from the destination; this may be increased if network delays are big
max_packet:	**<bytes>** - the maximum size of a single UDP packet (before compression); in some network environments it may be necessary to set this number to 16384, 8192 or less, if there is no transmission
max_bytes_ per_second:	**<number>** - limit on the number of bytes sent; default **0** = check disabled
rhost:	**<name | ip>** - server address
rport:	**<number>** - server UDP port
compress:	**<on | off>** - controls compression of the transmitted data; switching it on lowers network load but may cause raise processor load
protocol:	**<UDP | TCP>** - protocol to use
operate:	**<on | off>** - accept/send operate commands, controls possibility to write process point values back; values can be set only for process points originated by the Workstation on which this program runs
qual_map:	**<0 | 1>** - quality from values range
status_ pname:	**<name>** - name of IESS indicates status point
reinit_cycle:	**<n>** - **n=0**: no reinit, **n>0**: init in every **n** cycle
agnt_ timeout:	**<seconds>** - agent timeout (20 s); number of seconds after which Server will change status of this scanner from on-line to timedout when it stops updating process point values

3.2.10 EDS OPC UA ZE tab

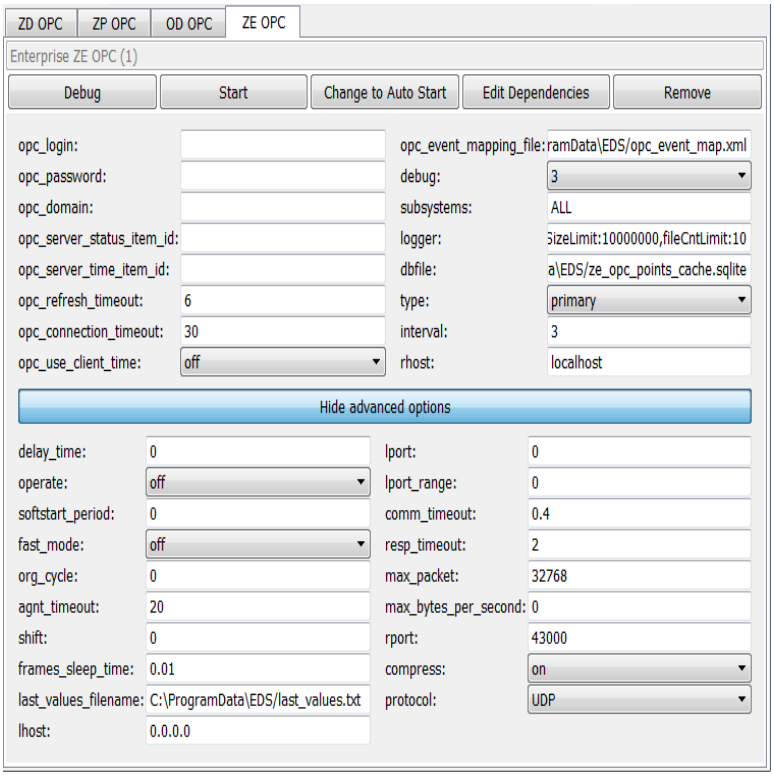


Figure 3.10 ZE Ovation tab

PARAMETER	DESCRIPTION
opc_login:	**<user>** - login to use for connection to OPC server
opc_password:	**<password>** - password to use for connection to OPC server
opc_domain:	**<domain>** - domain to use for connection to OPC server
opc_server_ status_item_id:	**<item id>** - id of OPC item which contains server status
opc_server_ time_item_id:	**<item id>** - id of OPC item which contains current server time. This is useful for calculating time shift between OPC client and server when **opc_use_client_time** is enabled.

Note: *These parameters should be obtained from the administrator of the OPC server to which EDS is connected. Often, their default values (empty) are enough for the connection to work properly, as OPC servers can be configured without the authorization requirement.*

PARAMETER	DESCRIPTION
opc_refresh_ timeout:	**<seconds>** - number of seconds after which inactive OPC item will be treated as timed out
opc_connection_ timeout:	**<seconds>** - number of seconds after which inactive OPC server will be treated as disconnected

PARAMETER	DESCRIPTION
opc_use_client_time:	**<off | on>** - determines the source of timestamps for OPC items **off** = use OPC item timestamps as returned by the OPC server (recommended) **on** = use client's time for OPC item timestamps
opc_event_mapping_file:	**<file path>** - path of XML file with definitions of event mapping rules Example file is placed in 'examples' folder in software installation catalogue. Within the XML file are comments describing parameters that can be configured. **event_map_basic.xml** is configured to be launched as default. Within this file, there are parameters for more advanced configuration (e.g. regular expressions for filtering or adding point prefixes) **event_map_basic.xml** example: **C:\Program Files\EDS92\OpcClient\examples\event_map_basic.xml**
debug:	Level of significance of messages to be logged; numbers have the following meaning: ■ **0** - Emergency, system is unusable ■ **1** - Alert, action must be taken immediately ■ **2** - Critical, critical conditions ■ **3** - Errors, error conditions ■ **4** - Warning, warning condition ■ **5** - Notice, normal but significant condition ■ **6** - Informational ■ **7** - Debugging EDS effectively uses levels from **3** to **7**. When **debug:** is set to **5**, then it results in a printout of messages of levels **5**,**4** and **3**. The default level is **3** (log errors only).
subsystems:	List of names of the EDS subsystems that need to be logged. A special keyword **ALL** selects all subsystems, which is the default and recommended setting for normal operation.
logger:	Logger initialization string. For more information see *1.4.7 Logger Initialization String on page 34*.
dbfile:	**<name>** - name of configuration file to stores point database cache.
type:	**<primary | backup | offline>** - source or output type; **primary** updates the process points, **backup** checks if a primary in on-line, **offline** synchronizes and does not update the process points (when primary timeouts, backup starts updating process point values).
interval:	**<seconds>** - seconds to sleep between updates; if negative, then it will synchronize with moments when when the time shift is a multiple of the given value, e.g. **interval=-5.0** will start update cycles every 5 seconds.
rhost:	**<name | ip>** - server address
delay_time:	**<seconds>** - start delay time to wait for dependent programs, default **0.0** sec This parameter is used to set waiting time for the completion of launching the OPC server, to which EDS Server is to be connected. It is used within systems where the OPC Server is on the same workstation as EDS Server and only to OPC Servers that malfunction in cases when the connection is executed too early.

PARAMETER	DESCRIPTION
operate:	**<on | off>** - accepts/sends operate commands, controls possibility to write process point values back; values can be set only for process points originated by the workstation on which this program runs.
softstart_period:	**<number>** - slows down the initialization process by a number of seconds. Multiple applications attempting to access Ovation at the same time may cause problems. Setting a value greater then **0** here can prevent that. Default value is **0**. This parameter is identical to **delay_time** parameter.
fast_mode:	**<on | off>** - send data without waiting for ACK [acknowledge]. This parameter is related to UDP communication with the EDS Server for current data. The feeder sends UDP frames with values for all points, in the meantime not waiting for the server confirmation for individual frames, but for the completion of cycle of sending all points which results in the server sending one joint confirmation. This means that there is no genuine unidirectional communication, yet the lack of confirmation for each single frame results in the cycle being faster and points being refreshed more often.
org_cycle:	**<0-255>** -sets data encoding, values **0** an **1** prompt encoding of all data, setting values from **2** to **255** prompt encoding data for every **n**-th point, where **n** is the value set.
agnt_timeout:	**<seconds>** - agent timeout, number of seconds after which EDS Server will change the status of this scanner from on-line to timedout when it stops updating process point values.
shift:	**<seconds>** - time shift for the update, can be used to determine exact moments of the updates.
frames_sleep_time:	**<seconds>** - sleep time after send frame, default 0.05 sec - UDP frames in network communication between software on same or different workstations
lhost:	**<name | ip>** - local host name or IP number; the recommended setting is **0.0.0.0**, as this opens local socket on all available network interfaces; it might, however, be necessary to limit communication to a specified network interface, which can be achieved by specifying appropriate value of this parameter.
lport:	**<number>** - local UDP port number. Selecting **0** results in automatic assignment of available port number by the operating system, which is the recommended option; it may be necessary to define a port number if there is a firewall between this application and the destination.
lport_range:	**<number>** - local UDP port range; allows to set the limit on used port numbers in order to narrow the port ranges opened In firewalls; specifying **lport=43000** and **lport_range=100** will make the application attempt to bind port **43000** - if it is already used and thus not available, it will proceed with **43001** and up to **43100**; first available port in this range will be used; if all ports are unavailable, the application will fail. If no firewalls in the network then this should be set to **0**.
comm_timeout:	**<seconds>** - communication timeout time in seconds when the application waits to receive acknowledge from the destination (by d efault it is set to **0.4** seconds).
resp_timeout:	**<seconds>** - response timeout (2.0 s). Time in seconds after which the application retransmits data packet, if it did not receive acknowledge from the destination; this may be increased if network delays are big

PARAMETER	DESCRIPTION
max_packet:	**<bytes>** - the maximum size of a single UDP packet (before compression); in some network environments it may be necessary to set this number to 16384, 8192 or less, if there is no transmission
max_bytes_per_second:	**<number>** - limit on the number of bytes sent default **0** = check disabled
rport:	**<number>** - server UDP port
compress:	**<on | off>** - controls compression of the transmitted data switching it on lowers network load but may cause raise processor load
protocol:	**<UDP | TCP>** - protocol to use

3.3 Enterprise OPC DCOM Server for Data Access

The DCOM Enterprise OPC Server has implemented OPC 2.0 Standard.

The Enterprise OPC Server is identified by name:

Enterprise OPC Server

and CLSID:

3bdfd250-2ac7-11d6-b36e-0080c8e7df60.

Enterprise OPC Server starts automatically after any OPC client request.

3.3.1 Installing Enterprise OPC DCOM Server

Package: EDS92OPCServer-**x**bit-rel.**x.x.x.x**.msi.

In order to install EDSOpcServer Interface package, it is necessary to login to the MS Windows environment with administrative privileges.

The software installs by default in the directory: **C:\Program Files\EDS92\OpcServer** .

da_srv.exe - is the Enterprise OPC Server program

opcsrvconfig.exe - is the configuration utility for the Enterprise OPC Server program

3.3.2 Configuring Enterprise OPC DCOM Server

The configuration utility **opcsrvconfig.exe** enables you to modify the communication parameters for the Enterprise OPC Server application. It is automatically started after successful installation of all package components. Its two tabs allow editing parameters of Enterprise OPC Server application.

The Enterprise OPC Server **Configuration** utility (**C:\Program Files\EDS92\OpcServer\opcsrvconfig.exe**) icons are located on the menu bar:

BUTTON	DESCRIPTION
	Reloads options from registers
	Saves options to registers

BUTTON	DESCRIPTION
	Imports from a file
	Exports to a file
	Starts the system interface for management of Services
	Display messages
	Opens **Help** window

Options for **Enterprise OPC Server Configuration**

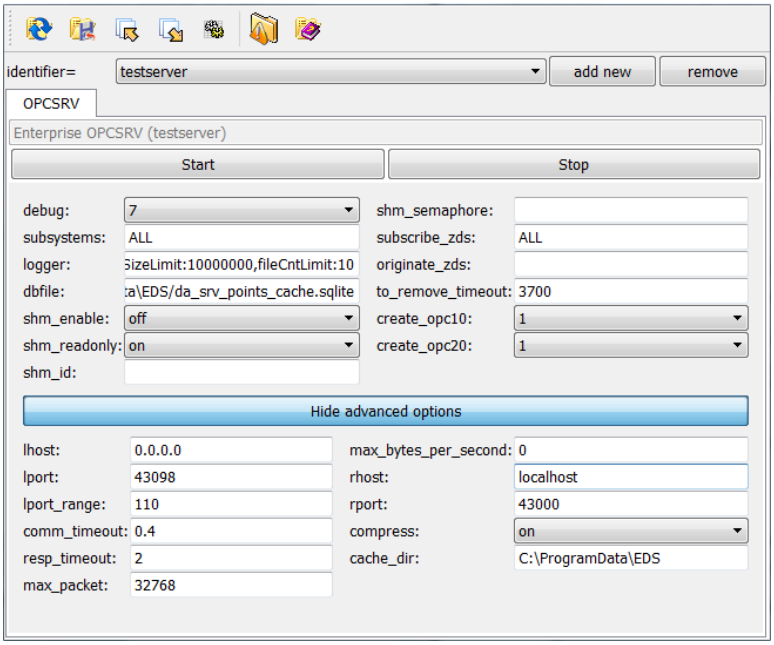


Figure 3.11 OPC Server configuration

To apply the configuration changes it is necessary to restart the OPC Server.

PARAMETER	DESCRIPTION
debug:	`<1 .. 7>` - debugging level; level of significance of messages to be logged; numbers have the following meaning: `0` Emergency, system is unusable `1` Alert, action must be taken immediately `2` Critical, critical conditions `3` Errors, error conditions `4` Warning, warning conditions `5` Notice, normal but significant condition `6` Informational `7` Debugging Effectively levels `3` to `7` are used. Selecting `debug=5` results in printout of messages
subsystems:	`<S1,S2,...>` - subsystems to be logged; list of names of the subsystems that need to be logged; special keyword `'ALL'` selects all subsystems, which is the default and recommended setting for normal operation

PARAMETER	DESCRIPTION
logger:	**<init_string>** - logger initialization string; example of Rotating Files Logger configuration: **rfile:5,fileName:/usr/eds/server/obj_srv.log,fileSizeLimit:1000000,fileCntLimit:3** Example configuration for console logger: **console:7** Example configuration for windows events: **sysevent:3** It is possible to use 2 or more loggers by separating by semicolon: Example use of console logger and windows events: **console:7; sysevent:3** Parameters description: **rfile:7** - log level 7 overwrites default debug= parameter **fileSizeLimit:1000000** - max size of created log file (in bytes) **fileCntLimit:3** - number of log files to created (**files: obj_srv.log, obj_srv.log.0, obj_srv.log.1** will be created and rotary overwrite. The most current is **obj_srv.log** always)
dbfile:	**<name>** - name of configuration file to stores point database cache.
shm_enable:	**<on | off>** - turns usage of shared memory **on** or **off** .
shm_ readonly:	**<on | off>** - if set to **on**, the application will not write any values to the shared memory, assuming some other process updates its contents.
shm_id:	**<identifier>** - shared memory identifier.
shm_ semaphore:	**<identifier>** - semaphore identifier associated with shared memory.
subscribe_ zds:	**<name>** - source names of points that are to be published by the OPC DA Server
originate_ zds:	**<name>** - source names of points for which the OPC DA Server will accept writes
to_remove_ timeout:	**<seconds>** - time after a group of not used group of points will be deleted (default 1 hour)
create_ opc10:	**<1 | 0>** - **1** = enables OPC 1.0 interface, **0** = disables OPC 1.0 interface
create_ opc20:	**<1 | 0>** - **1** = enables OPC 2.0 interface, **0** = disables OPC 2.0 interface
lhost:	**<name | ip>** - local host name or IP number; the recommended setting is **0.0.0.0**, as this opens local socket on all available network interfaces; it might, however, be necessary to limit communication to a specified network interface, what can be achieved by specifying appropriate value of this parameter
lport:	**<number>** - local UDP port number; selecting **0** results in automatic assignment of available port number by the operating system, what is the recommended option; it may be necessary to define a port number if there is a firewall between this application and the destination, which has been configured to pass through only specified port numbers for security reasons
lport_range:	**<number>** - local UDP port range, default **0**; allows to set the limit on used port numbers in order to narrow the port ranges opened In firewalls; specifying **lport=43000** and **lport_range=100** will make the application attempt to bind port 43000 - if it is already used and thus not available, it will proceed with 43001 and up to 43100; first available port in this range will be used; if all ports are not available, the application will fail. If no firewalls in the network then this should be **0**.

PARAMETER	DESCRIPTION
comm_ timeout:	**<seconds>** - communication timeout (1.0 s); time in seconds when the application waits to receive acknowledge from the destination; this parameter should
resp_ timeout:	**<seconds>** - response timeout (2.0 s); time in seconds after which the application retransmits data packet, if it did not receive acknowledge from the destination; this may be increased if network delays are big
max_packet:	**<bytes>** - the maximum size of a single UDP packet (before compression); in some network environments it may be necessary to set this number to 16384, 8192 or less, if there is no transmission
max_bytes_ per_second:	**<number>** - limit on the number of bytes sent; default 0 = check disabled
rhost:	**<name | ip>** - server address
rport:	**<number>** - server UDP port
compress:	**<on | off>** - controls compression of the transmitted data; switching it on lowers network load but may cause raise processor load
cache_dir:	**<directory>** - place where cache files are stored

3.4 Enterprise OPC UA Server for Data Access

The UA Enterprise OPC Server uses the newer Unified Architecture technology.

The UA Enterprise OPC Server is identified by name:

Enterprise OPC UA Server

Enterprise OPC UA Server starts automatically after any OPC client request.

3.4.1 Enterprise OPC Server Installation

Package: EDS92OPCSrvUa-**x**bit-rel.**x.x.x.x**.msi.

In order to install EDSOpcSrvUa Interface package, it is necessary to login to the MS Windows environment with administrative privileges.

The software installs by default in the directory: **C:\Program Files\EDS92\OpcSrvUa** .

opc_ua_srv.exe - is the Enterprise OPC UA Server program

opcuasrvconfig.exe - is the configuration utility for the Enterprise OPC UA Server program

3.4.2 Configuring Enterprise OPC UA Server

The configuration utility **opcuasrvconfig.exe** enables you to modify the communication parameters for the Enterprise OPC UA Server application. It is automatically started after successful installation of all package components. Its two tabs allow editing parameters of Enterprise OPC UA Server application.

The Enterprise OPC Server **Configuration** utility (**C:\Program Files\EDS92\OpcSrvUa\opcuasrvconfig.exe**) icons are located on the menu bar:

BUTTON	DESCRIPTION
	Reloads options from registers
	Saves options to registers
	Imports from a file
	Exports to a file
	Starts the system interface for management of Services
	Displays messages
	Opens **Help** window

Options for **Enterprise OPC Server Configuration**

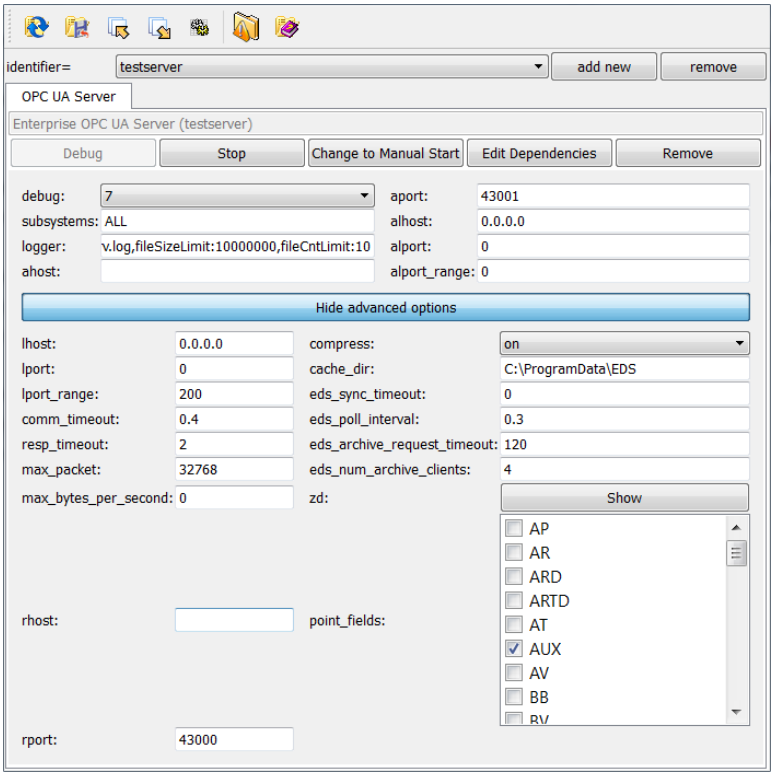


Figure 3.12 OPC Server configuration

To apply the configuration changes it is necessary to restart the OPC Server.

PARAMETER	DESCRIPTION
debug:	**<1 .. 7>** - debugging level; level of significance of messages to be logged; numbers have the following meaning: **0** Emergency, system is unusable **1** Alert, action must be taken immediately **2** Critical, critical conditions **3** Errors, error conditions **4** Warning, warning conditions **5** Notice, normal but significant condition **6** Informational **7** Debugging Effectively levels **3** to **7** are used. Selecting **debug=5** results in printout of messages
subsystems:	**<S1,S2,...>** - subsystems to be logged; list of names of the subsystems that need to be logged; special keyword **ALL** selects all subsystems, which is the default and recommended setting for normal operation
logger:	**<init_string>** - logger initialization string; example of Rotating Files Logger configuration: **rfile:5,fileName:/usr/eds/server/obj_srv.log,fileSizeLimit:1000000,fileCntLimit:3** Example configuration for console logger: **console:7** Example configuration for windows events: **sysevent:3** It is possible to use 2 or more loggers by separating by semicolon: Example use of console logger and windows events: **console:7; sysevent:3** Parameters description: **rfile:7** - log level 7 overwrites default debug= parameter **fileSizeLimit:1000000** - max size of created log file (in bytes) **fileCntLimit:3** - number of log files to created (**files: obj_srv.log, obj_srv.log.0, obj_srv.log.1** will be created and rotary overwrite. The most current is **obj_srv.log** always)
ahost:	**<name | ip>** - name or IP address of **ARCH Server**.
aport:	**<number>** - UDP port of **ARCH Server**.
alhost:	**<number>** - UDP port of **ARCH Server**.
alport:	**<number>** - local UDP port number for connections to **ARCH Server**.
alport_ range:	**<number>** - local UDP/TCP port range for connections to **ARCH Server**.
lhost:	**<name | ip>** - local host name or IP number; the recommended setting is **0.0.0.0**, as this opens local socket on all available network interfaces; it might, however, be necessary to limit communication to a specified network interface, what can be achieved by specifying appropriate value of this parameter
lport:	**<number>** - local UDP port number; selecting **0** results in automatic assignment of available port number by the operating system, what is the recommended option; it may be necessary to define a port number if there is a firewall between this application and the destination, which has been configured to pass through only specified port numbers for security reasons

PARAMETER	DESCRIPTION
lport_range:	**<number>** - local UDP port range, default **0**; allows to set the limit on used port numbers in order to narrow the port ranges opened In firewalls; specifying **lport=43000** and **lport_range=100** will make the application attempt to bind port 43000 - if it is already used and thus not available, it will proceed with 43001 and up to 43100; first available port in this range will be used; if all ports are not available, the application will fail. If no firewalls in the network then this should be **0**.
comm_ timeout:	**<seconds>** - communication timeout (1.0 s); time in seconds when the application waits to receive acknowledge from the destination; this parameter should
resp_ timeout:	**<seconds>** - response timeout (2.0 s); time in seconds after which the application retransmits data packet, if it did not receive acknowledge from the destination; this may be increased if network delays are big
max_packet:	**<bytes>** - the maximum size of a single UDP packet (before compression); in some network environments it may be necessary to set this number to 16384, 8192 or less, if there is no transmission
max_bytes_ per_second:	**<number>** - limit on the number of bytes sent; default 0 = check disabled
rhost:	**<name | ip>** - server address
rport:	**<number>** - server UDP port
compress:	**<on | off>** - controls compression of the transmitted data; switching it on lowers network load but may cause raise processor load
cache_dir:	**<directory>** - place where cache files are stored
eds_sync_ timeout:	**<seconds>** - EDS client synchronization timeout
eds_poll_ interval:	**<seconds>** - EDS client poll interval
eds_ archive_ request_ timeout:	**<seconds>** - EDS archive request timeout
eds_num_ archive_ clients:	**<number>** - number of concurrent ArchSrv clients
zd:	**<name>** - source names of points that are to be published by the OPC server
point_fields:	A list of EDS point fields provided by OPC server. Checking a given point field mark will activate it.

3.5 Accessing DCOM servers with an UA client

It is recommended to access DCOM servers with DCOM clients and UA servers with UA clients. Both clients are supplied in the EDS OPC Client package. However it is possible to access a DCOM server with an UA client. In order to do that, the following steps must be performed:

1. Obtain the URL address of the DCOM server. If it is not known, a number of scanner applications exist. Such an application can search the network for servers and provide the required URL.
2. Enter the server URL in the **opc_server_url** field in the OPC UA Client application.

3. Set the **zd** parameter in the client application to the same value as at the target server.
4. Set the rhost parameter appropriately to identify the server machine.

Such configuration should suffice. In some cases point mapping will be necessary. In order to set up point mapping, copy the contents of `C:\Program Files\EDS92\OpcClient\examples` (by default) to the `C:\Program Data\EDS` folder. These files are responsible for item mapping. For example modifying **zd_item_map.xml** by finding the line `<Regex PointIdMatch="(.*)" ItemIdFormat="ns=2;s=0:\1" />` and replacing `ns=2` and `s=0` with appropriate values will allow for desired mapping for **zd** data feeder.

3.6 Accessing UA servers with a DCOM client

It is recommended to access DCOM servers with DCOM clients and UA servers with UA clients. Both clients are supplied in the EDS OPC Client package. However it is possible to access an UA server with a DCOM client. In order to do that, the usage of a proxy server is required.

Note: *The information provided below is only an example and should be treated as suggestions rather than reliable information.*

To set up a connection perform the following steps:

1. Install an appropriate proxy program on the client machine.The recommended software for this is the **VISUAL_OPC_PROXY_NET** program by the company Technosoftware GmbH. By default the program is installed in the folder **C:\Program Files\Technosoftware GmbH\Visual OPC Proxy .NET**

Note: *This software is not a part of EDS and may require a separate license. The software by Technosoftware GmbH is used as an example. Any other program that performs the required functions may be used. As this example depends on outside software, this information may be outdated.*

2. Open the **VisualOpc40.ConfigurationTool.exe** executable located in the **VISUAL_OPC_ PROXY_NET** installation directory.
3. Click the **Manage** button and choose the **Visual OPC Proxy …** option from the menu.
4. In the newly opened window entitled **Configure COM Pseudo-Servers** click the right mouse button to open the context menu and select **Discover Servers…** .
5. In new window, in the **Host Name** field, enter the IP address of the desired OPC UA server and push the **Discover** button.
6. Choose the appropriate server address and in the newly opened window accept settings by pressing the **OK** button.
7. In the new window select DA to configure live data access or HDA to configure archive data access.
8. After finalizing configuration restart the **OPC UA Local Discovery Server** service.

CHAPTER 4

Gateways Package

IN THIS SECTION

4.1 Installing the EDS Gateways package.

The EDS Gateways package is a set of tools designed for:

- synchronizing online data,
- synchronizing historical data,
- reading and writing online values to files.

These functions are realized by different executables, installed by the following package:

EDS92GT-xbit-rel.x.x.x.x.msi

where **x** is the Operating System's bit version and **x.x.x.x** signifies the EDS software version. The installation is done by running the appropriate EDS package.

In order to install the EDS Gateways package, it is necessary to login to the MS Windows environment with administrative privileges. The software installs by default in the directory:

C:\Program Files\EDS92\GT

The package contains the following notable executables:

APPLICATION	NAME	DESCRIPTION
zd_file.exe	Data Source	Updates process point values in EDS from file
od_file.exe	Output Data	Writes process point values to file
zs_file.exe	Substitute Data	Substitute data feeder
gt_online.exe	Online Gateway	Online data gateway
gt_arch.exe	Historical Gateway	Historical data gateway
zd_modbus_tcp.exe	Modbus TCP	Modbus TCP online data feeder
zd_modbus_ser.exe	Modbus RTU	Modbus RTU online data feeder
zd_proxy.exe	Proxy feeder	Proxy feeder for one-way communication (see *A.4 One-way communication on page 139*)
zd_replay.exe	Replay feeder	Replays data from the historical database
zp_file.exe	Point Source	Imports the list of process points

4.2 Configuration

The configuration utility files (**fileconfig.exe**, **gtconfig.exe**, **mbconfig.exe**, **proxyconfig.exe** and **replayconfig.exe**) enable the user to modify the communication parameters for the applications. The tabs allow for editing the parameters of each application.

4.2.1 File configuration window

In order to open the File configuration window find and open the **fileconfig.exe** executable located by default in:

C:\Program Files\EDS92\GT

4.2.1.1 Source – ZD File tab

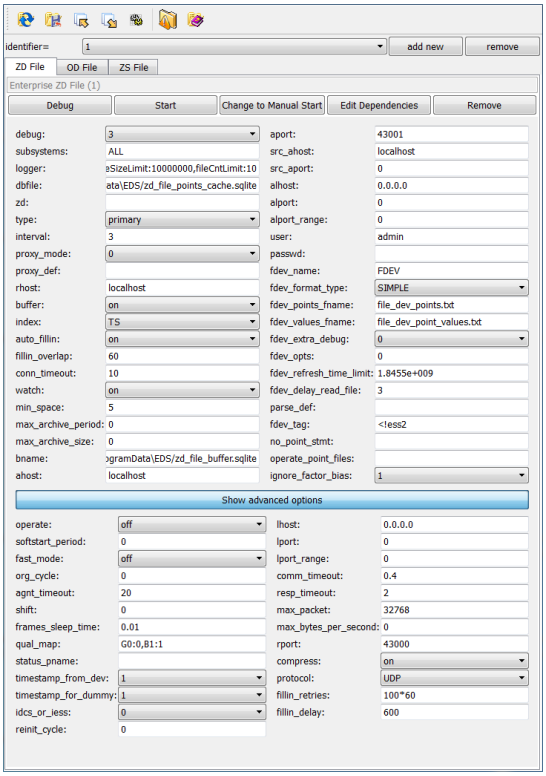


Figure 4.1 EDS Gateways Package configuration screen - Data Source ZD File tab

PARAMETER	DESCRIPTION
debug:	Level of significance of messages to be logged; numbers have the following meaning: ■ 0 - Emergency, system is unusable ■ 1 - Alert, action must be taken immediately ■ 2 - Critical, critical conditions ■ 3 - Errors, error conditions ■ 4 - Warning, warning condition ■ 5 - Notice, normal but significant condition ■ 6 - Informational ■ 7 - Debugging EDS effectively uses levels from 3 to 7. When **debug:** is set to 5, then it results in a printout of messages of levels 5,4 and 3. The default level is 3 (log errors only).
subsystems:	List of names of the EDS subsystems that need to be logged. A special keyword `ALL` selects all subsystems, which is the default and recommended setting for normal operation.
logger:	Logger initialization string. For more information see *1.4.7 Logger Initialization String on page 34*.
dbfile:	`<name>` - name of configuration file to store point database cache

PARAMETER	DESCRIPTION
type:	**<primary | backup | offline>** - source or output type; primary updates the process points, backup checks if a primary in on-line, offline synchronize and does not update the process points (when primary timeouts, backup starts updating process point values).
zd:	**<name1, name2,...>**- source name or names; Identifies which process points should be originated by this application; will be obsolete, use **org_ zd=** instead.
interval:	**<seconds>** - seconds to sleep between updates; if negative, then it will synchronize with moments when when the time shift is a multiple of the given value, e.g. **interval=-5.0** will start update cycles every 5 seconds.
proxy_mode:	**<0 | 1>** - **0** - normal mode, **1** - proxy mode.
proxy_def:	**<filename>** - file name with definition of points to manage
rhost:	**<name | ip>** - server address
buffer:	**<on | off>** - saves scanned values to an external file; if enabled (set to **on**), the scanner archives the values of process points that have the **AR=** parameter set to **L** or **F**.If disabled (set to **off**), process points are not archived.
index:	**<NONE | T | S | TS>** - forces creation of indices on archive data tables; **T** - timestamps, **S** - SIDs, **TS** - timestamps and SID's.
auto_fillin:	**<on | off>** - automatically fills in missing data on the Server.
fillin_overlap:	**<number>** - the number of seconds extending periods without connectivity.
conn_timeout:	**<number>** - the number of seconds that must pass before noticing lack of connection
watch:	**<on | off>** - deletes oldest archives when less than **min_space:** is available on the data file system.
min_space:	**<0.0 .. 95.0>** - minimum free space that must be left on the data file system (default **10**) expressed in percent.
max_archive_period:	**<n>** - keeps archives not older than **<n>** days; If set to **0**, checking the archives will not be performed.
max_archive_size:	**<n>** - keeps archives not bigger than **<n>** MB; If set to **0**, checking the archives will not be performed.
bname:	**<name>** - name of the file with buffered data.
ahost:	**<name | ip>** - name or IP address of **ARCH Server**.
aport:	**<number>** - UDP port of **ARCH Server**.
src_ahost:	**<name | ip>** - (wrapping source historian) address of **ARCH Server**.
src_aport:	**<number>** - (wrapping source historian) UDP port of **ARCH Server**.
alhost:	**<number>** - UDP port of **ARCH Server**.
alport:	**<number>** - local UDP port number for connections to **ARCH Server**.
alport_range:	**<number>** - local UDP/TCP port range for connections to **ARCH Server**.
user:	**<user>** - user name used for connections to **ARCH Server.**
passwd:	**<password>** - password used for connections to **ARCH Server**.

PARAMETER	DESCRIPTION
fdev_name:	**<name>** - local name of the device
fdev_format_type:	**<SIMPLE | NAMES | PMS | HTML>** - name of format type For fdev_format_type: **SIMPLE:** **file_dev_points.txt:** **fdev_a1 A** **fdev_a2 A** **fdev_b1 B** **fdev_b2 B** **fdev_p1 P** **fdev_p2 P** **file_dev_point_values.txt:** **33.763** **76.8** **0** **1** **AA** **FF** For fdev_format_type: **PMS:** **file_dev_point_values.txt:** **2-MAY-2003 13:23:20.00** **09LBA10CT201_XQ50 #** **12MAP20CP201_XQ50 1.56250E+00** **12LAE10CF901_XQ50 #** **12LAF10CF901_XQ50 0.00000E+00** **12MAC10CP205_XQ50 #** **12MAG20CT001_XQ50 1.93481E+02** **12PAB20CT002_XQ50 2.92480E+01** **12LCP01CF001_XQ50 0.00000E+00** **2-MAY-2003 13:23:20.09** For fdev_format_type: **NAMES:** **file_dev_points.txt is the copy or the same file like:** **file_dev_point_values.txt:** **a1 121.11 G** **a2 121.11 G** **b1 1 G** **b2 0 B** **p1 127 G** **p2 65535 F** For fdev_format_type: **HTML (only od_file):** **file_dev_points.txt** is the template to generate output file: **<!ess2 sprintf(%%4.2f,value("a1"))> <!ess2 quality("a1")>** **<!ess2 sprintf(%%4.2f,value("a2"))> <!ess2 quality("a2")>** **file_dev_point_values.txt** is the output file generated by template **99.99 G** **1.11 B**
fdev_points_fname:	**<name>** - name of file with IDCS point names
fdev_values_fname:	**<name>** - name of file with values of points
fdev_extra_debug:	**<1|0>** - **1** displays additional debug information

PARAMETER	DESCRIPTION
fdev_opts:	**<number>** - 8 bits opts: bit **0** decimal (1) : write bad point names bit **1** decimal (2) : save all items names bit **2** decimal (4) : load all items names bit **3** decimal (8) : allow write values file bit **4** decimal (16): allow write points file
fdev_refresh_time_limit:	**<time>** - maximum time (in seconds) to refresh **fdev_values_fname**; if exceed bad quality is set to **write**
fdev_delay_read_file:	**<time>** - time (in seconds) to wait after modifying **fdev_values_fname**
parse_def:	**<name>** - file name with shade date format definition for .txt mode
fdev_tag:	**<name>** - tag for HTML mode
no_point_stmt:	**<name>** - name for NO POINT statement
operate_point_files:	**<filenames>** - file names of points to operate command
ignore_factor_bias:	**<1 | 0>** - **1** = Don't calculate values by **F= B**= from **AUX** field
operate:	**<on | off>** - accepts/sends operate commands; controls possibility to write process point values back; values can be set only for process points originatingfrom the workstation on which this program runs.
softstart_period:	**<number>** - slows down the initialization process by a number of seconds. Multiple applications attempting to access Ovation at the same time may cause problems. Setting a value greater then **0** here can prevent that. Default value is **0**.
fast_mode:	**<on | off>** - sends data without waiting for ACK acknowlegement message.
org_cycle:	**<0-255>** -sets data encoding, values **0** an **1** prompt encoding of all data, setting values from **2** to **255** prompt encoding data for every **n**-th point, where **n** is the value set.
agnt_timeout:	**<seconds>** - agent timeout, number of seconds after which EDS Server will change the status of this scanner from on-line to timedout when it stops updating process point values.
shift:	**<seconds>** - time shift for the update, can be used to determine exact moments of the updates.
frames_sleep_time:	**<seconds>** - sleep time after frame is sent.
qual_map:	**<0 | 1>** - quality from values range
status_pname:	**<name>** - name of the point that indicates whether IESS is used.
timestamp_from_dev:	**<1 | 0>** - selecting **1** will set timestamp basing on the device time, selecting **0** will set timestamp basing on local time
timestamp_for_dummy:	**<1 | 0>** - selecting **1** will set timestamp for points not originating from the device, selecting **0** will not set a timestamp for those points.
reinit_cycle:	**<n>** - if the selected number is **0**, no reinitialisation will occur. If the selected number is greater than zero, reinitialisation will occur every **n**-th cycle.

PARAMETER	DESCRIPTION
lhost:	**<name | ip>** - local host name or IP number; the recommended setting is **0.0.0.0**, as this opens local socket on all available network interfaces; it might, however, be necessary to limit communication to a specified network interface, which can be achieved by specifying appropriate value of this parameter.
lport:	**<number>** - local UDP port number selecting **0** results in automatic assignment of available port number by the operating system, what is the recommended option; it may be necessary to define a port number if there is a firewall between this application and the destination, which has been configured to pass through only specified port numbers for security reasons
lport_range:	**<number>** - local UDP port range; allows to set the limit on used port numbers in order to narrow the port ranges opened In firewalls; specifying **lport=43000** and **lport_range=100** will make the application attempt to bind port **43000** - if it is already used and thus not available, it will proceed with **43001** and up to **43100**; first available port in this range will be used; if all ports are unavailable, the application will fail. If no firewalls in the network then this should be set to **0**.
comm_timeout:	**<seconds>** - communication timeout time in seconds when the application waits to receive acknowledge from the destination (by default it is set to **0.4** seconds).
resp_timeout:	**<seconds>** - response timeout time in seconds after which the application retransmits data packet. If it did not receive acknowledges from the destination. It may be increased if network delays are big (by default it is set to **2.0**).
max_packet:	**<bytes>** - a communication parameter (bytes). Specifies the maximum size of a single communication buffer (before compression). In some networks, may need to be decreased (for example to **16348**, **8192**, **4096** or **2048**).
max_bytes_per_second:	**<number>** - limit on the number of bytes sent (setting **0** disables the check).
rport:	**<number>** - server UDP port
compress:	**<on | off>** - controls compression of the transmitted data switching it on lowers network load but may cause raise in processor load.
protocol:	**<UDP | TCP>** - protocol to use.
fillin_retries:	**<spec>** - specifies how many times a fill-in operation for a given range should be performed.
fillin_delay:	**<number>** - number of seconds that must pass before starting an automatic fill-in.

4.2.1.2 Data Output – OD File tab

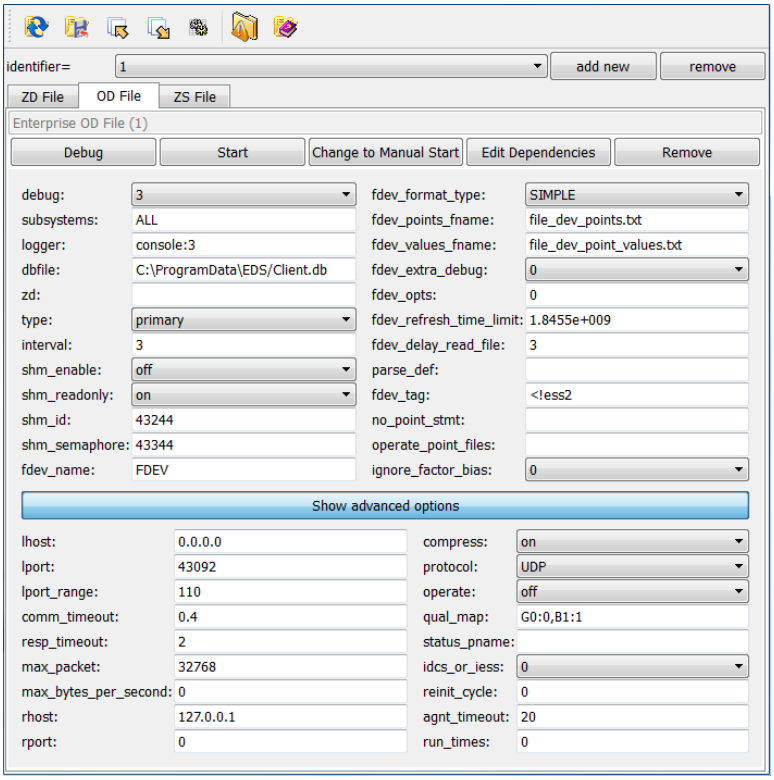


Figure 4.2 EDS Gateways Package configuration screen - Data Output OD File tab

PAPAMETER	DESCRIPTION
debug:	Level of significance of messages to be logged; numbers have the following meaning: ■ 0 - Emergency, system is unusable ■ 1 - Alert, action must be taken immediately ■ 2 - Critical, critical conditions ■ 3 - Errors, error conditions ■ 4 - Warning, warning condition ■ 5 - Notice, normal but significant condition ■ 6 - Informational ■ 7 - Debugging EDS effectively uses levels from 3 to 7. When **debug:** is set to 5, then it results in a printout of messages of levels 5,4 and 3. The default level is 3 (log errors only).
subsystems:	List of names of the EDS subsystems that need to be logged. A special keyword **ALL** selects all subsystems, which is the default and recommended setting for normal operation.
logger:	Logger initialization string. For more information see *1.4.7 Logger Initialization String on page 34*.
dbfile:	**<name>** - name of configuration file to store point database cache
zd:	**<name>** - source name for points

PAPAMETER	DESCRIPTION
type:	**<primary | backup | offline>** - source or output type **primary** updates the process points, **backup** checks if a primary in on-line, **offline** synchronizes and does not update the process points (when primary timeouts, backup starts updating process point values).
interval:	**<seconds>** - seconds to sleep between updates; if negative, then it will synchronize with moments when when the time shift is a multiple of the given value, e.g. **interval=-5.0** will start update cycles every 5 seconds.
shm_enable:	**<on | off>** - turns usage of shared memory **on** or **off** .
shm_readonly:	**<on | off>** - if set to **on**, the application will not write any values to the shared memory, assuming some other process updates its contents.
shm_id:	**<identifier>** - shared memory identifier.
shm_semaphore:	**<identifier>** - semaphore identifier associated with shared memory.
fdev_name:	**<name>** - local name of the device

PAPAMETER	DESCRIPTION
fdev_format_type:	**<SIMPLE | NAMES | PMS | HTML>** - name of format type For fdev_format_type: **SIMPLE:** **file_dev_points.txt:** **fdev_a1 A** **fdev_a2 A** **fdev_b1 B** **fdev_b2 B** **fdev_p1 P** **fdev_p2 P** **file_dev_point_values.txt:** **33.763** **76.8** **0** **1** **AA** **FF** For fdev_format_type: **PMS:** **file_dev_point_values.txt:** **2-MAY-2003 13:23:20.00** **09LBA10CT201_XQ50 #** **12MAP20CP201_XQ50 1.56250E+00** **12LAE10CF901_XQ50 #** **12LAF10CF901_XQ50 0.00000E+00** **12MAC10CP205_XQ50 #** **12MAG20CT001_XQ50 1.93481E+02** **12PAB20CT002_XQ50 2.92480E+01** **12LCP01CF001_XQ50 0.00000E+00** **2-MAY-2003 13:23:20.09** For fdev_format_type: **NAMES:** **file_dev_points.txt is the copy or the same file like:** **file_dev_point_values.txt:** **a1 121.11 G** **a2 121.11 G** **b1 1 G** **b2 0 B** **p1 127 G** **p2 65535 F** For fdev_format_type: **HTML (only od_file):** **file_dev_points.txt** is the template to generate output file: **<!ess2 sprintf(%%4.2f,value("a1"))> <!ess2 quality("a1")>** **<!ess2 sprintf(%%4.2f,value("a2"))> <!ess2 quality("a2")>** **file_dev_point_values.txt** is the output file generated by template **99.99 G** **1.11 B**
fdev_points_fname:	**<name>** - name of file with IDCS point names
fdev_values_fname:	**<name>** - name of file with values of points
fdev_extra_debug:	**<1|0>** - **1** displays additional debug information
fdev_opts:	**<number>** - 8 bits opts: bit **0** decimal (1) : write bad point names bit **1** decimal (2) : save all items names bit **2** decimal (4) : load all items names bit **3** decimal (8) : allow write values file bit **4** decimal (16): allow write points file

PAPAMETER	DESCRIPTION
fdev_refresh_time_ limit:	**<time>** - maximum time (in seconds) to refresh **fdev_values_fname**; if exceed bad quality is set to **write**
fdev_delay_read_file:	**<time>** - time (in seconds) to wait after modifying **fdev_values_fname**
parse_def:	**<name>** - file name with shade date format definition for .txt mode
fdev_tag:	**<name>** - tag for HTML mode
no_point_stmt:	**<name>** - name for NO POINT statement
operate_point_files:	**<filenames>** - file names of points to operate command
ignore_factor_bias:	**<1 | 0>** - **1** = Don't calculate values by **F= B**= from **AUX** field
lhost:	**<name | ip>** - local host name or IP number; the recommended setting is **0.0.0.0**, as this opens local socket on all available network interfaces; it might, however, be necessary to limit communication to a specified network interface, which can be achieved by specifying appropriate value of this parameter.
lport:	**<number>** - local UDP port number; selecting **0** results in automatic assignment of available port number by the operating system, which is the recommended option; it may be necessary to define a port number if there is a firewall between this application and the destination.
lport_range:	**<number>** - local UDP port range; allows to set the limit on used port numbers in order to narrow the port ranges opened In firewalls; specifying **lport=43000** and **lport_range=100** will make the application attempt to bind port **43000** - if it is already used and thus not available, it will proceed with **43001** and up to **43100**; first available port in this range will be used; if all ports are unavailable, the application will fail. If no firewalls in the network then this should be set to **0**.
comm_timeout:	**<seconds>** - communication timeout time in seconds when the application waits to receive acknowledge from the destination (by default it is set to **0.4** seconds).
resp_timeout:	**<seconds>** - response timeout time in seconds after which the application retransmits data packet. If it did not receive acknowledges from the destination. It may be increased if network delays are big (by default it is set to **2.0**).
max_packet:	**<bytes>** - a communication parameter (bytes). Specifies the maximum size of a single communication buffer (before compression). In some networks, may need to be decreased (for example to **16348**, **8192**, **4096** or **2048**).
max_bytes_per_ second:	**<number>** - limit on the number of bytes sent (setting **0** disables the check).
rhost:	**<name | ip>** - server address
rport:	**<number>** - server UDP port
compress:	**<on | off>** - controls compression of the transmitted data switching it on lowers network load but may cause raise in processor load.
protocol:	**<UDP | TCP>** - protocol to use
operate:	**<on | off>** - accepts/sends operate commands; controls possibility to write process point values back; values can be set only for process points originatingfrom the workstation on which this program runs.
qual_map:	**<0 | 1>** quality from the range of the values.

PAPAMETER	DESCRIPTION
status_pname:	**<name>** - name of IESS indicates status of the point.
reinit_cycle:	**<n>** - if the selected number is **0**, no reinitialisation will occur. If the selected number is greater than zero, reinitialisation will occur every **n**-th cycle.
agnt_timeout:	**<seconds>** - agent timeout, number of seconds after which EDS Server will change the status of this scanner from on-line to timedout when it stops updating process point values.
run_times:	**<n>** - setting **n=0**: runs the program always to infinity, setting **n>0**: runs **n** times

4.2.1.3 Substitute Data – ZS File tab

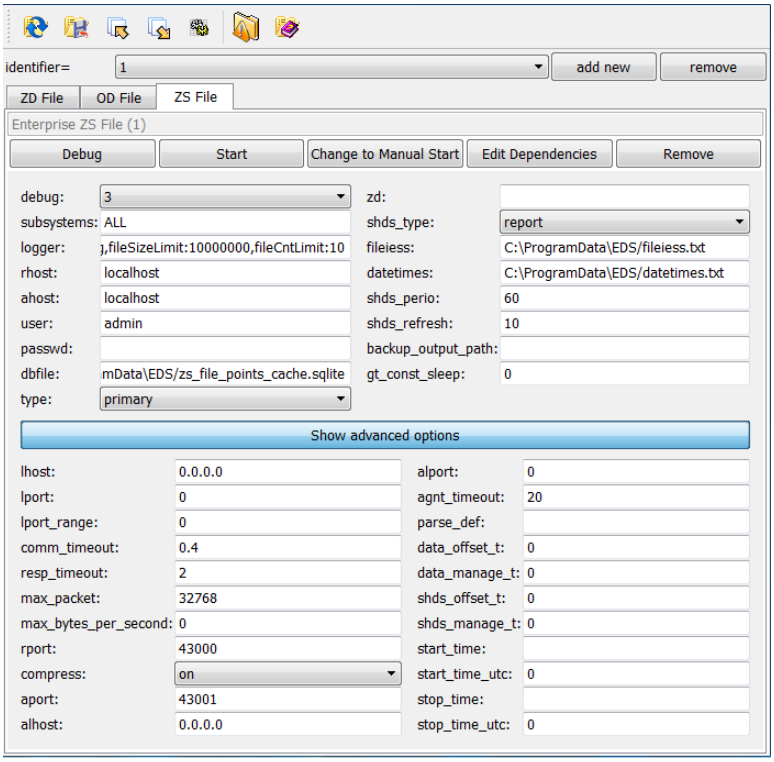


Figure 4.3 EDS Gateways Package configuration screen – Substitute Data ZS File tab

PARAMETER	DESCRIPTION
debug:	Level of significance of messages to be logged; numbers have the following meaning: ■ `0` - Emergency, system is unusable ■ `1` - Alert, action must be taken immediately ■ `2` - Critical, critical conditions ■ `3` - Errors, error conditions ■ `4` - Warning, warning condition ■ `5` - Notice, normal but significant condition ■ `6` - Informational ■ `7` - Debugging EDS effectively uses levels from `3` to `7`. When **debug:** is set to `5`, then it results in a printout of messages of levels `5`,`4` and `3`. The default level is `3` (log errors only).
subsystems:	List of names of the EDS subsystems that need to be logged. A special keyword `ALL` selects all subsystems, which is the default and recommended setting for normal operation.
logger:	Logger initialization string. For more information see *1.4.7 Logger Initialization String on page 34*.
rhost:	`<name | ip>` - server address
ahost:	`<name | ip>` - name or IP address of **ARCH Server**.
user:	`<user>` - user name used for connections to **ARCH Server.**
passwd:	`<password>` - password used for connections to **ARCH Server**.
dbfile:	`<name>` - name of configuration file to stores point database cache.
type:	`<primary | backup | offline>` - source or output type; primary updates the process points, backup checks if a primary in on-line, offline synchronize and does not update the process points (when primary timeouts, backup starts updating process point values).
zd:	`<name>` - source name for points
shds_type:	`<xml_t1 | txt | report | telog>` - type of program mode
fileiess:	`<name>` - file name with cross, line format: `~fname~iess~`, or shades values for report mode, or connection string from telog mode, eg. "`Provider=sqloledb;Auto Translate=True;Data Source=RHOST; Initial Catalog=DBNAME;Integrated Security=SSPI;Persist Security Info=False;`" or "`Provider=sqloledb;Auto Translate=True;Data Source=RHOST; Initial Catalog=DBNAME;Password=PASS;User ID=USER;`"
datetimes:	`<filename>` - name of the file which stores information about written data
shds_perio:	`<seconds>` - sleep time between read values from files or sources
shds_ refresh:	`<number>` - number of **shds_perio:** to integrate all data
backup_ output_path:	`<path>` - directory path of the managed files backups
gt_const_ sleep:	`<seconds>` - seconds to sleep after each point shade

PARAMETER	DESCRIPTION
lhost:	**<name | ip>** - local host name or IP number; the recommended setting is **0.0.0.0**, as this opens local socket on all available network interfaces; it might, however, be necessary to limit communication to a specified network interface, which can be achieved by specifying appropriate value of this parameter.
lport:	**<number>** - local UDP port number selecting **0** results in automatic assignment of available port number by the operating system, what is the recommended option; it may be necessary to define a port number if there is a firewall between this application and the destination, which has been configured to pass through only specified port numbers for security reasons
lport_range:	**<number>** - local UDP port range; allows to set the limit on used port numbers in order to narrow the port ranges opened In firewalls; specifying **lport=43000** and **lport_range=100** will make the application attempt to bind port **43000** - if it is already used and thus not available, it will proceed with **43001** and up to **43100**; first available port in this range will be used; if all ports are unavailable, the application will fail. If no firewalls in the network then this should be set to **0**.
comm_timeout:	**<seconds>** - communication timeout time in seconds when the application waits to receive acknowledge from the destination (by default it is set to **0.4** seconds).
resp_timeout:	**<seconds>** - response timeout time in seconds after which the application retransmits data packet. If it did not receive acknowledges from the destination. It may be increased if network delays are big (by default it is set to **2.0**).
max_packet:	**<bytes>** - a communication parameter (bytes). Specifies the maximum size of a single communication buffer (before compression). In some networks, may need to be decreased (for example to **16348**, **8192**, **4096** or **2048**).
max_bytes_per_second:	**<number>** - limit on the number of bytes sent (setting **0** disables the check).
rport:	**<number>** - server UDP port
compress:	**<on | off>** - controls compression of the transmitted data switching it on lowers network load but may cause raise in processor load.
aport:	**<number>** - UDP port of **ARCH Server**.
alhost:	**<number>** - UDP port of **ARCH Server**.
alport:	**<number>** - local UDP port number for connections to **ARCH Server**.
agnt_timeout:	**<seconds>** - agent timeout, number of seconds after which EDS Server will change the status of this scanner from on-line to timedout when it stops updating process point values.
parse_def:	**<name>** - file name with shade date format definition for .txt mode.
data_offset_t:	**<seconds>** - time filter to manage data from: **actual time** + **data_offset_t** to **actual time** + **data_offset_t** + **data_manage_t** **data_offset_t** may by positive or negative.
data_manage_t:	**<seconds>** - seconds for manage data time filter (see **data_offset_t**). If **<=0** then no filter applies.
shds_offset_t:	**<seconds>** - time filter to manage shades from: **actual time** + **data_offset_t** to **actual time** + **data_offset_t** + **data_manage_t** **shds_offset_t** may by positive or negative.
shds_manage_t:	**<seconds>** - seconds for manage shades time filter (see **shds_offset_t**). if **<=0** then no filter applies.

PARAMETER	DESCRIPTION
start_time:	**<YYYYMMDDHHMISS>** - work start time.
start_time_utc:	**<sec from 1970>** - work start time in UTC format.
stop_time:	**<YYYYMMDDHHMISS>** - work stop time.
stop_time_utc:	**<sec from 1970>** - work stop time in UTC format.

4.2.2 Gateways Configuration

In order to open the Gateways Configuration window find and open the **gtconfig.exe** executable located by default in:

C:\Program Files\EDS92\GT

4.2.2.1 ONLINE tab

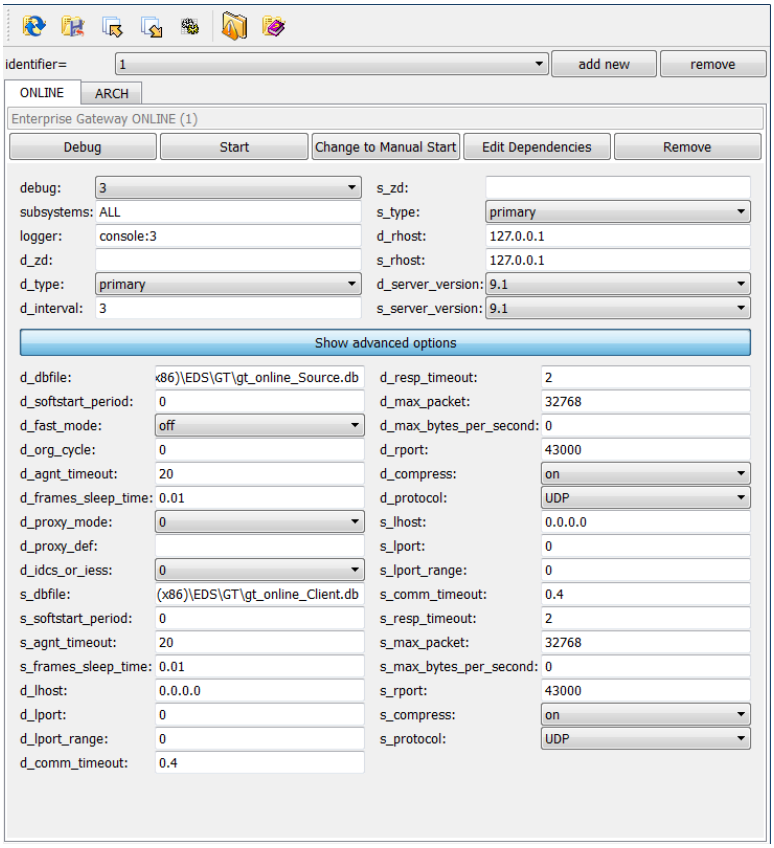


Figure 4.4 Gateways Configuration - ONLINE tab

PARAMETER	DESCRIPTION
debug:	Level of significance of messages to be logged; numbers have the following meaning: ■ `0` - Emergency, system is unusable ■ `1` - Alert, action must be taken immediately ■ `2` - Critical, critical conditions ■ `3` - Errors, error conditions ■ `4` - Warning, warning condition ■ `5` - Notice, normal but significant condition ■ `6` - Informational ■ `7` - Debugging EDS effectively uses levels from `3` to `7`. When **debug:** is set to `5`, then it results in a printout of messages of levels `5`,`4` and `3`. The default level is `3` (log errors only).
subsystems:	List of names of the EDS subsystems that need to be logged. A special keyword `ALL` selects all subsystems, which is the default and recommended setting for normal operation.
logger:	Logger initialization string. For more information see *1.4.7 Logger Initialization String on page 34*.
d_zd:	`<name1,name2,...>`- source name or names Identifies which process points should be originated by this application; will be obsolete, use **org_zd=** instead
d_type:	`<primary | backup | offline>` - source or output type `primary` updates the process points, `backup` checks if a primary in on-line, `offline` synchronizes and does not update the process points (when primary timeouts, backup starts updating process point values).
d_interval:	`<seconds>` - seconds to sleep between updates; if negative, then it will synchronize with moments when when the time shift is a multiple of the given value, e.g. **interval=**`-5.0` will start update cycles every 5 seconds.
s_zd:	`<name1, name2,...>` - source name or names; identifies which process points should be originated by this application; will be obsolete, user `org_zd=` instead
s_type:	`<primary | backup | offline>` - source or output type `primary` updates the process points, `backup` checks if a primary in on-line, `offline` synchronizes and does not update the process points (when primary timeouts, backup starts updating process point values).
d_rhost=	`<name | ip>` - Server address
s_rhost:	`<name | ip>` - server address
d_server_version:	`<version>` - version of driver for destination server (e.g. `9.0`)
s_server_version:	`<version>` - version of driver for source server (e.g. `9.0`)
d_dbfile:	`<name>` - name of configuration file to store point database cache.
d_softstart_ period:	`<number>` - slows down the initialization process by a number of seconds. Multiple applications attempting to access Ovation at the same time may cause problems. Setting a value greater then `0` here can prevent that. Default value is `0`.
d_fast_mode:	`<on | off>` - sends data without waiting for ACK acknowlegement message.

PARAMETER	DESCRIPTION
d_org_cycle:	**<0-255>** -sets data encoding, values **0** an **1** prompt encoding of all data, setting values from **2** to **255** prompt encoding data for every **n**-th point, where **n** is the value set.
d_agnt_timeout:	**<seconds>** - agent timeout, number of seconds after which the Server will change the status of this scanner from on-line to timedout when it stops updating process point values.
d_frames_sleep_time:	**<seconds>** - sleep time after frame is sent.
d_proxy_mode:	**<0 | 1>** - **0** - normal mode, **1** - proxy mode.
d_proxy_def:	**<filename>** - file name with definition of points to manage
s_dbfile:	**<name>** - name of configuration file to stores point database cache.
s_softstart_period:	**<number>** - slows down the initialization process by a number of seconds. Multiple applications attempting to access Ovation at the same time may cause problems. Setting a value greater then **0** here can prevent that. Default value is **0**.
s_agnt_timeout:	**<seconds>** - agent timeout, number of seconds after which the Server will change the status of this scanner from on-line to timedout when it stops updating process point values.
s_frames_sleep_time:	**<seconds>** - sleep time after frame is sent.
d_lhost:	**<name | ip>** - local host name or IP number; the recommended setting is **0.0.0.0**, as this opens local socket on all available network interfaces; it might, however, be necessary to limit communication to a specified network interface, what can be achieved by specifying appropriate value of this parameter
d_lport:	**<number>** - local UDP port number; selecting **0** results in automatic assignment of available port number by the operating system, which is the recommended option; it may be necessary to define a port number if there is a firewall between this application and the destination.
d_lport_range:	**<number>** - local UDP port range; allows to set the limit on used port numbers in order to narrow the port ranges opened In firewalls; specifying **lport=43000** and **lport_range=100** will make the application attempt to bind port **43000** - if it is already used and thus not available, it will proceed with **43001** and up to **43100**; first available port in this range will be used; if all ports are unavailable, the application will fail. If no firewalls in the network then this should be set to **0**.
d_comm_timeout:	**<seconds>** - communication timeout time in seconds when the application waits to receive acknowledge from the destination (by default it is set to **0.4** seconds).
d_resp_timeout:	**<seconds>** - response timeout time in seconds after which the application retransmits data packet. If it did not receive acknowledges from the destination. It may be increased if network delays are big (by default it is set to **2.0**).
d_max_packet:	**<bytes>** - a communication parameter (bytes). Specifies the maximum size of a single communication buffer (before compression). In some networks, may need to be decreased (for example to **16348**, **8192**, **4096** or **2048**).
d_max_bytes_per_second:	**<number>** - limit on the number of bytes sent (setting **0** disables the check).
d_rport:	**<number>** - Server UDP port

PARAMETER	DESCRIPTION
d_compress:	**<on | off>** - controls compression of the transmitted data switching it on lowers network load but may cause raise in processor load.
d_protocol:	**<UDP | TCP>** - protocol to use
s_lhost:	**<name | ip>** - local host name or IP number; the recommended setting is **0.0.0.0**, as this opens local socket on all available network interfaces; it might, however, be necessary to limit communication to a specified network interface, what can be achieved by specifying appropriate value of this parameter
s_lport:	**<number>** - local UDP port number selecting 0 results in automatic assignment of available port number by the operating system, what is the recommended option; it may be necessary to define a port numberif there is a firewall between this application and the destination, which has been configured to pass through only specified port numbers for security reasons
s_lport_range:	**<number>** - local UDP port range, default 0 Allows to set the limit on used port numbers in order to narrow the port ranges opened In firewalls; specifying lport=43000 and lport_range=100 will make the application attempt to bind port 43000 - if it is already used and thus not available, it will proceed with 43001 and up to 43100; first available port in this range will be used; if all ports are not available, the application will fail. If no firewalls in the network then this should be 0.
s_comm_timeout:	**<seconds>** - communication timeout (1.0 s) time in seconds when the application waits to receive acknowledge from the destination ; this parameter should
s_resp_timeout:	**<seconds>** - response timeout (2.0 s) time in seconds after which the application retransmits data packet, if it did not receive acknowledge from the destination ; this may be increased if network delays are big
s_max_packet:	**<bytes>** - the maximum size of a single UDP packet (before compression); in some network environments it may be necessary to set this number to 16384, 8192 or less, if there is no transmission
s_max_bytes_ per_second:	**<number>** - limit on the number of bytes sent default 0 = check disabled
s_rport:	**<number>** - Server UDP port
s_compress:	**<on | off>** - controls compression of the transmitted data switching it on lowers network load but may cause raise processor load
s_protocol:	**<UDP | TCP>** - protocol to use

4.2.2.2 ARCH tab

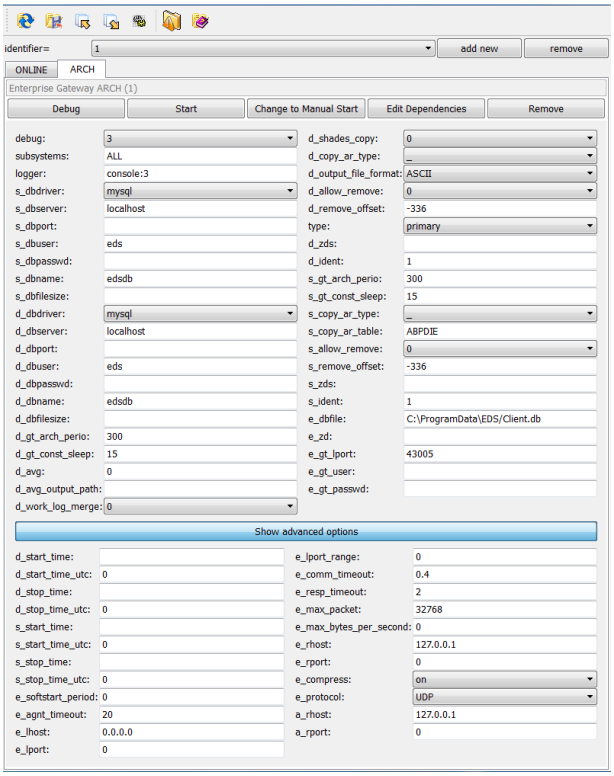


Figure 4.5 Gateways Configuration - ARCH tab

PARAMETER	DESCRIPTION
debug:	Level of significance of messages to be logged; numbers have the following meaning: ■ `0` - Emergency, system is unusable ■ `1` - Alert, action must be taken immediately ■ `2` - Critical, critical conditions ■ `3` - Errors, error conditions ■ `4` - Warning, warning condition ■ `5` - Notice, normal but significant condition ■ `6` - Informational ■ `7` - Debugging EDS effectively uses levels from `3` to `7`. When **debug:** is set to `5`, then it results in a printout of messages of levels `5`,`4` and `3`. The default level is `3` (log errors only).
subsystems:	List of names of the EDS subsystems that need to be logged. A special keyword `ALL` selects all subsystems, which is the default and recommended setting for normal operation.
logger:	Logger initialization string. For more information see *1.4.7 Logger Initialization String on page 34*.
s_dbdriver:	`<mysql | postgresql | sqlite | sqlserver>` - choice of the SQL database driver.
s_dbserver:	`<name | ip>` - name or IP address of the host where SQL engine runs.

PARAMETER	DESCRIPTION
s_dbport:	**<number>** - port for connection to SQL database.
s_dbuser:	**<name>** - name of user.
s_dpasswd:	**<password | @filename>** - user's password or name of file containing the password.
s_dname:	**<name>** - name of database.
s_dbfilesize:	**<size TB|GB|MB|KB>** - specified only for Sqlserver; size of database data file,expressed by an integer with suffixes **[KB,MB,GB,TB]**.
d_dbdriver:	**<mysql | postgresql | sqlite | sqlserver>** - choice of the SQL database driver.
d_dbserver	**<name | ip>** - name or IP address of the host where SQL engine runs.
d_dbport:	**<number>** - port for connection to SQL database.
d_dbuser:	**<name>** - name of user.
d_dbpasswd:	**<password | @filename>** - user's password or name of file containing the password.
d_dbname:	**<name>** - name of database.
d_dbfilesize:	**<size TB|GB|MB|KB>** - specified only for Sqlserver; size of database data file,expressed by an integer with suffixes **[KB,MB,GB,TB]**.
d_gt_arch_perio:	**<number>** - actual archive run period (in seconds).
d_gt_const_sleep:	**<number>** - old archives sleep time (in seconds).
d_avg:	**<number>** - average sample period.
d_avg_output_path:	**<path>** - path for avg output files.
d_work_log_merge:	**<0 | 1>** - work log merge: setting **0**does not merge, setting **1** merges.
d_shades_copy:	**<0 | 1>** - shades copy: **0**-no, **1**-yes
d_copy_ar_type:	**<_ | E | L | *>** - archive data points to copy **E** - external **L** - local ***** - all
d_output_file_ format:	**<ASCII | BIN>** - output format for avg files **ASCII** - plain text **BIN** - binary format
d_allow_remove:	**<0 | 1>** - allow to remove tables after copying: **0** - no remove; **1** - allow to remove data from hour tables with **remove_offset parameter**
d_remove_offset:	**<number>** - offset in hours for remove tables after copying, should be **<0** to remove past hour tables should be **==0** to remove just copied table should be **>0** to remove future tables (useful when replace old data)
type:	**<primary | backup | offline>** - source or output type; primary updates the process points, backup checks if a primary in on-line, offline synchronize and does not update the process points (when primary timeouts, backup starts updating process point values).
d_zds:	**<ZD1,ZD2,...>** - list of zd to be copied, if empty then all zd's will be copied

PARAMETER	DESCRIPTION
d_ident:	**<number>** - number identifying instance of gt_arch, used in primary and backup mode
s_gt_arch_perio:	**<number>** - actual archive run period (in seconds)
s_gt_const_sleep:	**<number>** - old archives sleep time (in seconds)
s_copy_ar_type:	**<_ | E | L | *>** - archive data points to copy **E** - external **L** - local ***** - all
s_copy_ar_table:	**<A | B | P | D | I | E>** - archive data tables to copy **A** - ANALOG **B** - BINARY **P** - PACKED **D** - DOUBLE **I** - INT64 **E** - EVENT
s_allow_remove:	**<0 | 1>** - allow to remove tables after copying: **0** - no remove; **1** - allow to remove after a period defined by **remove_offset**
s_remove_offset:	**<number>** - offset in hours to remove tables after copying, should be **<0** to remove past hour tables should be **==0** to remove just copied table should be **>0** to remove future tables (useful when replace old data)
s_zds:	**<ZD1,ZD2,...>** - list of zd to be copied, if empty then all zd's will be copied
s_ident:	**<number>** - number identifying instance of gt_arch, used in primary and backup mode
e_dbfile:	**<name>** - name of configuration file to stores point database cache.
e_zd:	**<name1, name2,...>**- source name or names; identifies which process points should be originated by this application; will be obsolete, user **org_zd=** instead
e_gt_lport:	**<number>** - Gateway UDP port
e_gt_user:	**<user>** - name of user
e_gt_passwd:	**<password>** - user's password
d_start_time:	**<YYYYMMDDHHMISS>** - work start time
d_start_time_utc:	**<sec from 1970>** - work start time in UTC format
d_stop_time:	**<YYYYMMDDHHMISS>** - work stop time
d_stop_time_utc:	**<sec from 1970>** - work stop time in UTC format
s_start_time:	**<YYYYMMDDHHMISS>** - work start time
s_start_time_utc:	**<sec from 1970>** - work start time in UTC format
s_stop_time:	**<YYYYMMDDHHMISS>** - work stop time
s_stop_time_utc:	**<sec from 1970>** - work stop time in UTC format
e_softstart_period:	**<number>** - slows down the initialization process by a number of seconds. Multiple applications attempting to access Ovation at the same time may cause problems. Setting a value greater then **0** here can prevent that. Default value is **0**.

PARAMETER	DESCRIPTION
e_agnt_timeout:	**<seconds>** - agent timeout, number of seconds after which the Server will change the status of this scanner from on-line to timedout when it stops updating process point values.
e_lhost:	**<name | ip>** - local host name or IP number; the recommended setting is **0.0.0.0**, as this opens local socket on all available network interfaces; it might, however, be necessary to limit communication to a specified network interface, what can be achieved by specifying appropriate value of this parameter
e_lport:	**<number>** - local UDP port number; selecting **0** results in automatic assignment of available port number by the operating system, which is the recommended option; it may be necessary to define a port number if there is a firewall between this application and the destination.
e_lport_range:	**<number>** - local UDP port range; allows to set the limit on used port numbers in order to narrow the port ranges opened In firewalls; specifying **lport=43000** and **lport_range=100** will make the application attempt to bind port **43000** - if it is already used and thus not available, it will proceed with **43001** and up to **43100**; first available port in this range will be used; if all ports are unavailable, the application will fail. If no firewalls in the network then this should be set to **0**.
e_comm_timeout:	**<seconds>** - communication timeout time in seconds when the application waits to receive acknowledge from the destination (by default it is set to **0.4** seconds).
e_resp_timeout:	**<seconds>** - response timeout time in seconds after which the application retransmits data packet. If it did not receive acknowledges from the destination. It may be increased if network delays are big (by default it is set to **2.0**).
e_max_packet:	**<bytes>** - a communication parameter (bytes). Specifies the maximum size of a single communication buffer (before compression). In some networks, may need to be decreased (for example to **16348**, **8192**, **4096** or **2048**).
e_max_bytes_per_second:	**<number>** - limit on the number of bytes sent (setting **0** disables the check).
e_rhost:	**<name | ip>** - server address
e_rport:	**<number>** - server UDP port
e_compress:	**<on | off>** - controls compression of the transmitted data switching it on lowers network load but may cause raise in processor load.
e_protocol:	**<UDP | TCP>** - protocol to use
a_rhost:	**<name | ip>** - server address
a_rport:	**<number>** - server UDP port

4.2.3 Defining Modbus address references

4.2.3.1 Addressing

The IDCS address takes the form **nXXyyyyy**, where:

- **n** - node number
- **XX** =

- IR (input registers)
- HR (holding registers)
- IS (input signals)
- CS (control signals)

- **yyyyy** - register number without the leading digit. part of the address does not have a fixed length. It can be, for example, `00003`, `0003`, `003`, `03`, `3`. The zeroes are usually added for convenience, as the number can range up to `10000`.

Example of referring to an address:

IESS	IDCS	MODBUS ADDRESS
Measurement 1	1IR00001	300001
Measurement 2	1IR00002	300002
Memory 1	1HR00001	400001
Memory 2	1HR00002	400002
DigIn 1	1IS00001	100001
DigIn 2	1IS00002	100002
Driver 1	1CS1	1
Driver 2	1CS2	2

4.2.3.2 The AUX field

The input in the **AUX** field modifies the data conversion. The commands used should be separated by spaces.

INPUT	DESCRIPTION
S16	16 bit register including sign
U16	16 bit register without sign
S32	32 bit register including sign
U32	32 bit register without sign
FL	float
UM10K	decreases the number 10000 times, 32 bit. e.g. will change 1 milion to 100. Useful in managing large numbers.
MS16	All these parameters are the same as the ones described above with the exception of the **M**- prefix. The **M**- prefix reverses the msb and lsb (more significant bit and less significant bit) in the data feed. In the normal order msb appears first and lsb second. If the m- prefix is used, the bits will be read in reversed order. Use it if your data is in the format lsb - msb.
MU16	
MS32	
MU32	
MFL	
MUM10K	

4.2.3.3 Scaling

The table below shows why scaling is necessary in data conversion. PLC/DCS does not hold, for instance, negative values. In order to retrive data from PLC/DCS, we must specify how the scaling will be executed, for example:

TEMPERATURE	PLC/DCS	EDS
100 °C	32000	100 °C
-50 °C	0	-50 °C

In order to execute scaling, two parametars must be specified:

PARAMETER	NAME	DESCRIPTION
F	**Factor**	factor for multiplication
B	**Bias**	adjustment of the 0 point, e.g. setting 5 will add 5 to all the values.

The scaling formula using the aforementioned values takes the following form:

EDS=PLC/DCS * F + B

In the example presented above, the parameters would be respectively:

F = 150/32000

B = -50

The parameters should be included in the **AUX** field. The **AUX** field essentially instructs the program how to read data from PLC/DCS.

In case of **IR** or **HR** values in IDCS, the process point must relate to a specific bit in the register. Example of supported syntax:

```
RT=BINARY IESS='HR0006.0' ZD='Mdb' IDCS='003HR0006.0' DESC='Holding
Register 6 bit 0'

RT=BINARY IESS='HR0006.1' ZD='Mdb' IDCS='003HR0006.1' DESC='Holding
Register 6 bit 1'

RT=BINARY IESS='HR0006.2' ZD='Mdb' IDCS='003HR0006.2' DESC='Holding
Register 6 bit 2'

RT=BINARY IESS='HR0006.3' ZD='Mdb' IDCS='003HR0006.3' DESC='Holding
Register 6 bit 3'

RT=BINARY IESS='HR0006.4' ZD='Mdb' IDCS='003HR0006.4' DESC='Holding
Register 6 bit 4'

RT=BINARY IESS='HR0006.8' ZD='Mdb' IDCS='003HR0006.8' DESC='Holding
Register 6 bit 8'
```

4.2.4 Serial Modbus Configuration

In order to open the Modbus Configuration window find and open the **mbconfig.exe** executable located by default in:

C:\Program Files\EDS92\GT

4.2.4.1 ZD TCP and OD TCP parameters

The parameters presented here apply to both ZD TCP and OD TCP tabs.

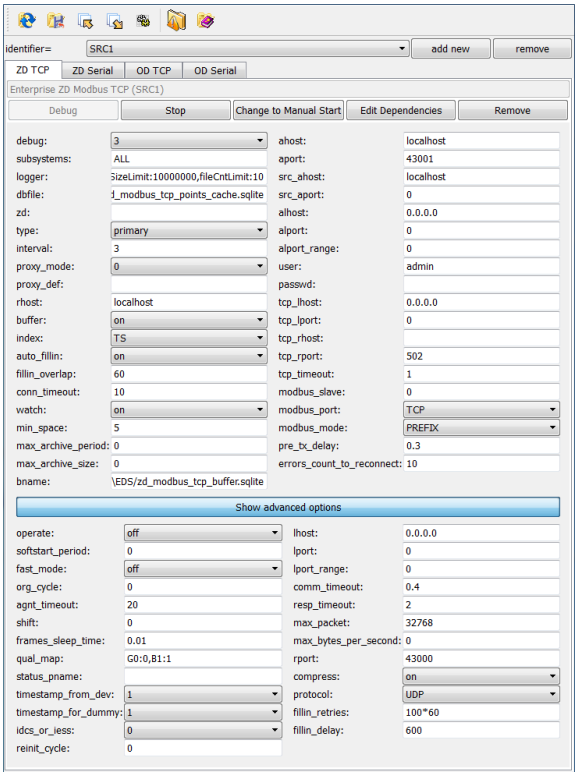


Figure 4.6 TCP, Serial Modbus configuration - TCP parameters

PARAMETER	DESCRIPTION
debug:	Level of significance of messages to be logged; numbers have the following meaning: ■ 0 - Emergency, system is unusable ■ 1 - Alert, action must be taken immediately ■ 2 - Critical, critical conditions ■ 3 - Errors, error conditions ■ 4 - Warning, warning condition ■ 5 - Notice, normal but significant condition ■ 6 - Informational ■ 7 - Debugging EDS effectively uses levels from 3 to 7. When **debug:** is set to 5, then it results in a printout of messages of levels 5,4 and 3. The default level is 3 (log errors only).
subsystems:	List of names of the EDS subsystems that need to be logged. A special keyword `ALL` selects all subsystems, which is the default and recommended setting for normal operation.
logger:	Logger initialization string. For more information see *1.4.7 Logger Initialization String on page 34*.
dbfile:	`<name>` - name of configuration file to store point database cache

PARAMETER	DESCRIPTION
zd:	**<name1>[,<name2>,...]** - source name or names; identifies which process points should be originated by this application; will be obsolete, use **org_zd=** instead
type:	**<primary | backup | offline>** - source or output type; primary updates the process points, backup checks if a primary in on-line, offline synchronize and does not update the process points (when primary timeouts, backup starts updating process point values).
interval:	**<seconds>** - seconds to sleep between updates; if negative, then it will synchronize with moments when when the time shift is a multiple of the given value, e.g. **interval=-5.0** will start update cycles every 5 seconds.
proxy_mode:	**<0 | 1>** - **0** - normal mode, **1** - proxy mode.
proxy_def:	**<filename>** - file name with definition of points to manage
rhost:	**<name | ip>** - Server address
buffer:	**<on | off>** - saves scanned values to an external file; if enabled (set to **on**), the scanner archives the values of process points that have the **AR=** parameter set to **L** or **F**.If disabled (set to **off**), process points are not archived.
index:	**<NONE | T | S | TS>** - forces creation of indices on archive data tables; **T** - timestamps, **S** - SIDs, **TS** - timestamps and SID's.
auto_fillin:	**<on | off>** - automatically fills in missing data on the Server.
fillin_overlap:	**<number>** - the number of seconds extending periods without connectivity.
conn_timeout:	**<number>** - the number of seconds that must pass before noticing lack of connection.
watch:	**<on | off>** - deletes oldest archives when less than **min_space:** is available on the data file system.
min_space:	**<0.0 .. 95.0>** - minimum free space that must be left on the data file system (default **10**) expressed in percent.
max_archive_period:	**<n>** - keeps archives not older than **<n>** days; If set to **0**, checking the archives will not be performed.
max_archive_size:	**<n>** - keeps archives not bigger than **<n>** MB; If set to **0**, checking the archives will not be performed.
bname:	**<name>** - name of the file with buffered data.
ahost:	**<name | ip>** - name or IP address of **ARCH Server**.
aport:	**<number>** - UDP port of **ARCH Server**.
src_ahost:	**<name | ip>** - (wrapping source historian) address of **ARCH Server**.
src_aport:	**<number>** - (wrapping source historian) UDP port of **ARCH Server**.
alhost:	**<number>** - UDP port of **ARCH Server**.
alport:	**<number>** - local UDP port number for connections to **ARCH Server**.
alport_range:	**<number>** - local UDP/TCP port range for connections to **ARCH Server**.
user:	**<user>** - user name used for connections to **ARCH Server.**

PARAMETER	DESCRIPTION
passwd:	**<password>** - password used for connections to **ARCH Server**.
tcp_lhost:	**<name | ip>** - optional, local name to use
tcp_lport:	**<number>** - optional, local port number
tcp_rhost:	**<name | ip>** - host to connect to
tcp_rport:	**<number>** - port to connect to
tcp_timeout:	**<seconds>** - timeout on communication
modbus_slave:	**<0 | 1>** - **0** - Master, **1** - Slave
modbus_port:	**<TCP | UDP | SERIAL>** - Port type: **TCP** or **UDP** or **SERIAL**
modbus_mode:	**<PREFIX | CRC | ASCII>** - **PREFIX** - 6 bytes prefix on frame begin; normal mode for TCP port, **CRC** - 2 bytes CRC on frame end; normal mode for serial port **ASCII** - modbus ascii mode
pre_tx_delay:	**<seconds>** - time in seconds to sleep before sending the Modbus frame
errors_count_to_ reconnect:	**<number>** - number of possible communication errors before reconnecting
operate:	**<on | off>** - accepts/sends operate commands; controls possibility to write process point values back; values can be set only for process points originating from the workstation on which this program runs.
softstart_period:	**<number>** - slows down the initialization process by a number of seconds. Multiple applications attempting to access Ovation at the same time may cause problems. Setting a value greater then **0** here can prevent that. Default value is **0**.
fast_mode:	**<on | off>** - sends data without waiting for ACK acknowlegement message.
org_cycle:	**<0-255>** -sets data encoding, values **0** an **1** prompt encoding of all data, setting values from **2** to **255** prompt encoding data for every **n**-th point, where **n** is the value set.
agnt_timeout:	**<seconds>** - agent timeout, number of seconds after which EDS Server will change the status of this scanner from on-line to timedout when it stops updating process point values.
shift:	**<seconds>** - time shift for the update, can be used to determine exact moments of the updates.
frames_sleep_time:	**<seconds>** - sleep time after frame is sent.
qual_map:	**<1 | 0>** - quality from values range
status_pname:	**<name>** - name of the point that indicates whether IESS is used.
timestamp_from_dev:	**<1 | 0>** - selecting **1** will set timestamp basing on the device time, selecting **0** will set timestamp basing on local time
timestamp_for_dummy:	**<1 | 0>** - selecting **1** will set timestamp for points not originating from the device, selecting **0** will not set a timestamp for those points.
reinit_cycle:	**<n>** - if the selected number is **0**, no reinitialisation will occur. If the selected number is greater than zero, reinitialisation will occur every **n**-th cycle.

PARAMETER	DESCRIPTION
lhost:	**<name | ip>** - local host name or IP number; the recommended setting is **0.0.0.0**, as this opens local socket on all available network interfaces; it might, however, be necessary to limit communication to a specified network interface, which can be achieved by specifying appropriate value of this parameter.
lport:	**<number>** - local UDP port number selecting **0** results in automatic assignment of available port number by the operating system, what is the recommended option; it may be necessary to define a port number if there is a firewall between this application and the destination, which has been configured to pass through only specified port numbers for security reasons
lport_range:	**<number>** - local UDP port range; allows to set the limit on used port numbers in order to narrow the port ranges opened In firewalls; specifying **lport=43000** and **lport_range=100** will make the application attempt to bind port **43000** - if it is already used and thus not available, it will proceed with **43001** and up to **43100**; first available port in this range will be used; if all ports are unavailable, the application will fail. If no firewalls in the network then this should be set to **0**.
comm_timeout:	**<seconds>** - communication timeout time in seconds when the application waits to receive acknowledge from the destination (by default it is set to **0.4** seconds).
resp_timeout:	**<seconds>** - response timeout time in seconds after which the application retransmits data packet. If it did not receive acknowledges from the destination. It may be increased if network delays are big (by default it is set to **2.0**).
max_packet:	**<bytes>** - a communication parameter (bytes). Specifies the maximum size of a single communication buffer (before compression). In some networks, may need to be decreased (for example to **16348**, **8192**, **4096** or **2048**).
max_bytes_per_second:	**<number>** - limit on the number of bytes sent (setting **0** disables the check).
rport:	**<number>** - server UDP port
compress:	**<on | off>** - controls compression of the transmitted data switching it on lowers network load but may cause raise in processor load.
protocol:	**<UDP | TCP>** - protocol to use.
fillin_retries:	**<spec>** - specifies how many times a fill-in operation for a given range should be performed.
fillin_delay:	**<number>** - number of seconds that must pass before starting an automatic fill-in.

4.2.4.2 ZD Serial and OD Serial parameters

The parameters presented here apply to both ZD Serial and OD Serial tabs.

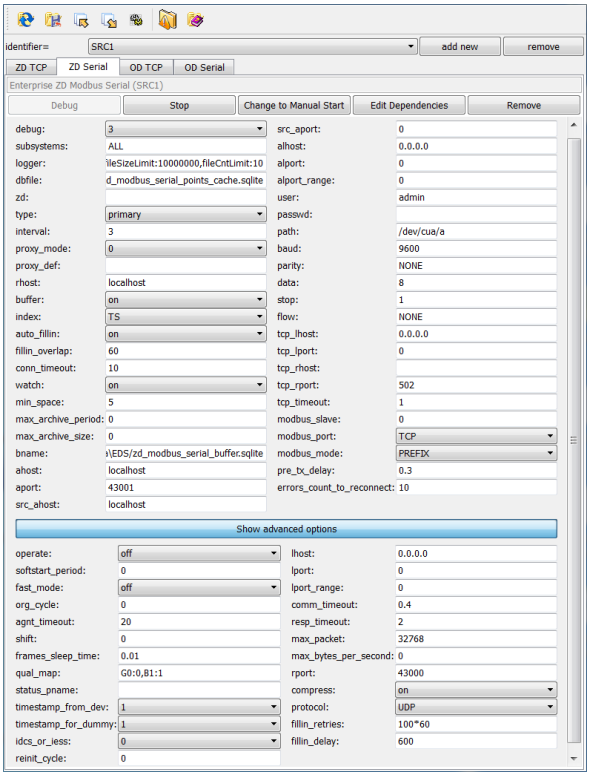


Figure 4.7 EDS Gateways Package configuration screen - Modbus RTU Serial

PARAMETER	DESCRIPTION
debug:	Level of significance of messages to be logged; numbers have the following meaning: ■ 0 - Emergency, system is unusable ■ 1 - Alert, action must be taken immediately ■ 2 - Critical, critical conditions ■ 3 - Errors, error conditions ■ 4 - Warning, warning condition ■ 5 - Notice, normal but significant condition ■ 6 - Informational ■ 7 - Debugging EDS effectively uses levels from 3 to 7. When **debug:** is set to 5, then it results in a printout of messages of levels 5,4 and 3. The default level is 3 (log errors only).
subsystems:	List of names of the EDS subsystems that need to be logged. A special keyword **ALL** selects all subsystems, which is the default and recommended setting for normal operation.
logger:	Logger initialization string. For more information see *1.4.7 Logger Initialization String on page 34*.
dbfile:	**<name>** - name of configuration file to store point database cache.

PARAMETER	DESCRIPTION
zd:	**<name1,name2,...>**- source name or names; identifies which process points should be originated by this application; will be obsolete, use **org_zd=** instead
type:	**<primary | backup | offline>** - source or output type; primary updates the process points, backup checks if a primary in on-line, offline synchronize and does not update the process points (when primary timeouts, backup starts updating process point values).
interval:	**<seconds>** - seconds to sleep between updates; if negative, then it will synchronize with moments when when the time shift is a multiple of the given value, e.g. **interval=-5.0** will start update cycles every 5 seconds.
proxy_mode:	**<0 | 1>** - **0** - normal mode, **1** - proxy mode.
proxy_def:	**<filename>** - file name with definition of points to manage
rhost:	**<name | ip>** - Server address
buffer:	**<on | off>** - saves scanned values to an external file; if enabled (set to **on**), the scanner archives the values of process points that have the **AR=** parameter set to **L** or **F**.If disabled (set to **off**), process points are not archived.
index:	**<NONE | T | S | TS>** - forces creation of indices on archive data tables; **T** - timestamps, **S** - SIDs, **TS** - timestamps and SID's.
auto_fillin:	**<on | off>** - automatically fills in missing data on the Server.
fillin_overlap:	**<number>** - the number of seconds extending periods without connectivity.
conn_timeout:	**<number>** - the number of seconds that must pass before noticing lack of connection.
watch:	**<on | off>** - deletes oldest archives when less than **min_space:** is available on the data file system.
min_space:	**<0.0 .. 95.0>** - minimum free space that must be left on the data file system (default **10**) expressed in percent.
max_archive_period:	**<n>** - keeps archives not older than **<n>** days; If set to **0**, checking the archives will not be performed.
max_archive_size:	**<n>** - keeps archives not bigger than **<n>** MB; If set to **0**, checking the archives will not be performed.
bname:	**<name>** - name of the file with buffered data.
ahost:	**<name | ip>** - name or IP address of **ARCH Server**.
aport:	**<number>** - UDP port of **ARCH Server**.
src_ahost:	**<name | ip>** - (wrapping source historian) address of **ARCH Server**.
src_aport:	**<number>** - (wrapping source historian) UDP port of **ARCH Server**.
alhost:	**<number>** - UDP port of **ARCH Server**.
alport:	**<number>** - local UDP port number for connections to **ARCH Server**.
alport_range:	**<number>** - local UDP/TCP port range for connections to **ARCH Server**.
user:	**<user>** - user name used for connections to **ARCH Server.**

PARAMETER	DESCRIPTION
passwd:	**<password>** - password used for connections to **ARCH Server**.
path:	**<path>** - path to serial device **/dev/cua/a**
baud:	**<baud>** - baud rate
parity:	**<0 | 1 | 2>** - parity bit: **0**-none, **1**-odd, **2**-even
data:	**<5 | 6 | 7 | 8>** - data bits
stop:	**<1 | 2>** - stop bits
flow:	**<flow control>** - type of flow control
tcp_lhost:	**<name | ip>** - optional, local name to use
tcp_lport:	**<number>** - optional, local port number
tcp_rhost:	**<name | ip>** - host to connect to
tcp_rport:	**<number>** - port to connect to
tcp_timeout:	**<seconds>** - timeout on communication
modbus_slave:	**<0 | 1>** - **0** - Master, **1** - Slave
modbus_port:	**<TCP | UDP | SERIAL>** - Port type: **TCP** or **UDP** or **SERIAL**
modbus_mode:	**<PREFIX | CRC | ASCII>** - **PREFIX** - 6 bytes prefix on frame begin; normal mode for TCP port, **CRC** - 2 bytes CRC on frame end; normal mode for serial port **ASCII** - modbus ascii mode
pre_tx_delay:	**<seconds>** - time in seconds to sleep before sending the Modbus frame
errors_count_to_ reconnect:	**<number>** - number of possible communication errors before reconnecting
operate:	**<on | off>** - accepts/sends operate commands; controls possibility to write process point values back; values can be set only for process points originating from the workstation on which this program runs.
softstart_period:	**<number>** - slows down the initialization process by a number of seconds. Multiple applications attempting to access Ovation at the same time may cause problems. Setting a value greater then **0** here can prevent that. Default value is **0**.
fast_mode:	**<on | off>** - sends data without waiting for ACK acknowlegement message.
org_cycle:	**<0-255>** -sets data encoding, values **0** an **1** prompt encoding of all data, setting values from **2** to **255** prompt encoding data for every **n**-th point, where **n** is the value set.
agnt_timeout:	**<seconds>** - agent timeout, number of seconds after which EDS Server will change the status of this scanner from on-line to timedout when it stops updating process point values.
shift:	**<seconds>** - time shift for the update, can be used to determine exact moments of the updates.
frames_sleep_time:	**<seconds>** - sleep time after frame is sent.
qual_map:	**<1 | 0>** - quality from values range
status_pname:	**<name>** - name of the point that indicates whether IESS is used.

PARAMETER	DESCRIPTION
timestamp_from_dev:	**<1 | 0>** - selecting **1** will set timestamp basing on the device time, selecting **0** will set timestamp basing on local time
timestamp_for_dummy:	**<1 | 0>** - selecting **1** will set timestamp for points not originating from the device, selecting **0** will not set a timestamp for those points.
reinit_cycle:	**<n>** - if the selected number is **0**, no reinitialisation will occur. If the selected number is greater than zero, reinitialisation will occur every **n**-th cycle.
lhost:	**<name | ip>** - local host name or IP number; the recommended setting is **0.0.0.0**, as this opens local socket on all available network interfaces; it might, however, be necessary to limit communication to a specified network interface, which can be achieved by specifying appropriate value of this parameter.
lport:	**<number>** - local UDP port number selecting **0** results in automatic assignment of available port number by the operating system, what is the recommended option; it may be necessary to define a port number if there is a firewall between this application and the destination, which has been configured to pass through only specified port numbers for security reasons
lport_range:	**<number>** - local UDP port range; allows to set the limit on used port numbers in order to narrow the port ranges opened In firewalls; specifying **lport=43000** and **lport_range=100** will make the application attempt to bind port **43000** - if it is already used and thus not available, it will proceed with **43001** and up to **43100**; first available port in this range will be used; if all ports are unavailable, the application will fail. If no firewalls in the network then this should be set to **0**.
comm_timeout:	**<seconds>** - communication timeout time in seconds when the application waits to receive acknowledge from the destination (by default it is set to **0.4** seconds).
resp_timeout:	**<seconds>** - response timeout time in seconds after which the application retransmits data packet. If it did not receive acknowledges from the destination. It may be increased if network delays are big (by default it is set to **2.0**).
max_packet:	**<bytes>** - a communication parameter (bytes). Specifies the maximum size of a single communication buffer (before compression). In some networks, may need to be decreased (for example to **16348**, **8192**, **4096** or **2048**).
max_bytes_per_second:	**<number>** - limit on the number of bytes sent (setting **0** disables the check).
rport:	**<number>** - server UDP port
compress:	**<on | off>** - controls compression of the transmitted data switching it on lowers network load but may cause raise in processor load.
protocol:	**<UDP | TCP>** - protocol to use.
fillin_retries:	**<spec>** - specifies how many times a fill-in operation for a given range should be performed.
fillin_delay:	**<number>** - number of seconds that must pass before starting an automatic fill-in.

4.2.5 Proxy feeder – ZD Proxy

In order to open the Gateways Proxy Configuration window find and open the **proxyconfig.exe** executable located by default in:

C:\Program Files\EDS92\GT

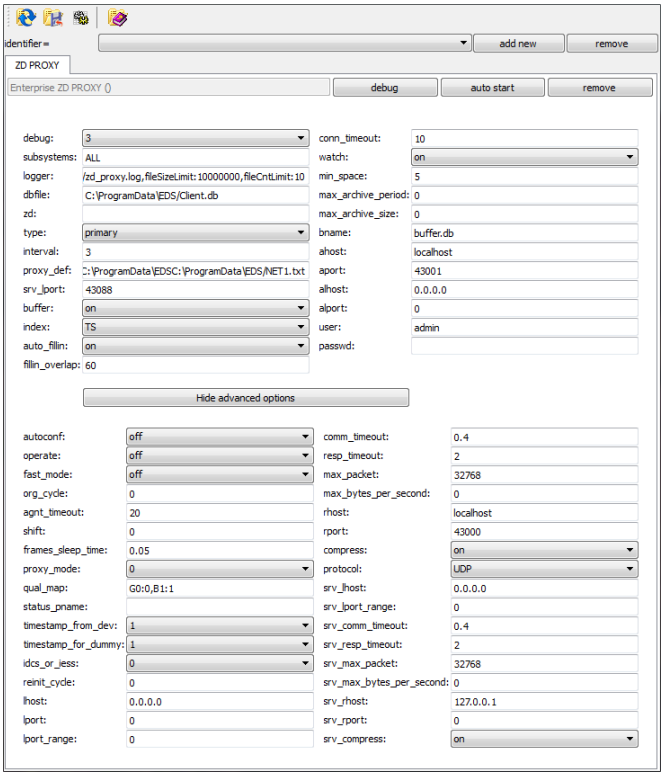


Figure 4.8 Gateways Configuration - ZD Proxy

PARAMETER	DESCRIPTION
debug:	Level of significance of messages to be logged; numbers have the following meaning: ■ 0 - Emergency, system is unusable ■ 1 - Alert, action must be taken immediately ■ 2 - Critical, critical conditions ■ 3 - Errors, error conditions ■ 4 - Warning, warning condition ■ 5 - Notice, normal but significant condition ■ 6 - Informational ■ 7 - Debugging EDS effectively uses levels from 3 to 7. When **debug:** is set to 5, then it results in a printout of messages of levels 5,4 and 3. The default level is 3 (log errors only).
subsystems:	List of names of the EDS subsystems that need to be logged. A special keyword **ALL** selects all subsystems, which is the default and recommended setting for normal operation.
logger:	Logger initialization string. For more information see *1.4.7 Logger Initialization String on page 34*.

PARAMETER	DESCRIPTION
dbfile:	**<name>** - name of configuration file to stores point database cache.
zd:	**<name1, name2, ...>**- source name or names; identifies which process points should be originated by this application; will be obsolete, use **org_zd=** instead
type:	**<primary | backup | offline>** - source or output type; primary updates the process points, backup checks if a primary in on-line, offline synchronize and does not update the process points (when primary timeouts, backup starts updating process point values).
interval:	**<seconds>** - seconds to sleep between updates; if negative, then it will synchronize with moments when when the time shift is a multiple of the given value, e.g. **interval=-5.0** will start update cycles every 5 seconds.
proxy_def:	**<filename>** - file name with definition of points to manage
srv_lport:	**<number>** - local UDP port number. Selecting **0** results in automatic assignment of available port number by the operating system, which is the recommended option; it may be necessary to define a port number if there is a firewall between this application and the destination.
buffer:	**<on | off>** - saves scanned values to an external file; if enabled (set to **on**), the scanner archives the values of process points that have the **AR=** parameter set to **L** or **F**.If disabled (set to **off**), process points are not archived.
index:	**<NONE | T | S | TS>** - forces creation of indices on archive data tables; **T** - timestamps, **S** - SIDs
auto_fillin:	**<on | off>** - automatically fills in missing data on the Server.
fillin_ overlap:	**<number>** - the number of seconds extending periods without connectivity.
conn_ timeout:	**<number>** - the number of seconds that must pass before noticing lack of connection.
watch:	**<on | off>** - deletes oldest archives when less than **min_space:** is available on the data file system.
min_space:	**<0.0 .. 95.0>** - minimum free space that must be left on the data file system (default **10**) expressed in percent.
max_ archive_ period:	**<n>** - keeps archives not older than **<n>** days; If set to **0**, checking the archives will not be performed.
max_ archive_ size:	**<n>** - keeps archives not bigger than **<n>** MB; If set to **0**, checking the archives will not be performed.
bname:	**<name>** - name of the file with buffered data.
ahost:	**<name | ip>** - name or IP address of **ARCH Server**.
aport:	**<number>** - UDP port of **ARCH Server**.
src_ahost:	**<name | ip>** - (wrapping source historian) address of **ARCH Server**.
src_aport	**<number>** - (wrapping source historian) UDP port of **ARCH Server**.
alhost:	**<number>** - UDP port of **ARCH Server**.
alport:	**<number>** - local UDP port number for connections to **ARCH Server**.

PARAMETER	DESCRIPTION
alport_ range:	**<number>** - local UDP/TCP port range for connections to **ARCH Server**.
user:	**<user>** - user name used for connections to **ARCH Server.**
passwd:	**<password>** - password used for connections to **ARCH Server**.
operate:	**<on | off>** - accepts/sends operate commands; controls possibility to write process point values back; values can be set only for process points originating from the workstation on which this program runs.
softstart_ period:	**<number>** - slows down the initialization process by a number of seconds. Multiple applications attempting to access Ovation at the same time may cause problems. Setting a value greater then **0** here can prevent that. Default value is **0**.
fast_mode:	**<on | off>** - sends data without waiting for ACK acknowlegement message.
org_cycle:	**<0-255>** -sets data encoding, values **0** an **1** prompt encoding of all data, setting values from **2** to **255** prompt encoding data for every **n**-th point, where **n** is the value set.
agnt_ timeout:	**<seconds>** - agent timeout, number of seconds after which EDS Server will change the status of this scanner from on-line to timedout when it stops updating process point values.
shift:	**<seconds>** - time shift for the update, can be used to determine exact moments of the updates.
frames_ sleep_time:	**<seconds>** - sleep time after frame is sent.
last_values_ filename:	**<filename>** - name of the file to store last valid values
proxy_ mode:	**<0 | 1>** - **0** - normal mode, **1** - proxy mode, only send values, no revise frames, ZP exports file, ZD use this file
qual_map:	**<0 | 1>** - quality from values range
status_ pname:	**<name>** - name of IESS indicates status point
timestamp_ from_dev:	**<1 | 0>** - selecting **1** will set timestamp basing on the device time, selecting **0** will set timestamp basing on local time
timestamp_ for_dummy:	**<1 | 0>** - selecting **1** will set timestamp for points not originating from the device, selecting **0** will not set a timestamp for those points.
reinit_cycle:	**<n>** - if the selected number is **0**, no reinitialisation will occur. If the selected number is greater than zero, reinitialisation will occur every **n**-th cycle.
lhost:	**<name | ip>** - local host name or IP number; the recommended setting is **0.0.0.0**, as this opens local socket on all available network interfaces; it might, however, be necessary to limit communication to a specified network interface, what can be achieved by specifying appropriate value of this parameter.
lport:	**<number>** - local UDP port number. Selecting **0** results in automatic assignment of available port number by the operating system, which is the recommended option; it may be necessary to define a port number if there is a firewall between this application and the destination.

PARAMETER	DESCRIPTION
lport_range:	**<number>** - local UDP port range; allows to set the limit on used port numbers in order to narrow the port ranges opened In firewalls; specifying **lport=43000** and **lport_range=100** will make the application attempt to bind port **43000** - if it is already used and thus not available, it will proceed with **43001** and up to **43100**; first available port in this range will be used; if all ports are unavailable, the application will fail. If no firewalls in the network then this should be set to **0**.
comm_timeout:	**<seconds>** - communication timeout time in seconds when the application waits to receive acknowledge from the destination (by default it is set to **0.4** seconds).
resp_timeout:	**<seconds>** - response timeout time in seconds after which the application retransmits data packet. If it did not receive acknowledges from the destination. It may be increased if network delays are big (by default it is set to **2.0**).
max_packet:	**<bytes>** - a communication parameter (bytes). Specifies the maximum size of a single communication buffer (before compression). In some networks, may need to be decreased (for example to **16348**, **8192**, **4096** or **2048**).
max_bytes_per_second:	**<number>** - limit on the number of bytes sent (setting **0** disables the check).
rhost:	**<name | ip>** - server address
rport:	**<number>** - server UDP port
compress:	**<on | off>** - controls compression of the transmitted data switching it on lowers network load but may cause raise in processor load.
protocol:	**<UDP | TCP>** - protocol to use
srv_lhost:	**<name | ip>** - local host name or IP number; the recommended setting is **0.0.0.0**, as this opens local socket on all available network interfaces; it might, however, be necessary to limit communication to a specified network interface, what can be achieved by specifying appropriate value of this parameter.
srv_lport_range:	**<number>** - local UDP port number. Selecting **0** results in automatic assignment of available port number by the operating system, which is the recommended option; it may be necessary to define a port number if there is a firewall between this application and the destination.
srv_comm_timeout:	**<number>** - local UDP port range; allows to set the limit on used port numbers in order to narrow the port ranges opened In firewalls; specifying **lport=43000** and **lport_range=100** will make the application attempt to bind port **43000** - if it is already used and thus not available, it will proceed with **43001** and up to **43100**; first available port in this range will be used; if all ports are unavailable, the application will fail. If no firewalls in the network then this should be set to **0**.
srv_resp_timeout:	**<seconds>** - communication timeout time in seconds when the application waits to receive acknowledge from the destination (by default it is set to **0.4** seconds).
srv_max_packet:	**<seconds>** - response timeout time in seconds after which the application retransmits data packet. If it did not receive acknowledges from the destination. It may be increased if network delays are big (by default it is set to **2.0**).
srv_max_bytes_per_second:	**<number>** - limit on the number of bytes sent (setting **0** disables the check).
srv_rhost:	**<name | ip>** - server address
srv_rport:	**<number>** - server UDP port

PARAMETER	DESCRIPTION
srv_ compress:	**<on | off>** - controls compression of the transmitted data switching it on lowers network load but may cause raise in processor load.
fillin_retries:	**<spec>** - specifies how many times a fillin operation for a given range should be performed
fillin_delay:	**<number>** - number of seconds that must pass before starting an automatic fill-in

4.2.6 Replay feeder – ZD Replay

In order to open the Gateways Replay Configuration window find and open the **replayconfig.exe** executable located by default in:

C:\Program Files\EDS92\GT

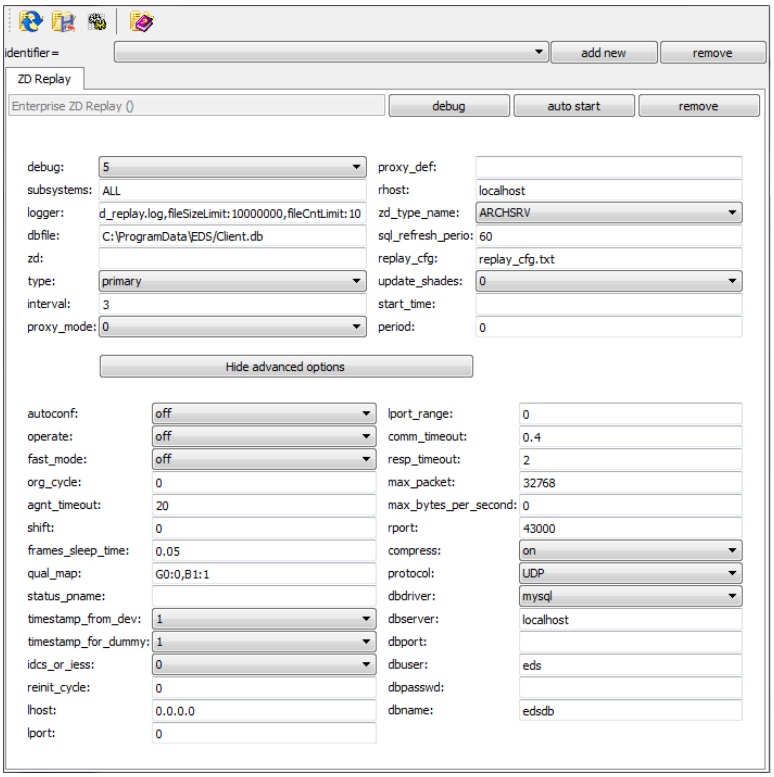


Figure 4.9 EDS Gateways Package - Replay feeder ZD_Replay

PARAMETER	DESCRIPTION
debug:	Level of significance of messages to be logged; numbers have the following meaning: ■ `0` - Emergency, system is unusable ■ `1` - Alert, action must be taken immediately ■ `2` - Critical, critical conditions ■ `3` - Errors, error conditions ■ `4` - Warning, warning condition ■ `5` - Notice, normal but significant condition ■ `6` - Informational ■ `7` - Debugging EDS effectively uses levels from `3` to `7`. When **debug:** is set to `5`, then it results in a printout of messages of levels `5`,`4` and `3`. The default level is `3` (log errors only).
subsystems:	List of names of the EDS subsystems that need to be logged. A special keyword `ALL` selects all subsystems, which is the default and recommended setting for normal operation.
logger:	Logger initialization string. For more information see *1.4.7 Logger Initialization String on page 34*.
dbfile:	`<name>` - name of configuration file to stores point database cache.
zd:	`<name1, name2, ...>`- source name or names; identifies which process points should be originated by this application; will be obsolete, use `org_zd=` instead
type:	`<primary | backup | offline>` - source or output type; primary updates the process points, backup checks if a primary in on-line, offline synchronize and does not update the process points (when primary timeouts, backup starts updating process point values).
interval:	`<seconds>` - seconds to sleep between updates; if negative, then it will synchronize with moments when when the time shift is a multiple of the given value, e.g. **interval=**`-5.0` will start update cycles every 5 seconds.
proxy_ mode:	`<0 | 1>` - `0` - normal mode, `1` - proxy mode.
proxy_def:	`<filename>` - file name with definition of points to manage
rhost:	`<name | ip>` - name or IP address of the server
zd_type_ name:	`< ORACLE | MYSQL | ARCHSRV >` - name of source data type
sql_refresh_ perio:	`<seconds>` - period between queries expressed in seconds
replay_cfg:	`<filename>` - configuration file name
update_ shades:	`<seconds>` - period between shade updates expressed in seconds.
start_time:	`<YYYYMMDDHHMISS>` - work start time.
period:	`<seconds>` - periodical cycle time expressed in seconds.
operate:	`<on | off>` - accepts/sends operate commands; controls possibility to write process point values back; values can be set only for process points originating from the workstation on which this program runs.
fast_mode:	`<on | off>` - sends data without waiting for ACK acknowlegement message.

PARAMETER	DESCRIPTION
org_cycle:	**<0-255>** -sets data encoding, values **0** an **1** prompt encoding of all data, setting values from **2** to **255** prompt encoding data for every **n**-th point, where **n** is the value set.
agnt_ timeout:	**<seconds>** - agent timeout, number of seconds after which EDS Server will change the status of this scanner from on-line to timedout when it stops updating process point values.
shift:	**<seconds>** - time shift for the update, can be used to determine exact moments of the updates.
frames_ sleep_time:	**<seconds>** - sleep time after frame is sent.
qual_map:	**<0 | 1>** - quality from values range
status_ pname:	**<name>** - name of the point that indicates whether IESS is used.
timestamp_ from_dev:	**<1 | 0>** - selecting **1** will set timestamp basing on the device time, selecting **0** will set timestamp basing on local time
timestamp_ for_dummy:	**<1 | 0>** - selecting **1** will set timestamp for points not originating from the device, selecting **0** will not set a timestamp for those points.
reinit_cycle:	**<n>** - if the selected number is **0**, no reinitialisation will occur. If the selected number is greater than zero, reinitialisation will occur every **n**-th cycle.
lhost:	**<name | ip>** - local host name or IP number; the recommended setting is **0.0.0.0**, as this opens local socket on all available network interfaces; it might, however, be necessary to limit communication to a specified network interface, which can be achieved by specifying appropriate value of this parameter.
lport:	**<number>** - local UDP port number. Selecting **0** results in automatic assignment of available port number by the operating system, which is the recommended option; it may be necessary to define a port number if there is a firewall between this application and the destination.
lport_range:	**<number>** - local UDP port range; allows to set the limit on used port numbers in order to narrow the port ranges opened In firewalls; specifying **lport=43000** and **lport_ range=100** will make the application attempt to bind port **43000** - if it is already used and thus not available, it will proceed with **43001** and up to **43100**; first available port in this range will be used; if all ports are unavailable, the application will fail. If no firewalls in the network then this should be set to **0**.
comm_ timeout:	**<seconds>** - communication timeout time in seconds when the application waits to receive acknowledge from the destination (by default it is set to **0.4** seconds).
resp_ timeout:	**<seconds>** - response timeout time in seconds after which the application retransmits data packet. If it did not receive acknowledges from the destination. It may be increased if network delays are big (by default it is set to **2.0**).
max_packet:	**<bytes>** - a communication parameter (bytes). Specifies the maximum size of a single communication buffer (before compression). In some networks, may need to be decreased (for example to **16348**, **8192**, **4096** or **2048**).
max_bytes_ per_second:	**<number>** - limit on the number of bytes sent (setting **0** disables the check).
rport:	**<number>** - server UDP port
compress:	**<on | off>** - controls compression of the transmitted data switching it on lowers network load but may cause raise in processor load.

PARAMETER	DESCRIPTION
protocol:	**<UDP | TCP>** - protocol to use
dbdriver:	**<mysql | postgresql | sqlite | sqlserver>** - SQL database driver
dbserver:	**<host>** - host where SQL engine runs
dbport:	**<port>** - connection port to SQL database
dbuser:	**<user>** - name of user
dbpasswd:	**<password | @filename>** - user's password or name of file containing the password
dbname:	**<name>** - name of database

4.3 Configuration example

Below a simple example of configuring EDS **Gateways** to transfer data between two machines is presented. The configuration parameters are left as default unless otherwise specified.

In this example the following parameters were used:

1. The configuration profile name was specified as **1**.
2. Parameters **d_zd** and **s_zd** were both specified as **testzd**. This is an arbitrary name and can be anything. Points are identified basing on this setting and thus it is advisable that the **zd** parameter is the same on both machines.
3. The **d_rhos**t parameter was specified as **s0129**. This is the example name of the machine to which data is sent within the network. Specify the target machine here.
4. The **s_rhost** parameter was specified as **s0035**. This is the example name of the machine from which data is taken within the network. Specify the source machine here.
5. The **debug** parameter was set to **7**. This allows for better monitoring of the program's working in logs.

This is a minimum, simple configuration, but it is functional and sufficient for sending points from one machine to another.

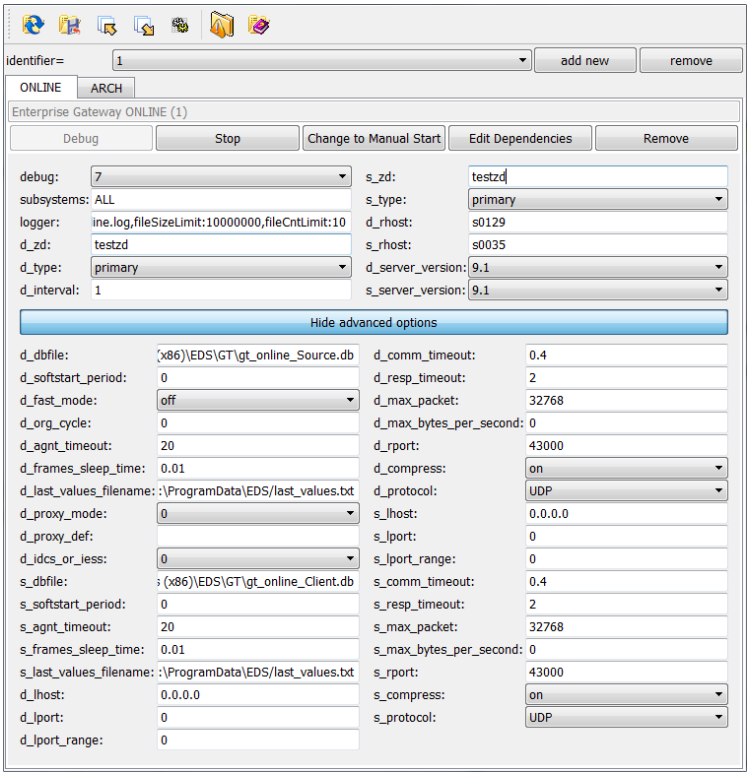


Figure 4.10 Gateways configuration example

Appendix A

IN THIS SECTION

A.1 WDPF 8 level record fields

This section describes the assignment of **WDPF 8** level point record fields to EDS point record fields. All references to **WDPF** record field names or fields in **C API** structures are underlined. If a **WDPF** record type does not contain the specified record field, then the corresponding field in EDS is left unchanged.

Assignment for all record types, empty fields signify that an EDS record field does not have a relevant counterpart in **WDPF**:

EDS RECORD FIELD NAME	WDPF RECORD FIELD NAME	REMARKS
IESS	**<prefix>PN<suffix>**	
ZD		configured in EDS
IDCS	**PN**	
DESC	**ED**	
AUX	**DROPSR aux_desc**	
AR		configured in EDS, optionally can be set basing on hsrpicfile
AP		configured in EDS
TG		configured in EDS, in WDPF8.x could be set basing on **alrm_char**
SG		configured in EDS
TS		time shown by EDS Server clock when reading sample on **WEStation** device.
TSS		time difference between EDS Server and **WEStation** device.

If a hsrpicfile source file is used to set the **AR**, **ARD** and **ARTD** parameters, then the following rules apply:

- each line starting with `;` is ignored
- by default **ARTD** is set to `P` and **ARD** to `1.0`
- a line starting with `/DEF` changes the default parameters as follows depending on the parameter set:

value after `DEADBAND=` is copied to **ARD**

input `DB_ALG=PCT_RANGE` sets `ARTD=P`

input `DB_ALG=STANDARD` sets `ARTD=V`

input `DB_ALG=RATIO` sets `ARTD=R`

Record types **AI**, **AL**, **AB**, **AC**, **AM** and **AX**

WDPF record types **AI**, **AL**, **AB**, **AC**, **AM** and **AX** are represented in EDS as **analog** points. Empty fields signify that an EDS record field does not have a relevant counterpart in **WDPF**:

EDS RECORD FIELD NAME	WDPF RECORD FIELD NAME	REMARKS
UN	**EU**	
DP	**FM**	
ARTD		configured in EDS, optionally can be set basing on parameter **DB_ALG=** `in hsrpicfile` basing on a source file.
ARD		configured in EDS, optionally can be read from the parameter **DEADBAND=** `in hsrpicfile`basing on a source file.
TB	**TB**	
BB	**TB**	
HL	**HL**	
LL	**LL**	
AV	**AV**	
ST		derived from AS
XST1	**AS**	
XST2	**LC, AW**	most sig. `16 bits = LC`, `least = AW`
AT	**MM, NN, AH, MN, LU**	converted to number of seconds since 1.1.1970 0:00:00

Record types **MV**, **MA**, **MP**

WDPF record types **MV**, **MA** and **MP** are represented in EDS as **analog** points. Empty fields signify that an EDS record field does not have a relevant counterpart in **WDPF**:

EDS RECORD FIELD NAME	WDPF RECORD FIELD NAME	REMARKS
ARTD		configured in EDS, optionally can be set basing on parameter **DB_ALG=** `in hsrpicfile` basing on a source file.
ARD		configured in EDS, optionally can be read from the parameter **DEADBAND=** `in hsrpicfile` basing on a source file.
AV	**TP**	
ST		Succesfull reading of **ST** signifies good quality.

Record types: **DI**, **DL**, **DC**, **DM** and **DX**

WDPF record types: **DI**, **DL**, **DC**, **DM** and **DX** are represented in EDS as **binary** points. Empty fields signify that an EDS record field does not have a relevant counterpart in **WDPF**:

EDS RECORD FIELD NAME	WDPF RECORD FIELD NAME	REMARKS
SD	**ST**	
RD	**RS**	
BV		equals bit `0` of **DS**
ST		derived from **DS**
XST1	**DS**	
XST2	**LC, DW**	most sig. `16 bits = LC, least = DW`
AT	**MM, NN, AH, MN, LU**	converted to number of seconds since 1.1.1970 0:00:00

Record types: **PB** and **PX**

WDPF record types **PX**, **PB** are represented in EDS as **packed** points. Empty fields signify that an EDS record field does not have a relevant counterpart in **WDPF**:

EDS RECORD FIELD NAME	WDPF RECORD FIELD NAME	REMARKS
PV	**AV**	
ST		derived from **AS**
XST1	**AS**	

Record types: **GP**

WDPF record type **GP** is represented in EDS as a **packed** point. Empty fields signify that an EDS record field does not have a relevant counterpart in **WDPF**:

EDS RECORD FIELD NAME	WDPF RECORD FIELD NAME	REMARKS
PV	**A3, A2**	most sig. `16 bits = A3, least = A2`
ST		derived from **DS**
XST1	**DS**	

Record types: **BG**, **BN** and **BX**

WDPF record types **BG**, **BN** and **BX** are represented in EDS as **packed** points. Empty fields signify that an EDS record field does not have a relevant counterpart in **WDPF**:

EDS RECORD FIELD NAME	WDPF RECORD FIELD NAME	REMARKS
PV	**A3, A2**	most sig. **16 bits = A3, least = A2**
ST		derived from **DS**
XST1	**DS**	

Record types: **VC**

WDPF record type **VC** is represented in EDS as a **packed** point. Empty fields signify that an EDS record field does not have a relevant counterpart in **WDPF**:

EDS RECORD FIELD NAME	WDPF RECORD FIELD NAME	REMARKS
PV	**A3, A2**	most sig. **16 bits = A3, least = A2**
ST		derived from **DS**
XST1	**DS**	

Record types: **DU**

WDPF record type **DU** is represented in **EDS** as a **packed** point. Empty fields signify that an EDS record field does not have a relevant counterpart in **WDPF**:

EDS RECORD FIELD NAME	WDPF RECORD FIELD NAME	REMARKS
AUX	**DROPSRaux_desc, TY**	
PV	**FA, FB, FC**	bits **31-16 FA**, bits **15-8 FB**, bits **7-0 FC**
ST		on successful read quality is Good, otherwise Bad
XST1	**FK, HC**	bits **31-16 FK**, bits **15-0 HC**
XST2	**FS, FO**	bits **31-16 FS**, bits **15-0 FO**

A.2 Ovation record fields

This section describes assignment of Ovation point record fields to EDS point record fields. All references to Ovation record field names or fields in C API structures are underlined. If an Ovation record type does not contain the specified record field, then corresponding field in EDS is left unchanged.

Assignment for all record types. Empty fields signify that an EDS record field does not have a relevant counterpart in Ovation:

EDS Record Field Name	OVATION Record Field Name	Remarks
IESS	**<prefix>PN<suffix>**	Default is a fully qualified point name
ZD		Configured in EDS
IDCS	**PN**	
DESC	**ED**	
AUX	**DROPDO**	
AR		Configured in EDS, optionally
AP		Can be set basing on hsrpicfile configured in EDS
TG		Configured in EDS
SG		Configured in EDS
TS		Time shown by EDS Server clock when reading sample on WEStation
TSS		Time difference between EDS Server and WEStation

It is possible to use hsrpicfile to set the **AR**, **ARD** and **ARTD** parameters - for details please refer to section *A.1 WDPF 8 level record fields on page 132*.

Record types: **LA**, **DA**, **SA** and **TA**

Ovation record types: **LA**, **DA**, **SA** and **TA** are represented in EDS as **analog** points. Empty fields signify that an EDS record field does not have a relevant counterpart in Ovation:

EDS Record Field Name	OVATION Record Field Name	Remarks
UN	**EU**	
DP	**FM**	
ARTD		configured in EDS, optionally can be set basing on parameter **DB_ALG=** `in hsrpicfile` basing on a source file.

EDS Record Field Name	OVATION Record Field Name	Remarks
ARD		configured in EDS, optionally can be read from the parameter **DEADBAND=** `in hsrpicfile` basing on a source file.
TB	**TV**	
BB	**TV**	
HL	**HL**	
LL	**LL**	
AV	**AV**	
ST		derived from **1W** (see below)
XST1	**1W**	
AT	**U6**	

Record types: **LD**, **DD**, **SD**, and **TD**

Ovation record types: **LD**, **DD**, **SD**, and **TD** are represented in EDS as **binary** points. Empty fields signify that an EDS record field does not have a relevant counterpart in Ovation:

EDS Record Field Name	OVATION Record Field Name	REMARKS
SD	**ST**	
RD	**RS**	
BV		equals bit 0 of **1W**
ST		derived from **1W**
XST1	**1W**	
AT	**U6**	

Record types: **LP**, **DP**, **SP**, **TP**, **RM** and **RN**

Ovation record types **LP**, **DP**, **SP**, **TP**, **RM** and **RN** are represented in EDS as **packed** points. Empty fields signify that an EDS record field does not have a relevant counterpart in Ovation:

EDS Record Field Name	OVATION Record Field Name	Remarks
PV	**A2**	
ST		derived from **1W**
XST1	**1W**	

Record types: **LC**

Ovation record type **LC** is represented in EDS as **packed** point. Empty fields signify that an EDS record field does not have a relevant counterpart in Ovation:

EDS Record Field Name	OVATION Record Field Name	Remarks
PV	**G1**	
ST		set by EDS

Record types: **DU**

Ovation record type **DU** is represented in EDS as **packed** point. Empty fields signify that an EDS record field does not have a relevant counterpart in Ovation:

EDS Record Field Name	OVATION Record Field Name	Remarks
AUX	**DROPDO TY=TY**	
PV	**FA, FB, FC**	**FA** corresponds to bits 31-16, **FB** to bits 15-8 and **FC** to bits 7-0.
ST		Successful reading of **ST** signifies good quality.
XST1	**FK, HC**	**FK** corresponds to bits 31-16 and **HC** to bits 15-0.
XST2	**FS, FO**	**FS** corresponds to bits 31-16 and **FO** to bits 15-0.

A.3 Mapping of Alarm Priorities

EDS alarm priority levels are similar to the ones in Ovation. Because WDPF 8 systems have different priority numbering, it was necessary to map them into EDS priority levels in order to secure consistency.

WDPF 8	OVATION	EDS	COLORS (DEFAULT)
3	1	1	Highest
	2	2	
	3	3	
2	4	4	
	5	5	
1	6	6	
	7	7	
0	8	8	Lowest

A.4 One-way communication

Note: *For a more detailed description please see the EDS Ovation One Way Communication Configuration manual*

One way communication setup provides ability to acquire process data from isolated subnets, separated from the EDS Server by network appliances preventing any transmission back to the DCS systems. The transmission can be blocked in two ways:

- by appropriate configuration of network devices like switches, routers, etc.
- by the devices which physically eliminate transmission in one direction, for example Net Optics Port Aggregator Tap

EDS software supports both solutions.

A.4.1 Import of points

If EDS Database Access is on the server side of the network (i.e. the isolating device is between EDS Database Access and the import agent), then one-way communication blocks its requests sent to the import agent (**ZP**). If EDS Database Access runs on the DCS side of network, then it is able to import points from the import agent, but is not able to communicate with the EDS Server. In both cases the standard EDS communication mechanisms do not work. The simplest solution to this problem is to run the import agent with the option **export_file=**`<filename>`, where `filename` is the name of the file, then copy (e.g. with an aid of pendrive) the file to the EDS Server and import the points from this file using EDS Database Access. To eliminate the need of manual transfer of the file, there are two applications designed to copy it using one-way communication mode:

- **send_files_udp** - sends specified file(s) over UDP
- **receive_files_udp** - receives file(s) sent by send_files_udp

The application **send_files_udp** can be run by cron periodically, what will transfer the file with the list of available process points to the EDS Server. Once there, you can use EDS Database Access to import/synchronize the point in the EDS database.

A.4.2 Online data

In order to enable one-way communication mode, you need to do the following:

- Set the parameter **proxy_mode=** to `using_idcs` in ZD configuration on the Ovation side.
- Set the parameter **srv_host** to the server address
- Set the parameter **srv_port** to the server port (by default is set to `43000`)
- Set the parameter **proxy_def=** to the text file containing all points imported from the Ovation network. The file name must be specified with full path. Since the feeder doesn't receive any information from the EDS Server, it can't get definition points from EDS side. The file specified by **proxy_def=** supplies this data. You can create such file by running the ZP feeder with the parameter **export_file=**`<filename>` where `filename` is the desired name of the file.
- Run zd_proxy on the EDS Server machine, which performs the following:
  - receives data with point values from Ovation
  - substitutes point system IDs so that they are conformant with the EDS Server database; points are matched by IESS fields basing on the information supplied by `proxy_def=` (the file specified here must be the same as the one specified for ZD).
  - sends them to the EDS Server

- Configure the transfer of the file specified in **proxy_def=** from the Ovation machine to the EDS Server. This is described below.

A.4.3 Transfer of files in one way communication

In order to send files in one-way communication mode, you need to configure two programs:

- **send_file_udp** - on the Ovation network side (the machine running the **ZD** feeder)
  - **rhost=** - EDS server host
  - **rport=** - listen port in receive_file_udp, defaults to 43187
  - **proxy_script=** - script run before sending
  - **proxy_start=** - starting date for sending data
  - **proxy_period=** - period of sending data beginning from proxy_start
  - **proxy_udp_period=** - interval between consecutive UDP messages
- **receive_file_udp** - on the EDS Server machine
  - **lport=** - listen port, defaults to 43187
  - **proxy_script=** - script run after receiving

This mechanism should typically be used to transfer:

- the point definition file specified in **proxy_def=** in **ZD** feeder
- process diagram sources

A.5 Ports in EDS

This section discuss the following topics:

- Communication (see *A.5.1 Communication on page 140*).
- Remote port (see *A.5.3 Remote port on page 141*).
- Local port (see *A.5.2 Local port on page 140*

A.5.1 Communication

In EDS we can view the communication between **ZD** and **SRV**:

```
ZD.lport <---> SRV.lport
```

A.5.2 Local port

SRV.lport is by default set to `43000` and has to be known to all applications which communicate with the **SRV**.

If there are no network constraints (like a firewall), **ZD.lport** can be any port number. By default it is `0` what is a special value telling the operating system to assign any available UDP port.

The communication between **ZD** and **SRV** works, because it is the **ZD** which sends the first message to **SRV** and **SRV** answers to the same **IP:PORT** which originated the message. **ZD.lport** can be set to some explicit value, however we need to remember that if such a particular port is not available, attempt to bind it will fail and whole application will not start. Therefore there is another parameter called **ZD.lport_range** described in the next section.

ZD.lport_range sets the number of consecutive port numbers to try in case bind fails. It begins with (**ZD.lport**) and stops when it comes to (**ZD.lport+ZD.lport_range**).

A.5.3 Remote port

Each of processes has its own local UDP socket (lport). The partner **UDP** port is called remote port (rport) and it differs from the point of view:

- From the point of view of **ZD**: `rport = Srv.lport`
- From the point of view of **SRV**: `rport = ZD.lport`