```
PK \square \square \emptyset K \cdot X^a EA9 \square \square \square \square  wallet-v4-code.fc#pragma version =0.2.0; ;; Wallet smart contract with plugins (slice, int) dict_get?(cell dict, int key_len,
slice index) asm(index dict key_len) "DICTGET" "NULLSWAPIFNOT"; (cell, int) dict_add_builder?(cell dict, int key_len, slice index, builder
value) asm(value index dict key_len) "DICTADDB"; (cell, int) dict_delete?(cell dict, int key_len, slice index) asm(index dict key_len) "DICTDEL";
() recv internal(int msg value, cell in msg cell, slice in msg) impure { var cs = in msg cell.begin parse(); var flags = cs~load uint(4); ;;
int_msg_info$0 ihr_disabled:Bool bounce:Bool bounced:Bool if (flags & 1) { ;; ignore all bounced messages return (); } if (in_msg.slice_bits() < 32)
 \{ ;; ignore simple transfers return (); \} int op = in msg~load uint(32); if (op != 0x706c7567) & (op != 0x64737472) <math>\{ ;; "plug" \& "dstr" ;; ignore all plus expressions and the context of the context
messages not related to plugins return (); } slice s_addr = cs~load_msg_addr(); (int wc, int addr_hash) = parse_std_addr(s_addr); slice wc_n_address
= begin_cell().store_int(wc, 8).store_uint(addr_hash, 256).end_cell().begin_parse(); var ds = get_data().begin_parse().skip bits(32 + 32 + 256); var
plugins = ds~load_dict(); var (_, success?) = plugins.dict_get?(8 + 256, wc_n_address); if ~(success?) { ;; it may be a transfer return (); } int query_id
= in_msg\sim load_uint(64); var msg = begin_cell(); if (op == 0x706c7567) { ;; request funds (int r_toncoins, cell r_extra) = (in_msg\sim load_grams(),
in_msg~load_dict()); [int my_balance, _] = get_balance(); throw_unless(80, my_balance - msg_value >= r_toncoins); msg = msg.store_uint(0x18, 6)
 .store_slice(s_addr) .store_grams(r_toncoins) .store_dict(r_extra) .store_uint(0, 4 + 4 + 64 + 32 + 1 + 1) .store_uint(0x706c7567 | 0x80000000, 32)
 .store_uint(query_id, 64); send_raw_message(msg.end_cell(), 64); } if (op == 0x64737472) { ;; remove plugin by its request plugins~dict_delete?(8 +
256, we n address); var ds = get data().begin parse().first bits(32 + 32 + 256); set data(begin cell().store slice(ds).store dict(plugins).end cell());
64 + 32 + 1 + 1) .store_uint(0x64737472 | 0x80000000, 32) .store_uint(query_id, 64); send_raw_message(msg.end_cell(), 64); } } } ()
recv external(slice in msg) impure { var signature = in msg~load bits(512); var cs = in msg; var (subwallet id, valid until, msg seqno) =
(cs~load_uint(32), cs~load_uint(32), cs~load_uint(32)); throw_if(36, valid_until <= now()); var ds = get_data().begin_parse(); var (stored_seqno,
stored_subwallet, public_key, plugins) = (ds~load_uint(32), ds~load_uint(32), ds~load_uint(256), ds~load_dict()); ds.end_parse(); throw_unless(33,
msg seqno == stored seqno); throw unless(34, subwallet id == stored subwallet); throw unless(35, check signature(slice hash(in msg), signature,
public_key)); accept_message(); set_data(begin_cell() .store_uint(stored_seqno + 1, 32) .store_uint(stored_subwallet, 32) .store_uint(public_key, 256)
 .store_dict(plugins) .end_cell()); commit(); cs~touch(); int op = cs~load_uint(8); if (op == 0) { ;; simple send while (cs.slice_refs()) { var mode =
cs~load_uint(8); send_raw_message(cs~load_ref(), mode); } return (); ;; have already saved the storage } if (op == 1) { ;; deploy and install plugin int
plugin_workchain = cs~load_int(8); int plugin_balance = cs~load_grams(); (cell state_init, cell body) = (cs~load_ref(), cs~load_ref()); int
plugin_address = cell_hash(state_init); slice wc_n_address = begin_cell().store_int(plugin_workchain, 8).store_uint(plugin_address,
256).end_cell().begin_parse(); var msg = begin_cell() .store_uint(0x18, 6) .store_uint(4, 3).store_slice(wc n address) .store_grams(plugin balance)
 .store\_uint(4+2+1, 1+4+4+64+32+1+1+1) .store\_ref(state\_init) .store\_ref(body); send\_raw\_message(msg.end\_cell(), 3); (plugins, integrated in the state of the state of
success?) = plugins.dict_add_builder?(8 + 256, wc_n_address, begin_cell()); throw_unless(39, success?); } if (op == 2) { ;; install plugin slice
wc n address = cs\simload bits(8 + 256); int amount = cs\simload grams(); int query id = cs\simload uint(64); (plugins, int success?) =
plugins.dict_add_builder?(8 + 256, wc_n_address, begin_cell()); throw_unless(39, success?); builder msg = begin_cell() .store_uint(0x18, 6)
.store\_uint(4, 3).store\_slice(wc\_n\_address) .store\_grams(amount) .store\_uint(0, 1 + 4 + 4 + 64 + 32 + 1 + 1) .store\_uint(0x6e6f7465, 32) ;; op
.store_uint(query_id, 64); send_raw_message(msg.end_cell(), 3); } if (op == 3) { ;; remove plugin slice wc_n_address = cs~load_bits(8 + 256); int
amount = cs~load_grams(); int query_id = cs~load_uint(64); (plugins, int success?) = plugins.dict_delete?(8 + 256, wc_n_address); throw_unless(39,
64 + 32 + 1 + 1) .store_uint(0x64737472, 32) ;; op .store_uint(query_id, 64); send_raw_message(msg.end_cell(), 3); } set_data(begin_cell(), 3); }
 .store_uint(stored_seqno + 1, 32) .store_uint(stored_subwallet, 32) .store_uint(public_key, 256) .store_dict(plugins) .end_cell()); } ;; Get methods int
seqno() method_id { return get_data().begin_parse().preload_uint(32); } int get_subwallet_id() method_id { return
get_data().begin_parse().skip_bits(32).preload_uint(32); } int get_public_key() method_id { var cs = get_data().begin_parse().skip_bits(64); return
cs.preload_uint(256); } int is_plugin_installed(int wc, int addr_hash) method_id { var ds = get_data().begin_parse().skip_bits(32 + 32 + 256); var
plugins = ds~load_dict(); var (_, success?) = plugins.dict_get?(8 + 256, begin_cell().store_int(wc, 8).store_uint(addr_hash,
256).end_cell().begin_parse()); return success?; } tuple get_plugin_list() method_id { var list = null(); var ds = get_data().begin_parse().skip_bits(32)
+32+256); var plugins = ds~load_dict(); do { var (wc_n_address, _, f) = plugins~dict::delete_get_min(8 + 256); if (f) { (int wc, int addr) =
(wc_n_address\sim load_int(8), wc_n_address\sim load_uint(256)); list = cons(pair(wc, addr), list); } until (~f); return list; } PK \square \square ØK · XøäCôk7k7
stdlib.fc;; Standard library for funC;; forall X -> tuple cons(X head, tuple tail) asm "CONS"; forall X -> (X, tuple) uncons(tuple list) asm
"UNCONS"; forall X -> (tuple, X) list_next(tuple list) asm( -> 1 0) "UNCONS"; forall X -> X car(tuple list) asm "CAR"; tuple cdr(tuple list) asm
"CDR"; tuple empty_tuple() asm "NIL"; forall X -> tuple tpush(tuple t, X value) asm "TPUSH"; forall X -> (tuple, ()) ~tpush(tuple t, X value) asm
"TPUSH"; forall X -> [X] single(X x) asm "SINGLE"; forall X -> X unsingle([X] t) asm "UNSINGLE"; forall X, Y -> [X, Y] pair(X x, Y y) asm
"PAIR"; forall X, Y -> (X, Y) unpair([X, Y] t) asm "UNPAIR"; forall X, Y, Z -> [X, Y, Z] triple(X x, Y y, Z z) asm "TRIPLE"; forall X, Y, Z -> (X, Y, Z)
Z, W) untuple4([X, Y, Z, W] t) asm "4 UNTUPLE"; forall X -> X first(tuple t) asm "FIRST"; forall X -> X second(tuple t) asm "SECOND"; forall X
-> X third(tuple t) asm "THIRD"; forall X -> X fourth(tuple t) asm "3 INDEX"; forall X, Y -> X pair_first([X, Y] p) asm "FIRST"; forall X, Y -> Y
pair_second([X, Y] p) asm "SECOND"; forall X, Y, Z -> X triple_first([X, Y, Z] p) asm "FIRST"; forall X, Y, Z -> Y triple_second([X, Y, Z] p) asm
"SECOND"; forall X, Y, Z -> Z triple third([X, Y, Z] p) asm "THIRD"; forall X -> X null() asm "PUSHNULL"; forall X -> (X, ()) ~impure touch(X
x) impure asm "NOP"; int now() asm "NOW"; slice my_address() asm "MYADDR"; [int, cell] get_balance() asm "BALANCE"; int cur_lt() asm
"LTIME"; int block lt() asm "BLOCKLT"; int cell hash(cell c) asm "HASHCU"; int slice_hash(slice s) asm "HASHSU"; int string_hash(slice s) asm
"SHA256U"; int check signature(int hash, slice signature, int public key) asm "CHKSIGNU"; int check data signature(slice data, slice signature, int
public_key) asm "CHKSIGNS"; (int, int, int) compute_data_size(cell c, int max_cells) impure asm "CDATASIZE"; (int, int, int)
slice compute data size(slice s, int max cells) impure asm "SDATASIZE"; (int, int, int, int, int) compute data size?(cell c, int max cells) asm
"CDATASIZEQ NULLSWAPIFNOT2 NULLSWAPIFNOT"; (int, int, int, int) slice_compute data size?(cell c, int max cells) asm "SDATASIZEQ
NULLSWAPIFNOT2 NULLSWAPIFNOT"; ;; () throw_if(int excno, int cond) impure asm "THROWARGIF"; () dump_stack() impure asm
"DUMPSTK"; cell get_data() asm "c4 PUSH"; () set_data(cell c) impure asm "c4 POP"; cont get_c3() impure asm "c3 PUSH"; () set_c3(cont c)
impure asm "c3 POP"; cont bless(slice s) impure asm "BLESS"; () accept_message() impure asm "ACCEPT"; () set_gas_limit(int limit) impure asm
"SETGASLIMIT"; () commit() impure asm "COMMIT"; () buy_gas(int gram) impure asm "BUYGAS"; int min(int x, int y) asm "MIN"; int max(int
x, int y) asm "MAX"; (int, int) minmax(int x, int y) asm "MINMAX"; int abs(int x) asm "ABS"; slice begin parse(cell c) asm "CTOS"; ()
end parse(slice s) impure asm "ENDS"; (slice, cell) load ref(slice s) asm(-> 1 0) "LDREF"; cell preload ref(slice s) asm "PLDREF"; ;; (slice, int)
~load_int(slice s, int len) asm(s len -> 1 0) "LDIX"; ;; (slice, int) ~load_uint(slice s, int len) asm( -> 1 0) "LDUX"; ;; int preload_int(slice s, int len)
asm "PLDIX"; ;; int preload uint(slice s, int len) asm "PLDUX"; ;; (slice, slice) load bits(slice s, int len) asm(s len -> 1 0) "LDSLICEX"; ;; slice
preload bits(slice s, int len) asm "PLDSLICEX"; (slice, int) load grams(slice s) asm(-> 1 0) "LDGRAMS"; slice skip bits(slice s, int len) asm
"SDSKIPFIRST"; (slice, ()) ~skip_bits(slice s, int len) asm "SDSKIPFIRST"; slice first_bits(slice s, int len) asm "SDCUTFIRST"; slice
skip_last_bits(slice s, int len) asm "SDSKIPLAST"; (slice, ()) ~skip_last_bits(slice s, int len) asm "SDSKIPLAST"; slice slice_last(slice s, int len)
asm "SDCUTLAST"; (slice, cell) load dict(slice s) asm(-> 1 0) "LDDICT"; cell preload dict(slice s) asm "PLDDICT"; slice skip dict(slice s) asm
"SKIPDICT"; (slice, cell) load_maybe_ref(slice s) asm( -> 1 0) "LDOPTREF"; cell preload_maybe_ref(slice s) asm "PLDOPTREF"; builder
store maybe ref(builder b, cell c) asm(c b) "STOPTREF"; int cell depth(cell c) asm "CDEPTH"; int slice_refs(slice s) asm "SREFS"; int
slice_bits(slice s) asm "SBITS"; (int, int) slice_bits_refs(slice s) asm "SBITREFS"; int slice_empty?(slice s) asm "SEMPTY"; int slice_data_empty?
(slice's asm "SDEP TO LOND TO SICE refs empty? (slice s) asm "SREMPTY"; int slice depth(slice s) asm "SDEP TO LOND TO SICE refs empty? (slice s) asm "SREMPTY"; int slice depth(slice s) asm "SDEP TO LOND TO SICE REPORT OF THE SICE REPORT OF T
"BREFS" "NULLSWAPIFNOT"; (ceil, "Religion of the control of the co
end_leel (bailder b) wilder store_ref(builder b, cell c) asm(c b) "STREF"; ;; builder store_uint(builder b, rent ixt (nt len) asm(x b len)
 "Sir value index dict key len) int(builder b, int x, int len) asm(x b len) "STIX"; builder store_slice(builder b, slice s) asmint TSLICER 6 builder 1
storted grams (builder b, cell c) asm(c b) "STDICT"; (slice, slice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d d a s lice) to aid (ax s g 7 a d a s lice) to aid (ax s g 7 a d a s lice) to aid (ax s g 7 a d a s lice) to aid (ax s g 7 a d a s lice) to aid (ax s g 7 a d a s lice) to aid (ax s g 7 a d a s lice) to aid (ax s g 7 a d a s lice) to aid (ax s g 7 a d a s lice) to aid (ax s g 7 a d a s lice) to aid (ax s g 7 a d a s lice) to aid (ax s g 7 a
1°0) ("LDMSGADDR"; tuple parse_addr(slice s) asm "PARSEMSGADDR"; (int, int) parse_std_addr(slice s) asm "REWRITES TDADDR"; (int, slice)
parsev_waterradidn(sligevs)ussmell'REWRITEVARADDR"; cell idict_set_ref(cell dict, int key_len, int index, cell value) as mell'REWRITEVARADDR"; cell idict_set_ref(cell dict, int key_len, int index, cell value) as mell'REWRITEVARADDR"; cell idict_set_ref(cell dict, int key_len, int index, cell value) as mell'REWRITEVARADDR"; cell idict_set_ref(cell dict, int key_len, int index, cell value) as mell'REWRITEVARADDR"; cell idict_set_ref(cell dict, int key_len, int index, cell value) as mell'REWRITEVARADDR"; cell idict_set_ref(cell dict, int key_len, int index, cell value) as mell'REWRITEVARADDR"; cell idict_set_ref(cell dict, int key_len, int index, cell value) as mell'REWRITEVARADDR"; cell idict_set_ref(cell dict, int key_len, int index, cell value) as mell'REWRITEVARADDR"; cell idict_set_ref(cell dict, int key_len, int index, cell value) as mell'REWRITEVARADDR"; cell idict_set_ref(cell dict, int key_len, int index, cell value) as mell'REWRITEVARADDR"; cell idict_set_ref(cell dict, int key_len, int index, cell value) as mell'REWRITEVARADDR"; cell idict_set_ref(cell dict, int key_len, int index, cell value) as mell'REWRITEVARADDR"; cell idict_set_ref(cell dict, int key_len, int index, cell value) as mell'REWRITEVARADDR.
"DICTISE TO LOCALITY OF THE PROPERTY OF THE PR
udict set ref (coll dict, int key len, int index, cell value) asm(value index dict key_len) "DICTUSETREF"; (cell, dict, sets ref cell dict, int
key len, int index, cell value) asm(value index dict key len) "DICTUSETREF"; cell idict_get_ref(cell dict, int key len, int index, cell value) asm(value index dict key len) "DICTUSETREF"; cell idict_get_ref(cell dict, int key len, int index, cell value) asm(value index dict key len) "DICTUSETREF"; cell idict_get_ref(cell dict, int key len, int index, cell value) asm(value index dict key len) "DICTUSETREF"; cell idict_get_ref(cell dict, int key len, int index, cell value) asm(value index dict key len) "DICTUSETREF"; cell idict_get_ref(cell dict, int key len, int index, cell value) asm(value index dict key len) "DICTUSETREF"; cell idict_get_ref(cell dict, int key len, int index dict key len) "DICTUSETREF"; cell idict_get_ref(cell dict, int key len, int index dict key len) "DICTUSETREF"; cell idict_get_ref(cell dict, int key len, int index dict key len) "DICTUSETREF"; cell idict_get_ref(cell dict, int key len, int index dict key le
keynlen "DICTIGETOPTREE"; (cell, int) idict get ref?(cell dict, int key len, int index) asm(index dict key len, "DICTIGETOPTREE"; (cell, int) idict get ref?(cell dict, int key len, int index) asm(index dict key len, "DICTIGETOPTREE"; (cell, int) idict get ref?(cell dict, int key len, int index) asm(index dict key len, int)
uldict iggererel income die index index die in
index cell value asm(value index dict key_len) "DICTISETGETOPTREF"; (cell, cell) udict_set_get_ref(cell dict_int key_len) int index cell value)
asm (value index white the volue of the value index) asm (value index) as m (value index) as m
"DICTIDEL"; (cell, int) udict delete? (cell dict, int key_len, int index) asm(index dict key_len) "DICTUDEL"; (slice, int) idict get? (cell dict, int
keyreletecint index) asm (index calict key_len) "DICTIGET" "NULLSWAPIFNOT"; (slice, int) udict_get?(cell dict, intekey blen in thindex) lasm (index) lasm (index)
dict key len, int index) signature (slice; hash(in msg), dict key len) dict key len, int index) signature (slice; hash(in msg), dict key len); accept message();
"DIGTIDE LOCK Tild" New Les Normanders (dictinkey) (cell, slice, int) udict delete get? (cell dict, int key len, int index) as meanders (dictinkey) (less) bre_uint (stored_seq
"DICTODEL GET" "NULLSWAPIFNOT"; (cell, (slice, int)) ~idict delete get?(cell dict, int key len, int index) agriculture get.
 "DIGTIDELGE Thas "NULLSWAPIFNOT"; (cell, (slice, int)) ~udict delete get?(cell dict, int key_len, int index) asm(index_dict keys) len) _cell());
 "DICTUPEL CELT" SWAPIFNOT"; cell udict set(cell dict, int key len, int index, slice value) asm(value index diet key len) "DICTUSET";
(sell_c) ~udict_set(cell_dict_int_key_len, int index, slice value) asm(value index dict key_len) "DICTUSET"; cell_idict_set(cell_dict_int_key_len, int
index, slices aluchusin (value index dict key len) "DICTISET"; (cell, ()) ~idict_set(cell dict, int key_len, int index, slices aluchusin (value index dict) asm(value index dict)
key len address in address in address in the len interior in the len index dict key len i
dictyind key neg-slice index; slice value) asm(value index dict key len) "DICTSET"; (cell, int) udict add?(cell dictp int key; deploint int dexalls dicted into the length of the length
value index dict key len) "DICTUADD"; (cell, int) udict replace? (cell dict, int key_len, int index, slice value) as m (value index dict key len) "DICTUADD"; (cell, int) udict replace? (cell dict, int key_len, int index, slice value) as m (value index dict key len) "DICTUADD"; (cell, int) udict replace? (cell dict, int key_len, int index, slice value) as m (value index dict key len) "DICTUADD"; (cell, int) udict replace? (cell dict, int key_len, int index, slice value) as m (value index dict key len) "DICTUADD"; (cell, int) udict replace?
keynberns," PICTEVIR) EPLACE"; (cell, int) idict add? (cell dict, int key len, int index, slice value) asm(value index strict keycelen) dyD4CETIAD Dety) cell,
int) natchord replace? (cell dict, int key len, int index, slice value) asm(value index dict key len) "DICTIREPLACE" cell dict, int key len, int index, slice value) asm(value index dict key len) "DICTIREPLACE" cell dict, int key len, int index, slice value) asm(value index dict key len) "DICTIREPLACE" cell dict, int key len, int index, slice value) asm(value index dict key len) "DICTIREPLACE" cell dict, int key len, int index, slice value) asm(value index dict key len) "DICTIREPLACE" cell dict, int key len, int index, slice value) asm(value index dict key len) "DICTIREPLACE" cell dict, int key len, int index, slice value) asm(value index dict key len) "DICTIREPLACE" cell dict, int key len, int index, slice value) asm(value index dict key len) "DICTIREPLACE" cell dict, int key len, int index, slice value) asm(value index dict key len) "DICTIREPLACE" cell dict, int key len, int index, slice value) asm(value index dict key len) "DICTIREPLACE" cell dict, int key len, int index, slice value) asm(value index dict key len) "DICTIREPLACE" cell dict, int key len, int index, slice value) asm(value index dict key len) "DICTIREPLACE" cell dict, int key len, int index, slice value) asm(value index dict key len) "DICTIREPLACE" cell dict, int key len, int index, slice value index dict key len, int index dict key len, in
key denpart builders value) asm(value index dict key len) "DICTUSETB"; (cell, ()) ~udict_set_builder(celledict_cint, keye lenpint index chailder
value) asin (value in the dex dict key len) "DICTUSETB"; cell idict set builder (cell dict, int key len, int index, builder value) to a len value in the len va
key Jen "DIGTISETB"; (cell, ()) ~idict set builder(cell dict, int key len, int index, builder value) asm(value index, dict key len) "DIGTISETB"; cell
dictiset third ended little the transfer len, slice index, builder value) asm(value index dict key_len) "DICTSETB"; (celling) "DICTS
key len, sice index, builder value) asm(value index dict key_len) "DICTSETB"; (cell, int) udict_add_builder?(cell_dict_sint/key_len_int) index, builder
value) asin (value asin (value) dict key_len) "DICTUADDB"; (cell, int) udict_replace_builder? (cell dict, int key_len, int liquid kalue) (asna value) (asna value) (asna value)
index dict key len, int index, builder value index dict key, len)
"DICTAD DEB'ge (cell, eint) cidict replace builder? (cell dict, int key len, int index, builder value) asm (value index dict leav messa general dict, int key len, int index, builder value) asm (value index dict leav messa general dict.)
(cell) i (op sticox64737472) udict delete get min(cell dict, int key len) asm(-> 0 2 1 3) "DICTUREMMIN" "NULLSWAP i NOT 2 (cell) fint, slice, int))
-udies::deleteleget(smirr(c)ell dict, int key len) asm(-> 0 2 1 3) "DICTUREMMIN" "NULLSWAPIFNOT2"; (cell, vinth slickesint)egin_cell());
idlet-deletes get find (cell dict, int key len) asm(-> 0 2 1 3) "DICTIREMMIN" "NULLSWAPIFNOT2"; (cell, (introvice) site of the control of the
- interpretation of the state o
dict delicted get with the delicted get with
ukdret relations gettern axi (coll, (int, stickes int) pin_cell());
store_slice(s_addr) .store_grams(0) int key_len) asm(-> 0 2 1 3) "DICTUREMMAX" "NULLSWAPIFNOT2"; (celling unit, celling unit(0x18,
idictodelete(0get7974xfdell dict, int key len) asm(-> 0 2 1 3) "DICTIREMMAX" "NULLSWAPIFNOT2"; (cell, (intstofrecivity))
diotd delete getgenax(coll diot), int key len) asm(-> 0 2 1 3) "DICTREMMAX" "NULLSWAPIFNOT2"; (cell, (slice) slice; int) (0x6e6f7465, 32);
~dict. deliecy-geternals (cell dict) int key len) asm(-> 0 2 1 3) "DICTREMMAX" "NULLSWAPIFNOT2"; (int, slice strift) unit cuest in fight (cell dict, int
keynden) as miles 12) 0/2) c'DICTUMIN" "NULLSWAPIFNOT2"; (int, slice, int) udict_get_max?(cell dict, int key_den) as m; (remb/2) b' DICTUMIN" "NULLSWAPIFNOT2"; (int, slice, int) udict_get_max?(cell dict, int key_den) as m; (remb/2) b' DICTUMAX"
"NUESLSWAPITATOTO VALCET, int) udict_get_min_ref?(cell dict, int key_len) asm (-> 1 0 2) "DICTUMINREF"_"NOTES WAPITATOTES WAPITATOTES WAPITATOTES WAPITATOTES WAPITATOTES.
int) was ctinged max of the company 
ktow weight as the control of the co
"NULLSWAPIFNOTO" (int, cell, int) idict_get_min_ref?(cell dict, int key_len) asm (-> 1 0 2) "DICTIMINREF" (int, cell, int) idict_get_min_ref?(cell dict, int key_len) asm (-> 1 0 2) "DICTIMINREF" (int, cell, int) idict_get_min_ref?(cell dict, int key_len) asm (-> 1 0 2) "DICTIMINREF" (int, cell, int) idict_get_min_ref?(cell dict, int key_len) asm (-> 1 0 2) "DICTIMINREF" (int, cell, int) idict_get_min_ref?(cell dict, int key_len) asm (-> 1 0 2) "DICTIMINREF" (int, cell, int) idict_get_min_ref?(cell dict, int key_len) asm (-> 1 0 2) "DICTIMINREF" (int, cell, int) idict_get_min_ref?(cell dict, int key_len) asm (-> 1 0 2) "DICTIMINREF" (int, cell, int) idict_get_min_ref?(cell dict, int key_len) asm (-> 1 0 2) "DICTIMINREF" (int, cell, int) idict_get_min_ref?(cell dict, int key_len) asm (-> 1 0 2) "DICTIMINREF" (int, cell, int) idict_get_min_ref?(cell dict, int key_len) asm (-> 1 0 2) "DICTIMINREF" (int, cell, int) idict_get_min_ref?(cell dict, int) idict_get_min_ref.(cell dict, int) id
intolicity_getgineax_ref?(cell dict, int key_len) asm (-> 1 0 2) "DICTIMAXREF" "NULLSWAPIFNOT2"; (int, slipeştine) widtet_get_next?(cell dict, int
key len, int pixot) asm (pixot dict key len -> 1 0 2) "DICTUGETNEXT" "NULLSWAPIFNOT2"; (int, slice, int) int pixot asm (pixot dict key len -> 1 0 2) "DICTUGETNEXT" "NULLSWAPIFNOT2"; (int, slice, int) int pixot asm (pixot dict key len -> 1 0 2) "DICTUGETNEXT" "NULLSWAPIFNOT2"; (int, slice, int) int pixot asm (pixot dict key len -> 1 0 2) "DICTUGETNEXT" "NULLSWAPIFNOT2"; (int, slice, int) int pixot dict key len -> 1 0 2) "DICTUGETNEXT" "NULLSWAPIFNOT2"; (int, slice, int) int pixot dict key len -> 1 0 2) "DICTUGETNEXT" "NULLSWAPIFNOT2"; (int, slice, int) int pixot dict key len -> 1 0 2) "DICTUGETNEXT" "NULLSWAPIFNOT2"; (int, slice, int) int pixot dict key len -> 1 0 2) "DICTUGETNEXT" "NULLSWAPIFNOT2"; (int, slice, int) int pixot dict key len -> 1 0 2) "DICTUGETNEXT" "NULLSWAPIFNOT2"; (int, slice, int) int pixot dict key len -> 1 0 2) "DICTUGETNEXT" "NULLSWAPIFNOT2"; (int, slice, int) int pixot dict key len -> 1 0 2) "DICTUGETNEXT" "NULLSWAPIFNOT2"; (int, slice, int) int pixot dict key len -> 1 0 2) "DICTUGETNEXT" "NULLSWAPIFNOT2"; (int, slice, int) int pixot dict key len -> 1 0 2) "DICTUGETNEXT" "NULLSWAPIFNOT2"; (int, slice, int) int pixot dict key len -> 1 0 2) "DICTUGETNEXT" "NULLSWAPIFNOT2"; (int, slice, int) int pixot dict key len -> 1 0 2) "DICTUGETNEXT" "NULLSWAPIFNOT2"; (int, slice, int) int pixot dict key len -> 1 0 2) "DICTUGETNEXT" "NULLSWAPIFNOT2"; (int, slice, int) int pixot dict key len -> 1 0 2) "DICTUGETNEXT" "NULLSWAPIFNOT2"; (int, slice, int) int pixot dict key len -> 1 0 2) "DICTUGETNEXT" "NULLSWAPIFNOT2"; (int, slice, int) int pixot dict key len -> 1 0 2) "DICTUGETNEXT" "NULLSWAPIFNOT2"; (int, slice, int) int pixot dict key len -> 1 0 2) "DICTUGETNEXT" "NULLSWAPIFNOT2"; (int, slice, int) int pixot dict key len -> 1 0 2) "DICTUGETNEXT" "NULLSWAPIFNOT2"; (int, slice, int) int pixot dict key len -> 1 0 2) "DICTUGETNEXT" "NULLSWAPIFNOT2"; (int, slice, int) int pixot dict key len -> 1 0 2) "DICTUGETNEXT" "NULLSWAPIFNOT2"; (int, slice, int) int pixot dict key len -> 1 0 2) "DICTUGETNEXT" "NULLSWAPIFN
keyenkemaint (pithot) asmit (pithot)
key fear int pitored asen (bivot dict key len -> 1 0 2) "DICTUGETPREV" "NULLSWAPIFNOT2"; (int, slice, int) indicted general dict, int
key_den, int private dict key_len -> 1 0 2) "DICTUGETPREVEQ" "NULLSWAPIFNOT2"; (int, slice, int) delict eget_coext3 (oelludict) int seq
key lesignatura (int, slice, int) idicti get steel (low of the control of the con
keyeden and pivot dict key_len -> 1 0 2) "DICTIGETNEXTEQ" "NULLSWAPIFNOT2"; (int, slice, int) idet get provide diction to get the get the get the diction to get the g
key_den(bingin) asin(photolicit key_len -> 1 0 2) "DICTIGETPREV" "NULLSWAPIFNOT2"; (int, slice, int) interpreted network key_len.
| seque + 1, | jet data().begin parse() preload_uint(32); } | key sen intripivot) asm(pivot dict key len -> 1 0 2) "DICTIGETPREVEQ" "NULLSWAPIFNOT2"; cell new_dict() asm(pivot) asm(pivot) linethodnid { return
dict. serrepty? (cellic) kasm "DICTEMPTY"; (slice, slice, int) pfxdict get? (cell dict, int key len, slice key) asm (keytalictekiey pless) (). skip_bits (32). preloa
"PEXITY COMMINITY OF LESWAPIFNOT2"; (cell, int) pfxdict_set?(cell dict, int key_len, slice key, slice value) asm(xalue key dict key len) method_id { cell(); cs~touch(); int op LESWAPIFNOT2"; (cell, int) pfxdict_set?(cell dict, int key_len, slice key, slice value) asm(xalue key dict key len)
"BFXDIOT(SET (or (cell) int) ipfxelict_delete?(cell dict, int key_len, slice key) asm(key dict key_len) "PFXDICTDE delete?(cell dict, int key_len, slice key) asm(key dict key_len) "PFXDICTDE delete?(cell dict, int key_len, slice key) asm(key dict key_len) "PFXDICTDE delete?(cell dict, int key_len, slice key) asm(key dict key_len) "PFXDICTDE delete?(cell dict, int key_len, slice key) asm(key dict key_len) "PFXDICTDE delete?(cell dict, int key_len, slice key) asm(key dict key_len) "PFXDICTDE delete?(cell dict, int key_len, slice key) asm(key dict key_len) "PFXDICTDE delete?(cell dict, int key_len, slice key) asm(key dict key_len) "PFXDICTDE delete?(cell dict, int key_len, slice key) asm(key dict key_len) "PFXDICTDE delete?(cell dict, int key_len, slice key) asm(key dict key_len) "PFXDICTDE delete?(cell dict, int key_len, slice key) asm(key dict key_len) "PFXDICTDE delete?(cell dict, int key_len, slice key) asm(key dict key_len) "PFXDICTDE delete?(cell dict, int key_len, slice key) asm(key dict key_len) "PFXDICTDE delete?(cell dict, int key_len, slice key) asm(key dict key_len, slice key
 "CONFIGOPTPAR'AM", ant cell_null?(cell c) asm "ISNULL"; () raw_reserve(int amount, int mode) impure asm "KAW RESERVE"; ()
rawd reserves extragint amount, int mode) impure asm "RAWRESERVEX"; () send raw message (cell msg, dint mode) impure asm
"SENDRAW (Start Code (cell new_code) impure asm "SETCODE"; int random() impure asm "RANDU256" data code (cell new_code) impure asm "SETCODE"; int random() impure asm "RANDU256" data code (cell new_code) impure asm "SETCODE"; int random() impure asm "RANDU256" data code (cell new_code) impure asm "SETCODE"; int random() impure asm "RANDU256" data code (cell new_code) impure asm "SETCODE"; int random() impure asm "RANDU256" data code (cell new_code) impure asm "SETCODE"; int random() impure asm "RANDU256" data code (cell new_code) impure asm "SETCODE"; int random() impure asm "RANDU256" data code (cell new_code) impure asm "SETCODE"; int random() impure asm "RANDU256" data code (cell new_code) impure asm "SETCODE"; int random() impure asm "RANDU256" data code (cell new_code) impure asm "SETCODE"; int random() impure asm "RANDU256" data code (cell new_code) impure asm "SETCODE"; int random() impure asm "RANDU256" data code (cell new_code) impure asm "SETCODE"; int random() impure asm "RANDU256" data code (cell new_code) impure asm "SETCODE"; int random() impure asm "RANDU256" data code (cell new_code) impure asm "SETCODE"; int random() impure asm "RANDU256" data code (cell new_code) impure asm "SETCODE"; int random() impure asm "RANDU256" data code (cell new_code) impure asm "SETCODE"; int random() impure asm "RANDU256" data code (cell new_code) impure asm "SETCODE"; int random() impure asm "RANDU256" data code (cell new_code) impure asm "SETCODE"; int random() impure asm "RANDU256" data code (cell new_code) impure asm "SETCODE"; int random() impure asm "RANDU256" data code (cell new_code) impure asm "SETCODE"; int random() impure asm "RANDU256" data code (cell new_code) impure asm "SETCODE"; int random() impure asm "RANDU256" data code (cell new_code) impure asm "SETCODE"; int random() impure asm "SETCODE"; int random() impure asm "SETCODE" data code (cell new_code) impure a
"RAND"; () randomize(int x) impure asm "RANDSEED"; int set seed() impure asm "SETRAND"; () randomize(int x) impure asm "ADDRAND"; ()
rangonyechtin inspulsed sint (8) Littime" "ADDRAND"; builder store_coins(builder b, int x) asm "STVARUINT16" (1) inspulsed sint (8) Littime" "ADDRAND"; builder store_coins(builder b, int x) asm "STVARUINT16" (1) inspulsed sint (8) Littime" "ADDRAND"; builder store_coins(builder b, int x) asm "STVARUINT16" (1) inspulsed sint (8) Littime" "ADDRAND"; builder store_coins(builder b, int x) asm "STVARUINT16" (1) inspulsed sint (8) Littime" "ADDRAND"; builder store_coins(builder b, int x) asm "STVARUINT16" (2) Inspulsed sint (8) Littime" (1) Inspulsed sint (8) Littime (1) Inspulsed sint (8) Littime (1) Inspulsed sint (8) Littime (1) Littime 
> LIVARUINT 16"; int equal slices (slice a, slice b) asm "SDEQ"; int builder null? (builder b) asm "ISNULL"; builder store builder (builder to,
```

beither from castor STBR to PK | | | | ØK·XaÈA9© | © | wallet-v4-code.fcPK | | | ØK·XøäCôk7k7 Ø stdlib: FFR for Lett (). bei From parse ()); return