# ESL Diagnostic Pack

**ESL Diagnostic Pack** provides a view on the running ESL and JAVA scripts in an application. It is accessible from D2000 CNF Objects -> **ESL Diagnostic Pack** or D2000 GrEditor, **Configuration D2000** -> **ESL Diagnostic Pack**.

Each script is interpreted within a process, which is DODM model. Depending on a configuration, these processes can be identical to the processes that run in an operating system (HIP, EVH, SES, HIS) or they are just DODM processes (SEE).

The physical processes are monitored on **Physical process** tab.



Description of columns:

| | |
|---|---|
| ID (HOBJ) | Unique identifier of object of *Process* type in DODM model. |
| NAME | Unique name of object of *Process* type in DODM model. |
| IP | IP of computer where the process is running. |
| AllocatedMermory | Memory allocated to the given process. |

When right clicking on any view, the popup menu appears with one option **Copy**. It enables to copy a displayed list into clipboard.



*JVM Instances* tab provides view on JVM instances:

JVM parameters:

| | |
|---|---|
| Physical process | Name of physical process in which JVM is created. |
| Allocated memory | Amount of allocated memory in the Java Virtual Machine (total - free). |
| Total memory | Total amount of memory in the JVM. |
| Free memory | Amount of free memory in the JVM. |
| Max memory | Maximum amount of memory that Java Virtual Machine will attempt to use. |

*DODM Process* tab provides view on DODM processes.



Description of columns:

| | |
|---|---|
| ID (HOBJ) | Unique identifier of object of *Process* type in DODM model. |
| NAME | Unique name of object of *Process* type in DODM model. |
| Physical process | Name of physical process, in which DODM process is executed. |
| D2000 HI | Name of D2000 HI, if the process is used for interpretation of ESL scripts opened in the given HIP process. |

If *Physical process* is of SES type, the value in *NAME* column is different than in *Physical process* column.

*Event Definitions* tab provides view on the definition of objects of *Event* or *Picture* type, as they are placed within the logical DODM processes.

The process of EVH type contains the definition of all its children (objects of *Event* type), or definition of objects of *Event* type, which were opened by OPENEVENT in the given process.

The process of HIS type contains the definition of currently opened pictures (client part of script) and definition of objects of *Event* type (client part of script), which were opened by OPENEVENT in the given process.

The process of SEE type contains the definition of server script of currently opened pictures and the objects of Event type on a dedicated HIS process.

Description of columns:

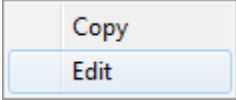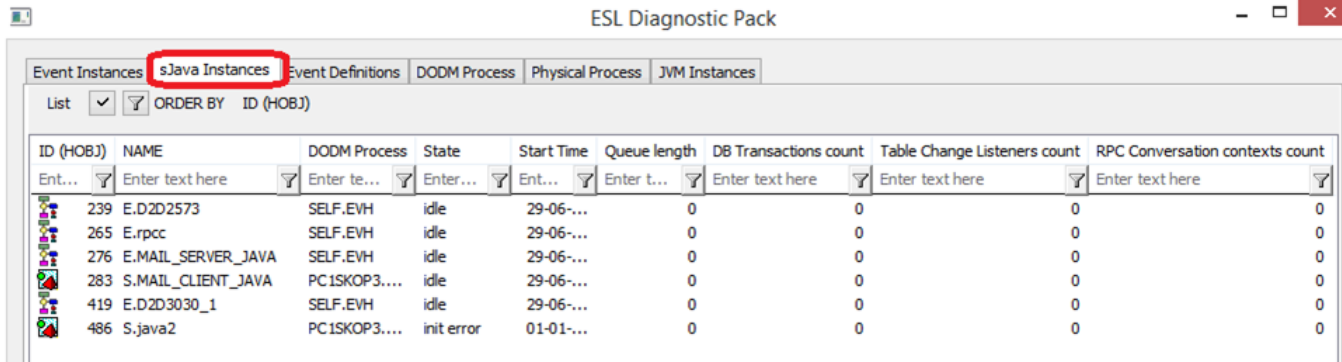| | |
|---|---|
| ID (HOBJ) | If an object has been opened multiply (OPEN action or OPENEVENT action), the column contains its dynamic HOBJ. This is a return value of %GetSelfHBJ (@FALSE). |
| NAME | Name of object containing ESL script. |
| Physical process DODM process | Name of objects of Process type (with suffix EVH, HIS, SEE, SES), which contains the given definition. |
| D2000 HI | Name of D2000 HI if DODM process is used for interpretation of ESL scripts opened in the appropriate HIP process. |
| Kernel instance ID | True number of instance. Value is 0 (zero), if the object is not opened multiply. If it has been opened as instance, in this column there is listed its real instance. This is the return value of function %GetSelfInstanceId(@TRUE). |
| Logical instance ID | Required number of instance. The value is 0 (zero), if the object is not opened multiply. If it has been opened as instance, in this column there is listed its required instance. This is the return value of function %GetSelfInstanceId(@FALSE). |
| BASE ID (HOBJ) | HOBJ of object that contains the given ESL script. This is a return value of %GetSelfHBJ(@TRUE) function. |
| EventType | Configuration parameter. Possible values TRIGGEREVENT, SERVEREVENT, CLIENTANDSERVEREVENT, UNITEVENT. |
| Queue Length | For object of SERVEREVENT or CLIENTANDSERVEREVENT type (this includes also a picture), the value means the number of requests (e.g. calls of RPC procedures, changes watched by ON CHANGE, ...) in a queue. For the event of TRIGGEREVENT type and the permission of request queue for start, the value means the number of requests for start. |
| Openned by OPEN EVENT | A flag that indicates whether the instance was opened by OPENEVENT action. If yes, it may be closed or restarted after pressing right mouse button and selecting an appropriate option. |
| lastInstanceId | The existence of definition of Event object on EVH process not necessarily mean the existence of its instance. This condition can occur with the Event object that is started by changing the value of object (trigger). In other cases, usually, just one instance of ESL script exists toward one definition. Each running instance in the process is defined by the unique identifier, which is displayed in this column. Its value can be retrieved by calling %GetEventInstanceId(). |
| Number of Instances | Number of instances of object of Event type that were started from given definition. Usually, it is 0 or 1. Higher value may be achieved when using ENABLE script action. |

After pressing right-mouse button, the menu with two options appears. The **Edit** option is available for object of *Picture* type only in D2000 GrEditor.

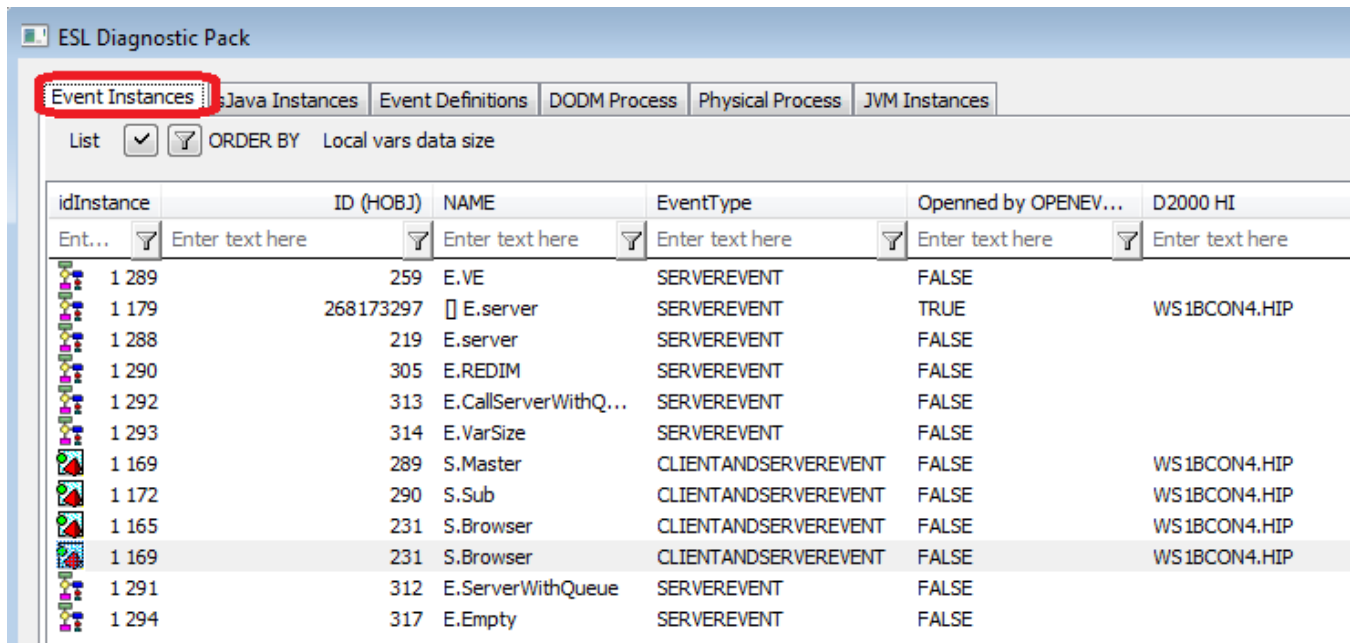*sJava Instances* tab contains the list of all sJAVA instances.



The following data of each instance are displayed:

| ID (HOBJ) | HOBJ of object which the given instance belongs to. |
|---|---|
| NAME | Name of object which the given instance belongs to. |
| DODM process | Name of object of *Process* type (with suffix EVH, HIS, SEE), which contains the given instance. |
| State | **Init_Error** – sJava instance failed to start due to a compiler error or an error generated by JVM when creating an instance class (e.g. failed to find the dependent classes - classNotFound java exception ...).<br>**IDLE** – the request queue waiting for processing is empty and no request is processed (changing the value of object, request to execute RPC, ...).<br>**Running** – the request is processed.<br>**SyncWaiting** – passive waiting on finishing the synchronous action (e.g. reading data from archive, synchronous calling the RPC, ...) during processing the request. |
| Start Time | Creation time of instance(start of script). |
| QueueLength | Unprocessed requests in the queue. |
| DB Transactions count | Opened database transaction count (created by calling the dbTransOpen method). |
| Table Change Listeners count | Total count of listeners registered by calling the addDBTableListener method. |
| RPC Conversation contexts count | The registered RPC conversations defined in the application. |

*Event Instances* tab provides the most interesting view on the system in term of diagnostic.



This tab displays all running instances of ESL scripts in the system. Each instance contains these attributes:

| idInstance | Unique identifier of instance. It may be get by calling the %GetEventInstanceId(). |
|---|---|

| ID (HOBJ) | If the object was opened multiply (OPEN script action or OPENEVENT), the column contains its dynamic HOBJ. This is a return value of %GetSelfHBJ (@FALSE). |
|---|---|
| NAME | Name of object. |
| EventType | Configuration parameter. Possible values TRIGGEREVENT, SERVEREVENT, CLIENTANDSERVEREVENT, UNITEVENT. |
| Kernel instance ID | True number of instance. Value is 0 (zero), if the object is not opened multiply. If it has been opened as instance, in this column there is listed its real instance. This is the return value of function %GetSelfInstanceId(@TRUE). |
| Logical instance ID | Required number of instance. The value is 0 (zero), if the object is not opened multiply. If it has been opened as instance, in this column there is listed its required instance. This is the return value of function %GetSelfInstanceId(@FALSE). |
| Modify time stamp | Time stamp from the configuration of object. |
| Start Time | Creation time of instance (start of script). |
| Current Line | Currently executed line. |
| CNT count | (+) Number of created data containers. |
| CNT dataSize | (+) Memory size which is occupied by data from containers. |
| Local vars data size | (+) Memory size which is occupied by all local variables (they size may be found in the ESL Diagnostic Pack - Instance). |
| DB Connects count | (+) Number of database connections that are created by DB_CONNECT, PG_CONNECT and SQL_CONNECT. |
| DB Transactions count | (+) Number of database transactions opened by DB_TRANS_OPEN. |
| Files count | (+) Number of files opened or created by %FIO_* script action. |
| Object lists count | (+) Number of opened by LST_CREATE. |
| Status | Current script status (status ES_WAIT_EXTERNAL_RQ means the passive waiting). |
| Physical process, DODM process | Name of objects of *Process* type (with suffix EVH, HIS, SEE, SES), which contains the given instance. |
| Queue Length | For object of SERVEREVENT or CLIENTANDSERVEREVENT type (this includes also a picture), the value means the number of requests (e.g. calls of RPC procedures, changes watched by ON CHANGE, ...) in a queue. For the event of TRIGGEREVENT type and the permission of request queue for start, the value means the number of requests for start. |
| XML Docs count | (+) Number of opened XML documents. |
| DBS objects count | (+) Number of database objects with which data were exchanged (read/write). |
| D2000 HI | Name of D2000 HI if DODM process is used for interpretation of ESL scripts opened in the appropriate HIP process. |

The columns with (+) contains the total count of elements which includes also possible instances of UNIT events !!!!

This menu appears after the right-clicking over the instance.
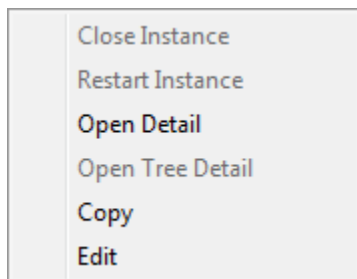


The option *Close Instance* and *Restart Instance* is available only for instances that were opened by OPENEVENT script action and enables to close the instance or close and reopen (the required number of instance will remain).

The option *Open Detail* opens detail view on the instance (ESL Diagnostic Pack – Instance).

The option *Open Tree Detail* is available only for the script of picture. It opens detailed view on all opened pictures and events in the D2000 HI.

Example:
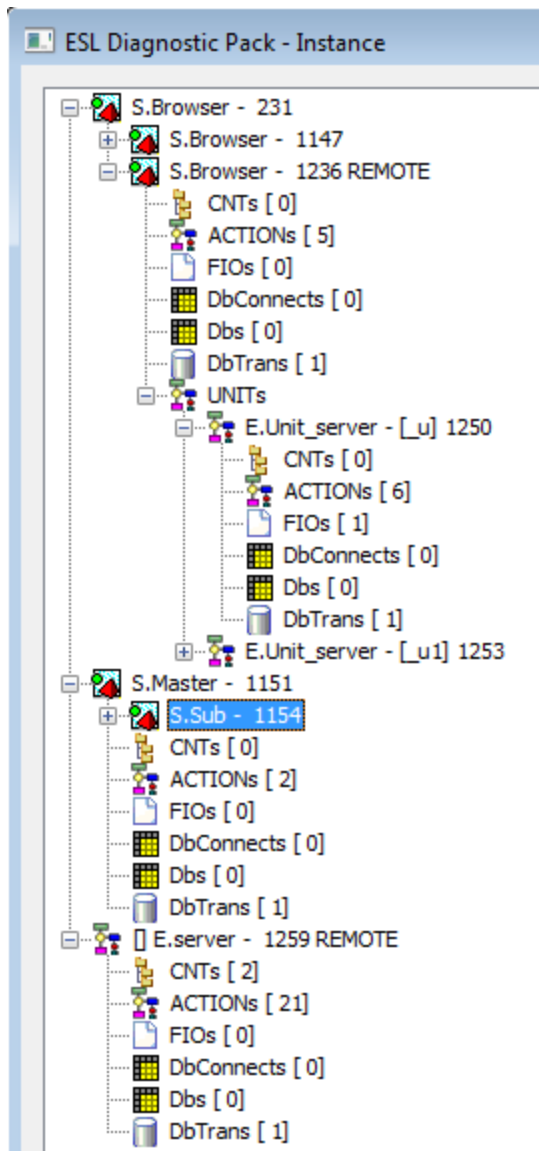
On the picture you may see that two pictures are opened in D2000 HI - S.Browser and S.Master.

The picture S.Browser has both a local and remote script (the remote one is specified by the word REMOTE). After the name, there is always the attribute *i dInstance* for better identification of instance. The remote script uses two event of UNIT type, which are declared in ESL script as _u and _u1.

The picture S.Master contains the subpicture S.Sub.

In D2000 HI, there is opened the object of Event type - E.server. The name contains REMOTE flag, so it is started as the remote script. After right-clicking on the event or picture, this popup menu opens:
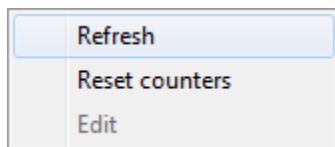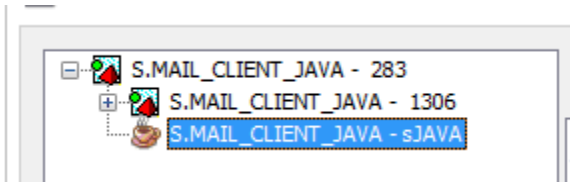


Refresh – refreshes data on instance of script or picture.

Reset counters – resets the counters in the lists of CNTs, ACTIONs, DbConnects, Dbs and DbTrans (in the text below).

If some picture has only JAVA script, this information will display:
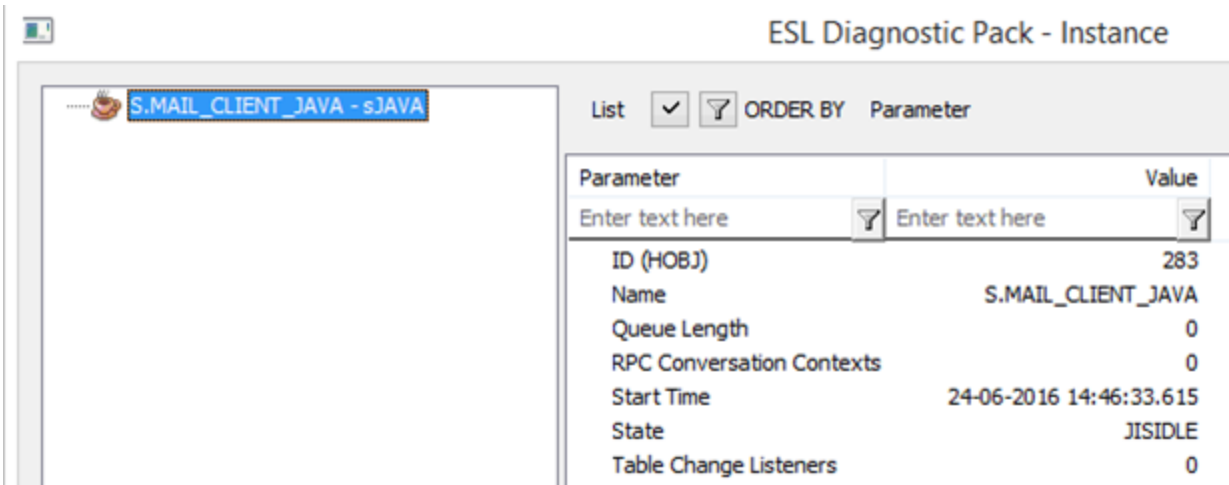
If the picture has both ESL script and JAVA scrip, both will be displayed:



Each instance describes in detail its status by the following items (the number in the square brackets represents the number of instances for each item).

## CNTs  - data containers

After double-clicking on the data container, the line of its creation appears.
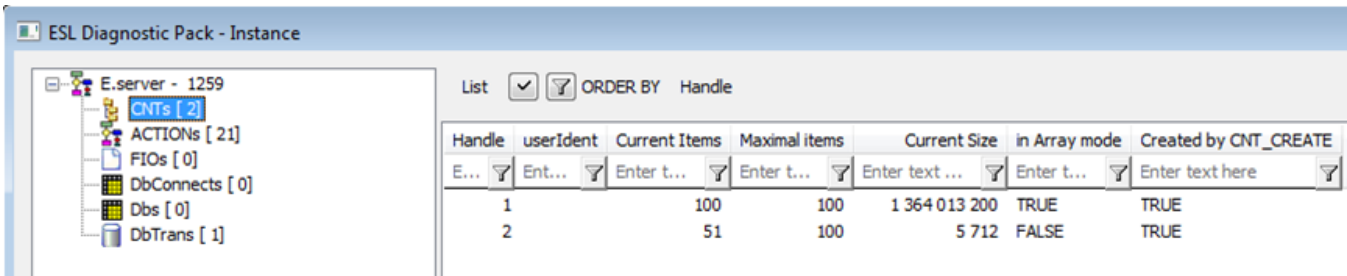


Description of columns:

| | |
|---|---|
| Handle | Unique identifier of data container. The value is the same as value returned by CNT_CREATE. |
| userIdent | Value of string, defined by a user, within CNT_DEBUG. |
| Current Items | Current items in the container. |
| Maximal Items | Maximum items in the container (it may be reset by the option *Reset counters*). |
| In Array mode | Flag that indicates the existence of internal array (see CNT_CNVTOARRAY script action). |
| Created by CNT_CREATE | Described the way of creation of container. TRUE – CNT_CREATE script action. FALSE – GETARCHARR_TO_CNT script action. |

## ACTIONs  - the actions in ESL script

Each action of ESL script is je characterized as stated on the picture below:

Script E.CallServerWithQueue:

```
 9
10    BEGIN
11      INT _i
12      RECORD (SD._System_Proces) _rec
13
14      FOR _i = 1 TO 10 DO_LOOP
15        REDIM _rec[2*_i]
16        DELAY 500[ms]
17        CALL [E.ServerWithQueue] Message  ON SELF.EVH
18      END_LOOP
19
20    END
21
```

Script E. ServerWithQueue:

```
 8
 9    RPC PROCEDURE Message
10      DELAY 1[s]
11    END Message
```

Description of columns:

| Line | Line on which the action occurs. |
|---|---|
| Action | Identification of action type. |
| Counter | Number of usage of action (it may be reset by the option *Reset counters*). |
| Counter passive | Number of passive wait states. This value increases by 1 after each passive waiting on the action to carry out the activity which caused. The counter can be higher than Counter. In the example above, the DELAY and RPC CALL SYNC action cause it (to reset it, use the option *Reset counters*). |
| Duration | Total duration of the active execution (it may be reset by the option *Reset counters*). |
| DurationPassive | Total duration of the passive waiting for execution of required activity (it may be reset by the option *Reset counters*). |
| maxDuration | Maximum duration of the active execution (it may be reset by the option *Reset counters*). |
| maxDurationPassive | Maximum duration of the passive waiting for execution of required activity (it may be reset by the option *Reset counters*). |
| actionInfo | Variable information which depends on type of action. Only two actions give the information: <br> • REDIM - displays the required size for 10 latest executions. <br> • VAR - declaration of the local variable. It displays the name of local variable and the current size of necessary memory (in case of recursion and the local variable it is the memory consumption for all its instances). |

# FIOs  - opened files

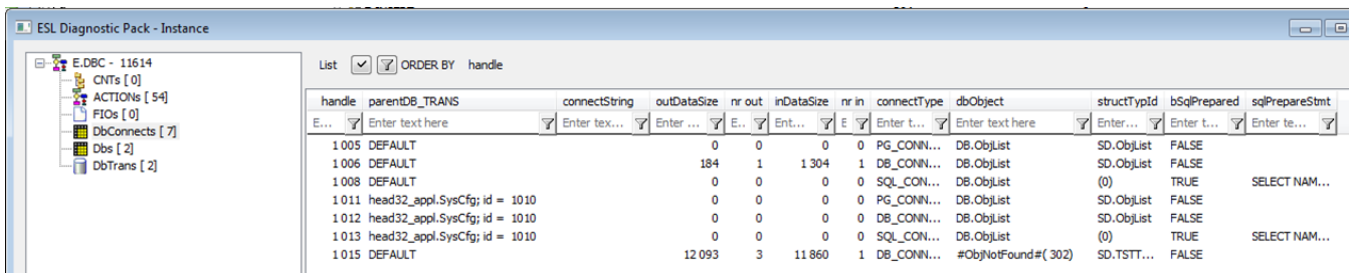After double-clicking on the file, the line of its creation appears.
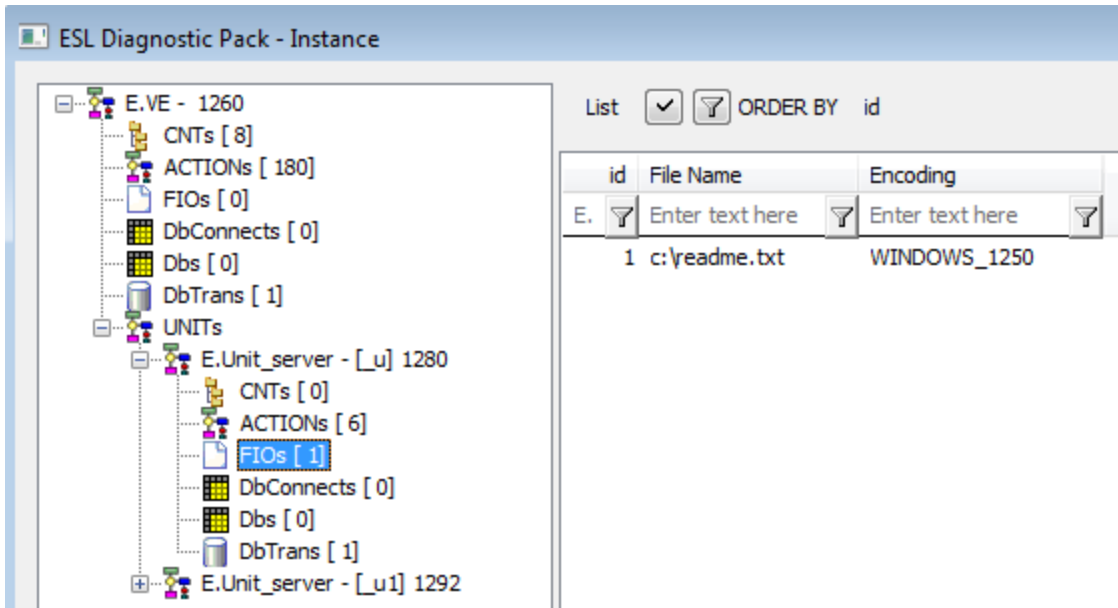
Description of columns:

| Id | Unique identifier of file. The value is equal to the value that is returned by the appropriate %FIO_* action. |
|---|---|
| File Name | Name of file. |
| Encoding | File encoding. |

## DbConnects - database connections that were created by DB_CONNECT, PG_CONNECT and SQL_CONNECT

DAfter double-clicking on the file, the line of its creation appears.



Description of columns:

| Handle | Unique identifier of connection. The value is equal to a value that returns the appropriate action. |
|---|---|
| parentDB_TRANS | DEFAULT = if the connection to a database is created by so-called Automatic connection, or name of object of Database type that was used to establish the database transaction by DB_TRANS_OPEN. |
| connectString | The value that is valid when establishing connection by SQL_CONNECT with connectStringom. |
| outDataSize, inDataSize | Amount of data which were sent or received within the given connection. |
| nr out, nr in | Number of output or input operations. |
| connectType | Depending on the method of establishing the connection, it is the value PG_CONNECT, DB_CONNECT or SQL_CONNECT. |
| dbObject | Name of object of Table type, which was used when establishing the connection (if only HOBJ of object is available, the following message appears: #ObjNotFound#(302), but HOBJ=302). |
| structTypId | Name of structure definition assigned to the appropriate object of Table type. |
| bSqlPrepared | Flag that indicates whether SQL_PREPARE is active on the given connection. |
| sqlPrepareStmt | Form of SQL command. |

## Dbs - database objects that was read by script or recorded data

This view summarizes quantity of transferred data with regard to the objects of *Table* type regardless of way or connection on which the transfer occurred.



Description of columns:

| Table | Name of object of *Table* type (if only HOBJ is available, the following message appears: #ObjNotFound#(302), but HOBJ=302). |
|---|---|
| outDataSize, inDataSize | The amount of data that were sent or received. |
| nr out, nr in | Number of output or input operations. |

## DbTrans - database transactions opened by DB_TRANS_OPEN

After double-clicking on the file, the line of its creation appears.



Description of columns:

| Handle | Unique identifier of connection. The value is equal to a value that returns the appropriate action DB_TRANS_OPEN. |
|---|---|
| dbObject | Name of object of *Database* type, for which the transaction was opened. |
| subConnectsNr | Number of connections of type DB_CONNECT, PG_CONNECT, SQL_CONNECT established within the transaction. |
| outDataSize, inDataSize | The amount of data that were sent or received within connection. |
| nr out, nr in | Number of output or input operations. |

## UNITs - instances of Units

In the list of instances of UNIT events you may see similar information as in the list of event instance.

---

ⓘ **Related pages:**

Ponuka Objekty