

GHS_Record_Inset_

Version 100

Procedure Description

21. December 2005


GreenHouse

Software & Consulting

Karl-Heinz Weber

Heinrichstrasse 12

D-45711 Datteln/Horneburg

GHS_Record_Inset_

GHS_Record_Inset_ encodes/decodes fields in a data record.

```
Error := GHS_Record_Inset_(Buf:BufLen
                           ,Mask,MaskLen
                           ,Key:KeyLen
                           ,Seed,SeedLen
                           ,Direction
                           ,SegNum);
```

Error, Int, returned value

```
0    operation successful
1    BufLen exceeds max size, which is actually 8192
2    MaskLen is an odd number
3    KeyLen is not 16
4    SeedLen is not 12
5    A Mask field is NOT within size of Buf
6    A Mask field is not in ascending order
7    BufLen negative
8    Error while allocating the extended segment
9    Decode error: First byte of first mask field is not encoded
10   Decode error: Last byte of last mask field is not encoded
```

Buf:BufLen, String:EXT:Ref:*:INT:Value input,output:input
 BufLen is 0 .. 8192

Mask, Int:EXT:Ref* input

MaskMaskLen, INT:Value input
 MaskLen always is an even number

Key:KeyLen, String:EXT:Ref:*:INT:Value input:input
 KeyLen has to be 16

Seed, Int:EXT:Ref:* input,output

SeedLen, INT:Value input
 SeedLen has to be 12

Direction, INT:Value, input
 0 Decode
 1 Encode

SegNum, Int:EXT:Ref:1 input,output
 -1 use the first free available segment
 >-1 use that segment

Mask:MaskLen **input**

Int:EXT:Ref*

Mask is a set of number pairs, defining the fields, that have to be encoded/decoded.

A number pair defines the

- start byte address of the field (first byte starts with 0)
- the length of the field (in bytes)

The smallest start address is zero (0), the largest start address is BufLen-1.

e.g.:

0,15 Is a 15 byte string, starting at position 0, and ending at position 14
19,7 is a 7 byte string, starting at position 19 and ending at position 25

Mask can be defined like this:

```
Literal MaskLen = 4; ! Number of words in Mask  
Int .Mask[0:MaskLen-1] := [0,15,19,7]; ! Mask number pairs
```

This defines the two fields from above.

The total number of pairs is limited by the size of the record.

In case MaskLen is an odd number, error 2 is returned.

Mask fields have to be within the range of BufLen (0 – BufLen-1). A field, exceeding BufLen, causes error 5 to be returned.

The Mask fields have to be in ascending order. In case a field is found, that does not follow this requirement, or in case fields overlap, error 6 is returned.

MaskLen **input**

Int:Value

MaskLen defines the length in words of the INT-array Mask, and therefore always is an even number.

Direction

input

INT:Value

Defines the direction (encode, decode) of the operation.

Encode causes the procedure to translate clear text into a set of 'random' numbers.
Decode causes the procedure to translate a set of 'random numbers' back into the clear text.

Valid values are:

0 Decode
1 Encode

SegNum

input, output

Int:EXT:Ref:1

Defines the segment number of an extended segment to be used by DES.
The used DES procedure needs an extended segment. This is allocated with the first call, and kept until the process is stopped

To allow an optimum on flexibility, two methods of allocating the segment are supported:

1. The procedure can use the first unused segment of its process environment.
To make the procedure aware of this, the initial value has to be -1.
The segment number is returned, where bit zero (SegNum.<0> = sign bit) is set.
2. The user defined segment is used. On return, the sign bit is set.

Do NOT touch the returned value, and provided it with all subsequent calls.

Values are:

-1 the first free available segment is used
>-1 the specified segment is used

Example

```
Struct .EXT DataRecord;
  Begin
    String  FirstName[0:19];      ! 0,20
    String  LastName[0:19];      ! 20,20
    Int(32) Salary                ! 40, 4
    Int     Age;                  ! 44, 2
    String  SocialNumber[0:31]   ! 46,32
    Struct  Address;
      Begin
        String  Street[0:19];    ! 78,20
        String  City[0:19];     ! 98,20
        Int     ZIPCode;        ! 118, 2
        Int(32) PhoneNumber;    ! 120, 4
      End;
    .
    .
    .
  End;

Literal  Encode    = 1,
         Decode    = 0,
         NumFields = 3;

String  .EXT Seed[0:7];
String  .EXT Key[0:15] := "1qaSDF5&7lkjPo=?";
Int     .EXT Mask[(NumFields*2)-1] := [$OFFSET(DataRecord.Salary)
                                       , $LEN(DataRecord.Salary)
                                       , $OFFSET(DataRecord.Age)
                                       , $LEN(DataRecord.Age)
                                       , $OFFSET(DataRecord.SocialNumber)
                                       , $LEN(DataRecord.SocialNumber)];

Int     SegNum := -1;
.
! Lots of code . . .
.
! Adjust DES key
Use I;
For I := 0 to 15 do
  Key[I] := Key[I] `<<'1;
Drop I;
.
! Lots of code . . .
.
! Encode record fields
Error := GHS_Record_Inset_(DataRecord:$LEN(DataRecord)
                          ,Mask:NumFields*2
                          ,Key:16
                          ,Seed,8
                          ,Encode
                          ,SegNum);
```

Use

Encode:

The ENCODE operation returns an operation specific SEED value.

To decode any byte from the encoded record, this SEED is needed!

It is highly recommended, to encode ALL fields, that need to be encoded, within one call.

Encoding different fields by different procedure calls results in a set of different returned SEEDs!

You have to keep track of the SEEDs, and the related encode operation!

For simplicity reasons it is recommended, to encode all fields within one operation, which produces one SEED for this record.

Decode:

The DECODE operation needs the SEED, generated by the ENCODE operation when encoding the filed(s), that have to be decoded.

Single fields can be decoded, even single bytes of encoded fields.

Dependencies

GHS_Record_Inset_ uses procedures from the DES FreeWare suite, provided by GreenHouse (GHS), as well as procedures from the GHS Library.

Delivery

GHS_Record_Inset_ comes in two flavors:

1. TAL code 100 to be used with BIND, or the SEARCH compiler directive
2. pTAL code 700, ready to be used with nld

File name is: STREAM

It as well comes with a file, containing the external declarations.

File Name is: STREAMEX

Performance

Throughput depends on the position of the last byte that has to be encoded/decoded: Accessing fields at the beginning of the record is much faster than accessing fields at the end.

To speed up the cipher process it is highly recommended to have the fields to be encoded/decoded at the beginning of the record, and the plan text at the end.

INSET Program

INSET is a TAL/pTAL program, allowing the encoding/decoding of all records of a file.

Command syntax is:

```
[run] INSET -H[ELP]
```

where

-H[ELP] causes INSET to display its syntax.

```
[run] INSET/IN <file>,OUT <file>/ENCODE|DECODE  
      ,MASK <mask>  
      [,KEY <key>]
```

where

IN <file> is any ENSCRIBE type file.
Maximum record size for the ENCODE operation depends on the in file type, and is 8 bytes less of the maximum record size.

OUT <file> is the name of the out file, which has to exist.
For ENCODE, the record size has to be 12 bytes bigger than in IN file, for DECODE, the record size has to be 12 bytes smaller than of IN file.
These 12 bytes are needed to store the record specific SEED, and other operation attributes.
Do NOT change them.

ENCODE|DECODE tells the program, what to do

MASK <mask> Keyword, plus a set of number pairs, defining the fields to be encoded/decoded.

e.g.

0,7,15,4,30,17 which means:

0, 7 start at byte 0 and process 7 bytes (0 .. 6)

15, 4 start at byte 15 and process 4 bytes (15 .. 18)

29,17 start at byte 29 and process 17 bytes (29 .. 45)

KEY <key> keyword, plus a 16 byte user password in quotes.
When missing, the user is prompted to enter a 16 byte long key.

e.g.:

```
run inset/in infile,out outfile/encode,key 1qayxcvbnmkjhgf5,mask 0,7,15,4,29,17
```

The order of keywords is arbitrary.

Availability*Produkt:*

The product (procedure) will be made available in a customized and time limited, but fully functional version. This allows a one month test.

Test program:

The test program will be restricted to 1.000 records. This limit will be removed, when the product is purchased.