File created: 16-May-90 16:27:39 **{DSK}<usr>local>lde>lispcore>sources>ERROR-RUNTIME.;2**

changes to: (VARS ERROR-RUNTIMECOMS)

previous date: 5-Feb-88 15:54:20 {DSK}<usr>local>lde>lispcore>sources>ERROR-RUNTIME.;1

Read Table: XCL

Package: INTERLISP

Format: XCCS


```
(RPAQQ ERROR-RUNTIMECOMS
       ((COMS

;;; Internal functions.

              (FUNCTIONS SI::CONDITION-CASE-ERROR CONDITION-HANDLER CONDITION-REPORTER %PRINT-CONDITION
                     CONDITIONS::%RESTART-PRINTER CONDITIONS::%RESTART-DEFAULT-REPORTER REPORT-CONDITION
                     CONDITION-PARENT)
              (VARIABLES *CONDITION-HANDLER-BINDINGS* *PROCEED-CASES*)
              (FUNCTIONS CHECK-TYPE-FAIL ECASE-FAIL ASSERT-FAIL)
              (FUNCTIONS MAKE-INTO-CONDITION RAISE-SIGNAL DEFAULT-HANDLE-CONDITION DEFAULT-PROCEED-REPORTER
                     CONDITIONS::DEFAULT-RESTART-REPORTER DEFAULT-PROCEED-TEST TEST-PROCEED-CASE
                     WALK-PROCEED-CASES SI::INVOKE-ACTUAL-RESTART))
        (COMS

;;; Exported symbols. Anything here that's not in CL should be in XCL.

              (VARIABLES CONDITIONS:*BREAK-ON-SIGNALS* *BREAK-ON-WARNINGS* XCL:*CURRENT-CONDITION*)
              (FUNCTIONS MAKE-CONDITION SIGNAL CL:ERROR CL:CERROR CL:WARN CL:BREAK CONDITIONS:INVOKE-DEBUGGER)
              (FUNCTIONS CONDITIONS:FIND-RESTART CONDITIONS:COMPUTE-RESTARTS CONDITIONS:INVOKE-RESTART
                     CONDITIONS:INVOKE-RESTART-INTERACTIVELY))
        (PROP FILETYPE ERROR-RUNTIME)))


;;; Internal functions.

(CL:DEFUN SI::CONDITION-CASE-ERROR (SI::REAL-SELECTOR SI::POSSIBILITIES)
   (CL:ERROR "Unexpected selector in ~S." 'CONDITION-CASE SI::REAL-SELECTOR SI::POSSIBILITIES))


(DEFMACRO CONDITION-HANDLER (XCL::CONDITION-TYPE)
   `(GETPROP ,XCL::CONDITION-TYPE '%CONDITION-HANDLER))


(DEFMACRO CONDITION-REPORTER (XCL::CONDITION-TYPE)
   `(GETPROP ,XCL::CONDITION-TYPE '%CONDITION-REPORTER))


(CL:DEFUN %PRINT-CONDITION (CONDITION STREAM LEVEL)
   (DECLARE (IGNORE LEVEL))
   (CL:IF *PRINT-ESCAPE*
       (CL:FORMAT STREAM "#<Condition ~S @ ~O,~O>" (CL:TYPE-OF CONDITION)
              (\\HILOC CONDITION)
              (\\LOLOC CONDITION))
       (REPORT-CONDITION CONDITION STREAM)))


(CL:DEFUN CONDITIONS::%RESTART-PRINTER (CONDITIONS:RESTART STREAM CONDITIONS::LEVEL)
   (CL:IF *PRINT-ESCAPE*
       (CL:FUNCALL CL::%DEFAULT-PRINT-FUNCTION CONDITIONS:RESTART STREAM CONDITIONS::LEVEL)
       (LET ((CONDITIONS::REPORTER (OR (CONDITIONS::RESTART-REPORT CONDITIONS:RESTART)
                                       (CONDITIONS::DEFAULT-RESTART-REPORT (CONDITIONS:RESTART-NAME
                                                                              CONDITIONS:RESTART))
                                       (CL:RETURN-FROM CONDITIONS::%RESTART-PRINTER (
                                                                          CONDITIONS::DEFAULT-RESTART-REPORTER
                                                                             CONDITIONS:RESTART STREAM))
                                       )))
            (CL:IF (CL:STRINGP CONDITIONS::REPORTER)
                (CL:PRINC CONDITIONS::REPORTER STREAM)
                (CL:FUNCALL CONDITIONS::REPORTER STREAM)))))


(CL:DEFUN CONDITIONS::%RESTART-DEFAULT-REPORTER (CONDITIONS:RESTART STREAM)
   (CL:FUNCALL (CONDITIONS::DEFAULT-RESTART-REPORT (CONDITIONS:RESTART-NAME CONDITIONS:RESTART))
          CONDITIONS:RESTART STREAM))


(CL:DEFUN REPORT-CONDITION (CONDITION STREAM)
   (CL:DO* ((TYPE (CL:TYPE-OF CONDITION)
                  (CONDITION-PARENT TYPE))
            (REPORTER (CONDITION-REPORTER TYPE)
                   (CONDITION-REPORTER TYPE)))
           ((NULL TYPE)
```

```
            (CL:BREAK "No report function found for ~S." CONDITION))
       (CL:WHEN REPORTER
           (RETURN (CL:IF (CL:STRINGP REPORTER)
                       (CL:PRINC REPORTER STREAM)
                       (CL:FUNCALL REPORTER CONDITION STREAM))))))


(CL:DEFUN CONDITION-PARENT (TYPE)
   (LET ((PARENT (GETSUPERTYPE TYPE)))
        (CL:IF (EQ PARENT 'CL::STRUCTURE-OBJECT)
            NIL
            PARENT)))


(CL:DEFVAR *CONDITION-HANDLER-BINDINGS* NIL
   "Condition handler binding stack")


(CL:DEFVAR *PROCEED-CASES* NIL
   "Active proceed case stack")


(CL:DEFUN CHECK-TYPE-FAIL (PROCEEDABLE PLACE VALUE DESIRED-TYPE MESSAGE)
   (CONDITIONS:RESTART-CASE (CL:ERROR 'XCL:TYPE-MISMATCH :NAME PLACE :VALUE VALUE :EXPECTED-TYPE DESIRED-TYPE
                                 :MESSAGE MESSAGE)
          (STORE-VALUE (NEW)
                 :REPORT
                 (LAMBDA (STREAM)
                   (CL:FORMAT STREAM "Change the value of ~A" PLACE))
                 :INTERACTIVE
                 (LAMBDA NIL
                   (CL:FORMAT *QUERY-IO* "Enter a new value to store into ~A: " PLACE)
                   (LIST (CL:EVAL (CL:READ *QUERY-IO*))))
                 :FILTER
                 (LAMBDA NIL
                   (AND PROCEEDABLE (TYPEP XCL:*CURRENT-CONDITION* 'XCL:TYPE-MISMATCH)))
                 NEW)))


(CL:DEFUN ECASE-FAIL (PROCEEDABLE PLACE VALUE SELECTORS)
   (CONDITIONS:RESTART-CASE (CL:IF (EQL PLACE VALUE)
                                (CL:ERROR "~S is ~?." VALUE "~#[wrong~;not ~S~;neither ~S nor ~S~:;not~@{~#[~;
                                       or~] ~S~^,~}~]" SELECTORS)
                                (CL:ERROR "The value of ~S, ~S,~&is ~?." PLACE VALUE "~#[wrong~;not ~S~;neither
                                       ~S nor ~S~:;not~@{~#[~; or~] ~S~^,~}~]" SELECTORS))
          (STORE-VALUE (V)
                 :FILTER
                 (LAMBDA NIL PROCEEDABLE)
                 :REPORT
                 (LAMBDA (STREAM)
                   (CL:FORMAT STREAM "Change the value of ~A" PLACE))
                 :INTERACTIVE
                 (LAMBDA NIL
                   (CL:FORMAT *QUERY-IO* "Enter a new value to store into ~A: " PLACE)
                   (LIST (CL:EVAL (CL:READ *QUERY-IO*))))
                 V)))


(CL:DEFUN ASSERT-FAIL (STRING &REST ARGS)
   (PROCEED-CASE (CL:ERROR 'XCL:ASSERTION-FAILED :FORMAT-STRING STRING :FORMAT-ARGUMENTS ARGS)
          (CONDITIONS:CONTINUE NIL :REPORT "Re-test assertion")))


(CL:DEFUN MAKE-INTO-CONDITION (DATUM DESIRED-TYPE ARGS)
   (CL:ETYPECASE DATUM
       (CONDITION DATUM)
       (CL:SYMBOL (CL:IF (CL:SUBTYPEP DATUM 'CONDITION)
                      (CL:APPLY 'MAKE-CONDITION DATUM ARGS)
                      (CL:ERROR "~S is not a condition type." DATUM)))
       (STRING (MAKE-CONDITION DESIRED-TYPE :FORMAT-STRING DATUM :FORMAT-ARGUMENTS ARGS))))


(CL:DEFUN RAISE-SIGNAL (CONDITION)
   (CL:WHEN (TYPEP CONDITION CONDITIONS:*BREAK-ON-SIGNALS*)
          (CL:BREAK "Condition ~S is about to be signalled." CONDITION))
   (LET ((*CONDITION-HANDLER-BINDINGS* *CONDITION-HANDLER-BINDINGS*))
        (CL:FLET ((TRY-TO-HANDLE (CONDITION TYPE-SPEC HANDLER)
                         (CL:MACROLET ((WITHOUT-HANDLERS (&BODY BODY)
                                              `(LET (*CONDITION-HANDLER-BINDINGS*)
                                                    ,@BODY)))
                                (CL:WHEN (PROCEED-CASE (WITHOUT-HANDLERS (TYPEP CONDITION TYPE-SPEC))
                                               (PROCEED NIL :REPORT "Skip the bad handler binding" NIL))
                                       (CL:FUNCALL HANDLER CONDITION)))))
             (WHILE *CONDITION-HANDLER-BINDINGS*
                DO (LET ((BINDING (CL:POP *CONDITION-HANDLER-BINDINGS*)))
                         (IF (EQ (CL:FIRST BINDING)
                                 :MULTIPLE-HANDLER-BINDINGS)
```

```
                              THEN (CL:POP BINDING)
                                   (WHILE BINDING DO (TRY-TO-HANDLE CONDITION (CL:POP BINDING)
                                                                              (CL:POP BINDING)))
                             ELSE (TRY-TO-HANDLE CONDITION (CAR BINDING)
                                                           (CDR BINDING))))
                  FINALLY (DEFAULT-HANDLE-CONDITION CONDITION)))
        CONDITION))


(CL:DEFUN DEFAULT-HANDLE-CONDITION (CONDITION)
   (CL:DO ((TYPE (CL:TYPE-OF CONDITION)
                 (CONDITION-PARENT TYPE)))
          ((NULL TYPE))
       (LET ((HANDLER (CONDITION-HANDLER TYPE)))
            (CL:WHEN HANDLER (CL:FUNCALL HANDLER CONDITION)))))


(CL:DEFUN DEFAULT-PROCEED-REPORTER (PC STREAM)
   (CL:FORMAT STREAM "Proceed-type: ~A" (PROCEED-CASE-NAME PC)))


(CL:DEFUN CONDITIONS::DEFAULT-RESTART-REPORTER (CONDITIONS:RESTART STREAM)
   (CL:FORMAT STREAM "Restart type: ~A" (CONDITIONS:RESTART-NAME CONDITIONS:RESTART)))


(DEFMACRO DEFAULT-PROCEED-TEST (XCL::PROCEED-TYPE)
   `(GETPROP ,XCL::PROCEED-TYPE '%DEFAULT-PROCEED-TEST))


(CL:DEFUN TEST-PROCEED-CASE (PC &AUX FILTER)
   (COND
      ((CL:SETF FILTER (CONDITIONS::RESTART-TEST PC))
       (CL:FUNCALL FILTER))
      ((CONDITIONS:RESTART-NAME PC)
       (CL:IF (CL:SETF FILTER (DEFAULT-PROCEED-TEST (CONDITIONS:RESTART-NAME PC)))
           (CL:FUNCALL FILTER)
          T))
      (T                                                          ; unnamed proceed case with no explicit test
         T)))


(CL:DEFUN WALK-PROCEED-CASES (PROCEED-CASES PRED)
   (CL:FLET ((CONVERT-PROCEED-CASE (PC BLIP)
                    (CL:IF (NULL (CONDITIONS::RESTART-TAG PC))
                        (LET ((NEW (CONDITIONS::COPY-RESTART PC)))
                             (CL:SETF (CONDITIONS::RESTART-TAG NEW)
                                      BLIP)
                             NEW)
                        PC)))
         (CL:DO ((TAIL PROCEED-CASES (CDR TAIL)))
                ((NULL TAIL)
                 NIL)
             (CL:MACROLET ((PROCESS-THING (THING BLIP)
                                  `(LET ((PC (CONVERT-PROCEED-CASE ,THING ,BLIP)))
                                        (CL:WHEN (CL:FUNCALL PRED PC)
                                                 (CL:RETURN-FROM WALK-PROCEED-CASES PC)))))
                   (LET ((OBJECT (CAR TAIL)))
                        (CL:IF (CL:CONSP OBJECT)
                            (CL:DO ((THINGS OBJECT (CDR THINGS)))
                                   ((NULL THINGS))
                                (PROCESS-THING (CAR THINGS)
                                               TAIL))
                            (PROCESS-THING OBJECT TAIL)))))))


(CL:DEFUN SI::INVOKE-ACTUAL-RESTART (SI::RESTART SI::ARGUMENTS)
   (COND
      ((NULL (CONDITIONS::RESTART-FUNCTION SI::RESTART))
       (CL:THROW (CONDITIONS::RESTART-TAG SI::RESTART)
              (CONS (CONDITIONS::RESTART-SELECTOR SI::RESTART)
                    SI::ARGUMENTS)))
      ((EQ (CONDITIONS::RESTART-SELECTOR SI::RESTART)
           'SI::COMPLEX-RESTART-MARKER)
       (CL:APPLY (CONDITIONS::RESTART-FUNCTION SI::RESTART)
              SI::ARGUMENTS))
      (T (CL:ERROR "Malformed restart object ~S." SI::RESTART))))
```

;;; Exported symbols. Anything here that's not in CL should be in XCL.

```
(CL:DEFVAR CONDITIONS:*BREAK-ON-SIGNALS* NIL)


(CL:DEFVAR *BREAK-ON-WARNINGS* NIL
   "If true, calls to WARN will cause a break as well as logging the warning.")
```

```
(CL:DEFVAR XCL:*CURRENT-CONDITION* NIL
   "The condition currently being signalled")


(CL:DEFUN MAKE-CONDITION (TYPE &REST XCL::SLOT-INITIALIZATIONS)
   "Create a condition object of the specified type."
   (CL:APPLY (CL::STRUCTURE-CONSTRUCTOR TYPE)
          XCL::SLOT-INITIALIZATIONS))


(CL:DEFUN SIGNAL (XCL::DATUM &REST XCL::ARGS)
   (LET ((XCL:*CURRENT-CONDITION* (MAKE-INTO-CONDITION XCL::DATUM 'SIMPLE-CONDITION XCL::ARGS)))
        (RAISE-SIGNAL (CL:SETQ *LAST-CONDITION* XCL:*CURRENT-CONDITION*))
        (CL:RETURN-FROM SIGNAL XCL:*CURRENT-CONDITION*)))


(CL:DEFUN CL:ERROR (CL::DATUM &REST CL::ARGS)
   ;; In Xerox Common Lisp, as with Interlisp, errors may not enter the debugger if they are simple, defined by ENTER-DEBUGGER-P
   (LET ((XCL:*CURRENT-CONDITION* (MAKE-INTO-CONDITION CL::DATUM 'SIMPLE-ERROR CL::ARGS)))
        (RAISE-SIGNAL (CL:SETQ *LAST-CONDITION* XCL:*CURRENT-CONDITION*))
                                                                ; may just unwind.
        (RESETLST
            (LET ((PRINTMSG T)
                  (ERRORPOS (FIND-DEBUGGER-ENTRY-FRAME 'CL:ERROR)))
                 (DECLARE (CL:SPECIAL PRINTMSG ERRORPOS))
                 (RESETSAVE NIL (LIST 'RELSTK ERRORPOS))
                 (COND
                    ((NULL (ENTER-DEBUGGER-P HELPDEPTH ERRORPOS XCL:*CURRENT-CONDITION*))
                     ;; says not to enter debugger
                     (COND
                        (PRINTMSG                               ; print message if no break is to occur.
                                  (CL:PRINC XCL:*CURRENT-CONDITION* *ERROR-OUTPUT*)))
                     (ERROR!)))
                 (DEBUGGER :CONDITION XCL:*CURRENT-CONDITION* :AT (STKNAME ERRORPOS))))))


(CL:DEFUN CL:CERROR (CL::PROCEED-FORMAT-STRING CL::DATUM &REST CL::ARGUMENTS)
   (LET ((XCL:*CURRENT-CONDITION* (MAKE-INTO-CONDITION CL::DATUM 'SIMPLE-ERROR CL::ARGUMENTS)))
        (PROCEED-CASE (CL:ERROR XCL:*CURRENT-CONDITION*)
               (CONDITIONS:CONTINUE NIL :REPORT (CL:APPLY (FUNCTION CL:FORMAT)
                                                          T CL::PROCEED-FORMAT-STRING CL::ARGUMENTS)
                      (CL:RETURN-FROM CL:CERROR XCL:*CURRENT-CONDITION*)))))


(CL:DEFUN CL:WARN (CL::DATUM &REST CL::ARGUMENTS)
   (LET ((XCL:*CURRENT-CONDITION* (MAKE-INTO-CONDITION CL::DATUM 'SIMPLE-WARNING CL::ARGUMENTS)))
        (CL:UNLESS (TYPEP XCL:*CURRENT-CONDITION* 'WARNING)
            (CL:CERROR "Signal and report the condition anyway" 'XCL:TYPE-MISMATCH :NAME 'CL::CONDITION :VALUE
                  XCL:*CURRENT-CONDITION* :EXPECTED-TYPE 'WARNING))
        (CL:WHEN *BREAK-ON-WARNINGS* (CL:BREAK "Warning: ~A" XCL:*CURRENT-CONDITION*))
        (PROCEED-CASE (PROGN (RAISE-SIGNAL XCL:*CURRENT-CONDITION*)
                             (CL:FORMAT *ERROR-OUTPUT* "~&Warning: ~A~%" XCL:*CURRENT-CONDITION*)
                             NIL)
              (CONDITIONS:MUFFLE-WARNING NIL :REPORT "Don't print the warning" NIL))))


(CL:DEFUN CL:BREAK (&OPTIONAL (CL::FORMAT-STRING "Break.")
                              &REST CL::FORMAT-ARGUMENTS)
   ;; Want to try and get some indication of which break you're returning from.
   (PROCEED-CASE (CONDITIONS:INVOKE-DEBUGGER (MAKE-CONDITION 'SIMPLE-CONDITION :FORMAT-STRING
                                                            CL::FORMAT-STRING :FORMAT-ARGUMENTS CL::FORMAT-ARGUMENTS))
          (CONDITIONS:CONTINUE NIL :REPORT "Return from BREAK" (CL:RETURN-FROM CL:BREAK NIL))))


(CL:DEFUN CONDITIONS:INVOKE-DEBUGGER (CONDITION)
   ;; always enter debugger, never return
   (DEBUGGER :CONDITION CONDITION))


(CL:DEFUN CONDITIONS:FIND-RESTART (CONDITIONS::IDENTIFIER)
   (CL:FLET ((CONDITIONS::SAME-RESTART (CONDITIONS::IDENTIFIER CONDITIONS::PROTOTYPE)))
          (CL:ETYPECASE CONDITIONS::IDENTIFIER
              (NULL (CL:ERROR "~S is an invalid argument to ~S;~%    use ~S instead" NIL
                           'CONDITIONS:FIND-RESTART 'CONDITIONS:COMPUTE-RESTARTS))
              (CONDITIONS:RESTART (WALK-PROCEED-CASES *PROCEED-CASES*
                                                     #'(CL:LAMBDA (CONDITIONS:RESTART)
                                                              (AND (OR (EQ CONDITIONS::IDENTIFIER CONDITIONS:RESTART)
                                                                       (AND (CONDITIONS::RESTART-TAG CONDITIONS::IDENTIFIER)
                                                                            (EQ (CONDITIONS:RESTART-NAME
                                                                                      CONDITIONS::IDENTIFIER)
                                                                                (CONDITIONS:RESTART-NAME CONDITIONS:RESTART)
                                                                                ))
```

```
                                                    (EQ (CONDITIONS::RESTART-TAG
                                                            CONDITIONS::IDENTIFIER)
                                                        (CONDITIONS::RESTART-TAG CONDITIONS:RESTART)
                                                        )
                                                    (EQ (CONDITIONS::RESTART-SELECTOR
                                                            CONDITIONS::IDENTIFIER)
                                                        (CONDITIONS::RESTART-SELECTOR
                                                            CONDITIONS:RESTART))
                                                    (EQ (CONDITIONS::RESTART-TEST
                                                            CONDITIONS::IDENTIFIER)
                                                        (CONDITIONS::RESTART-TEST CONDITIONS:RESTART
                                                            ))
                                                    (EQ (CONDITIONS::RESTART-REPORT
                                                            CONDITIONS::IDENTIFIER)
                                                        (CONDITIONS::RESTART-REPORT
                                                            CONDITIONS:RESTART))
                                                    (EQ (CONDITIONS::RESTART-INTERACTIVE-FN
                                                            CONDITIONS::IDENTIFIER)
                                                        (CONDITIONS::RESTART-INTERACTIVE-FN
                                                            CONDITIONS:RESTART))
                                                    (EQ (CONDITIONS::RESTART-FUNCTION
                                                            CONDITIONS::IDENTIFIER)
                                                        (CONDITIONS::RESTART-FUNCTION
                                                            CONDITIONS:RESTART))))
                                                (TEST-PROCEED-CASE CONDITIONS:RESTART)))))
            (CL:SYMBOL (WALK-PROCEED-CASES *PROCEED-CASES* #'(CL:LAMBDA (CONDITIONS:RESTART)
                                                                (AND (EQ (CONDITIONS:RESTART-NAME
                                                                            CONDITIONS:RESTART)
                                                                        CONDITIONS::IDENTIFIER)
                                                                    (TEST-PROCEED-CASE
                                                                        CONDITIONS:RESTART)))))))


(CL:DEFUN CONDITIONS:COMPUTE-RESTARTS ()
   (LET ((CONDITIONS::FOUND NIL))
        (WALK-PROCEED-CASES *PROCEED-CASES* #'(CL:LAMBDA (CONDITIONS:RESTART)
                                                (CL:WHEN (CL:CATCH 'SI::SKIP-PROCEED-CASE (
                                                                                            TEST-PROCEED-CASE
                                                                                                CONDITIONS:RESTART
                                                                                                    ))
                                                    (CL:PUSH CONDITIONS:RESTART CONDITIONS::FOUND))
                                                NIL))
        (CL:NREVERSE CONDITIONS::FOUND)))


(CL:DEFUN CONDITIONS:INVOKE-RESTART (CONDITIONS:RESTART &REST CONDITIONS::ARGUMENTS)
   (LET ((CONDITIONS::R (CONDITIONS:FIND-RESTART CONDITIONS:RESTART)))
        (CL:IF (NULL CONDITIONS::R)
            (CL:ERROR 'XCL:BAD-PROCEED-CASE :NAME CONDITIONS:RESTART)
            (SI::INVOKE-ACTUAL-RESTART CONDITIONS::R CONDITIONS::ARGUMENTS))))


(CL:DEFUN CONDITIONS:INVOKE-RESTART-INTERACTIVELY (CONDITIONS:RESTART)
   (LET ((CONDITIONS::R (CONDITIONS:FIND-RESTART CONDITIONS:RESTART)))
        (CL:IF (NULL CONDITIONS::R)
            (CL:ERROR 'XCL:BAD-PROCEED-CASE :NAME CONDITIONS:RESTART)
            (LET ((CONDITIONS::I-FN (CONDITIONS::RESTART-INTERACTIVE-FN CONDITIONS:RESTART)))
                 (SI::INVOKE-ACTUAL-RESTART CONDITIONS::R (CL:IF CONDITIONS::I-FN (CL:FUNCALL CONDITIONS::I-FN))
                       )))))

(PUTPROPS ERROR-RUNTIME FILETYPE CL:COMPILE-FILE)

(PUTPROPS ERROR-RUNTIME COPYRIGHT ("Venue & Xerox Corporation" 1986 1987 1988 1990))
```

FUNCTION INDEX

VARIABLE INDEX

MACRO INDEX

PROPERTY INDEX