Mapping Specification for DWG/DXF (MSD)

AutoLISP Code Samples

March, 2008

The code samples contained herein are intended as learning tools and demonstrate basic coding techniques used to implement MSD. They were created with AutoLISP™ using the Microsoft Notepad text editor. Error handling has been omitted for clarity.

This document assumes the reader has a working knowledge of reading and writing to the DWG and or DXF format. All samples are provided "as-is" and without warranty of any kind, expressed or implied. ESRI does not assume any responsibility for their successful operation.

Contents

Reporting the PRJ string embedded in the DWG

```
;;DEFINE THE COMMAND-LINE ALIAS
(defun c:CSLIST ()(ListESRICoordinateSystem))

;;DEFINE THE FUNCTION

;;VARIABLES
;;$entPrj = coordinate system xrecord object (association list)

(defun ListESRICoordinateSystem ( / $entPrj)

   ;ECHO COMMAND TO SCREEN
   (prompt "\nLIST COORDINATE SYSTEM")
   (princ)

   ;GET PRJ OBJECT
   (setq $entPrj (dictsearch (namedobjdict)"ESRI_PRJ"))

   ;LIST CONTENTS OF PRJ OBJECT TO TEXT SCREEN
   (if (/= $entPrj nil)

      (progn

         ;GET THE WKT STRING
         (cdr (assoc 1 $entPrj))

      );_ progn

      (progn

         ;PROMPT USER IF PRJ OBJECT DOES NOT EXIST
         (prompt " (0 found)")
         (princ)

      );_progn

   );_ if LIST CONTENTS OF PRJ OBJECT TO TEXT SCREEN

);_ defun main function
```

Working with Feature Classes

Writing a Feature Class to the Named Object Dictionary

```
;;DEFINE THE COMMAND-LINE ALIAS
(defun c:FCNEW ()(DefineESRIFeatureClass))

;;DEFINE THE FUNCTION

;;VARIABLES
;;$features = features dictionary name (string)
;;$featureClass = feature class dictionary name (string)
;;$featureType = feature type xrecord name (string)
;;$featureQuery = feature query xrecord name (string)

;;$objFeatures  = root-level dictionary (entity name)
;;$objFeatureClass = feature class dictionary entry (entity)
```

```
;;$objFeatureType = feature type dictionary entry (entity name)
;;$objFeatureQuery = feature query dictionary entry (entity name)
;;$objList = dictionary entry list (list construct)
;;$objResB = data pair (for xrecords)
;;$objXrecd = xrecord object
;;$objDict = dictionary object

(defun DefineESRIFeatureClass ( / $features $featureClass $featureType
                                  $featureQuery $objFeatures $objFeatureClass
                                  $objFeatureType $objFeatureQuery $objList
                                  $objResB $objXrecd $objDict
                                )

   ;ECHO COMMAND TO SCREEN
   (prompt "\nCREATE FEATURE CLASS")

   ;SET NAME FOR ROOT-LEVEL FEATURES DICTIONARY
   (setq $features "ESRI_FEATURES")

   ;FIND/ADD THE ROOT-LEVEL DICTIONARY
   (if (= (dictsearch (namedobjdict) $features) nil)

      ;CREATE NEW DICTIONARY IF NOT FOUND
      (progn

         ;CREATE A DICTIONARY ENTRY LIST
         (setq $objList(list '(0 . "DICTIONARY") '(100 . "AcDbDictionary")))

         ;CREATE A DICTIONARY OBJECT
         (setq $objDict (entmakex $objList))

         ;ADD THE ESRI_FEATURES ROOT-LEVEL DICTIONARY TO THE NOD
         (setq $objFeatures (dictadd (namedobjdict) $features $objDict))

      );_ progn CREATE NEW DICTIONARY IF NOT FOUND

      ;ELSE GET EXISTING DICTIONARY
      (setq $objFeatures (handent(cdr (assoc 5
                                        (dictsearch (namedobjdict) $features)
                                     )
                                  )
                         )
      )

   );_if FIND/ADD THE ROOT-LEVEL DICTIONARY

   ;SET NAME FOR FEATURE CLASS DICTIONARY
   (setq $featureClass "MyFeatureClass")

   ;FIND/ADD A FEATURE CLASS DICTIONARY
   (if (= (dictsearch $objFeatures $featureClass) nil)

      ;CREATE FEATURE CLASS IF NOT FOUND
      (progn

         ;CREATE A DICTIONARY ENTRY LIST
         (setq $objList(list '(0 . "DICTIONARY") '(100 . "AcDbDictionary")))

         ;CREATE A DICTIONARY OBJECT
         (setq $objDict (entmakex $objList))

         ;ADD THE NEW FEATURE CLASS DICTIONARY TO THE ESRI_FEATURES DICTIONARY
         (setq $objFeatureClass (dictadd $objFeatures $featureClass $objDict))
```

```
        ;PRINT FEATURE CLASS NAME TO SCREEN
        (prompt (strcat "\nFeature class: " $featureClass " (new) \n" ))

    );progn

    (progn

    ;ELSE GET THE EXISTING FEATURE CLASS DICTIONARY
    (setq $objFeatureClass (handent(cdr (assoc 5
                                            (dictsearch $objFeatures $featureClass)
                                        )
                                    )
                            )
    )

    ;PRINT FEATURE CLASS NAME TO SCREEN
    (prompt (strcat "\nFeature class: " $featureClass " (already exists) \n" ))

    );_ progn

);_ if FIND/ADD A FEATURE CLASS DICTIONARY

;;ADD/REPLACE FEATURE TYPE XRECORD

;SET OBJECT NAME TO FEATURETYPE
(setq $FeatureType "FeatureType")

;CONSTRUCT THE FEATURE DATA PAIR
(setq $objResB (cons 1 "Polyline"))

;MAKE THE XRECORD ASSOCIATION LIST
(setq $objList (append (list '(0 . "Xrecord")'(100 . "AcDbXrecord"))
                       (list $objResB)
               )
)

;MAKE THE XRECORD OBJECT
(setq $objXrecd (entmakex $objList))

;REMOVE THE EXISTING FEATURETYPE XRECORD OBJECT
(if (dictsearch $objFeatureClass $FeatureType)
       (dictremove $objFeatureClass $FeatureType)
)

;ADD THE FEATURE TYPE XRECORD OBJECT TO THE FEATURE CLASS DICTIONARY
(setq $objFeatureType (dictadd $objFeatureClass $FeatureType $objXrecd))

;;ADD/REPLACE FEATURE QUERY XRECORD

;SET OBJECT NAME TO FEATUREQUERY
(setq $featureQuery "FeatureQuery")

;CONSTRUCT THE LAYER NAME DATA PAIR
(setq $objResB (cons 8 "Layer1,Layer2"))

;MAKE THE XRECORD ASSOCIATION LIST
(setq $objList (append (list '(0 . "Xrecord")'(100 . "AcDbXrecord"))
                       (list $objResB)
               )
)

;MAKE THE XRECORD OBJECT
```

```
    (setq $objXrecd (entmakex $objList))

    ;REMOVE EXISTING XRECORD OBJECT
    (if (dictsearch $objFeatureClass $featureQuery)
        (dictremove $objFeatureClass $featureQuery)
    )

    ;ADD THE NEW FEATURE QUERY XRECORD OBJECT TO THE FEATURE CLASS DICTIONARY
    (setq $objFeatureQuery (dictadd $objFeatureClass $featureQuery $objXrecd))

    ;PRINT LAYERS, LINETYPES AND COLOR TO SCREEN
    (prompt (strcat "Query:"
                    "\n Layers = " (cdr $objResB)
                    "\n Linetypes = not specified"
                    "\n Color = not specified \n"
            )
    )
    (princ)

);_ defun main function
```

Adding Fields to a Feature Class

This code sample requires the feature class *MyFeatureClass* created by the previous code sample.

```
;;DEFINE THE COMMAND-LINE ALIAS
(defun c:FCFLDS ()(AddFieldsESRIFeatureClass))

;;DEFINE THE FUNCTION

;;VARIABLES
;;$features = name for root-level features dictionary (string)
;;$featureClass = name for features class dictionary (string)
;;$featureAttributes  = name for attributes dictionary (string)
;;$objFeatures  = root-level dictionary entry (entity name)
;;$objFeatureClass = feature class dictionary entry (entity name)
;;$objDict = dictionary object (entity name)
;;$objFeatureAttributes = esri_attributes extension dictionary (entity name)
;;$fieldnameInt = attribute field name, short integer (string)
;;$fieldnameLong = attribute field name, long integer (string)
;;$fieldnameReal = attribute field name, real number (string)
;;$fieldnameText = attribute field name, text (string)
;;$objResB = data pair (list construct)
;;$objList = dictionary entry list (list construct)
;;$objXrecd = xrecord object (entity name)
;;$objFieldInt = object added to the dictionary (entity name)
;;$objFieldLong = object added to dictionary (entity name)
;;$objFieldReal = object added to dictionary (entity name)
;;$objFieldText = object added to dictionary (entity name)

(defun AddFieldsESRIFeatureClass ( / $features $featureClass $featureAttributes
                                     $objFeatures $objFeatureClass $objDict
                                     $objFeatureAttributes $fieldnameInt
                                     $fieldnameLong $fieldnameReal $fieldnameText
                                     $objResB $objList $objXrecd $objFieldInt
                                     $objFieldLong $objFieldReal $objFieldText
                                   )

    ;ECHO COMMAND TO SCREEN
    (prompt "ADD ATTRIBUTES TO FEATURE CLASS")
    (princ)
```

```
;SET OBJECT NAME FOR THE ROOT-LEVEL DICTIONARY
(setq $features "ESRI_FEATURES")

;SET OBJECT NAME FOR THE FEATURE CLASS
(setq $featureClass "MyFeatureClass")

;SET OBJECT NAME FOR THE ATTRIBUTE DICTIONARY
(setq $featureAttributes "ESRI_Attributes")

;FIND THE ROOT-LEVEL DICTIONARY
(if (dictsearch (namedobjdict) $features)

   ;CREATE/ADD ATTRIBUTES TO THE FEATURE CLASS DICTIONARY
   (progn

      ;GET THE ENTITY NAME FOR THE ROOT-LEVEL DICTIONARY
      (setq $objFeatures
         (handent(cdr (assoc 5 (dictsearch (namedobjdict) $features))))
      )

      ;GET THE FEATURE CLASS DICTIONARY
      (if (dictsearch $objFeatures $featureClass)

         ;GET THE FEATURE CLASS DICTIONARY
         (progn
            ;GET THE ENTITY NAME OF THE FEATURE CLASS DICTIONARY
            (setq $objFeatureClass
               (handent(cdr (assoc 5 (dictsearch $objFeatures $featureClass))))
            )

            ;FIND/ADD THE ATTRIBUTES DICTIONARY
            (if (= (dictsearch $objFeatureClass $featureAttributes) nil)

               ;CREATE A NEW DICTIONARY IF NOT FOUND
               (progn

                  ;CREATE A DICTIONARY ENTRY LIST
                  (setq $objList
                     (list '(0 . "DICTIONARY") '(100 . "AcDbDictionary"))
                  )

                  ;CREATE A DICTIONARY OBJECT FOR ATTRIBUTES
                  (setq $objDict (entmakex $objList))

                  ;ADD THE ATTRIBUTES DICTIONARY TO THE FEATURE CLASS DICTIONARY
                  (setq $objFeatureAttributes
                     (dictadd $objFeatureClass $featureAttributes $objDict)
                  )
               );_ progn

               ;ELSE GET THE EXISTING ATTRIBUTES DICTIONARY
               (setq $objFeatureAttributes
                  (handent(cdr (assoc 5
                                 (dictsearch $objFeatureClass $featureAttributes)
                               )
                         )
                  )
               )

            );_ FIND/ADD THE ATTRIBUTES DICTIONARY

            ;SET ATTRIBUTE FIELD NAMES
            (setq $fieldnameInt "IntField")
```

```
(setq $fieldnameLong "LongField")
(setq $fieldnameReal "RealField")
(setq $fieldnameText "TextField")

;SET ATTRIBUTE DEFAULT VALUES
(setq $fieldDefaultInt 99)
(setq $fieldDefaultLong 900000)
(setq $fieldDefaultReal 99.999)
(setq $fieldDefaultText "SAMPLE STRING")

;CREATE/ADD FIELD DEFINITION FOR SHORT INTEGER
(if (= (dictsearch $objFeatureAttributes $fieldnameInt) nil)

    (progn

       ;CONSTRUCT THE LIST
       (setq $objResB (cons 70 $fieldDefaultInt))

       ;CREATE THE XRECORD LIST
       (setq $objList
          (append (list '(0 . "Xrecord")'(100 . "AcDbXrecord"))
                  (list $objResB)
          )
       )

       ;MAKE THE XRECORD OBJECT
       (setq $objXrecd (entmakex $objList))

       ;ADD THE XRECORD OBJECT TO THE ATTRIBUTES DICTIONARY
       (setq $objFieldInt
          (dictadd $objFeatureAttributes $fieldnameInt $objXrecd)
       )

       ;INFORM USER
       (prompt (strcat "\n Fieldname = " $fieldnameInt
                       "\n    Data type = Short Integer"
                       "\n    Default value = " (itoa $fieldDefaultInt)

               )
       )

    );_ progn

    (prompt (strcat "\n Fieldname " $fieldnameInt " already exists"))

);_ if CREATE/ADD FIELD DEFINITION FOR SHORT INTEGER

;CREATE/ADD A FIELD DEFINITION FOR LONG INTEGER
(if (= (dictsearch $objFeatureAttributes $fieldnameLong) nil)

    (progn

       ;CONSTRUCT THE DATA LIST
       (setq $objResB (cons 90 $fieldDefaultLong))

       ;CREATE THE XRECORD LIST
       (setq $objList
          (append (list '(0 . "Xrecord")'(100 . "AcDbXrecord"))
                  (list $objResB)
          )
       )

       ;MAKE THE XRECORD OBJECT
```

```
            (setq $objXrecd (entmakex $objList))

            ;ADD THE XRECORD OBJECT TO THE ATTRIBUTES DICTIONARY
            (setq $objFieldInt
               (dictadd $objFeatureAttributes $fieldnameLong $objXrecd)
            )

            ;INFORM USER
            (prompt (strcat "\n Fieldname = " $fieldnameLong
                            "\n    Data type = Long Integer"
                            "\n    Default value = "
                                    (itoa $fieldDefaultLong)
                    )
            )

        );_ progn

        (prompt (strcat "\n Fieldname " $fieldnameLong " already exists"))

    );_ if CREATE/ADD A FIELD DEFINITION FOR LONG INTEGER

    ;CREATE/ADD A FIELD DEFINITION REAL NUMBER
    (if (= (dictsearch $objFeatureAttributes $fieldnameReal) nil)

        (progn

            ;CONSTRUCT THE LIST
            (setq $objResB (cons 40 $fieldDefaultReal))

            ;MAKE THE XRECORD LIST
            (setq $objList
               (append (list '(0 . "Xrecord")'(100 . "AcDbXrecord"))
                       (list $objResB))
               )

            ;MAKE THE XRECORD OBJECT
            (setq $objXrecd (entmakex $objList))

            ;ADD THE XRECORD OBJECT TO THE ATTRIBUTES DICTIONARY
            (setq $objFieldInt
               (dictadd $objFeatureAttributes $fieldnameReal $objXrecd)
            )

            ;INFORM USER
            (prompt (strcat "\n Fieldname = " $fieldnameReal
                            "\n    Data type = Real Number"
                            "\n    Default value = "
                               (rtos $fieldDefaultReal 2 3)
                    )
            )
        );_ progn

        (prompt (strcat "\n Fieldname " $fieldnameReal " already exists"))

    );_ if CREATE/ADD A FIELD DEFINITION REAL NUMBER

    ;CREATE/ADD A FIELD DEFINITION FOR TEXT
    (if (= (dictsearch $objFeatureAttributes $fieldnameText) nil)

        (progn

            ;CONSTRUCT THE LIST
            (setq $objResB (cons 1 $fieldDefaultText))
```

```
                        ;MAKE THE XRECORD LIST
                        (setq $objList
                           (append (list '(0 . "Xrecord")'(100 . "AcDbXrecord"))
                                   (list $objResB)
                           )
                        )

                        ;MAKE THE XRECORD OBJECT
                        (setq $objXrecd (entmakex $objList))

                        ;ADD THE XRECORD OBJECT TO THE ATTRIBUTES DICTIONARY
                        (setq $objFieldInt
                           (dictadd $objFeatureAttributes $fieldnameText $objXrecd)
                        )

                        ;INFORM USER
                        (prompt (strcat "\n Fieldname = " $fieldnameText
                                        "\n    Data type = Text"
                                        "\n    Default value = " $fieldDefaultText
                                )
                        )

                     );_ progn

                     (prompt (strcat "\n Fieldname " $fieldnameText " already exists"))

                  );_ if CREATE/ADD A FIELD DEFINITION FOR TEXT

               (princ)

               );_ progn GET THE FEATURE CLASS DICTIONARY

               ;ELSE INFORM USER FEATURE CLASS NOT FOUND
               (progn

                  (prompt (strcat " " $featureClass " not found"))
                  (princ)

               );_ progn

            );_ if GET THE FEATURE CLASS DICTIONARY

         );_ progn CREATE/ADD ATTRIBUTES TO THE FEATURE CLASS DICTIONARY

         ;ELSE PROMPT USER NOT FOUND
         (progn

            (prompt (strcat " Dictionary " $features " not found"))
            (princ)

         );_ progn

      );_ if FIND THE ROOT-LEVEL DICTIONARY

);_ defun main function
```

Reporting the Feature Classes saved in the Named Object Dictionary

```
;;DEFINE THE COMMAND=LINE ALIAS
```

```
(defun c:FCLIST ()(ListESRIFeatures))
;;DEFINE THE MAIN FUNCTION

;;VARIABLES
;;$objFeatures  = esri features dictionary (entity name)
;;$entFeature  = esri features dictionary (assocation list)
;;$objFeatureClass = feature class dictionary (entity name)
;;$featureClass = feature class name (string)
;;$featureType = feature type (string)
;;$entFeatureQuery = feature query xrecord (association list)
;;$entPointer = entity pointer (integer)
;;$entCounter = feature class counter (integer)
;;$entTotal = total entities in association list (integer)
;;$entCode = result buffer group code (integer)
;;$entLayer = query layers
;;$entLineType = query line type
;;$entColor = query color

(defun ListESRIFeatures ( / $objFeatures $entFeature $objFeatureClass $featureClass
                           $featureType $entFeatureQuery $entPointer $entTotal
                           $entCode $entLayer $entLineType $entColor $entCounter
                         )

   ;ECHO COMMAND TO SCREEN
   (prompt "LIST ALL FEATURE CLASSES")

   ;GET THE ENTRIES FROM THE ROOT-LEVEL FEATURES DICTIONARY
   (if (/= (setq $entFeatures (dictsearch (namedobjdict) "ESRI_FEATURES")) nil)

      (progn

         ;GET THE ENTITY NAME FOR FEATURES DICTIONARY OBJECT
         (setq $objFeatures (handent(cdr (assoc 5 $entFeatures))))

         ;GET TOTAL ENTITIES IN THE FEATURES DICTIONARY
         (setq $entTotal (length $entFeatures))

         ;SET POINTER
         (setq $entPointer 0)

         ;SET COUNTER
         (setq $entCounter 0)

         ;STEP THROUGH EACH ENTITY
         (while (< $entPointer $entTotal)

            ;GET DATA IF DXF GROUP CODE = 3
            (if (= (setq $entCode (car (nth $entPointer $entFeatures))) 3)

               (progn

                  ;INCREMENT THE FEATURE CLASS COUNTER
                  (setq $entCounter (+ $entCounter 1))

                  ;GET THE FEATURE CLASS NAME
                  (setq $featureClass (cdr (nth $entPointer $entFeatures)))

                  ;PRINT FEATURE CLASS NAME TO SCREEN
                  (prompt (strcat "\nFeature class: " $featureClass "\n" ))

                  ;GET THE ENTITY NAME FOR THE FEATURE CLASS DICTIONARY
                  (setq $objFeatureClass
                     (handent(cdr (assoc 5 (dictsearch $objFeatures $featureClass))))
```

```
                )

                ;GET THE FEATURE TYPE
                (setq $featureType
                   (cdr (assoc 1 (dictsearch $objFeatureClass "FeatureType")))
                )

                ;PRINT FEATURE TYPE TO SCREEN
                (if (= $featureType nil)
                       (prompt "Type: not found \n")
                       (prompt (strcat "Type: " $featureType "\n"))
                );_ if

                ;GET FEATURE QUERY
                (if (= (setq $entFeatureQuery
                           (dictsearch $objFeatureClass "FeatureQuery")
                       )
                     nil
                    )

                   (prompt "Query: not found \n")

                   (progn
                      ;GET LAYERS, LINETYPES AND COLOR
                      (if (= (setq $entLayer (cdr (assoc 8 $entFeatureQuery))) nil)
                             (setq $entLayer "not specified")
                      )
                      (if (= (setq $entLineType
                                 (cdr (assoc 6 $entFeatureQuery))) nil)
                            (setq $entLineType "not specified")
                      )

                      (if (= (setq $entColor (cdr (assoc 62 $entFeatureQuery))) nil)
                             (setq $entColor "not specified")
                      )

                      ;PRINT LAYERS, LINETYPES AND COLOR TO SCREEN
                      (prompt (strcat "Query:"
                                      "\n Layers = " $entLayer
                                      "\n Linetypes = " $entLineType
                                      "\n Color = " $entColor "\n"
                              )
                      )
                   );_ progn

                );if_ GET FEATURE QUERY

                ;CALL SUBROUTINE TO GET ATTRIBUTES
                ($getattributes $objFeatureClass)

             );_ progn

          );_ if GET DATA IF DXF GROUP CODE = 3

         ;INCREMENT THE POINTER
         (setq $entPointer (+ $entPointer 1))

       );_ while STEP THROUGH EACH ENTITY

      (if (= $entCounter 0) (prompt " (0 found)"))

    );_ progn
```

```
      (prompt " (0 found)")

   );_ if GET THE ESRI FEATURES DICTIONARY

      (princ)

);defun

;; DEFINE SUBROUTINE TO GET ATTRIBUTES

;;PARAMETERS
;;$objFeatureClass = feature class dictionary (entity name)

;;VARIABLES
;;$objAttributes = attributes dictionary (entity name)
;;$entAttributes = attributes dictionary (association list)
;;$entTotalAtt = total entities in attribute dictionary
;;$entPointerAtt = entity pointer (integer)
;;$entCounterAtt = attribute counter (integer)
;;$entCodeAtt = entity group code (integer)
;;$entXrecord = attribute xrecord (association list)
;;$entField = field data (result buffer)
;;$fieldName = fieldname (string)
;;$fieldType = field type (string)
;;$fieldValue = field value (varies)
;;$listAttributes = attribute collection (list array)
;;$listHeaders = ui headers (list array)
;;$listPointer = ui print list pointer

(defun $getAttributes ($objFeatureClass / $objAttributes $entAttributes $entTotalAtt
                                          $entPointerAtt $entCounterAtt $entCodeAtt
                                          $entXrecord $entField $fieldName $fieldType
                                          $fieldValue $listHeaders $listPointer
$listAttributes
                      )

   ;GET ATTRIBUTES
   (if (= (setq $entAttributes (dictsearch $objFeatureClass "ESRI_Attributes")) nil)

      (prompt "Attributes: (0 found) \n")

      (progn

         ;GET NUMBER OF ENTITIES IN THE ATTRIBUTES DICTIONARY
         (setq $entTotalAtt (length $entAttributes))

         ;GET THE ENTITY NAME OF THE ATTRIBUTE DICTIONARY
         (setq $objAttributes (handent (cdr (assoc 5 $entAttributes))))

         ;SET POINTER
         (setq $entPointerAtt 0)

         ;SET COUNTER
         (setq $entCounterAtt 0)

         ;PROCESS EACH ENTITY
         (while (< $entPointerAtt $entTotalAtt)

            ;GET ENTITY GROUP CODE
            (setq $entCodeAtt (car (nth $entPointerAtt $entAttributes)))

            ;GET THE OBJECT IF GROUP CODE = 3 ATTRIBUTE
            (if (= $entCodeAtt 3)
```

```
        (progn

           ;INCREMENT THE ATTRIBUTE COUNTER
           (setq $entCounterAtt (+ $entCounterAtt 1))

           ;GET THE FIELD NAME
           (setq $fieldName (cdr (nth $entPointerAtt $entAttributes)))

           ;GET THE XRECORD
           (setq $entXrecord (dictsearch $objAttributes $fieldName))

           ;GET THE DATA TYPE AND VALUE
           (cond

              ((/= (setq $entField (assoc 1 $entXrecord)) nil)
                   (setq $fieldType "Text"
                         $fieldValue (cdr $entField)
                   )
              )
              ((/= (setq $entField (assoc 40 $entXrecord)) nil)
                   (setq $fieldType "Real Number"
                         $fieldValue (rtos (cdr $entField))
                   )
              )
              ((/= (setq $entField (assoc 70 $entXrecord)) nil)
                   (setq $fieldType "Short Integer"
                         $fieldValue (itoa (cdr $entField))
                   )
              )
              ((/= (setq $entField (assoc 90 $entXrecord)) nil)
                   (setq $fieldType "Long Integer"
                         $fieldValue (itoa (cdr $entField))
                   )
              )

           );_ cond GET THE DATA TYPE AND VALUE

           ;COLLECT ATTRIBUTES IN A LIST ARRAY
           (if (= $entCounterAtt 1)

              (setq $listAttributes
                 (list (list $fieldName $fieldType $fieldValue))
              )
              (setq $listAttributes
                 (append $listAttributes
                    (list (list $fieldName $fieldType $fieldValue))
                 )
              )
           );_ if COLLECT ATTRIBUTES IN A LIST ARRAY
        );_ progn
     );_ if GET THE OBJECT IF GROUP CODE = 3 ATTRIBUTE

     ;INCREMENT THE POINTER
     (setq $entPointerAtt (+ $entPointerAtt 1))

  );_ while PROCESS EACH ENTITY

  ;SET HEADER ARRAY
  (setq $listHeaders '("\n Fieldname = "
                       "\n   Data type = "
                        "\n   Default value = "
                       )
```

```
         )

         ;SET PRINTER COUNTER
         (setq $listPointer 0)

         ;PRINT ATTRIBUTE LIST TO SCREEN
         (prompt (strcat "Attributes:(" (itoa $entCounterAtt) " found)"))

         (while (< $listPointer $entCounterAtt)

            (prompt (strcat (nth 0 $listHeaders)
                            (car (nth $listPointer $listAttributes))
                            (nth 1 $listHeaders)
                            (cadr (nth $listPointer $listAttributes))
                            (nth 2 $listHeaders)
                            (caddr (nth $listPointer $listAttributes))
                    )
            )

            (setq $listPointer (+ $listPointer 1))
            (princ)

         );_ while PRINT ATTRIBUTES TO SCREEN

      );_ progn

   );if_ GET ATTRIBUTES

);_defun SUBROUTINE TO GET ATTRIBUTES
```

Selecting Entities Belonging a Feature Class

This code sample requires the feature class *MyFeatureClass* created by previous code samples as well as entities on *layer1* and *layer2*.

```
;;DEFINE THE COMMAND-LINE ALIAS
(defun c:FCSEL ()(SelectESRIFeatureClass))

;;DEFINE THE FUNCTION

;;VARIABLES
;;$features = name for root-level features dictionary (string)
;;$featureClass = name for features class dictionary (string)
;;$featureAttributes  = name for attributes dictionary (string)
;;$featureType = feature type (string)
;;$objFeatures  = root-level dictionary entry (entity name)
;;$objFeatureClass = feature class dictionary entry (entity name)
;;$entFeatureType = feature type xrecord (association list)
;;$qryLayers = query layers data pair (Result buffer)
;;$gisPoint = entity filter for point
;;$gisPolyline = entity filter for polyline
;;$gisPolygon = entity filter polygon
;;$gisAnnotation = entity filter for annoation
;;$gisMultiPatch = entity filter for multipatch
;;$gisFilter = entity filter

(defun SelectESRIFeatureClass ( / $features $featureClass $featureAttributes
                                  $featureType $objFeatures $objFeatureClass
                                  $entFeatureType $qryLayers $gisPoint
                                  $gisPolyline $gisPolygon $gisAnnotation
                                  $gisMultiPatch $gisFilter
                                )
```

```
;SET THE OBJECT NAME FOR ROOT-LEVEL DICTIONARY
(setq $features "ESRI_FEATURES")

;SET OBJECT NAME FOR THE FEATURE CLASS DICTIONARY
(setq $featureClass "MyFeatureClass")

;SET OBJECT NAME FOR THE ATTRIBUTE DICTIONARY
(setq $featureAttributes "ESRI_Attributes")

;PREDEFINE THE ARCGIS FEATURE TYPE ENTITY FILTERS
(setq $gisPoint (list '(-4 . "<or")
                      '(0 . "POINT")'(0 . "INSERT")'(0 . "SHAPE")
                      '(0 . "HATCH")'(0 . "PROXY")
                    '(-4 . "or>")
              )
)

(setq $gisPolyline (list '(-4 . "<or")
                         '(0 . "ARC") '(0 . "CIRCLE") '(0 . "ELLIPSE")
                         '(0 . "LINE")'(0 . "MLINE")'(0 . "*POLYLINE")
                         '(0 . "RAY")'(0 . "SPLINE")'(0 . "XLINE")
                         '(0 . "TRACE")'(0 . "SOLID")'(0 . "FACE")
                       '(-4 . "or>")
                 )
)

(setq $gisPolygon (list '(-4 . "<or")
                         '(0 . "CIRCLE")'(0 . "SOLID")'(0 . "ELLIPSE")
                         '(0 . "FACE")'(-4 . "<and")'(0 . "*POLYLINE")
                         '(70 . 1)'(-4 . "and>")
                            '(-4 . "<and")
                               '(0 . "MLINE")'(70 . 1)
                            '(-4 . "and>")
                       '(-4 . "or>")
                )
)

(setq $gisAnnotation (list '(-4 . "<or")
                           '(0 . "TEXT")'(0 .  "MTEXT") '(0 .  "ATTRIBUTE")
                           '(0 .  "ATTDEF")
                         '(-4 . "or>")
                    )
)

(setq $gisMultiPatch $gisPolyline)

;GET THE ROOT-LEVEL FEATURES DICTIONARY
(if (dictsearch (namedobjdict) $features)

   (progn

      ;GET THE ENTITY NAME OF THE FEATURES DICTIONARY
      (setq $objFeatures
         (handent(cdr (assoc 5 (dictsearch (namedobjdict) $features))))
      )

      ;GET THE FEATURE CLASS
      (if (dictsearch $objFeatures $featureClass)

         ;SET THE APPROPRIATE FILTERS THEN SELECT THE QUALIFYING MEMBERS
         (progn
```

```
        ;GET THE ENTITY NAME OF THE FEATURE CLASS DICTIONARY
        (setq $objFeatureClass
           (handent(cdr (assoc 5 (dictsearch $objFeatures $featureClass))))
        )

        ;GET THE FEATURETYPE
        (if (setq $entFeatureType(dictsearch $objFeatureclass "FeatureType"))

           (setq $featureType (cdr (assoc 1 $entFeatureType)))
           (setq $featureType "")

        );_ if GET/SET THE FEATURETYPE

        ;GET THE FEATUREQUERY
        (if (setq $objFeatureQuery
               (dictsearch $objFeatureclass "FeatureQuery")
            )
           (setq $qryLayers (assoc 8 $objFeatureQuery))
           (setq $qryLayers "")
        );_ if

        ;GET THE APPROPRIATE TYPE-FILTER AND ADD THE LAYER QUERY
        (cond

           ((= $featureType "Point")
            (setq $gisFilter (append $gisPoint (list $qryLayers)))
           )

           ((= $featureType "Polyline")
            (setq $gisFilter (append $gisPolyline (list $qryLayers)))
           )

           ((= $featureType "Polygon")
            (setq $gisFilter (append $gisPolygon (list $qryLayers)))
           )

           ((= $featureType "Multipatch")
            (setq $gisFilter (append $gisMultiPatch (list $qryLayers)))
           )

           ((= $featureType "Annotation")
            (setq $gisFilter (append $gisAnnotation (list $qryLayers)))
           )

           (t nil)

        );_ cond

        ;CREATE THE SELECTION SET
        (setq $ObjSSet (ssget "X" $gisFilter))

        ;RUN THE COMMAND "SELECT"
        (prompt (strcat " Selecting entities defined as " $featureClass "\n"))
        (command ".SELECT" $ObjSSet pause)

    );_ progn SET THE APPROPRIATE FILTERS THEN SELECT THE QUALIFYING MEMBERS

    ;ELSE PROMPT USER THAT FEATURE CLASS NOT FOUND
    (progn

       (prompt (strcat " feature class " $featureClass " not found"))
       (princ)
```

```
            );_ progn

          );_ if GET THE FEATURE CLASS

       );_ prog

       ;ELSE PROMPT USER NOT FOUND
       (progn

          (prompt (strcat " Dictionary " $features " not found"))
          (princ)

       );progn

   );_ if GET THE ROOT-LEVEL DICTIONARY

);_ defun main function
```

Deleting a Feature Class from the Named Object Dictionary

```
;;DEFINE THE COMMAND-LINE ALIAS
(defun c:FCDEL ()(DeleteESRIFeatureClass))

;;DEFINE THE FUNCTION

;;VARIABLES
;;$features = features dictionary name (string)
;;$featureClass = feature class dictionary name (string)
;;$objFeatures  = root level features dictionary (entity name)
;;$objFeatureClass = feature class dictionary (entiy name)

(defun DeleteESRIFeatureClass ( / $features $featureClass $objFeatures
$objFeatureClass)

   ;ECHO COMMAND TO SCREEN
   (prompt "\nDELETE FEATURE CLASS")

   ;SET OBJECT NAME FOR ROOT-LEVEL DICTIONARY
   (setq $features "ESRI_FEATURES")

   ;SET OBJECT NAME FOR FEATURE CLASS
   (setq $featureClass "MyFeatureClass")

   ;FIND THE FEATURES DICTIONARY
   (if (dictsearch (namedobjdict) $features)

      ;GET THE FEATURES DICTIONARY
      (progn

         (setq $objFeatures
            (handent(cdr (assoc 5 (dictsearch (namedobjdict) $features))))
         )

         ;GET THE FEATURE CLASS DICTIONARY
         (if (dictsearch $objFeatures $featureClass)

            (progn

               ;DELETE THE FEATURE CLASS DICTIONARY
               (setq $objFeatureClass (dictremove $objFeatures $featureClass))
               (prompt (strcat "\n" $featureClass " (deleted) \n"))
```

```
               (princ)

            );_ progn

            ;ELSE PROMPT USER NOT FOUND
            (progn

               (prompt (strcat "\n" $featureClass " (not found) \n"))
               (princ)

            );_ progn

         );_ if GET THE FEATURE CLASS

      );_ progn GET THE ESRI_FEATURES DICTIONARY

      ;ELSE INFORM USER THAT DICTIONTARY NOT FOUND
      (progn

         (prompt (strcat "\n" $features " (not found)"))
         (prompt (strcat "\n" $featureClass " (not found) \n"))
         (princ)

      );_ progn

   );_ if FIND THE FEATURES DICTIONARY

);_ defun main function
```

Working with Attributes on Entities

Attaching Attributes to an Entity

To attach attributes to a specific entity, follow the example that adds fields to a feature class. The only difference is that you use the *ESRI_Attributes* dictionary in the entity's extension dictionary instead of the feature class dictionary.

Reporting Attributes Attached to Entities

```
;;DEFINE THE COMMAND-LINE ALIAS
(defun c:ATLIST()(ListESRIAttributes))

;;VARIABLES
;;$objSelect = selected object (entity name)
;;$objDictionary = extension dictionary object (entity name)
;;$objAttributes = attributes dictionary (entity name)
;;$entAttributes = attributes dictionary (association list)
;;$entSelect = select object (association list)
;;$entType = entity type (string)
;;$entTotalAtt = total entities in attribute dictionary
;;$entPointerAtt = entity pointer (integer)
;;$entCounterAtt = attribute counter (integer)
;;$entCodeAtt = entity group code (integer)
;;$entXrecord = attribute xrecord (association list)
;;$entField = field data (result buffer)
;;$fieldName = fieldname (string)
```

```
;;$fieldType = field type (string)
;;$fieldValue = field value (varies)
;;$listAttributes = attribute collection (list array)
;;$listPointer = ui print list pointer

;;DEFINE THE FUNCTION

(defun ListESRIAttributes( / $objSelect $objAttributes $entAttributes
                             $entSelect $entType $entTotalAtt
                             $entPointerAtt $entCounterAtt $entCodeAtt
                             $entXrecord $entField $fieldName $fieldType
                             $fieldValue $listAttributes $listPointer
                         )

   ;ECHO COMMAND TO SCREEN
   (prompt "\nLIST ATTRIBUTES FROM FEATURE ENTITY")

   ;SELECT THE ENTITY
   (if (/= (setq $objSelect(entsel)) nil)

      (progn

         ;GET ENTITY LIST
         (setq $entSelect (entget (car $objSelect)))

         ;GET ENTITY TYPE
         (setq $entType (cdr (assoc 0 $entSelect )))

         ;GET THE ESRI EXTENSION DICTIONARY USING DXF CODE 360 HARD-OWNER ID/HANDLE
         (if (and
                (/= (setq $objDictionary (cdr (assoc 360 $entSelect))) nil)
                (/= (setq $entAttributes (dictsearch $objDictionary "ESRI_Attributes")
                )
             nil)
            )

            (progn

               ;GET NUMBER OF ENTITIES IN THE ATTRIBUTES DICTIONARY
               (setq $entTotalAtt (length $entAttributes))

               ;GET THE ENTITY NAME OF THE ATTRIBUTE DICTIONARY
               (setq $objAttributes (handent (cdr (assoc 5 $entAttributes))))

               ;SET POINTER
               (setq $entPointerAtt 0)

               ;SET COUNTER
               (setq $entCounterAtt 0)

               ;PROCESS EACH ENTITY
               (while (< $entPointerAtt $entTotalAtt)

                  ;GET ENTITY GROUP CODE
                  (setq $entCodeAtt (car (nth $entPointerAtt $entAttributes)))

                  ;GET THE OBJECT IF GROUP CODE = 3 ATTRIBUTE
                  (if (= $entCodeAtt 3)

                     (progn

                        ;INCREMENT THE ATTRIBUTE COUNTER
                        (setq $entCounterAtt (+ $entCounterAtt 1))
```

```
                ;GET THE FIELD NAME
                (setq $fieldName (cdr (nth $entPointerAtt $entAttributes)))

                ;GET THE XRECORD
                (setq $entXrecord (dictsearch $objAttributes $fieldName))

                ;GET THE DATA TYPE AND VALUE
                (cond

                   (((/= (setq $entField (assoc 1 $entXrecord)) nil)
                        (setq $fieldType "Text"
                              $fieldValue (cdr $entField)
                        )
                   )
                   (((/= (setq $entField (assoc 40 $entXrecord)) nil)
                        (setq $fieldType "Real Number"
                              $fieldValue (rtos (cdr $entField))
                        )
                   )
                   (((/= (setq $entField (assoc 70 $entXrecord)) nil)
                        (setq $fieldType "Short Integer"
                              $fieldValue (itoa (cdr $entField))
                        )
                   )
                   (((/= (setq $entField (assoc 90 $entXrecord)) nil)
                        (setq $fieldType "Long Integer"
                              $fieldValue (itoa (cdr $entField))
                        )
                   )

                );_ cond GET THE DATA TYPE AND VALUE

                ;COLLECT ATTRIBUTES IN A LIST ARRAY
                (if (= $entCounterAtt 1)

                   (setq $listAttributes
                      (list (list $fieldName $fieldType $fieldValue))
                   )
                   (setq $listAttributes
                      (append $listAttributes
                         (list (list $fieldName $fieldType $fieldValue))
                      )
                   )

                );_ if COLLECT ATTRIBUTES IN A LIST ARRAY
             );_ progn
          );_ if GET THE OBJECT IF GROUP CODE = 3 ATTRIBUTE

          ;INCREMENT THE POINTER
          (setq $entPointerAtt (+ $entPointerAtt 1))

       );_ while PROCESS EACH ENTITY

       ;PRINT ENTITY TYPE TO THE TEXT SCREEN
       (textscr)
       (prompt (strcat "\n" "Entity: " $entType "\n"))

       ;SET PRINTER COUNTER
       (setq $listPointer 0)

       ;PRINT ATTRIBUTE LIST TO SCREEN
       (prompt (strcat "Attributes:(" (itoa $entCounterAtt) " found)"))
```

```
                (while (< $listPointer $entCounterAtt)

                   (prompt (strcat "\n " (car (nth $listPointer $listAttributes))
                                   " = " (caddr (nth $listPointer $listAttributes))
                           )
                   )

                   (setq $listPointer (+ $listPointer 1))
                   (princ)

                );_ while PRINT ATTRIBUTES TO SCREEN

             );_ progn STEP THROUGH THE EXTENSION DICTIONARY

             ;INFORM USER IF NO ATTRIBUTES FOUND
             (prompt "0 attributes found")

          );_ if GET THE ESRI EXTENSION DICTIONARY

       );_ progn GET ENTITY LIST

       ;INFORM USER IF NO ENTITY SELECTED
       (prompt " 0 selected")

    );_ if SELECT ENTITY

    (princ)

);_ defun main function
```