CEEFRST—Free heap storage

CEEFRST—Free heap storage

CEEFRST frees storage previously allocated by CEEGTST or by a language intrinsic function. Normally, you do not need to call CEEFRST because Language Environment automatically returns all heap storage to the operating system when the enclave terminates. However, if you are allocating a large amount of heap storage, you should free the storage when it is no longer needed. This freed storage then becomes available for later requests for heap storage, thus reducing the total amount of storage needed to run the application.

All requests for storage are conditional. If storage is not available, the feedback code (fc) is set and returned to you, but the thread does not abend. An attempt to free storage that was already marked as free produces no action and returns a non-CEE000 symbolic feedback code. An attempt to free storage at anything other than a valid starting address produces no action and returns a non-CEE000 symbolic feedback code. The application does not abend.

However, if you call CEEFRST for an invalid address, and you had specified TRAP(OFF), your application can abend. The reaction of Language Environment to this is undefined. Also, partial freeing of an allocated area is not supported.

When storage is allocated by CEEGTST, its allocated size is used during free operations. Storage allocated by CEEGTST, but not explicitly freed, is automatically freed at enclave termination.

CEEFRST generates a system-level free storage call to return a storage increment to the operating system only when:

- The last heap element within an increment is being freed, and
- The HEAP runtime option or a call to CEECRHP specifies FREE (note that KEEP is the IBM-supplied default setting for the initial heap).

Otherwise, the freed storage is simply added to the free list; it is not returned to the operating system until termination. The out-of-storage condition can cause freeing of empty increments even when KEEP is specified.

Read syntax diagramSkip visual syntax diagram

address (input) A fullword address pointer. address is the address returned by a previous CEEGTST call or a language intrinsic function such as ALLOCATE or `malloc()`. The storage at this address is deallocated.

fc (output) A 12-byte feedback code, optional in some languages, that indicates the result of this service. If you choose to omit this parameter, refer to Invoking callable services for the appropriate syntax to indicate that the feedback code was omitted.

The following feedback codes can result from this service:

Code	Severity	Message number	Message text
CEE000	0	—	The service completed successfully.
CEE0P2	4	0802	Heap storage control information was damaged.
CEE0PA	3	0810	The storage address in a free storage (CEEFRST) request was not recognized, or heap storage (CEECZST) control information was damaged.

Usage notes

- If you specify heap_free_value in the STORAGE runtime option, all freed storage is overwritten with heap_free_value. Otherwise, it is simply marked as available.

  Portions of the freed storage area can be used to hold internal storage manager control information. These areas are overwritten, but not with heap_free_value.

- The heap identifier is inferred from the address of the storage to be freed. The storage is freed from the heap in which it was allocated.

- The address passed as the argument is a dangling pointer after a call to CEEFRST. The storage freed by this operation can be reallocated on a subsequent CEEGTST call. If the pointer is not reassigned, any further use of it causes unpredictable results.

- z/OS UNIX considerations—CEEFRST applies to the enclave. One thread can allocate storage, and another can free it.

For more information

- See CEEGTST—Get heap storage for more information about the CEEGTST callable service.
- See HEAP for further information about the HEAP runtime option.

- See CEECRHP—Create new additional heap for more information about the CEECRHP callable service.
- See STORAGE for further information about the STORAGE runtime option.

Examples

1. An example of CEEFRST called by C/C++:

```
/*Module/File Name: EDCFRST   */

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <leawi.h>
#include <ceeedcct.h>

int main(void) {

   _INT4 heapid, size;
   _POINTER address;
   _FEEDBACK fc;
 /*  .
      .
      .  */
   heapid = 0;    /* get storage from initial heap   */
   size = 4000;   /* number of bytes of heap storage */

   /* obtain the storage using CEEGTST */
   CEEGTST(&heapid,&size,&address,&fc);

   /* check the first 4 bytes of the feedback token */
   /* (0 if successful) */
   if ( _FBCHECK ( fc , CEE000 ) != 0 ) {
      printf("CEEGTST failed with message number %d\n",
             fc.tok_msgno);
      exit(99);
   }
 /*  .
      .
      .  */

   /* free the storage that was previously obtained */
   /* using CEEGTST */
   CEEFRST(&address,&fc);
```

```
   /* check the first 4 bytes of the feedback token */
   /* (0 if successful) */
   if ( _FBCHECK ( fc , CEE000 ) != 0 ) {
      printf("CEEFRST failed with message number %d\n",
             fc.tok_msgno);
      exit(99);
   }
 /*  .
      .
      .  */
}
```

2. An example of CEEFRST called by COBOL:

```
CBL LIB,QUOTE
      *Module/File Name: IGZTFRST
      ********************************************
      **                                        **
      ** IGZTFRST - Call CEEFRST to free heap   **
      **            storage                     **
      **                                        **
      ** In this example, a call is made to     **
      ** CEEGTST to obtain 4000 bytes of storage **
      ** from the initial heap (HEAPID = 0).    **
      ** A call is then made to CEEFRST to free **
      ** the storage.                           **
      **                                        **
      ********************************************
       IDENTIFICATION DIVISION.
       PROGRAM-ID. IGZTFRST.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01  HEAPID                  PIC S9(9) BINARY.
       01  STGSIZE                 PIC S9(9) BINARY.
       01  ADDRSS                  USAGE IS POINTER.
       01  FC.
           02  Condition-Token-Value.
           COPY  CEEIGZCT.
               03  Case-1-Condition-ID.
                   04  Severity    PIC S9(4) BINARY.
                   04  Msg-No      PIC S9(4) BINARY.
               03  Case-2-Condition-ID
                         REDEFINES Case-1-Condition-ID.
                   04  Class-Code  PIC S9(4) BINARY.
                   04  Cause-Code  PIC S9(4) BINARY.
               03  Case-Sev-Ctl    PIC X.
               03  Facility-ID     PIC XXX.
```

```
     02  I-S-Info            PIC S9(9) BINARY.
 PROCEDURE DIVISION.
 PARA-CBLFRST.

** Specify 0 to get storage from the initial
**    heap.
** Specify 4000 to get 4000 bytes of storage.
** Call CEEGTST to obtain storage.
     MOVE 0 TO HEAPID.
     MOVE 4000 TO STGSIZE.

     CALL "CEEGTST" USING HEAPID , STGSIZE ,
         ADDRSS , FC.
     IF CEE000 of FC  THEN
         DISPLAY "Obtained " STGSIZE " bytes of"
             " storage at location " ADDRSS
             " from heap number " HEAPID
     ELSE
         DISPLAY "CEEGTST failed with msg "
             Msg-No of FC UPON CONSOLE
         STOP RUN
     END-IF.
** To free storage, use the address returned
**     by CEECRHP in the call to CEEFRST.
     CALL "CEEFRST" USING ADDRSS , FC.
     IF CEE000 of FC THEN
         DISPLAY "Returned " STGSIZE " bytes of"
             " storage at location " ADDRSS
             "  to  heap number " HEAPID
     ELSE
         DISPLAY "CEEFRST failed with msg "
             Msg-No of FC UPON CONSOLE
     END-IF.
     GOBACK.
```

3. An example of CEEFRST called by PL/I:

```
*PROCESS MACRO;
 /*Module/File Name: IBMFRST                        */
 /************************************************/
 /**                                              **/
 /** Function: CEEFRST - free heap storage        **/
 /**                                              **/
 /** This example calls CEEGTST to obtain storage **/
 /** from the initial heap, and then calls        **/
 /** CEEFRST to discard it.                       **/
 /**                                              **/
```

```
/*************************************************/
PLIFRST: PROC OPTIONS(MAIN);

   %INCLUDE  CEEIBMAW;
   %INCLUDE  CEEIBMCT;

   DCL ADDRSS   POINTER;
   DCL HEAPID   REAL FIXED BINARY(31,0);
   DCL STGSIZE  REAL FIXED BINARY(31,0);
   DCL 01 FC,                        /* Feedback token */
          03 MsgSev    REAL FIXED BINARY(15,0),
          03 MsgNo     REAL FIXED BINARY(15,0),
          03 Flags,
             05 Case      BIT(2),
             05 Severity  BIT(3),
             05 Control   BIT(3),
          03 FacID     CHAR(3),    /* Facility ID */
          03 ISI    /* Instance-Specific Information */
                      REAL FIXED BINARY(31,0);

   DCL 01 FC2,                       /* Feedback token */
          03 MsgSev    REAL FIXED BINARY(15,0),
          03 MsgNo     REAL FIXED BINARY(15,0),
          03 Flags,
             05 Case      BIT(2),
             05 Severity  BIT(3),
             05 Control   BIT(3),
          03 FacID     CHAR(3),    /* Facility ID */
          03 ISI   /* Instance-Specific Information */
                      REAL FIXED BINARY(31,0);

   HEAPID = 0; /* get storage from the initial heap */

   STGSIZE = 4000;   /* get 4000 bytes of storage   */

   /* Call CEEGTST to obtain the storage           */
   CALL CEEGTST ( HEAPID, STGSIZE, ADDRSS, FC );
   IF  FBCHECK( FC, CEE000)  THEN  DO;
      PUT SKIP LIST( 'Obtained ' || STGSIZE
         || ' bytes of storage at location '
         || DECIMAL( UNSPEC( ADDRSS ) )
         || ' from heap ' || HEAPID );
      END;

   ELSE  DO;
      DISPLAY( 'CEEGTST failed with msg '
```

```
         || FC.MsgNo );
      STOP;
      END;  /* Call CEEFRST with the address returned from */
   /*    CEEGTST to free the storage allocated by   */
   /*    the call to CEEGTST                        */
   CALL CEEFRST ( ADDRSS, FC2 );
   IF  FBCHECK( FC2, CEE000)  THEN  DO;
      PUT SKIP LIST( 'Storage block at location '
         || DECIMAL( UNSPEC( ADDRSS ) ) || ' freed');
      END;
   ELSE  DO;
      DISPLAY( 'CEEFRST failed with msg '
         || FC2.MsgNo );
      STOP;
      END;

END PLIFRST;
```

Parent topic: Callable services