

Akcie v skripte

Akcie v skriptoch

Skript je tvorený postupnosťou akcií, ktoré sa po aktivácii vykonávajú. Akcie sa zapisujú v prostredí editora akcií. Asi zápisu uvedené v hranatých zátvorkách [] sú nepovinné. Znak | (t.j. ALEBO) vyjadruje možnosť alternatívneho zápisu.

Typy akcií

Zoznam jednotlivých typov akcií. Nasledujúci zoznam popisuje základnú množinu akcií, ktoré sú prístupné v oboch aplikáciách skriptu (objekt typu [Event](#), [Aktívna schéma](#)).

Akcie možno rozdeliť do týchto základných kategórií (typov):

- [priraovacie akcie](#)
- [akcie pre prístup k databáze](#)
- [akcie pre obsluhu chybových stavov](#)
- [riadiace akcie](#)
- [akcie pre komunikáciu s operátorom](#)
- [akcie pre prácu s archívom](#)
- [akcie pre synchronizovanie vykonávania akcií skriptov](#)
- [riadenie alarmov](#)
- [akcie pre prácu so štruktúrami](#)
- [akcie pre prácu s dátovými kontajnermi](#)
- [akcie pre riadenie prístupových práv](#)
- [akcie pre prácu so zoznamami objektov](#)
- [ostatné akcie](#)

Poznámka: V systéme D2000 je možné implementovať aplikáciu funkcionality použitím jazyka JAVA. Ekvivalenty k ESL akciám sú uvedené v samostatnej on-line dokumentácii, ktorá sa nachádza v podadresári [Help programového adresára](#) systému D2000.

Priraovacie akcie

Priraovacie akcie umožňujú meniť hodnoty a odkazy na objekty:

1. objektov v systéme
 2. lokálnych premenných
- [Priradenie](#)
 - [SET WITH](#)
 - [SET AS \[DIRECT\]](#)
 - [SET BIND](#)

Akcie pre prístup k databáze

Pre prácu s tabuľkou databázy sú určené skupiny akcií, ktoré používajú rôzne spôsoby prístupu k dátam.

Ak pri práci s databázou nastane chyba, číselný kód, ktorý ju bližšie popisuje, je možné získať volaním funkcie [%GetLastExtErrorCode](#). Podrobnejšie informácie o chybe je možné získať funkciou [%GetLastExtErrorMsg](#).

Prvý spôsob využíva existenciu [kúľ](#) alebo podmienku WHERE. Podľa spôsobu zápisu akcie umožňuje zápis/ítanie jedného alebo viacerých riadkov.

- [DB_DELETE](#)
- [DB_CONNECT](#)
- [DB_DISCONNECT](#)
- [DB_INSERT](#)
- [DB_INSUPD](#)
- [DB_READ](#)
- [DB_READ_BLOB](#)
- [DB_SET_PROCESS_PARAMS](#)
- [DB_UPDATE](#)
- [DB_UPDATE_BLOB](#)

[Príklad práce s tabuľkou \(akcie DB_...\)](#)

Druhý spôsob pracuje pomocou stránkovania, pričom veľkosť stránky (počet riadkov v tabuľke) je voliteľný.

- [PG_CONNECT](#)
- [PG_DISCONNECT](#)
- [PG_READ](#)
- [PG_INSERT](#)
- [PG_DELETE](#)
- [PG_UPDATE](#)

Príklad práce s tabuľkou (akcie PG_ ...)

Tretí spôsob je modifikácia prvého. [Modifikácia spoíva v tom](#), že práca s tabuľkou nevyžaduje akcie typu CONNECT a DISCONNECT.

- DBS_DELETE
- DBS_INSERT
- DBS_INSUPD
- DBS_READ
- DBS_READ_BLOB
- DBS_UPDATE
- DBS_UPDATE_BLOB

Posledný spôsob umožňuje využiť celú škálu **SQL** príkazov pri práci s databázou.

- SQL_CONNECT
- SQL_DISCONNECT
- SQL_EXEC_DIRECT
- SQL_EXEC_PROC
- SQL_SELECT

Ľadenie databázy prostredníctvom príkazu **SELECT**.

- SQL_BINDIN
- SQL_FETCH
- SQL_FREE
- SQL_PREPARE

Príklad práce s databázou (akcie SQL_ ...)

Pre riadenie [transakčného spracovania príkazov](#) (v zmysle databázovej transakčnosti) je možné použiť nasledovné akcie:

- DB_TRANS_OPEN
- DB_TRANS_COMMIT
- DB_TRANS_ROLLBACK
- DB_TRANS_CLOSE

Príklad transakčnej práce s databázou

Vynútenie obnovy (refresh) zobrazených dát v pohľadoch užívateľa v procese [D2000 HI](#) (napríklad zobrazova typu [Browser](#)) je možné vyvolať akciou:

- DB_REFRESH_TABLE

Zaregistrovanie procedúry, ktorá sa zavola po

- zmene dát v tabuľke,
- zrušení tabuľky,
- pri prepínaní aktívnej a pasívnej inštancie procesu DbManager,
- obnove dát akciou [DB_REFRESH_TABLE](#),

je možné akciou:

- [ON DB_CHANGE](#)

[Prenos handle na databázové spojenie medzi bežiacimi ESL skriptami](#)

Konverzia a reprezentácia hodnôt v databáze

Zápis

Hodnota premennej (vo väčšine prípadov) políka lokálnej štruktúrovanej premennej je pri zápise do databázy konvertovaná tak, že ak je **platná**, je zapísaná normálnym spôsobom. Ak je **neplatná** je do databázy vložená NULL hodnota. Pri type **TEXT** toto pravidlo funguje rovnako, okrem databázy ORACLE, kde je prázdny text reprezentovaný databázovou hodnotou NULL rovnako ako neplatná hodnota.

Ľadenie

Po preĽadení hodnoty z databázy (ktorá je rôzna od NULL) prebehne konverzia hodnoty na požadovaný typ, ktorý je daný typom premennej, do ktorej je umiestnený výsledok Ľadenia. V prípade úspešnej konverzie je výsledná hodnota platná. Pri Ľadení NULL hodnoty je výsledná hodnota neplatná. Pri type **TEXT** je NULL hodnota v databáze konvertovaná na platný prázdny textový reazec. Jediná výnimka je Ľadenie prostredníctvom akcie [DB_READ/DBS_READ](#) na platforme ORACLE OCI, kedy je NULL hodnota konvertovaná na **neplatnú** hodnotu.

Tabuľka znázorňuje výsledok zápisu a Ľadenia textovej hodnoty v závislosti na databázovej platforme.

DBS_INSERT - zápis textovej hodnoty do databázy (D2Value -> DBValue).

PG_READ, BrowserRead - íťanie textovej hodnoty z databázy akciou **PG_READ** alebo do zobrazovaa **Browser** (dáta zverejnené cez **OnFetchDone**) (DBValue -> D2Value).

DB_READ - íťanie textovej hodnoty prostredníctvom akcie **DB_READ** (DBValue -> D2Value).

Databáza	DBS_INSERT	PG_READ, BrowserRead	DB_READ
Sybase 12/PostgreSQL	"Text" -> "Text"	"Text" -> "Text"	"Text" -> "Text"
dbmanager.exe	"" -> ""	"" -> ""	"" -> ""
	Invalid->NULL	NULL->""	NULL->""
ORACLE OCI	"Text" -> "Text"	"Text" -> "Text"	"Text" -> "Text"
dbmanager_ora.exe	"" -> NULL		
	Invalid->NULL	NULL->""	NULL->Invalid
ORACLE ODBC	"Text" -> "Text"	"Text" -> "Text"	"Text" -> "Text"
dbmanager.exe	"" -> NULL		
	Invalid->NULL	NULL->""	NULL->""

Akcie pre obsluhu chybových stavov

- [EXCEPTION_HANDLER](#)
- [ON ERROR](#)
- [RESUME](#)
- [RETRY](#)

Riadiace akcie

Riadiace akcie sú akcie, ktoré ovplyvujú tok riadenia (poradie vykonávania akcií) v skripte.

- [BEGIN](#)
- [CALL](#) - lokálne volanie procedúr
- [CALL](#) - vzdialené volanie procedúr
- [CALL](#) - volanie Public procedúr
- [DELAY](#)
- [DO_LOOP, EXIT_LOOP, END_LOOP](#)
- [ENABLE](#)
- [END](#)
- [END procedure](#)
- [EVENT](#)
- [GOSUB](#)
- [GOTO](#)
- [IF GOTO](#)
- [IF THEN \[ELSE\] ENDIF](#)
- [IMPLEMENTATION](#)
- [ON DB_CHANGE](#)
- [ON GOTO](#)
- [ON CHANGE](#)
- [OnExternalEvent](#)
- [PRAGMA](#)
- [PROCEDURE](#)
- [RETURN](#)
- [Riadiace funkcie](#) (bez návratovej hodnoty)
- [WAIT](#)

Akcie pre komunikáciu s operátorom

Nasledovné akcie umožňujú implementovať v skripte dialóg s operátorom, alebo ovláda schémy na pracovnej konzole operátora. Tu je vhodné použiť preddefinovanú lokálnu premennú **_FROM_HIP**. Ak je skript spustený zo schémy ([pripojený grafický objekt na ovládanie](#)), je automaticky táto lokálna premenná asociovaná s procesom, z ktorého bol skript odštartovaný. Toto umožňuje adresne pracovať s týmto procesom:

- otvára, zatvára schémy,
- posíla správy pre operátora,
- smerovať akciu QUERY.

- [CLOSE](#)
- [MESSAGE](#)
- [OPEN](#)
- [OPENEVENT](#)
- [QUERY](#)

Akcie pre synchronizovanie vykonávania akcií skriptov

Akcie GETACCESS a RELEASEACCESS umožňujú navzájom synchronizovať vykonávanie akcií v:

1. rôznych inštanciách eventov v rámci jedného procesu [D2000 Event Handler](#),
2. rôznych skriptoch aktívnych schém v rámci jedného procesu [D2000 HI](#),
3. rôznych skriptoch alebo globálne v aplikácii v inštanciách eventov alebo skriptov aktívnych schém.

Poskytujú určitú formu komunikácie medzi skriptami.

- [GETACCESS](#)
- [RELEASEACCESS](#)

Akcie pre prácu s archívom

Ľatanie/zapisovanie dát z/do archívu, mazanie dát v archíve.

- [CALCARCHEXPR](#)
- [CALCONDEMANDSTAT](#)
- [CALCSTATFUNC](#)
- [CALCSTATFUNCARR](#)
- [DELETEARCHDATA](#)
- [GETARCHARR](#)
- [GETARCHARR_TO_CNT](#)
- [GETARCHCOL](#)
- [GETARCHROW](#)
- [GETARCHSTRUCT](#)
- [GETARCHVAL](#)
- [INSERTARCHARR](#)
- [UPDATEARCHVAL](#)

Ovládanie alarmov

Ovládanie systémových alebo procesných alarmov.

- [BLOCK](#)
- [KVIT](#)
- [UNBLOCK](#)
- [UNBLOCK_ALL](#)

Akcie pre prácu so štruktúrami

Pri práci s rozsiahlymi lokálnymi štruktúrami je občas potrebné štruktúru utriediť, vložiť alebo zmazať riadok, alebo vyhľadať riadok. ESL toto umožňuje, ale tieto operácie si vyžadujú prechod celej štruktúry cyklom. Toto môže byť asovo náročné. ESL preto definuje nasledujúce akcie, ktoré uvedené inštalácie vykonávajú optimálnejšie:

- [COPYCOL](#)
- [COPYCOLIDX](#)
- [DELETE](#)
- [EXPORT_CSV](#)
- [EXPORT_CSV_TEXT](#)
- [FIND_TRUE](#)
- [GETCOLTIME](#)
- [GETROWDESC](#)
- [IMPORT_CSV](#)
- [INSERT](#)
- [SETCOLTIME](#)
- [SORT](#)
- [TRANSCOLTOROW](#)
- [TRANSROWTOCOL](#)

Akcie pre prácu s dátovými kontajnermi

Akcie umožňujú prácu so skladom hodnôt, tzv. *dátovým kontajnerom* (interná dátová štruktúra).

- [CNT_CNVTOARRAY](#)
- [CNT_CREATE](#)
- [CNT_DEBUG](#)
- [CNT_DELETE](#)
- [CNT_DESTROY](#)
- [CNT_FIND](#)
- [CNT_GETITEM](#)
- [CNT_GETKEY](#)

- [CNT_GETNR](#)
- [CNT_INSERT](#)

Akcie **CNT_GETNR**, **CNT_CNVTOARRAY** a **CNT_GETITEM** umožňujú prezeranie hodnôt z kontajnera pomocou indexu. Akcia **CNT_CNVTOARRAY** interne vytvorí pole, v ktorom sú všetky hodnoty usporiadané podľa kúľ vzostupne. Index je poradové číslo hodnoty v rámci poľa. Pole zaniká zmazaním alebo vložení hodnoty do kontajnera.

Vznik kontajnera zabezpečujú akcie **CNT_CREATE** - vone použitý kontajner, alebo **GETARCHARR_TO_CNT**. Druhý typ kontajnera je plnený stránkami, ktoré obsahujú dáta prečítané z archívu.

Takto koncipovaný prístup do archívu je efektívnejší (spotreba pamäte a rýchlosť) ako použitie akcie GETARCHARR. Pre kontajner vytvorený akciou GETARCHARR_TO_CNT sú použité len akcie CNT_GETNR, CNT_FIND a CNT_DESTROY ([príklad](#)).

[Prenos dátových kontajnerov medzi bežiacimi ESL skriptami](#)

Akcie pre riadenie prístupových práv

Akcie umožňujú nastavenie prístupových práv počas behu systému.

- [REBUILD_ACC](#)
- [RES_GROUP_DELETE](#)
- [RES_GROUP_DELETE_ALL](#)
- [RES_GROUP_INSERT](#)
- [RES_GROUP_QUERY](#)

Akcie pre prácu so zoznamami objektov

Akcie pre prácu so zoznamami objektov umožňujú:

- vytvorí zoznam objektov podľa zadaných kritérií,
- prejsť na prvú, predošlú, nasledujúcu alebo poslednú stránku zoznamu,
- prejsť na stránku určenú jej poradovým číslom,
- zistiť celkový počet objektov zoznamu (pozor! nejde o počet objektov na stránke),
- zatvoriť zoznam.

Po vytvorení zoznamu sú ihneď prístupné dáta na prvej stránke.

Taktiež prechod na prvú, predošlú, nasledujúcu, poslednú alebo poradovým číslom zadanú stránku sprístupuje dáta danej stránky.

Každý záznam v zozname objektov predstavuje jednoznačný identifikátor objektu, názov objektu, popis objektu, typ objektu, počet riadkov a počet stĺpcov.

- [LST_CREATE](#)
- [LST_CLOSE](#)
- [LST_GO_NEXT](#)
- [LST_GO_PREV](#)
- [LST_GO_LAST](#)
- [LST_GO_FIRST](#)
- [LST_GO_PAGE](#)
- [LST_GETINFO](#)

[Príklad](#) práce so zoznamami objektov (akcie LST_...).

Ostatné akcie

- [COMMAND](#)
- [COPYOBJECT](#)
- [DELETEOBJECT](#)
- [FIND_FILES](#)
- [GETOLDVAL](#)
- [GETPOINTADR](#)
- [LOG](#)
- [LOGEX](#)
- [PLAY](#)
- [READLOG](#)
- [REDIM](#)
- [RUN](#)
- [RUNEX](#)
- [SETDT_LINEOBJ](#)