

# Ladenie a debugovanie

## D2000 DBManager - ladenie a debugovanie

Ladiace informácie a informácie o chybách procesu **D2000 DBManager** sa zobrazujú v okne procesu a ukladajú sa do logovacieho súboru v podadresári **T RACE aplikného adresára**. Ak sa proces **D2000 DBManager** spúša ako **SELF.DBM**, vytvorí sa logovacie súbory **DBManager.log** a **DBManager\_ERR.log**. Ak sa spúša ako **MENO.DBM**, vytvorí sa logovacie súbory **DBManager\_meno.log** a **DBManager\_meno\_ERR.log**. Ak vekos logovacieho súboru dosiahne preddefinovanú hodnotu 10 MB, bude premenovaný na **DBManager\_meno\_prev.log** (**DBManager\_meno\_ERR\_prev.log**) a vytvorí sa nový logovací súbor. Predchádzajúci súbor **DBManager\_meno\_prev.log** (**DBManager\_meno\_ERR\_prev.log**) sa zmaže.

Ak má objekt typu **Databáza** zadanú nenulovú hodnotu parametra **Vekos logovacieho súboru**, ladiace a chybové informácie akcií vykonávaných na tomto objekte sa budú zapisovať do logovacieho súboru databázy. Názov súboru je **meno\_databazy.log**, pričom v mene databázy sú bodky nahradené znakom **"\_"** (ak objekt typu **Databáza** sa volá **DB.Test**, názov súboru bude **DB\_Test.log**). Pre samotné chybové informácie je názov súboru **meno\_databazy\_ERR.log**.

Ak vekos logovacieho súboru dosiahne vekos rovnú hodnotu parametra **Vekos logovacieho súboru**, bude premenovaný na **meno\_databazy\_prev.log** a vytvorí sa nový logovací súbor. Predchádzajúci súbor **meno\_databazy\_prev.log** bude zmazaný. Rovnako to platí pre súbor **meno\_databazy\_ERR.log**.

Proces D2000 DBManager ponúka niekoľko úrovní debugovania:

- Výpisy chýb
- Výpisy akcií
- Výpisy akcií trvajúcich dlhšie ako zadaný počet sekúnd
- Výpisy so zapnutou ladiacou kategóriou **DBG.DBMANAGER**
- Výpisy so zapnutou ladiacou kategóriou **DBG.DBMANAGER.DATA**
- Výpisy so zapnutou ladiacou kategóriou **DBG.DBMANAGER.DBCTX**
- Výpisy so zapnutou ladiacou kategóriou **DBG.DBMANAGER.DBCTX.CSV**
- Výpisy so zapnutou ladiacou kategóriou **DBG.DBMANAGER.SQL\_CONNECT**
- Výpisy so zapnutým štartovacím parametrom **/DBD<pocet\_poziadaviek>**
- Tell príkaz **SHOW\_HANDLE**
- Tell príkaz **SHOW\_CONNECT**
- Tell príkaz **SET\_WATCHDOG**
- Tell príkaz **SET\_WATCHDOG\_QUEUE**
- Tell príkaz **TIME\_STATISTICS**
- Optimalizácia a ladenie
- **SV\_System\_DBMDbPerf**

### Výpisy chýb

Chyby sa zapisujú do logovacieho súboru koniaceho **\_ERR.log** vždy. Pri chybovej správe je väčšinou uvedený aj SQL príkaz, ktorý ju spôsobil.

Príklad:

```
12:40:08.760 25.02 *** Error in con 24:
12:40:08.766 25.02 con 24:SELF.EVH;E.Test_JS_Column_Types( 1928) 2448;: 36
```

```
12:40:08.771 25.02 con 24:[InsertRecord] Execute: INSERT INTO "ROVE_OD"."TEST_COLUMN_TYPES" ("ID","ColumnName","ColumnDate") VALUES
(:1,:2,:3) ,row 1,Exception name:
OCI.THICK.LIB_ERROR
Message: ORA-00001: porušenie jedineného obmedzenia (ROVE_OD.SYS_C00112480)
```

```
Call stack traceback locations:
0x453d18 0x4542d1 0x4509dc 0x991db4 0x999458 0x906822 0x8ef428 0x9d187c 0x76b6652b
```

**Poznámka 1:** Ak skript vykonávajúci akciu, počas ktorej sa vyskytla chyba, je v režime ladenia, tak informácia o chybe sa odošle do okna **ESL Editor** a zobrazí sa v záložke **Debug**.

**Poznámka 2:** Jednotlivé ESL akcie pre prácu s databázou (cez proces D2000 DBManager) si ukladajú informáciu, odkiaľ boli volané z ESL kódu. Pri výskyte chyby je potom vypísaná aj táto informácia (v príklade na začiatku logu) okrem samotnej informácie o zistenej chybe.

### Výpisy akcií

Podrobnejšie výpisy do logovacieho súboru sa zapisujú po zapnutí voby **Debug** v konfigurácii objektu typu **Databáza**. Log obsahuje informácie o jednotlivých akciách ESL (typ akcie, začiatok a koniec), o databáze a spojení, na ktorom akcia prebieha.

Príklad:

```
14:18:59.677 Db TestDb con 2:PG_INSERT END
14:19:01.099 Db TestDb con 2:PG_INSERT BEG
```

Log ukazuje začiatok (BEGIN) a koncový (END) stav akcie **PG\_INSERT** na databáze (**Db**) s menom **TestDb**. Táto akcia sa vykonala prostredníctvom spojenia (**con**) . 2 a trvala 422 milisekúnd.

**Poznámka:** Ak sa log ukladá do logovacieho súboru databázy a nie do spoločného logovacieho súboru DBManagera (vi parameter **Vekos logovacieho súboru**), výpis neobsahuje názov databázy. Periodicky (každých 60 sekúnd) sa do logu ukladá watchdog správa (WD) a informované výpisy o databáze: **09:00:27.786 Db TestDb WD: 5 (4/1/0) cons: avail- 3,normal- 2, handles- 4**

Log informuje, že databáza *TestDb* má pä spojení, z toho štyri sú netrtransakné, jedno transakné a žiadne rezervované pre browser. Tri spojenia sú voné (a *vail*) a dve sa používajú (*normal*).

Zoznam všetkých možných stavov je zobrazený v tabuके:

| Stav   | Popis                                                              |
|--------|--------------------------------------------------------------------|
| Init   | Spojenie je v inicializanej fáze (vytvára sa). Prechodný stav.     |
| Avail  | Spojenie je voné.                                                  |
| Normal | Spojenie sa používa.                                               |
| Live   | Spojenie prechádza zo stavu Avail do stavu Normal. Prechodný stav. |
| Die    | Spojenie sa zaviera. Prechodný stav.                               |
| Zombie | Spojenie je zavreté, obslužný thread koní. Prechodný stav.         |

*Handles* udáva poet deskriptorov, ktoré má DBManager otvorené. Môžu existova štyri typy deskriptorov:

- **deskriptor tabuky** sa otvára z procesu [D2000 HI](#) cez zoznam tabuliek, v browseri alebo zo skriptu volaniami [DB\\_CONNECT](#), [PG\\_CONNECT](#), [S QL\\_CONNECT](#), [SQL\\_PREPARE](#) a na krátku dobu (poas trvania akcie) aj volaniami [DBS\\_\\*](#)
- **deskriptor transakcie** sa otvára volaním akcie [DB\\_TRANS\\_OPEN](#)
- **deskriptor spojenia** sa otvára volaním akcie [SQL\\_CONNECT](#), ak prvý parameter je *connectString*
- **deskriptor databázy** sa otvára volaním akcie [SQL\\_CONNECT](#), ak prvý parameter *dbObjIdent* je typu [Databáza](#)

Pokia má niektorá tabuka nakonfigurovaný parameter *asová hbka* a proces [D2000 DBManager](#) z nej periodicky maže staré záznamy, logovací súbor bude obsahova aj údaje o zaiatku a konci periodického mazania:

```
10:29:32.068 11.08 Db TestDB table Time_Test periodic delete BEG
10:29:32.162 11.08 Db TestDB table Time_Test periodic delete END
```

## Výpisy akcií trvajúcich dlhšie ako zadaný poet sekúnd

Výpisy o trvaní operácií dlhších ako zadaný poet sekúnd sa do logovacieho súboru zapisujú po nastavení parametra [Loguj dlhšie operácie ako \(sec\)](#) na nenulovú hodnotu v konfigurácii objektu typu *Databáza*. Log obsahuje informácie o spojení, na ktorom akcia prebehla, lokalizáciu akcie v ESL skripte, volajúcu správu DbManager-a a as trvania akcie v milisekundách.

Príklad:

```
19:27:37.286 01.07 con 1:DBS_READ BEG
19:27:37.297 01.07 con 1:DBS_READ END
19:27:37.298 01.07 con 1:DBS_READ
SELF.EVH;E.Test( 1934) 1230;: 16
- M.DB.CDB_CTRLMSG - 122[ms]
```

## Výpisy so zapnutou ladiacou kategóriou DBG.DBMANAGER

Podrobnejšie výpisy do súboru po zapnutí ladiacej kategórie *DBG.DBMANAGER* pomocou procesu [D2000 System Console](#) v okne [Debug info](#). Každý výpis akcie obsahuje navyše informácie o volaných procedúrach v procese [D2000 DBManager](#), ktoré sú užitočné pre vývojárov, a texty vykonávaných SQL príkazov.

Príklad ukazuje výpis o vykonaní akcie [SQL\\_PREPARE](#):

```
12:05:55.301 22.08 Db TestDB con 1:SQL_PREPARE BEG
12:05:55.302 22.08 PrepareSqlStmt: SELECT ID_MATERIAL,Name FROM material
12:06:30.028 22.08 PrepareSqlStmt end
12:06:30.029 22.08 Db TestDB con 1:SQL_PREPARE END
```

Podrobnejšie príklady výpisov s odlišnosťami pre ODBC a OCI verziu DBManagera nájdete v [príkladoch výpisov logovacieho súboru](#).

## Výpisy so zapnutou ladiacou kategóriou DBG.DBMANAGER.DATA

Podrobnejšie výpisy do súboru po zapnutí ladiacej kategórie *DBG.DBMANAGER.DATA* pomocou procesu [D2000 System Console](#) v okne [Debug info](#). Každý výpis vykonávanej akcie obsahuje navyše výpis hodnôt, ktoré sa zapísali do databázy alebo sa z nej naítali.

Príklad ukazuje akciu zobrazenie prvej stránky tabuky *MAT\_GROUP* v zobrazovai typu *Browser* v [D2000 HI](#):

```

10:35:22.601 25.08 Db MyDB con 5:GetDatabaseData START
10:35:22.602 25.08 Open database DSN: TENTO
10:35:23.727 25.08 Open database DSN: TENTO end
10:35:23.728 25.08 PrepareSelectStmt: SELECT "ID_GROUP","NAME" FROM "SKVI"."MAT_GROUP"
10:35:23.736 25.08 PrepareSelectStmt end
10:35:23.737 25.08 ReadPage (Direction: 2): SELECT "ID_GROUP","NAME" FROM "SKVI"."MAT_GROUP"
10:35:23.741 25.08 ReadPage data row 1: 7; 'wood';
10:35:23.742 25.08 ReadPage data row 2: 1; 'plastic';
10:35:23.742 25.08 ReadPage data row 3: 2; 'metal';
10:35:23.743 25.08 ReadPage data row 4: 3; 'paper';
10:35:23.743 25.08 ReadPage data row 5: 6; 'water';
10:35:23.744 25.08 ReadPage data row 6: 4; 'other';
10:35:23.745 25.08 ReadPage end
10:35:23.746 25.08 Db MyDB con 5:GetDatabaseData END

```

Podrobnejšie príklady výpisov s odlišnosťami pre ODBC a OCI verziu DBManagera nájdete v [príkladoch výpisov logovacieho súboru](#).

## Výpisy so zapnutou ladiacou kategóriou DBG.DBMANAGER.DBCTX

Podrobnejšie výpisy po zapnutí ladiacej kategórie DBG.DBMANAGER.DBCTX pomocou procesu [D2000 System Console](#) v okne [Debug info](#).

Pri modifikovaní kontextu procesu viazaného v procese D2000 DBManager (ESL akciou DB\_SET\_PROCESS\_PARAMS) sa vypisujú informácie do hlavného logu daného procesu D2000 DBManager, ktoré sú užité pre vývojárov. Nasledujúce výpisy sú teda nezávislé na vobe [Debug](#) v konfigurácii objektu typu *Databáza* (nie sú zapisované do adresára *Trace*).

Príklad ukazuje výpis po vykonaní akcie DB\_SET\_PROCESS\_PARAMS pri nastavovaní parametra:

```

[01-08-2013 13:14:13] DBCTX: Call Chain of DB_SET_PROCESS_PARAMS - SELF.EVH;E.SetParams( 1347485) 8735;; 12
[01-08-2013 13:14:13] DBCTX: CTX created for idOwner = 55
[01-08-2013 13:14:13] DBCTX: INSERT parameter "NAME=SELF.DBM" of idOwner = 55

```

**Poznámka:** Výpis o vytvorení kontextu ("CTX created ...") sa zobrazí len pri prvom volaní akcie od naštartovania procesu.

Príklad ukazuje výpis po vykonaní akcie DB\_SET\_PROCESS\_PARAMS pri vymazávaní parametra:

```

[01-08-2013 13:16:53] DBCTX: Call Chain of DB_SET_PROCESS_PARAMS - SELF.EVH;E.SetParams( 1347485) 8936;; 12
[01-08-2013 13:16:53] DBCTX: DELETE parameter "NAME" of idOwner = 55

```

Podrobnejšie výpisy do logovacieho súboru sa zapisujú po zapnutí voby [Debug](#) v konfigurácii objektu typu *Databáza*. Log obsahuje informácie o jednotlivých parametroch nastavených v kontexte daného klienta spúšajúceho DB akcie cez proces [D2000 DBManager](#).

Príklad ukazuje výpis pri vykonaní akcie SQL\_SELECT, ak boli predtým nastavené parametre kontextu procesu cez akciu DB\_SET\_PROCESS\_PARAMS:

```

15:01:14.114 01.08 con 1:SQL_SELECT BEG
15:01:14.121 01.08 con 1: D2000_PROCESS_PARAMS - SET PARAMS "NAME=VERSION27" of dbCtxId = 55
15:01:14.125 01.08 con 1:SQL_SELECT END

```

## Výpisy so zapnutou ladiacou kategóriou DBG.DBMANAGER.DBCTX.CSV

Podrobnejšie CSV výpisy po zapnutí ladiacej kategórie DBG.DBMANAGER.DBCTX.CSV pomocou procesu [D2000 System Console](#) v okne [Debug info](#). Do logovacieho súboru sa zapisujú až po zapnutí voby [Debug](#) v konfigurácii objektu typu *Databáza*. Log obsahuje štruktúrované CVS informácie o jednotlivých parametroch nastavených v kontexte daného klienta spúšajúceho DB akcie cez proces [D2000 DBManager](#).

CSV hlavička:

```

TIMESTAMP;D2000_PROCESS_PARAMS;CID;COMMAND;TID;CURRENT_TASK;DB_CTX_ID;DB_CTX_ON;MC1;MC2;OPERATION;SUCCESS
set ;={TRUE/FALSE};
del ;={TRUE/FALSE}

```

CSV stpce: asová peiatka, konštanta "D2000\_PROCESS\_PARAMS", identifikátor spojenia, DB príkaz, identifikátor internej transakcie, názov task-u, ID klienta, existencia kontextu (TRUE/FALSE); modifikované poítadlo pred a po (MC1 a MC2), operácia a jej úspešnosť. Operácie sú bu modifikácia parametrov (set) alebo rušenie kontextu na danom spojení (del).

Príklad 1:

```

15:28:12.899 01.08 con 1:SQL_SELECT BEG
15:28:12.905 01.08 ;D2000_PROCESS_PARAMS;1;SQL_SELECT;1005;tdb_task_000000000905BB40;55;TRUE;0;1;set
"NAME=VERSION27";TRUE
15:28:12.908 01.08 con 1:SQL_SELECT END

```

Príklad 2:

```

15:28:22.051 01.08 con 1:SQL_SELECT BEG
15:28:22.054 01.08 ;D2000_PROCESS_PARAMS;1;SQL_SELECT;1006;tdb_task_000000000905BB40;55;TRUE;1;2;set;TRUE
15:28:22.056 01.08 con 1:SQL_SELECT END

```

## Výpisy so zapnutou ladiacou kategóriou DBG.DBMANAGER.SQL\_CONNECT

Zapnutie tejto ladiacej kategórie bude mať rovnaký účinok, ako pridanie "DEBUG" do parametra *connectString* vo všetkých volaniach akcie [SQL\\_CONNECT](#), teda zapisovanie ladiacich informácií akcie [SQL\\_CONNECT](#) a všetkých ďalších akcií, ktoré používajú toto spojenie, do logu procesu **D2000 DBManager**.

**Poznámka:** Ladiaca kategória [DBG.DBMANAGER.SQL\\_CONNECT](#) neovplyvuje tie akcie [SQL\\_CONNECT](#), ktoré používajú odkaz na objekt typu [Databáza](#) alebo [Tabuka](#), lebo na tie sa vzahuje parameter [Debug](#) v konfiguracii okna objektu [Databáza](#). Ovplyvuje iba akcie [SQL\\_CONNECT](#), ktoré na spojenie používajú *connectString*, t.j. textový pripojovací reazec, ktorého debugovanie sa nedá zapnúť v procese **D2000 CNF**, ale iba pridaním [DEBUG](#) do parametra *connectString*.

## Výpisy so zapnutým štartovacím parametrom [/DBD<pocet\\_poziadaviek>](#)

Vo verzii 7.01.023 bolo implementované ladenie výkonnosti automatických (netransakčných) spojení. Ak je spustený **DBManager** s parametrom [/DBD<pocet\\_poziadaviek>](#), do logu po štarte zapíše riadok:  
*Performance logging is ON.*

Ak na niektorom z [automatických spojení](#) prekroí počet požiadaviek vo fronte hodnotu *pocet\_poziadaviek*, v logovacom súbore (log databázy, ak je zadaná hodnota [Vekos logovacieho súboru](#), inak log **DBManager**a) vznikne zápis obsahujúci výkonnostné varovanie s časom spojenia a počet požiadaviek vo fronte: *10:32:12.983 14.02 con 1:performance warning: 2 requests*

Toto varovanie znamená, že prichádzajúca netransakčná požiadavka bude musieť čakať, kým sa spracujú požiadavky pred ňou. Pokiaľ sa takéto varovania objavujú častejšie, kvôli zvýšeniu "priepustnosti" je vhodné nakonfigurovať väčší počet [automatických spojení](#). Odporúčaná hodnota *<pocet\_poziadaviek>* je 1, t.j. spúšťanie **DBManager**a s parametrom [/DBD1](#).

**Poznámka 1:** Pokiaľ je nastavený parameter [Debug](#), spracovanie požiadaviek je pomalšie, keďže sa vypisujú ladiace informácie do logu. Vtedy treba zvážiť počet [automatických spojení](#) alebo zvýšiť hodnotu *<pocet\_poziadaviek>* na [/DBD2](#), aby sa výkonnostné varovania neobjavovali.

**Poznámka 2:** Transakčné spojenia výkonnostne ladiť netreba, keďže jedno transakčné spojenie sa v súčasnosti používa iba sekvencne z jedného eventu.

## Tell príkaz [SHOW\\_HANDLE](#)

Interaktívne zisovanie otvorených [deskriptorov](#) pomocou Tell príkazu [SHOW\\_HANDLE](#). Syntax príkazu je:

*SHOW\_HANDLE [HOBJ] alebo [maska [INFO]]*

Voliteľné parametre môžu byť:

- **HOBJ** objektu [Databáza](#), ktorého otvorené deskriptory sa majú zobraziť
- **HOBJ** objektu [Tabuka](#), ktorého otvorené deskriptory sa majú zobraziť
- **HOBJ** objektu [Definícia štruktúry](#), pričom sa majú zobraziť otvorené deskriptory tabuliek, ktoré používajú túto definíciu štruktúry
- maska názvu tabuky, ktorej otvorené deskriptory sa majú zobraziť
- maska informácie, odkiaľ bol deskriptor otvorený, pokiaľ je uvedené kľúčové slovo *INFO*

Pokiaľ je Tell príkaz **SHOW\_HANDLE** volaný bez parametrov, vypíše sa informácia o všetkých otvorených deskriptoroch. Výpis obsahuje pre každý deskriptor názov databázy, číslo spojenia, typ a názov deskriptora a informáciu, odkiaľ bol deskriptor otvorený. Výpis typu a názvu deskriptora je pre rôzne typy deskriptorov zobrazený v tabuľke:

| Typ deskriptora       | Výpis typu     | Názov deskriptora                                                                                              |
|-----------------------|----------------|----------------------------------------------------------------------------------------------------------------|
| deskriptor tabuky     | <i>table</i>   | Meno objektu typu <i>Tabuka</i> .                                                                              |
| deskriptor transakcie | <i>trans</i>   | Meno objektu typu <i>Databáza</i> , na ktorej je transakcia otvorená.                                          |
| deskriptor spojenia   | <i>connect</i> | Parameter <i>connectString</i> akcie <a href="#">SQL_CONNECT</a> .                                             |
| deskriptor databázy   | <i>dbase</i>   | Meno objektu typu <i>Databáza</i> , ktorý bol parametrom <i>dbObjIdent</i> akcie <a href="#">SQL_CONNECT</a> . |

Príklad výpisu:

```
CONO connection established (IPC_TCPIP)
Receiv TELL Command : SHOW_HANDLE
=====
->Db TestDB con 1:table MAT_GROUP: <HI mycomp,S.Test_DBmanager>
->Db TestDB con 2:trans TestDB: <S.Test_DBmanager: 220>
=====
```

**D2000 DBManager** má dva otvorené deskriptory. Prvý je na spojení . 1, deskriptor je typu *deskriptor tabuky* a otvorený je v browseri tabuka *MAT\_GROUP*. Deskriptor je otvorený zo schémy *S.Test\_DBmanager* z procesu **D2000 HI**, ktorý beží na počítači *mycomp*. Druhý deskriptor je na spojení . 2, deskriptor je typu *deskriptor transakcie* a otvorený je z 220-ého riadku skriptu v schéme *S.Test\_DBmanager*.

## Tell príkaz [SHOW\\_CONNECT](#)

Interaktívne zisovanie otvorených spojení pomocou Tell príkazu [SHOW\\_CONNECT](#). Syntax príkazu je:

*SHOW\_CONNECT [HOBJ [ID]] alebo [maska [ID]] [DETAIL]*

Voliteľné parametre sú:

- HOBJ objektu *Databáza*, ktorého otvorené spojenia sa majú zobraziť
- maska názvu objektu *Databáza*, ktorého otvorené spojenia sa majú zobraziť
- ID - číslo spojenia alebo číslo transakcie (vrátené ako parameter *handleIdent\_Int* akcie [DB\\_TRANS\\_OPEN](#)), ktoré sa má zobraziť
- parameter "DETAIL" spôsobí, že za výpisom spojenia bude nasledovať výpis deskriptorov (vo formáte ako pre [SHOW\\_HANDLE](#)) otvorených na tomto spojení

Výpis obsahuje:

- pre každú databázu, vyhovujúcu vstupnému filtru, jej názov a počet otvorených spojení
- pre každé spojenie:
  - názov databázy
  - číslo spojenia
  - stav spojenia
  - počet otvorených deskriptorov
  - dobu od poslednej vykonanej operácie (alebo slovo *busy* a informácie o vnútornom stave spojenia, ak na spojení práve prebieha operácia. Príklad: *busy (U\_EXECDIRECT1/D\_EXECDIRECT2)*)
  - informáciu, či je spojenie transakčné, netransakčné alebo netransakčné rezervované pre browsery
  - pre transakčné spojenie aj číslo transakcie (parameter *handleIdent\_Int* akcie [DB\\_TRANS\\_OPEN](#))
- pre každý deskriptor (parameter "DETAIL") výpis vo formáte [SHOW\\_HANDLE](#)

Príklad výpisu (spojenia v databázach záujmujúcich sa na Te spolu s výpisom deskriptorov):

```
Receive TELL Command : SHOW_CONNECT Te* DETAIL
=====
->Db TestDB 2 cons
->Db TestDB con 1:normal, 1 handles, idle 04:24:682, non-transact
->Db TestDB con 1:table MAT_GROUP: <HI mycomp,S.Test_DBmanager>
->Db TestDB con 2:normal, 2 handles, idle 20:037, transact 1053
->Db TestDB con 2:trans TestDB: <S.Test_DBmanager: 220>
->Db TestDB con 2:table MAT_GROUP: <S.Test_DBmanager: 109>
```

**Poznámka:** Pre spojenia, ktoré sú v [status](#) *avail*, udáva informácia o transakčnosti a číslo transakcie starý údaj (transakcia bola už ukončená). Takéto spojenie môže byť [recyklované](#) podľa potreby ako transakčné alebo netransakčné.

## Tell príkaz SET\_WATCHDOG

SQL operácie, ktorých samotné vykonávanie trvalo dlhší čas, je možné sledovať tell príkazom [SET\\_WATCHDOG](#). Syntax príkazu je:

*SET\_WATCHDOG database\_mask seconds*

Povinné parametre sú:

- maska názvu objektu *Databáza*, ktorého spojenia sa majú monitorovať
- počet sekúnd, po uplynutí ktorých je databázová operácia vypísaná, pokiaľ ešte stále trvá. Hodnota 0 vypína sledovanie doby vykonávania SQL operácií.

Pri prvom spustení Tell príkazu [SET\\_WATCHDOG](#) s nenulovým parametrom *seconds*, ktorého maske vyhovuje aspoň jedna databáza, je spustený sledovací task, ktorý každú sekundu skontroluje, či spojenie na niektorej sledovanej databáze neprekročilo nastavený čas vykonávania (parameter *seconds*). Ak sa tak stane, táto informácia sa zapisuje do logu. Výpis obsahuje:

- názov databázy (pokiaľ nemá Databáza nakonfigurovaný vlastný log)
- číslo spojenia
- dĺžka trvania operácie
- informácie o vnútornom stave spojenia (pre vývojárov D2000)

Príklad výpisu (z logu databázy, takže názov databázy sa neuvádza):

```
16:45:28.763 24.11 Performance WD: con 3: operation lasts 27 sec in U_EXECDIRECT1/D_EXECDIRECT6
16:45:29.767 24.11 Performance WD: con 3: operation lasts 28 sec in U_EXECDIRECT1/D_EXECDIRECT6
16:45:30.782 24.11 Performance WD: con 3: operation lasts 29 sec in U_EXECDIRECT1/D_EXECDIRECT6
16:45:31.798 24.11 Performance WD: con 3: operation lasts 30 sec in U_EXECDIRECT1/D_EXECDIRECT6
16:45:34.829 24.11 Performance WD: con 3: operation lasts 2 sec in U_EXECDIRECT1/D_EXECDIRECT6
```

Po skončení príkazu sa vypíšu do chybového logu databázy podrobné informácie o príkaze. Príklady:

```

17:54:00.116 16.02 con 1:Query execution duration 00:00:01.103 SQL_CONNECT TransactId 10319, dbTransId 0, Handle
1104492764, connectString {}, DbTableId 1473 {DB.MATERIAL}, Comment {NS1PHUM3_HI.HIS;S.sql_TEST( 9473) 10864;
BT_connect_OnClick: 248}
17:54:18.926 16.02 con 1:Query execution duration 00:00:01.105 SQL_PREPARE TransactId 10320, dbTransId-1, Handle
10319, Statment {SELECT ID_MATERIAL FROM material }, bBindIn FALSE, FetchSize 1, colNr 17, Comment {NS1PHUM3_HI.
HIS;S.sql_TEST( 9473) 10864;BT_prepare_OnClick: 279}
17:55:59.113 16.02 con 2:Query execution duration 00:00:01.110 DB_TRANS_OPEN TransactId 10325, Comment
{NS1PHUM3_HI.HIS;S.sql_TEST( 9473) 10864;BT_db_trans_open_OnClick: 543}

```

Význam jednotlivých polí:

- SQL\_CONNECT / SQL\_PREPARE / DB\_TRANS\_OPEN .. - vykonávaná databázová akcia
- TransactId - číslo D2000 transakcie
- dbTransId - číslo databázovej transakcie. Ak je väčšie ako 0, databázová akcia je transakčná (prebieha v kontexte DB\_TRANS\_OPEN). Ak je nulové alebo záporné, operácia je netransakčná.
- Handle - číslo handle v rámci DBManagera
- DbTableId - HOBJ a meno D2000 objektu *Tabuka*
- Comment - postupnosť volania procedúr v rámci ESL skriptov (call chain)
- connectString, Statement, bBindIn, FetchSize, colNr, MaxRows .. parametre špecifické pre jednotlivé databázové akcie

**Poznámka:** Spustenie sledovania SQL operácií je zapísané v logu DBManagera. Výpis obsahuje zoznam databáz, ktoré vyhovujú maske *database\_mask*. Pri prvom volaní tell príkazu [SET\\_WATCHDOG](#) obsahuje výpis aj informáciu o spustení sledovacieho tasku.

```

16:45:27.735 24.11 =====
16:45:27.736 24.11 ->Db DBC_ROVE_OD 2 cons
16:45:27.737 24.11 ->Db DBC_KOMP_OD 3 cons
16:45:27.742 24.11 Starting performance watchdog task for database operations
16:45:27.743 24.11 =====

```

Pokiaľ bol na všetkých databázach nastavený parameter *seconds* na hodnotu 0, sledovací task prestane monitorovať databázy a v logu sa objaví hláška:

```
17:37:41.588 24.11 Performance watchdog: going to sleep (no more databases to monitor)
```

Pri ďalšom zapnutí sledovania už sledovací task beží (takže ho nie je potrebné spúšať) a v logu sa objaví iba hláška o aktivácii:

```
18:45:28.749 24.11 Performance watchdog: starting monitoring
```

## Tell príkaz SET\_WATCHDOG\_QUEUE

Databázové akcie, ktorých vykonávanie **vítane asu stráveného vo frontách DBManagera** trvalo dlhší čas, je možné sledovať tell príkazom [SET\\_WATCHDOG\\_QUEUE](#). Syntax príkazu je:

```
SET_WATCHDOG_QUEUE database_mask seconds
```

Povinné parametre sú:

- maska názvu objektu *Databáza*, ktorého spojenia sa majú monitorovať
- minimálna dĺžka trvania databázových akcií, ktorá sa má sledovať.  
Hodnota 0 vypína sledovanie doby vykonávania databázových akcií.

Rozdiel medzi [SET\\_WATCHDOG](#) a SET\_WATCHDOG\_QUEUE je tento:

príkaz SET\_WATCHDOG sleduje operácie, ktorých vykonávanie v databáze trvalo dlhšie ako špecifikovaný čas. Pokiaľ ale viacero klientov zdieľa to isté automatické (netransakčné) spojenie, z pohľadu klienta môže databázová akcia trvať dlho kvôli tomu, že akákoľvek vo fronte, v ktorej je pred ňou ešte jedna alebo viacero databázových akcií od iných klientov. Z toho dôvodu bol implementovaný príkaz SET\_WATCHDOG\_QUEUE, ktorý monitoruje celkový čas vybavenia databázovej akcie od vloženia do príslušnej fronty až po ukončenie.

Po skončení databázovej akcie sa vypíšu do chybového logu databázy podrobné informácie o príkaze. Príklady:

```
18:01:21.579 16.02 con 1:Query total duration 00:00:01.130, execution 00:00:00.105 SQL_EXEC_PROC TransactId 10331, dbTransId-1, Statement {{ call TESTF_IN_OUT (?) }}, Comment {NS1PHUM3_HI.HIS;S.sql_TEST( 9473) 11332; BT_exec_proc_OnClick: 626}
18:02:39.149 16.02 con 1:Query total duration 00:00:01.103, execution 00:00:00.202 SQL_CONNECT TransactId 10335, dbTransId 0, Handle 1104492765, connectString {}, DbTableId 1473 {DB.MATERIAL}, Comment {NS1PHUM3_HI.HIS;S.sql_TEST( 9473) 11453;BT_connect_OnClick: 248}
18:02:42.437 16.02 con 1:Query total duration 00:00:01.103, execution 00:00:00.035 SQL_DISCONNECT TransactId 10335, dbTransId-1, Comment {NS1PHUM3_HI.HIS;S.sql_TEST( 9473) 11453;BT_connect_OnClick: 248}
18:02:51.163 16.02 con 1:Query total duration 00:00:01.105, execution 00:00:00.654 SQL_PREPARE TransactId 10337, dbTransId-1, Handle 10336, Statement {SELECT ID_MATERIAL FROM material }, bBindIn FALSE, FetchSize 1, colNr 17, Comment {NS1PHUM3_HI.HIS;S.sql_TEST( 9473) 11453;BT_prepare_OnClick: 279}
18:02:54.275 16.02 con 1:Query total duration 00:00:01.110, execution 00:00:00.239 SQL_FETCH TransactId 10338, dbTransId-1, Handle 10336, MaxRows 100, Comment {NS1PHUM3_HI.HIS;S.sql_TEST( 9473) 11453;BT_fetch_OnClick: 299}
18:02:57.009 16.02 con 1:Query total duration 00:00:01.104, execution 00:00:00.109 SQL_FREE Handle 10336, Comment {NS1PHUM3_HI.HIS;S.sql_TEST( 9473) 11453;BT_free_OnClick: 325}
```

Význam jednotlivých polí: vi popis v rámci sekcie [SET\\_WATCHDOG](#).

## Tell príkaz TIME\_STATISTICS

Tell príkaz vypíše štatistiky trvania jednotlivých typov databázových akcií pre jednotlivé databázy alebo pre jednotlivé tabučky (pokiaľ je špecifikovaný parameter DETAIL). Pre každý typ databázovej akcie (otvorenie browsera, transakcie, trieda akcií zahŕajúca DB\_\* a DBS\_\* akcie at) sa vypíše celkový počet vykonaných akcií, celkové a priemerné trvanie a maximum trvania jednej akcie spolu s komentárom (ESL call chain).

Štatistiky s nulovým potom vykonaní sa nevypisujú.

Pokiaľ je špecifikovaný aj parameter DETAIL, po výpise štatistík pre databázu nasleduje výpis štatistík pre všetky tabučky, ktorých rodičom je databáza.

Príklad výpisu:

```
09:36:28.023 21.02 =====
09:36:28.026 21.02 ->Db MesDB total time 000 00:00:40.572
09:36:28.029 21.02 Operation BROWSER_OPEN executions 3 total duration 000 00:00:03.686, average 000 00:00:
01.229, maximum 000 00:00:01.468 by
09:36:28.032 21.02 Operation DB_CONNECT executions 2 total duration 000 00:00:02.207, average 000 00:00:01.104,
maximum 000 00:00:01.104 by NS1PHUM3_HI.HIS;S.Test_Dbmanagera( 9424) 10510;DB_CONNECT_OnClick: 109

09:36:28.034 21.02 Operation DB_CONTROL executions 15 total duration 000 00:00:16.589, average 000 00:00:01.106,
maximum 000 00:00:01.129 by con 1:DBS_READ
NS1PHUM3_HI.HIS;S.Test_DbRefresh( 10105) 10429;StressTest_OnClick: 20

09:36:28.037 21.02 Operation DB_REFRESH_TABLE executions 2 total duration 000 00:00:02.526, average 000 00:00:
01.263, maximum 000 00:00:01.266 by Triggered by commit of NS1PHUM3_HI.HIS;S.Test_DbRefresh( 10105) 10429;
StressTest_OnClick: 10

09:36:28.040 21.02 Operation DB_TRANS_OPEN executions 4 total duration 000 00:00:04.426, average 000 00:00:
01.106, maximum 000 00:00:01.107 by NS1PHUM3_HI.HIS;S.Test_Dbmanagera( 9424) 10479;TRANS_OPEN_OnClick: 227

09:36:28.043 21.02 Operation DB_TRANS_CONTROL executions 2 total duration 000 00:00:02.301, average 000 00:00:
01.150, maximum 000 00:00:01.197 by NS1PHUM3_HI.HIS;S.Test_DbRefresh( 10105) 10429;StressTest_OnClick: 35

09:36:28.046 21.02 Operation DB_TRANSACT_ABORT executions 8 total duration 000 00:00:08.837, average 000 00:00:
01.105, maximum 000 00:00:01.107 by
09:36:28.049 21.02 ->Table DB.MAT_GROUP total time 000 00:00:04.420
09:36:28.052 21.02 Operation BROWSER_OPEN executions 1 total duration 000 00:00:01.107, average 000 00:00:
01.107, maximum 000 00:00:01.107 by
09:36:28.054 21.02 Operation DB_CONTROL executions 3 total duration 000 00:00:03.313, average 000 00:00:01.104,
maximum 000 00:00:01.105 by NS1PHUM3_HI.HIS;S.Test_Dbmanagera( 9424) 10479;DB_CONNECT_OnClick: 109

09:36:28.057 21.02 ->Table DB.MATERIAL total time 000 00:00:16.913
09:36:28.060 21.02 Operation BROWSER_OPEN executions 1 total duration 000 00:00:01.111, average 000 00:00:
01.111, maximum 000 00:00:01.111 by
09:36:28.063 21.02 Operation DB_CONTROL executions 12 total duration 000 00:00:13.276, average 000 00:00:01.106,
maximum 000 00:00:01.129 by con 1:DBS_READ
NS1PHUM3_HI.HIS;S.Test_DbRefresh( 10105) 10429;StressTest_OnClick: 20

09:36:28.065 21.02 Operation DB_REFRESH_TABLE executions 2 total duration 000 00:00:02.526, average 000 00:00:
01.263, maximum 000 00:00:01.266 by Triggered by commit of NS1PHUM3_HI.HIS;S.Test_DbRefresh( 10105) 10429;
StressTest_OnClick: 10

09:36:28.069 21.02 =====
```

## Optimalizácia a ladenie

[Parametre databázy](#) je možné nastavovať aj v procese [D2000 CNF](#). Spojenie vytvorené akciou [SQL\\_CONNECT](#) obsahuje v konektovacom stringu nasledovné parametre:

| Názov parametra v CNF | Názov parametra v konektovacom stringu |
|-----------------------|----------------------------------------|
|-----------------------|----------------------------------------|

|                                      |           |
|--------------------------------------|-----------|
| Predpripravené spojenia              | PRE=...   |
| Maximum spojení                      | MAX=...   |
| Maximum automatických spojení        | NTC=...   |
| Rezervovaných browser spojení        | BRC=...   |
| Zatvor nepoužívané spojenia po (sek) | CLOSE=... |
| Ukoni DBManager po timeoute (min)    | WDC=...   |
| Debug                                | DEBUG     |
| Off                                  | OFF       |
| Maximum vrátených riadkov            | MR=...    |
| Prázdné operácie po dobe neinnosti   | EO=...    |

Príklad nastavenia doplnkových parametrov databázy v jej konektovacom stringu:

DEBUG;PRE=10;MAX=120;CLOSE=300;NTC=20;WDC=7;MR=1000

Databáza bude mať zapnuté vypisovanie debugovacích informácií do súboru *DBManager.log*, po štarte procesu **D2000 DBManager** sa vytvorí 10 spojení na databázu, maximálny počet spojení je 120, ak zatvorenia nepoužívaného spojenia je 300 sekúnd a maximálny počet automatických spojení je 20, vnútorný watchdog ukoní proces **D2000 DBManager**, ak je niektoré spojenie na databázu nefunkčné (zamrznuté) viac ako 7 minút.

- Z dôvodov optimalizácie rýchlosti a vzájomného neblokovania sa klientov vytvára proces **D2000 DBManager** viac ako jedno automatické spojenie na databázu. Automatické spojenia môžu byť vytvárané ihneď po štarte procesu **D2000 DBManager**, takže sú už predpripravené na použitie (keďže napr. u databázy Oracle môže trvať vytvorenie spojenia aj 1 sekundu alebo dlhšie, o sa prejaví oneskorením v skriptoch ESL). Poet predpripravených spojení je možné nastaviť pomocou procesu **D2000 CNF** na objekte typu Databáza pomocou parametra **PRE** (Preconnects). Maximálny počet spojení je možné obmedziť parametrom **MAX** (Max connects), pričom hodnota 0 obmedzenie vypína. Ak proces **D2000 DBManager** spotrebuje všetky spojenia, tak ďalšie volania snažiac sa o vytvorenie spojenia (**DB\_TRANS\_OPEN**) vrátia chybu **DBM\_MAX\_CONNECTIONS**.
- Pomocou parametra **NTC** je možné nastaviť maximálny počet spojení automaticky vytvorených. Pokiaľ parameter **NTC** nie je zadáný, tak:
  - na automatické spojenia sa použije najviac 1/4 z Max connects
  - ak je Max<4, vytvorí sa 1 automatické spojenie
  - ak je Max=0, vytvorí sa najviac 5 automatických spojení (prednastavená hodnota)

Ak už proces **D2000 DBManager** dosiahol maximálny zadáný počet automatických spojení, po ďalšej požiadavke (napr. netransakčný **SQL\_CONNECT**, **PG\_CONNECT** alebo **DB\_CONNECT**) sa nevytvorí nové spojenie, ale použije sa jedno z existujúcich. Parameter **NTC** má význam ani nie tak pre proces **D2000 DBManager**, ako pre databázu, ku ktorej sa pripája, aby nevyperal limit spojení, ktoré má databáza nastavené.

- Z dôvodu ďalšieho zrýchlenia práce, proces **D2000 DBManager** recykluje existujúce spojenia. Ak bolo spojenie uzavreté (napr. **DB\_TRANS\_CLOSE**) alebo nie je používané (vzniklo ako dôsledok **SQL\_CONNECT** a prebehlo **SQL\_DISCONNECT**), tak je označené ako voné. Voné spojenia môžu byť opätovne využité, o ušetrí sa pripájania sa k databáze a urýchli prácu skriptov. Voné spojenia, ktoré nepatria medzi predpripravené a nie sú použité, budú ukonené po uplynutí času, ktorý je daný v parametri **Close** v sekundách (predpripravené spojenia sa neukonujú). Nastavením tohto parametra na zápornú hodnotu je možné prikázať procesu **D2000 DBManager**, aby voné spojenia neukonoval - v takomto prípade bude ich množstvo iba stúpať a bude odzrkadľovať stav počas maximálnej záťaže procesu **D2000 DBManager**.
- Kvôli zrýchleniu operácií v prístroji **browser** je od verzie **D2000 8.00.009** možné vyhraďovať pre tieto operácie (otváranie browsera, prechod medzi stránkami a akcia **DB\_REFRESH\_TABLE**) spojenia pomocou parametra **BRC - Rezervované browser spojenia**. Výhodou je, že na týchto vyhradených spojeniach nebudú vykonávané iné netransakčné operácie (ktoré môžu trvať niekoľko sekúnd a viac) a akcia **DB\_REFRESH\_TABLE** ako aj prechod medzi stránkami bude rýchlejší, ako keď sú browser operácie vykonávané na automatických spojeniach. **Poznámka:** Ak je hodnota parametra **BRC - Rezervované browser spojenia** rovná nule, browser operácie sú vykonávané na automatických spojeniach, ako to bolo pred implementáciou tohto parametra.
- Ladiace a informané výpisy: Proces **D2000 DBManager** umožňuje zobraziť niekoľko úrovní ladiacich výpisov, ktoré sú popísané v **samostatnom dokumente**.
- Ukonovanie procesu **D2000 DBManager** v dôsledku "zamrznutia" niektorého spojenia: **D2000 DBManager** obsahuje vnútorný watchdog, ktorý každú minútu zisťuje, či niektoré spojenie na databázu nie je "zamrznuté" (t.j. obsluhový thread sa nevrátil z volania ODBC rozhrania). V takomto prípade zapíše informáciu do **logovacieho súboru**. Ak watchdog zistí takúto situáciu n-krát po sebe (pričom n je nastavené parametrom **WDT**), ukoní proces **D2000 DBManager**. Prednastavená hodnota **WDT=0** znamená, že proces **D2000 DBManager** sa neukonuje.
- Nastavovanie parametrov pre spojenia vytvárané pomocou príkazu **SQL\_CONNECT** s použitím tzv. konektovacieho stringu: keďže tieto spojenia sa neviažu na objekt typu Databáza, ale priamo sa odvolávajú v konektovacom stringu na DSN, nastavenie parametrov **Debug** a **Connect timeout** je možné realizovať priamo v konektovacom stringu.

Príklad: "UID=meno užívateľa;PWD=heslo užívateľa;DSN=meno DSN;ACD;Debug;CLOSE=300"

Vytvorené spojenie bude mať zapnuté vypisovanie ladiacich informácií a nepoužívané spojenie sa zavrie po 300 sekundách. Voné spojenie je možné znovu použiť iba vtedy, ak má rovnaký konektovací string ako vytvárané spojenie.

Ladiace a informané výpisy týchto spojení, ktoré sú "nezávislé" (independent) na objektoch typu Databáza, vyzerajú nasledovne:

```
08:59:45.710 Indep# 3 DSN=TestX:SQL_FETCH BEG
08:59:45.710 Indep# 3 DSN=TestX:SQL_FETCH END
```

log informuje, že "nezávislé" spojenie . 3, ktorého DSN je "TestX", vykonalo akciu SQL\_FETCH.

Ak aspoň jedno existujúce nezávislé spojenie má zapnuté vypisovanie ladiacich informácií prostredníctvom parametra **Debug** v konektovacom stringu, tak pri periodickom vypisovaní sa v logu objavujú aj informácie o nezávislých spojeniach:

```
09:14:27.752 Db Independent connects WD: 1 (1/0/0) cons:normal- 1,
```

log informuje o tom, že existuje jediné nezávislé spojenie a to je používané.  
ísla v zátvorkách udávajú poty netransakčných, transakčných a browser spojení.

## SV.\_System\_DBMdbPerf

---

Z hore uvedeného vyplýva, že v každom ase existujú pre každý objekt typu *Databáza* tri skupiny spojení (databázových pripojení a tým zároveň aj obslužných taskov), pričom ich množstvo v rámci skupiny je nejakým spôsobom obmedzené (alebo nie je podľa spôsobu nastavenia).

Možné typy spojení sú:

- **transakčné** (vznikajú v ESL volaním akcie [DB\\_TRANS\\_OPEN](#)),
- **netransakčné automatické** (NTC),
- v prípade rezervácie vyhradené **netransakčné browser spojenia** (BTC).

V ase zvýšenej záťaže, dochádza k stavu, že požiadaviek je viac ako pripojení, a preto sú tieto požiadavky zaraované do fronty podľa ich typu. Štruktúrovaná premenná "SV.\_System\_DBMdbPerf" je určená práve pre sledovanie stavu jednotlivých frontov.

Každý riadok štruktúrovanej premennej popisuje všetky tri fronty tromi veličinami, ktoré sa aktualizujú s periódou 10[s] (pozri [SV.\\_System\\_DBMdbPerf](#)).

- **ConNr** - počet pripojení v rámci skupiny,
- **CurrRq** - aktuálny počet požiadaviek na skupinu,
- **MaxRq** - maximálny počet požiadaviek na skupinu počas uplynulej periódy.

Sledovaním priebehu počtu požiadaviek vo fronte je jednoducho možné zistiť, či je počet pripojení v rámci skupiny dostatočný, ako rýchlo dokáže skupina obsluhovať požiadavky a či je potrebné navýšiť počet pripojení v rámci skupiny (navýšenie počtu pripojení nemusí v zásade zvýšiť počet spracovaných požiadaviek za jednotku času, ak je napríklad preťažený databázový server, ...).

Hodnota v stĺpci "Database" objektu SV.\_System\_DBMdbPerf identifikuje:

- meno objektu typu *Databáza* alebo
- meno objektu typu *Proces* s príponou DBM (DbManager). V tomto prípade hodnoty v riadku predstavujú sumárne počty požiadaviek a pripojení pre všetky databázy, ktoré príslušný proces obsluhuje.