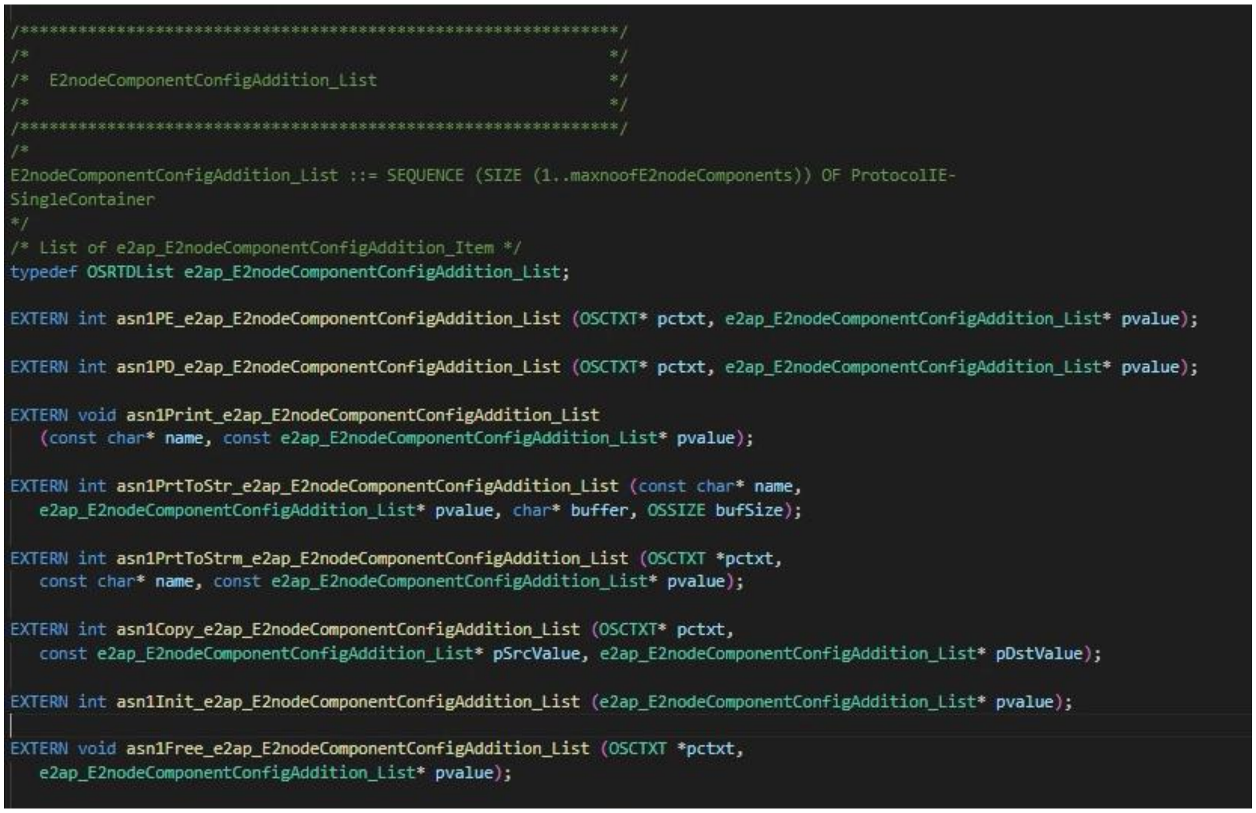
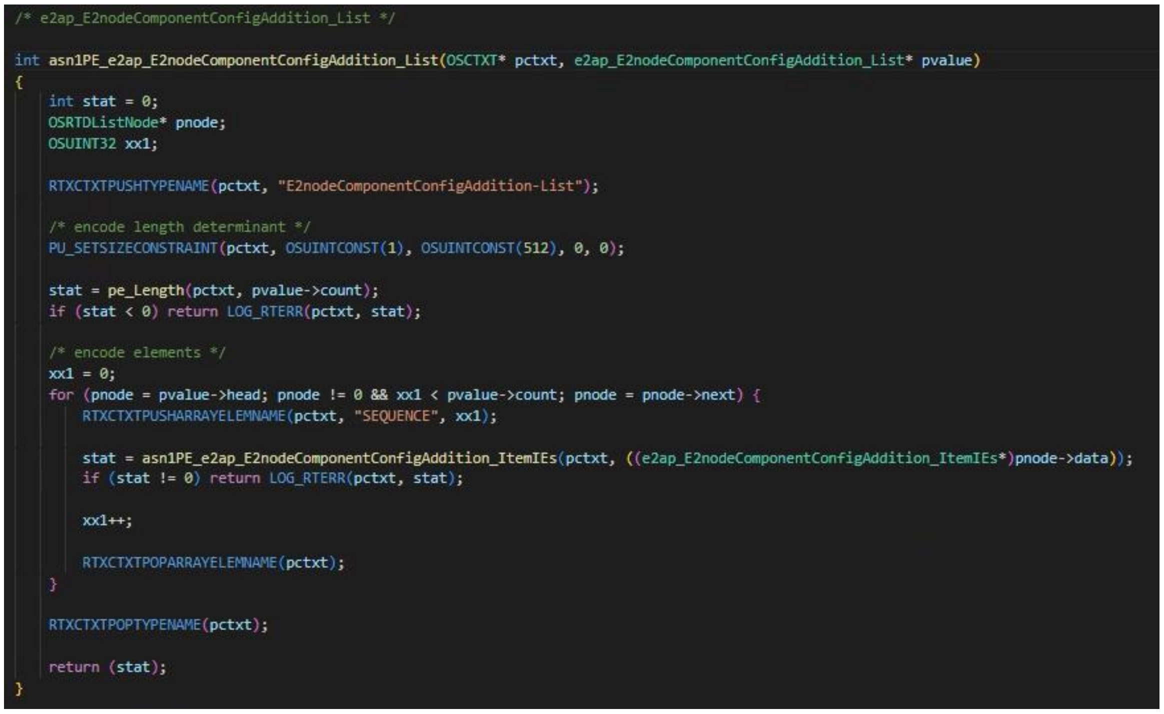
```
E2nodeComponentConfigAddition-List ::= SEQUENCE
(SIZE(1..maxofE2nodeComponents)) OF ProtocolIE

SingleContainer { {E2nodeComponentConfigAddition-ItemIEs} }


E2nodeComponentConfigAddition-ItemIEs  E2AP-PROTOCOL-IES ::= {

 { ID id-E2nodeComponentConfigAddition-Item CRITICALITY reject  TYPE
E2nodeComponentConfigAddition

Item   PRESENCE mandatory },

 ...

}


E2nodeComponentConfigAddition-Item ::= SEQUENCE {

 e2nodeComponentInterfaceType  E2nodeComponentInterfaceType,

 e2nodeComponentID    E2nodeComponentID,

 e2nodeComponentConfiguration  E2nodeComponentConfiguration,

 ...

}
```

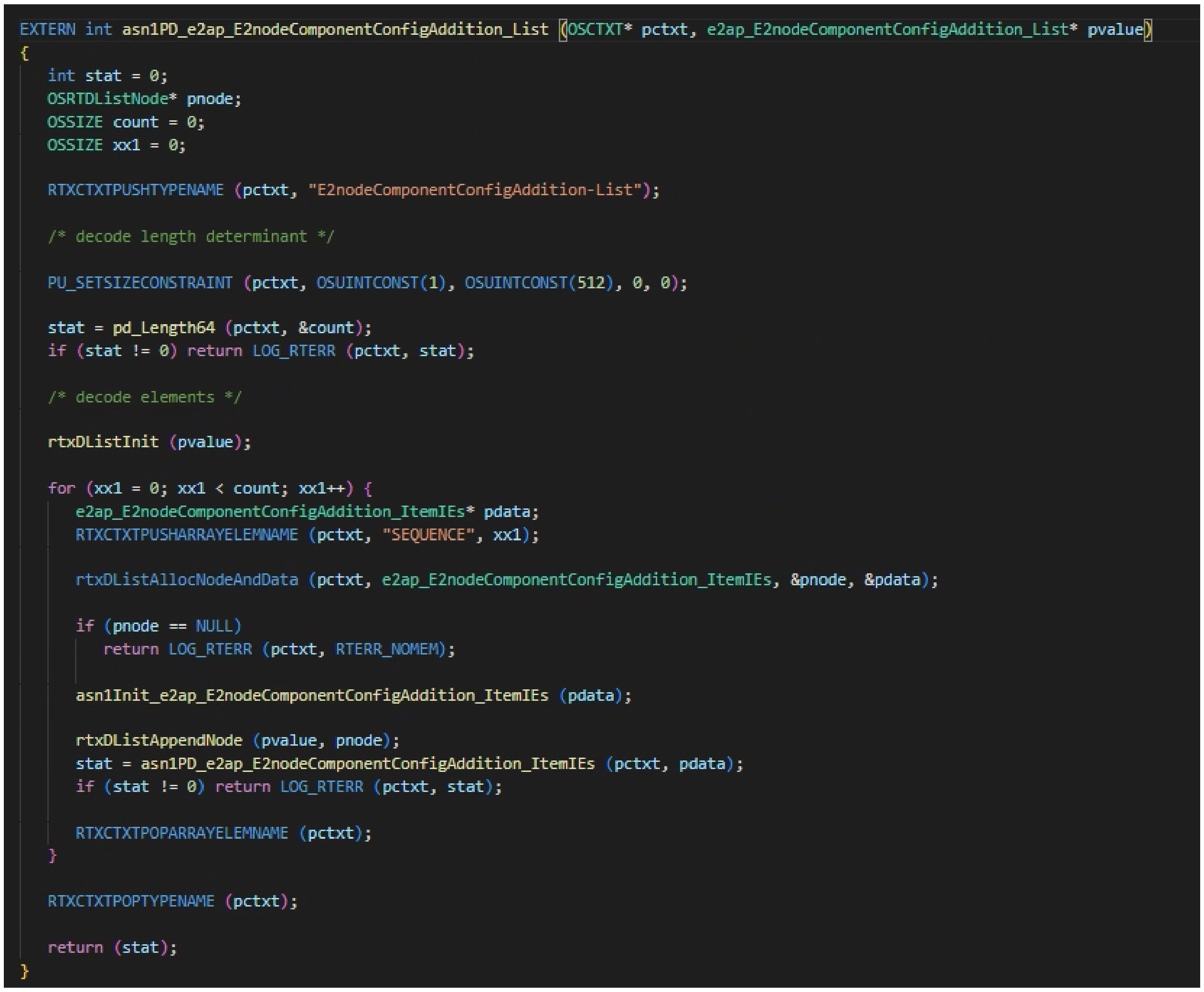
```
E2nodeComponentConfigAddition-List ::= SEQUENCE
(SIZE(1..maxofE2nodeComponents)) OF ProtocolIE

SingleContainer { {E2nodeComponentConfigAddition-ItemIEs} }
```

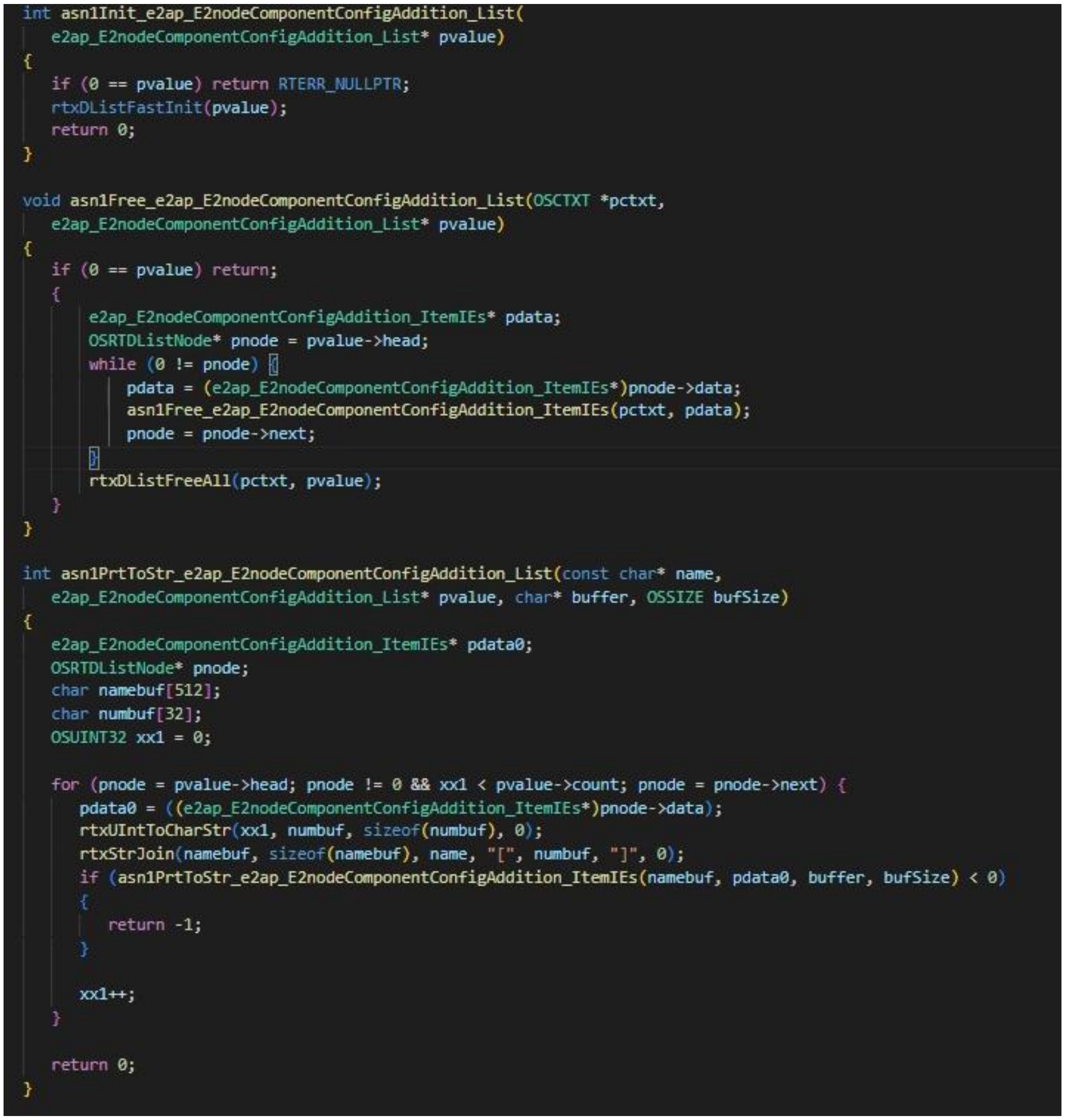


```c
/**************************************************************/
/*                                                            */
/*  E2nodeComponentConfigAddition_List                        */
/*                                                            */
/**************************************************************/
/*
E2nodeComponentConfigAddition_List ::= SEQUENCE (SIZE (1..maxnoofE2nodeComponents)) OF ProtocolIE-
SingleContainer
*/
/* List of e2ap_E2nodeComponentConfigAddition_Item */
typedef OSRTDList e2ap_E2nodeComponentConfigAddition_List;

EXTERN int asn1PE_e2ap_E2nodeComponentConfigAddition_List (OSCTXT* pctxt, e2ap_E2nodeComponentConfigAddition_List* pvalue);

EXTERN int asn1PD_e2ap_E2nodeComponentConfigAddition_List (OSCTXT* pctxt, e2ap_E2nodeComponentConfigAddition_List* pvalue);

EXTERN void asn1Print_e2ap_E2nodeComponentConfigAddition_List
   (const char* name, const e2ap_E2nodeComponentConfigAddition_List* pvalue);

EXTERN int asn1PrtToStr_e2ap_E2nodeComponentConfigAddition_List (const char* name,
   e2ap_E2nodeComponentConfigAddition_List* pvalue, char* buffer, OSSIZE bufSize);

EXTERN int asn1PrtToStrm_e2ap_E2nodeComponentConfigAddition_List (OSCTXT *pctxt,
   const char* name, const e2ap_E2nodeComponentConfigAddition_List* pvalue);

EXTERN int asn1Copy_e2ap_E2nodeComponentConfigAddition_List (OSCTXT* pctxt,
   const e2ap_E2nodeComponentConfigAddition_List* pSrcValue, e2ap_E2nodeComponentConfigAddition_List* pDstValue);

EXTERN int asn1Init_e2ap_E2nodeComponentConfigAddition_List (e2ap_E2nodeComponentConfigAddition_List* pvalue);

EXTERN void asn1Free_e2ap_E2nodeComponentConfigAddition_List (OSCTXT *pctxt,
   e2ap_E2nodeComponentConfigAddition_List* pvalue);
```



```c
/* e2ap_E2nodeComponentConfigAddition_List */

int asn1PE_e2ap_E2nodeComponentConfigAddition_List(OSCTXT* pctxt, e2ap_E2nodeComponentConfigAddition_List* pvalue)
{
    int stat = 0;
    OSRTDListNode* pnode;
    OSUINT32 xx1;

    RTXCTXTPUSHTYPENAME(pctxt, "E2nodeComponentConfigAddition-List");

    /* encode length determinant */
    PU_SETSIZECONSTRAINT(pctxt, OSUINTCONST(1), OSUINTCONST(512), 0, 0);

    stat = pe_Length(pctxt, pvalue->count);
    if (stat < 0) return LOG_RTERR(pctxt, stat);

    /* encode elements */
    xx1 = 0;
    for (pnode = pvalue->head; pnode != 0 && xx1 < pvalue->count; pnode = pnode->next) {
        RTXCTXTPUSHARRAYELEMNAME(pctxt, "SEQUENCE", xx1);

        stat = asn1PE_e2ap_E2nodeComponentConfigAddition_ItemIEs(pctxt, ((e2ap_E2nodeComponentConfigAddition_ItemIEs*)pnode->data));
        if (stat != 0) return LOG_RTERR(pctxt, stat);

        xx1++;

        RTXCTXTPOPARRAYELEMNAME(pctxt);
    }

    RTXCTXTPOPTYPENAME(pctxt);

    return (stat);
}
```


```c
EXTERN int asn1PD_e2ap_E2nodeComponentConfigAddition_List (OSCTXT* pctxt, e2ap_E2nodeComponentConfigAddition_List* pvalue)
{
   int stat = 0;
   OSRTDListNode* pnode;
   OSSIZE count = 0;
   OSSIZE xx1 = 0;

   RTXCTXTPUSHTYPENAME (pctxt, "E2nodeComponentConfigAddition-List");

   /* decode length determinant */

   PU_SETSIZECONSTRAINT (pctxt, OSUINTCONST(1), OSUINTCONST(512), 0, 0);

   stat = pd_Length64 (pctxt, &count);
   if (stat != 0) return LOG_RTERR (pctxt, stat);

   /* decode elements */

   rtxDListInit (pvalue);

   for (xx1 = 0; xx1 < count; xx1++) {
      e2ap_E2nodeComponentConfigAddition_ItemIEs* pdata;
      RTXCTXTPUSHARRAYELEMNAME (pctxt, "SEQUENCE", xx1);

      rtxDListAllocNodeAndData (pctxt, e2ap_E2nodeComponentConfigAddition_ItemIEs, &pnode, &pdata);

      if (pnode == NULL)
         return LOG_RTERR (pctxt, RTERR_NOMEM);

      asn1Init_e2ap_E2nodeComponentConfigAddition_ItemIEs (pdata);

      rtxDListAppendNode (pvalue, pnode);
      stat = asn1PD_e2ap_E2nodeComponentConfigAddition_ItemIEs (pctxt, pdata);
      if (stat != 0) return LOG_RTERR (pctxt, stat);

      RTXCTXTPOPARRAYELEMNAME (pctxt);
   }

   RTXCTXTPOPTYPENAME (pctxt);

   return (stat);
}
```



```
int asn1Init_e2ap_E2nodeComponentConfigAddition_List(
   e2ap_E2nodeComponentConfigAddition_List* pvalue)
{
   if (0 == pvalue) return RTERR_NULLPTR;
   rtxDListFastInit(pvalue);
   return 0;
}

void asn1Free_e2ap_E2nodeComponentConfigAddition_List(OSCTXT *pctxt,
   e2ap_E2nodeComponentConfigAddition_List* pvalue)
{
   if (0 == pvalue) return;
   {
      e2ap_E2nodeComponentConfigAddition_ItemIEs* pdata;
      OSRTDListNode* pnode = pvalue->head;
      while (0 != pnode) {
         pdata = (e2ap_E2nodeComponentConfigAddition_ItemIEs*)pnode->data;
         asn1Free_e2ap_E2nodeComponentConfigAddition_ItemIEs(pctxt, pdata);
         pnode = pnode->next;
      }
      rtxDListFreeAll(pctxt, pvalue);
   }
}

int asn1PrtToStr_e2ap_E2nodeComponentConfigAddition_List(const char* name,
   e2ap_E2nodeComponentConfigAddition_List* pvalue, char* buffer, OSSIZE bufSize)
{
   e2ap_E2nodeComponentConfigAddition_ItemIEs* pdata0;
   OSRTDListNode* pnode;
   char namebuf[512];
   char numbuf[32];
   OSUINT32 xx1 = 0;

   for (pnode = pvalue->head; pnode != 0 && xx1 < pvalue->count; pnode = pnode->next) {
      pdata0 = ((e2ap_E2nodeComponentConfigAddition_ItemIEs*)pnode->data);
      rtxUIntToCharStr(xx1, numbuf, sizeof(numbuf), 0);
      rtxStrJoin(namebuf, sizeof(namebuf), name, "[", numbuf, "]", 0);
      if (asn1PrtToStr_e2ap_E2nodeComponentConfigAddition_ItemIEs(namebuf, pdata0, buffer, bufSize) < 0)
      {
         return -1;
      }

      xx1++;
   }

   return 0;
}
```

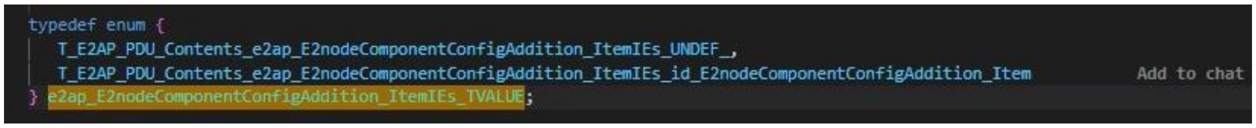
E2nodeComponentConfigAddition-ItemIEs E2AP-PROTOCOL-IES ::= {

{ ID id-E2nodeComponentConfigAddition-Item CRITICALITY reject TYPE E2nodeComponentConfigAddition

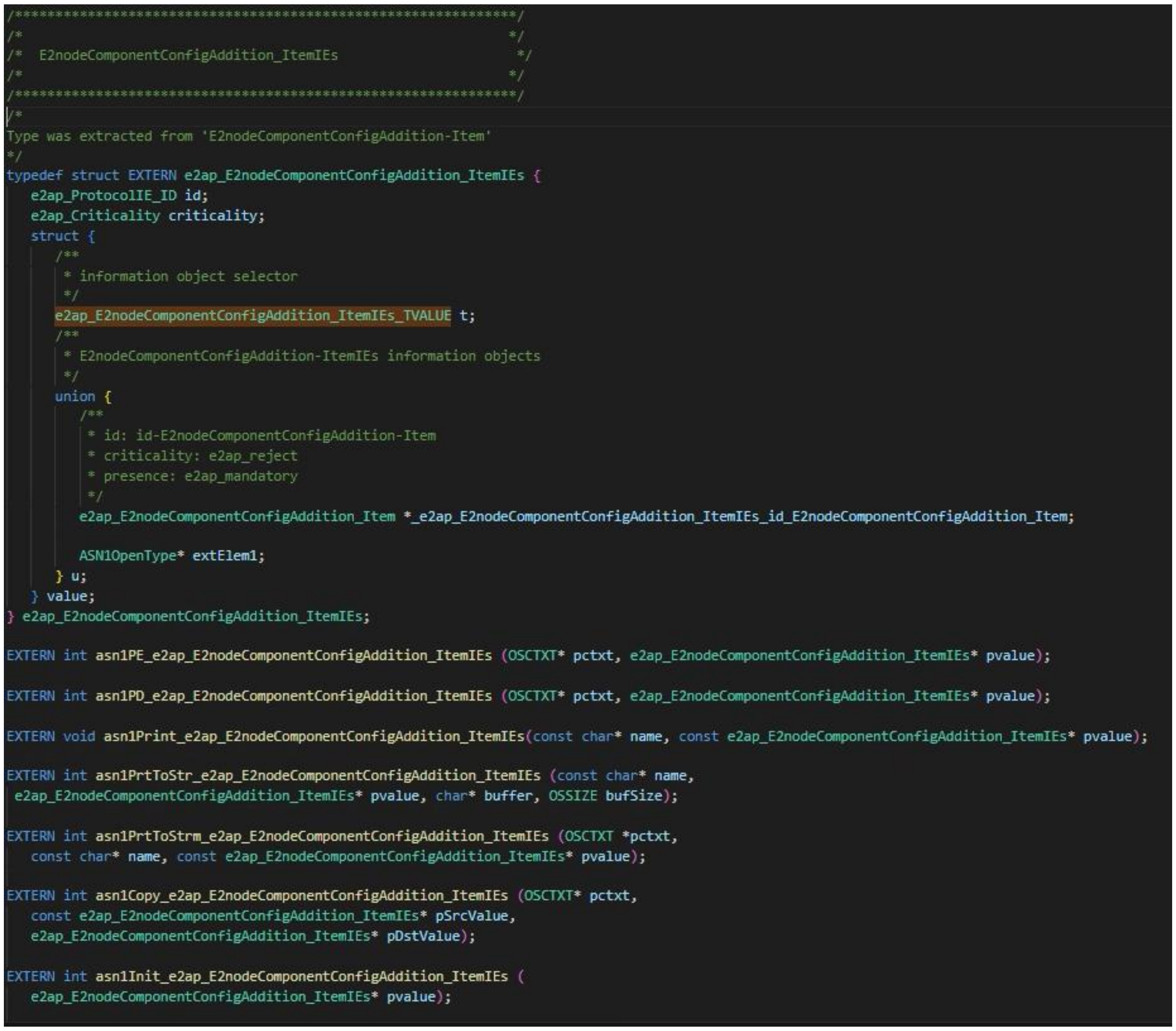
Item   PRESENCE mandatory },

...

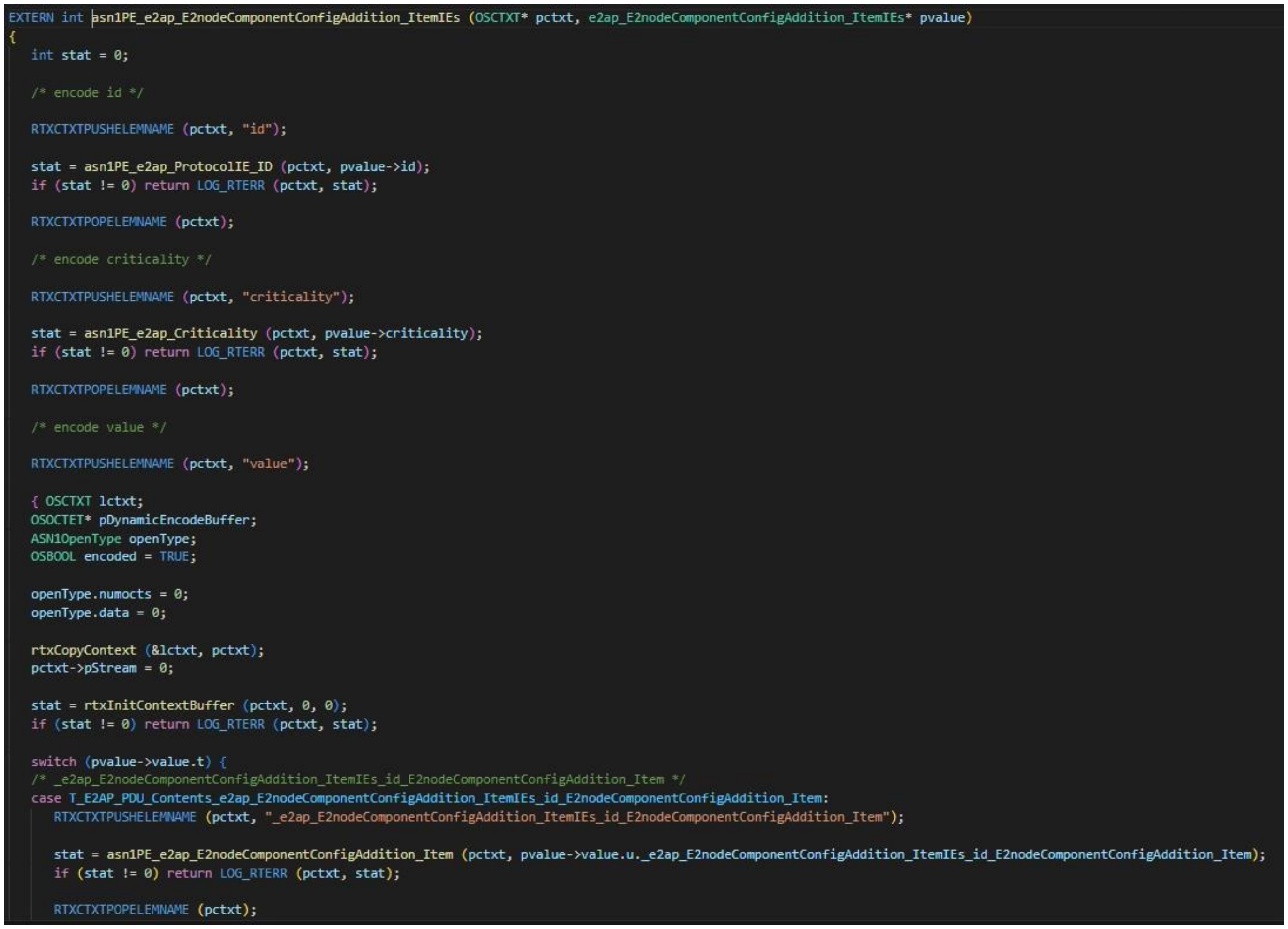
}



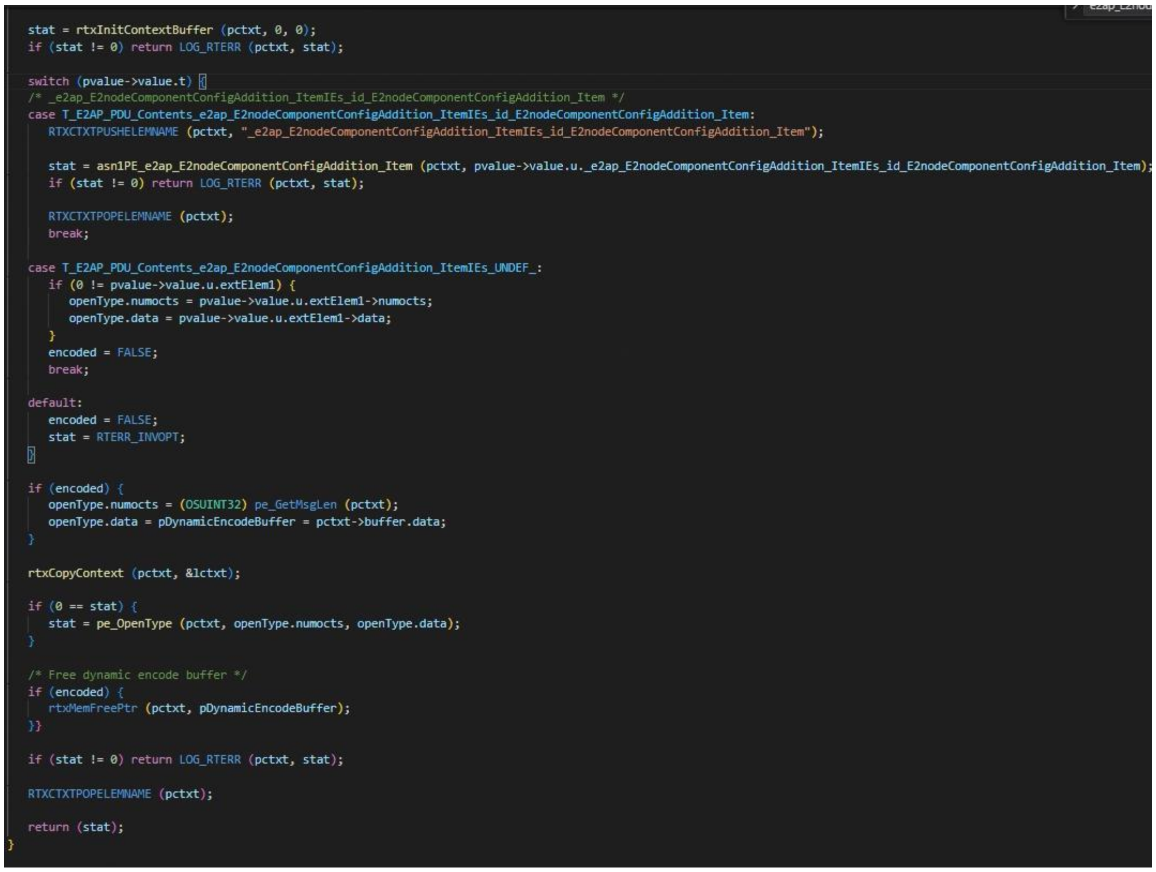
```
typedef enum {
   T_E2AP_PDU_Contents_e2ap_E2nodeComponentConfigAddition_ItemIEs_UNDEF_,
   T_E2AP_PDU_Contents_e2ap_E2nodeComponentConfigAddition_ItemIEs_id_E2nodeComponentConfigAddition_Item
} e2ap_E2nodeComponentConfigAddition_ItemIEs_TVALUE;
```



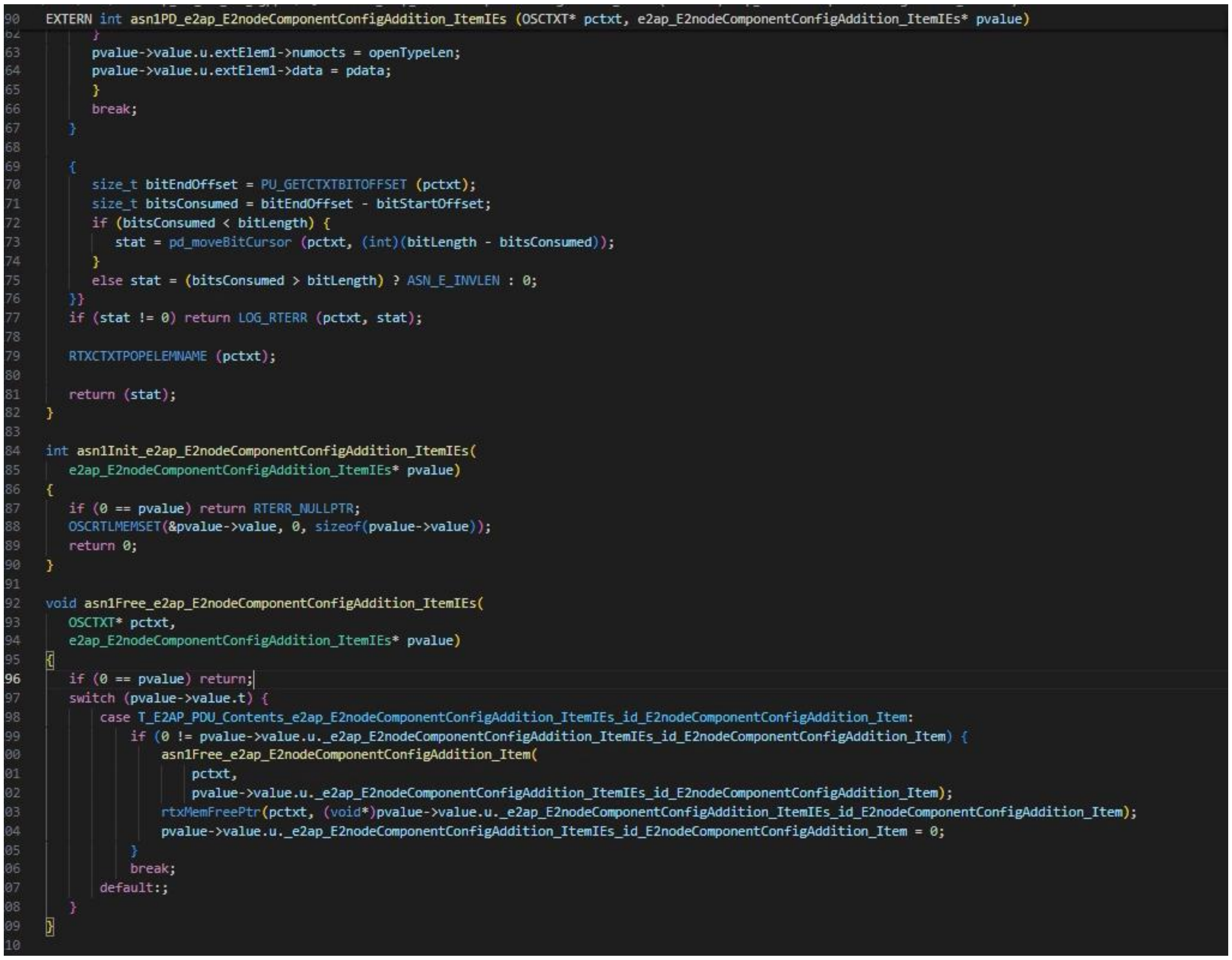
```
/**************************************************************/
/*                                                            */
/*  E2nodeComponentConfigAddition_ItemIEs                     */
/*                                                            */
/**************************************************************/
/*
Type was extracted from 'E2nodeComponentConfigAddition-Item'
*/
typedef struct EXTERN e2ap_E2nodeComponentConfigAddition_ItemIEs {
   e2ap_ProtocolIE_ID id;
   e2ap_Criticality criticality;
   struct {
      /**
       * information object selector
       */
      e2ap_E2nodeComponentConfigAddition_ItemIEs_TVALUE t;
      /**
       * E2nodeComponentConfigAddition-ItemIEs information objects
       */
      union {
         /**
          * id: id-E2nodeComponentConfigAddition-Item
          * criticality: e2ap_reject
          * presence: e2ap_mandatory
          */
         e2ap_E2nodeComponentConfigAddition_Item *_e2ap_E2nodeComponentConfigAddition_ItemIEs_id_E2nodeComponentConfigAddition_Item;

         ASN1OpenType* extElem1;
      } u;
   } value;
} e2ap_E2nodeComponentConfigAddition_ItemIEs;

EXTERN int asn1PE_e2ap_E2nodeComponentConfigAddition_ItemIEs (OSCTXT* pctxt, e2ap_E2nodeComponentConfigAddition_ItemIEs* pvalue);

EXTERN int asn1PD_e2ap_E2nodeComponentConfigAddition_ItemIEs (OSCTXT* pctxt, e2ap_E2nodeComponentConfigAddition_ItemIEs* pvalue);

EXTERN void asn1Print_e2ap_E2nodeComponentConfigAddition_ItemIEs(const char* name, const e2ap_E2nodeComponentConfigAddition_ItemIEs* pvalue);

EXTERN int asn1PrtToStr_e2ap_E2nodeComponentConfigAddition_ItemIEs (const char* name,
 e2ap_E2nodeComponentConfigAddition_ItemIEs* pvalue, char* buffer, OSSIZE bufSize);

EXTERN int asn1PrtToStrm_e2ap_E2nodeComponentConfigAddition_ItemIEs (OSCTXT *pctxt,
   const char* name, const e2ap_E2nodeComponentConfigAddition_ItemIEs* pvalue);

EXTERN int asn1Copy_e2ap_E2nodeComponentConfigAddition_ItemIEs (OSCTXT* pctxt,
   const e2ap_E2nodeComponentConfigAddition_ItemIEs* pSrcValue,
   e2ap_E2nodeComponentConfigAddition_ItemIEs* pDstValue);

EXTERN int asn1Init_e2ap_E2nodeComponentConfigAddition_ItemIEs (
   e2ap_E2nodeComponentConfigAddition_ItemIEs* pvalue);
```



```
EXTERN int asn1PE_e2ap_E2nodeComponentConfigAddition_ItemIEs (OSCTXT* pctxt, e2ap_E2nodeComponentConfigAddition_ItemIEs* pvalue)
{
   int stat = 0;

   /* encode id */

   RTXCTXTPUSHELEMNAME (pctxt, "id");

   stat = asn1PE_e2ap_ProtocolIE_ID (pctxt, pvalue->id);
   if (stat != 0) return LOG_RTERR (pctxt, stat);

   RTXCTXTPOPELEMNAME (pctxt);

   /* encode criticality */

   RTXCTXTPUSHELEMNAME (pctxt, "criticality");

   stat = asn1PE_e2ap_Criticality (pctxt, pvalue->criticality);
   if (stat != 0) return LOG_RTERR (pctxt, stat);

   RTXCTXTPOPELEMNAME (pctxt);

   /* encode value */

   RTXCTXTPUSHELEMNAME (pctxt, "value");

   { OSCTXT lctxt;
   OSOCTET* pDynamicEncodeBuffer;
   ASN1OpenType openType;
   OSBOOL encoded = TRUE;

   openType.numocts = 0;
   openType.data = 0;

   rtxCopyContext (&lctxt, pctxt);
   pctxt->pStream = 0;

   stat = rtxInitContextBuffer (pctxt, 0, 0);
   if (stat != 0) return LOG_RTERR (pctxt, stat);

   switch (pvalue->value.t) {
   /* _e2ap_E2nodeComponentConfigAddition_ItemIEs_id_E2nodeComponentConfigAddition_Item */
   case T_E2AP_PDU_Contents_e2ap_E2nodeComponentConfigAddition_ItemIEs_id_E2nodeComponentConfigAddition_Item:
      RTXCTXTPUSHELEMNAME (pctxt, "_e2ap_E2nodeComponentConfigAddition_ItemIEs_id_E2nodeComponentConfigAddition_Item");

      stat = asn1PE_e2ap_E2nodeComponentConfigAddition_Item (pctxt, pvalue->value.u._e2ap_E2nodeComponentConfigAddition_ItemIEs_id_E2nodeComponentConfigAddition_Item);
      if (stat != 0) return LOG_RTERR (pctxt, stat);

      RTXCTXTPOPELEMNAME (pctxt);
```

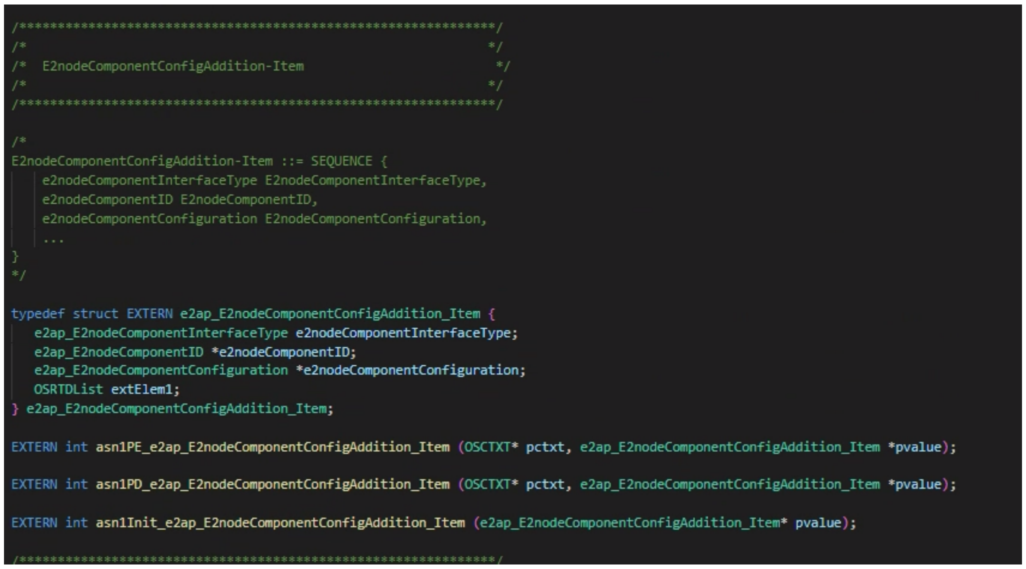


```
   stat = rtxInitContextBuffer (pctxt, 0, 0);
   if (stat != 0) return LOG_RTERR (pctxt, stat);

   switch (pvalue->value.t) {
   /* _e2ap_E2nodeComponentConfigAddition_ItemIEs_id_E2nodeComponentConfigAddition_Item */
   case T_E2AP_PDU_Contents_e2ap_E2nodeComponentConfigAddition_ItemIEs_id_E2nodeComponentConfigAddition_Item:
      RTXCTXTPUSHELEMNAME (pctxt, "_e2ap_E2nodeComponentConfigAddition_ItemIEs_id_E2nodeComponentConfigAddition_Item");

      stat = asn1PE_e2ap_E2nodeComponentConfigAddition_Item (pctxt, pvalue->value.u._e2ap_E2nodeComponentConfigAddition_ItemIEs_id_E2nodeComponentConfigAddition_Item);
      if (stat != 0) return LOG_RTERR (pctxt, stat);

      RTXCTXTPOPELEMNAME (pctxt);
      break;

   case T_E2AP_PDU_Contents_e2ap_E2nodeComponentConfigAddition_ItemIEs_UNDEF_:
      if (0 != pvalue->value.u.extElem1) {
         openType.numocts = pvalue->value.u.extElem1->numocts;
         openType.data = pvalue->value.u.extElem1->data;
      }
      encoded = FALSE;
      break;

   default:
      encoded = FALSE;
      stat = RTERR_INVOPT;
   }

   if (encoded) {
      openType.numocts = (OSUINT32) pe_GetMsgLen (pctxt);
      openType.data = pDynamicEncodeBuffer = pctxt->buffer.data;
   }

   rtxCopyContext (pctxt, &lctxt);

   if (0 == stat) {
      stat = pe_OpenType (pctxt, openType.numocts, openType.data);
   }

   /* Free dynamic encode buffer */
   if (encoded) {
      rtxMemFreePtr (pctxt, pDynamicEncodeBuffer);
   }}

   if (stat != 0) return LOG_RTERR (pctxt, stat);

   RTXCTXTPOPELEMNAME (pctxt);

   return (stat);
}
```

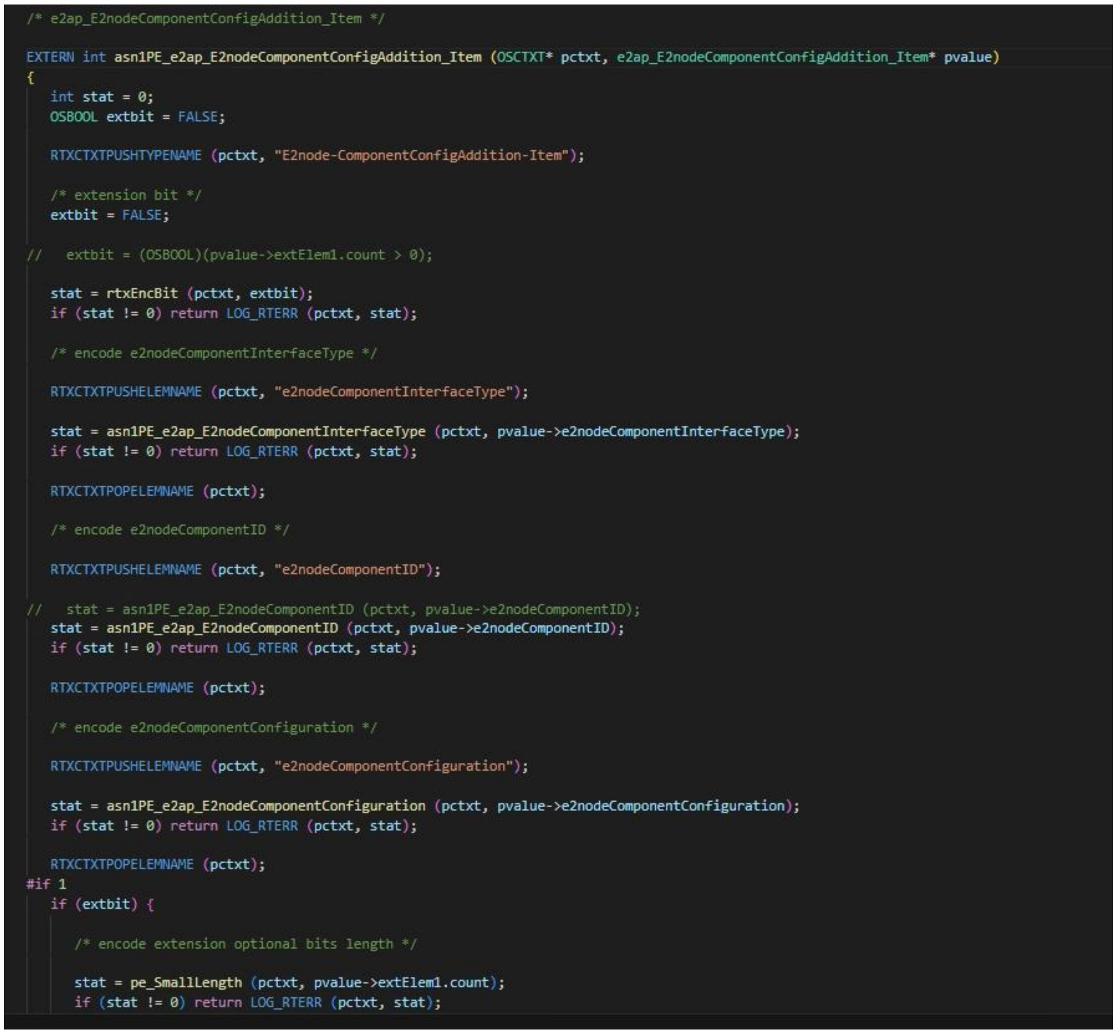


```c
EXTERN int asn1PD_e2ap_E2nodeComponentConfigAddition_ItemIEs (OSCTXT* pctxt, e2ap_E2nodeComponentConfigAddition_ItemIEs* pvalue)
        }
        pvalue->value.u.extElem1->numocts = openTypeLen;
        pvalue->value.u.extElem1->data = pdata;
        }
        break;
    }

    {
        size_t bitEndOffset = PU_GETCTXTBITOFFSET (pctxt);
        size_t bitsConsumed = bitEndOffset - bitStartOffset;
        if (bitsConsumed < bitLength) {
            stat = pd_moveBitCursor (pctxt, (int)(bitLength - bitsConsumed));
        }
        else stat = (bitsConsumed > bitLength) ? ASN_E_INVLEN : 0;
    }}
    if (stat != 0) return LOG_RTERR (pctxt, stat);

    RTXCTXTPOPELEMNAME (pctxt);

    return (stat);
}

int asn1Init_e2ap_E2nodeComponentConfigAddition_ItemIEs(
    e2ap_E2nodeComponentConfigAddition_ItemIEs* pvalue)
{
    if (0 == pvalue) return RTERR_NULLPTR;
    OSCRTLMEMSET(&pvalue->value, 0, sizeof(pvalue->value));
    return 0;
}

void asn1Free_e2ap_E2nodeComponentConfigAddition_ItemIEs(
    OSCTXT* pctxt,
    e2ap_E2nodeComponentConfigAddition_ItemIEs* pvalue)
{
    if (0 == pvalue) return;
    switch (pvalue->value.t) {
        case T_E2AP_PDU_Contents_e2ap_E2nodeComponentConfigAddition_ItemIEs_id_E2nodeComponentConfigAddition_Item:
            if (0 != pvalue->value.u._e2ap_E2nodeComponentConfigAddition_ItemIEs_id_E2nodeComponentConfigAddition_Item) {
                asn1Free_e2ap_E2nodeComponentConfigAddition_Item(
                    pctxt,
                    pvalue->value.u._e2ap_E2nodeComponentConfigAddition_ItemIEs_id_E2nodeComponentConfigAddition_Item);
                rtxMemFreePtr(pctxt, (void*)pvalue->value.u._e2ap_E2nodeComponentConfigAddition_ItemIEs_id_E2nodeComponentConfigAddition_Item);
                pvalue->value.u._e2ap_E2nodeComponentConfigAddition_ItemIEs_id_E2nodeComponentConfigAddition_Item = 0;
            }
            break;
        default:;
    }
}
```

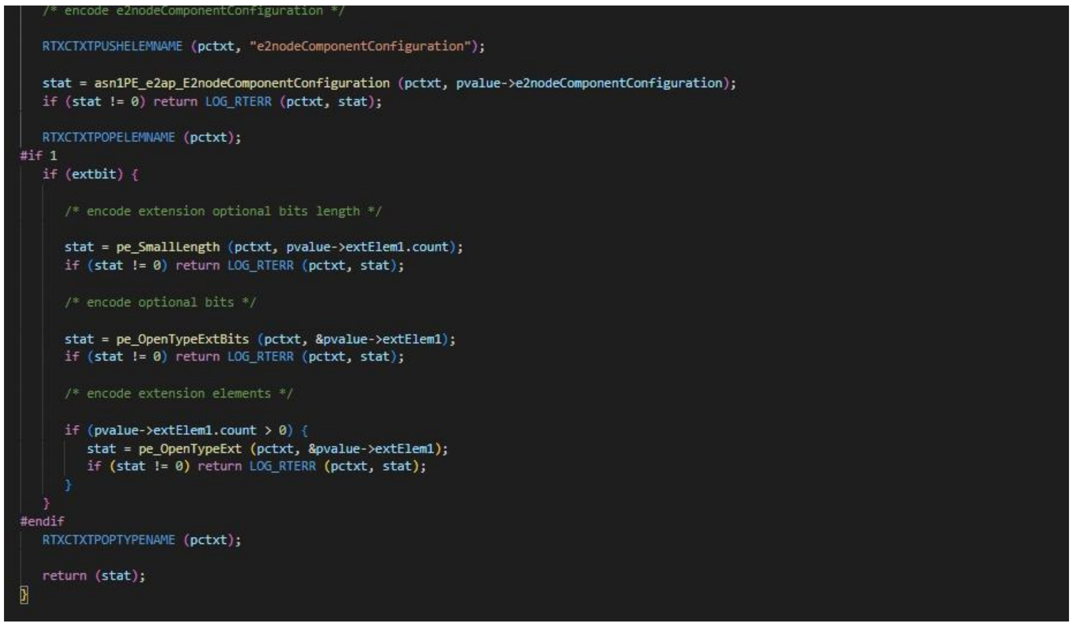
```
E2nodeComponentConfigAddition-Item ::= SEQUENCE {
 e2nodeComponentInterfaceType  E2nodeComponentInterfaceType,
 e2nodeComponentID    E2nodeComponentID,
 e2nodeComponentConfiguration  E2nodeComponentConfiguration,
 ...
}
```



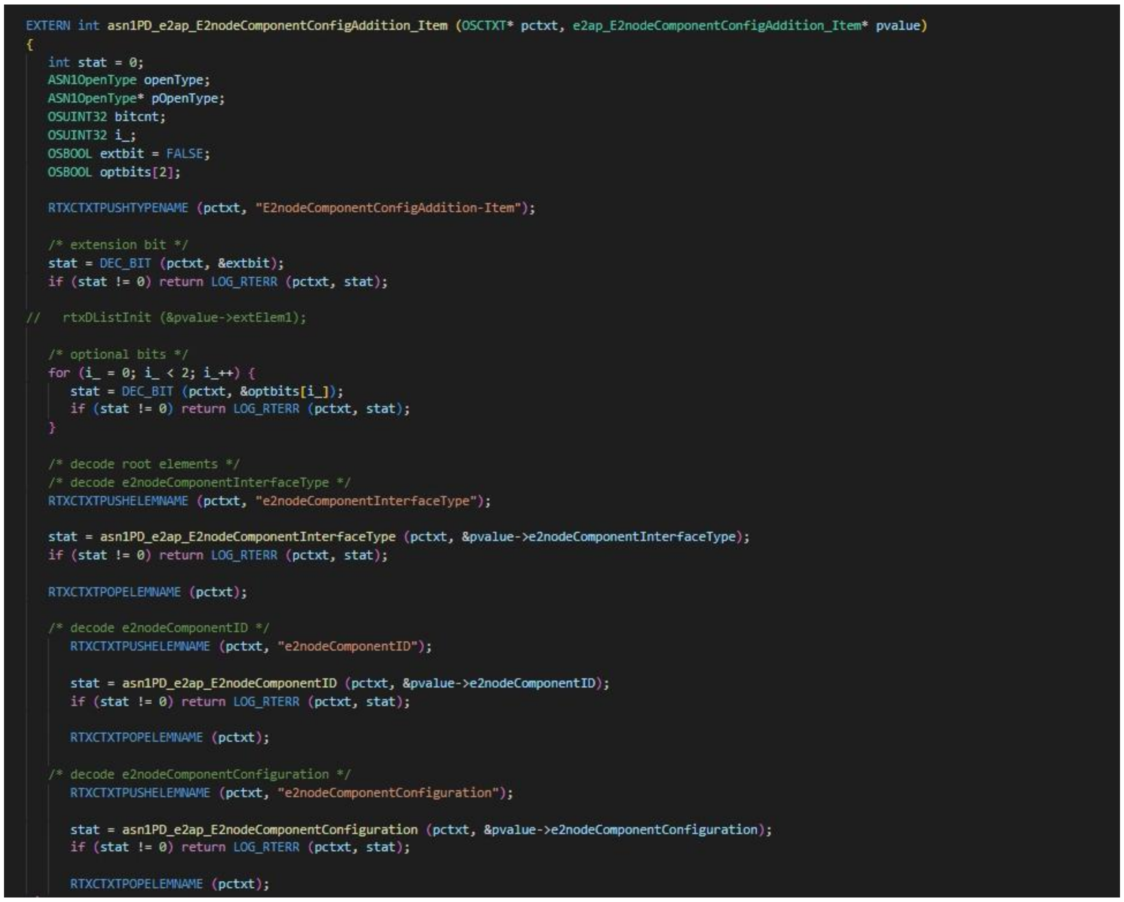
```
/***************************************************************/
/*                                                             */
/*  E2nodeComponentConfigAddition-Item                          */
/*                                                             */
/***************************************************************/

/*
E2nodeComponentConfigAddition-Item ::= SEQUENCE {
    e2nodeComponentInterfaceType E2nodeComponentInterfaceType,
    e2nodeComponentID E2nodeComponentID,
    e2nodeComponentConfiguration E2nodeComponentConfiguration,
    ...
}
*/

typedef struct EXTERN e2ap_E2nodeComponentConfigAddition_Item {
   e2ap_E2nodeComponentInterfaceType e2nodeComponentInterfaceType;
   e2ap_E2nodeComponentID *e2nodeComponentID;
   e2ap_E2nodeComponentConfiguration *e2nodeComponentConfiguration;
   OSRTDList extElem1;
} e2ap_E2nodeComponentConfigAddition_Item;

EXTERN int asn1PE_e2ap_E2nodeComponentConfigAddition_Item (OSCTXT* pctxt, e2ap_E2nodeComponentConfigAddition_Item *pvalue);

EXTERN int asn1PD_e2ap_E2nodeComponentConfigAddition_Item (OSCTXT* pctxt, e2ap_E2nodeComponentConfigAddition_Item *pvalue);

EXTERN int asn1Init_e2ap_E2nodeComponentConfigAddition_Item (e2ap_E2nodeComponentConfigAddition_Item* pvalue);

/***************************************************************/
```



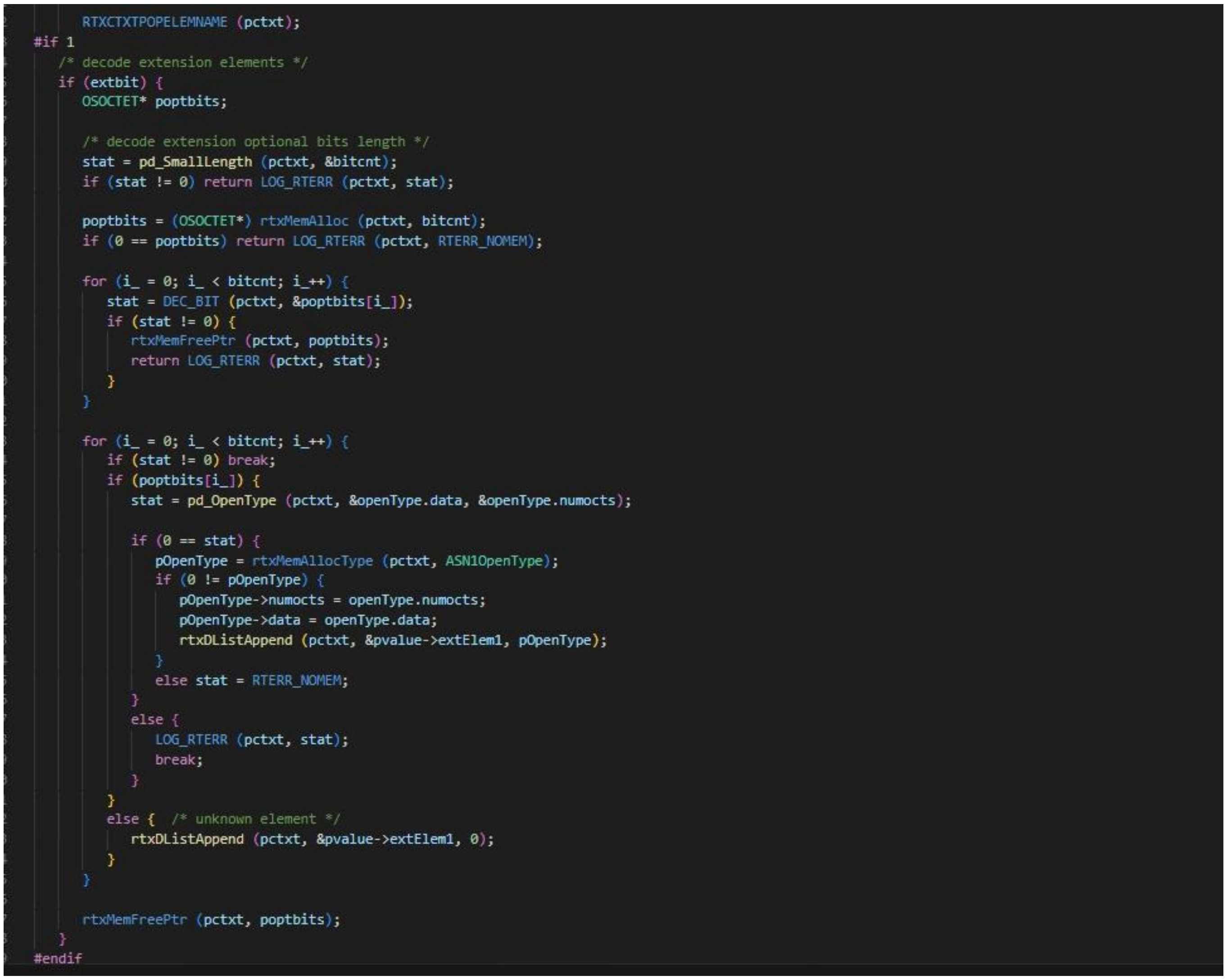
```
/* e2ap_E2nodeComponentConfigAddition_Item */

EXTERN int asn1PE_e2ap_E2nodeComponentConfigAddition_Item (OSCTXT* pctxt, e2ap_E2nodeComponentConfigAddition_Item* pvalue)
{
   int stat = 0;
   OSBOOL extbit = FALSE;

   RTXCTXTPUSHTYPENAME (pctxt, "E2node-ComponentConfigAddition-Item");

   /* extension bit */
   extbit = FALSE;

//   extbit = (OSBOOL)(pvalue->extElem1.count > 0);

   stat = rtxEncBit (pctxt, extbit);
   if (stat != 0) return LOG_RTERR (pctxt, stat);

   /* encode e2nodeComponentInterfaceType */

   RTXCTXTPUSHELEMNAME (pctxt, "e2nodeComponentInterfaceType");

   stat = asn1PE_e2ap_E2nodeComponentInterfaceType (pctxt, pvalue->e2nodeComponentInterfaceType);
   if (stat != 0) return LOG_RTERR (pctxt, stat);

   RTXCTXTPOPELEMNAME (pctxt);

   /* encode e2nodeComponentID */

   RTXCTXTPUSHELEMNAME (pctxt, "e2nodeComponentID");

//   stat = asn1PE_e2ap_E2nodeComponentID (pctxt, pvalue->e2nodeComponentID);
   stat = asn1PE_e2ap_E2nodeComponentID (pctxt, pvalue->e2nodeComponentID);
   if (stat != 0) return LOG_RTERR (pctxt, stat);

   RTXCTXTPOPELEMNAME (pctxt);

   /* encode e2nodeComponentConfiguration */

   RTXCTXTPUSHELEMNAME (pctxt, "e2nodeComponentConfiguration");

   stat = asn1PE_e2ap_E2nodeComponentConfiguration (pctxt, pvalue->e2nodeComponentConfiguration);
   if (stat != 0) return LOG_RTERR (pctxt, stat);

   RTXCTXTPOPELEMNAME (pctxt);
#if 1
   if (extbit) {

      /* encode extension optional bits length */

      stat = pe_SmallLength (pctxt, pvalue->extElem1.count);
      if (stat != 0) return LOG_RTERR (pctxt, stat);
```

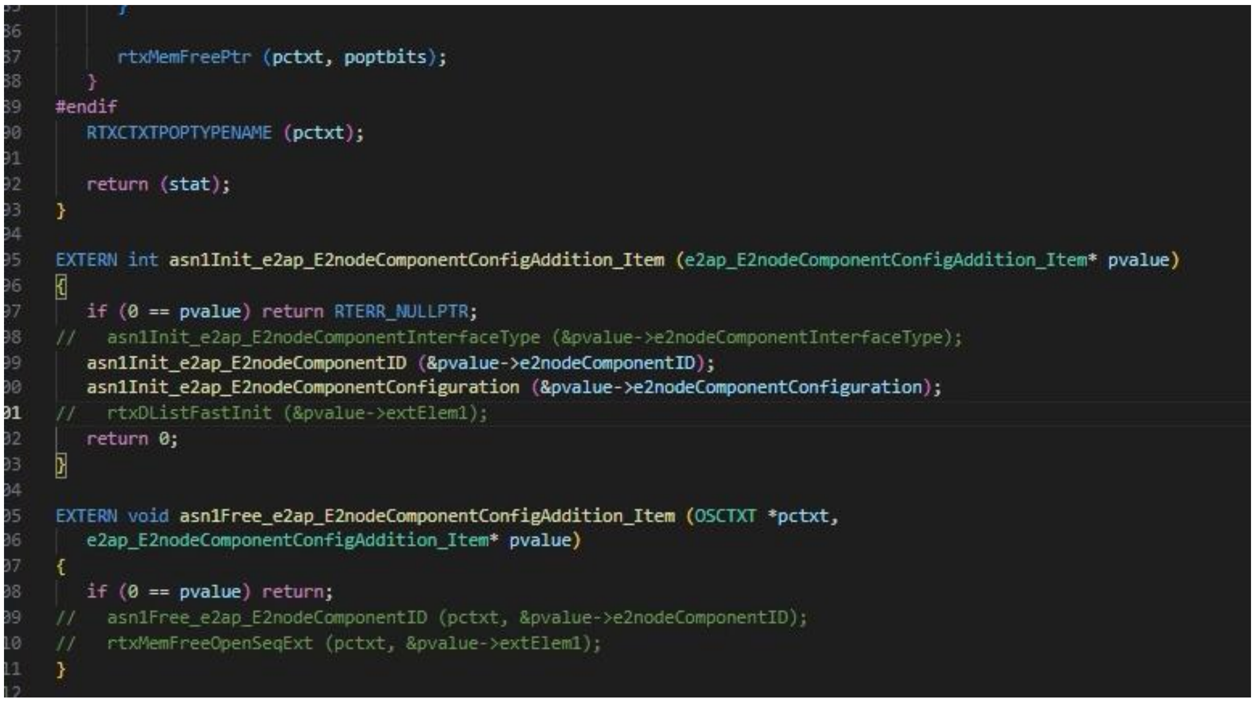


```
   /* encode e2nodeComponentConfiguration */

   RTXCTXTPUSHELEMNAME (pctxt, "e2nodeComponentConfiguration");

   stat = asn1PE_e2ap_E2nodeComponentConfiguration (pctxt, pvalue->e2nodeComponentConfiguration);
   if (stat != 0) return LOG_RTERR (pctxt, stat);

   RTXCTXTPOPELEMNAME (pctxt);
#if 1
   if (extbit) {

      /* encode extension optional bits length */

      stat = pe_SmallLength (pctxt, pvalue->extElem1.count);
      if (stat != 0) return LOG_RTERR (pctxt, stat);

      /* encode optional bits */

      stat = pe_OpenTypeExtBits (pctxt, &pvalue->extElem1);
      if (stat != 0) return LOG_RTERR (pctxt, stat);

      /* encode extension elements */

      if (pvalue->extElem1.count > 0) {
         stat = pe_OpenTypeExt (pctxt, &pvalue->extElem1);
         if (stat != 0) return LOG_RTERR (pctxt, stat);
      }
   }
#endif
   RTXCTXTPOPTYPENAME (pctxt);

   return (stat);
}
```



```c
EXTERN int asn1PD_e2ap_E2nodeComponentConfigAddition_Item (OSCTXT* pctxt, e2ap_E2nodeComponentConfigAddition_Item* pvalue)
{
   int stat = 0;
   ASN1OpenType openType;
   ASN1OpenType* pOpenType;
   OSUINT32 bitcnt;
   OSUINT32 i_;
   OSBOOL extbit = FALSE;
   OSBOOL optbits[2];

   RTXCTXTPUSHTYPENAME (pctxt, "E2nodeComponentConfigAddition-Item");

   /* extension bit */
   stat = DEC_BIT (pctxt, &extbit);
   if (stat != 0) return LOG_RTERR (pctxt, stat);

//   rtxDListInit (&pvalue->extElem1);

   /* optional bits */
   for (i_ = 0; i_ < 2; i_++) {
      stat = DEC_BIT (pctxt, &optbits[i_]);
      if (stat != 0) return LOG_RTERR (pctxt, stat);
   }

   /* decode root elements */
   /* decode e2nodeComponentInterfaceType */
   RTXCTXTPUSHELEMNAME (pctxt, "e2nodeComponentInterfaceType");

   stat = asn1PD_e2ap_E2nodeComponentInterfaceType (pctxt, &pvalue->e2nodeComponentInterfaceType);
   if (stat != 0) return LOG_RTERR (pctxt, stat);

   RTXCTXTPOPELEMNAME (pctxt);

   /* decode e2nodeComponentID */
      RTXCTXTPUSHELEMNAME (pctxt, "e2nodeComponentID");

      stat = asn1PD_e2ap_E2nodeComponentID (pctxt, &pvalue->e2nodeComponentID);
      if (stat != 0) return LOG_RTERR (pctxt, stat);

      RTXCTXTPOPELEMNAME (pctxt);

   /* decode e2nodeComponentConfiguration */
      RTXCTXTPUSHELEMNAME (pctxt, "e2nodeComponentConfiguration");

      stat = asn1PD_e2ap_E2nodeComponentConfiguration (pctxt, &pvalue->e2nodeComponentConfiguration);
      if (stat != 0) return LOG_RTERR (pctxt, stat);

      RTXCTXTPOPELEMNAME (pctxt);
```


```
      RTXCTXTPOPELEMNAME (pctxt);
#if 1
   /* decode extension elements */
   if (extbit) {
      OSOCTET* poptbits;

      /* decode extension optional bits length */
      stat = pd_SmallLength (pctxt, &bitcnt);
      if (stat != 0) return LOG_RTERR (pctxt, stat);

      poptbits = (OSOCTET*) rtxMemAlloc (pctxt, bitcnt);
      if (0 == poptbits) return LOG_RTERR (pctxt, RTERR_NOMEM);

      for (i_ = 0; i_ < bitcnt; i_++) {
         stat = DEC_BIT (pctxt, &poptbits[i_]);
         if (stat != 0) {
            rtxMemFreePtr (pctxt, poptbits);
            return LOG_RTERR (pctxt, stat);
         }
      }

      for (i_ = 0; i_ < bitcnt; i_++) {
         if (stat != 0) break;
         if (poptbits[i_]) {
            stat = pd_OpenType (pctxt, &openType.data, &openType.numocts);

            if (0 == stat) {
               pOpenType = rtxMemAllocType (pctxt, ASN1OpenType);
               if (0 != pOpenType) {
                  pOpenType->numocts = openType.numocts;
                  pOpenType->data = openType.data;
                  rtxDListAppend (pctxt, &pvalue->extElem1, pOpenType);
               }
               else stat = RTERR_NOMEM;
            }
            else {
               LOG_RTERR (pctxt, stat);
               break;
            }
         }
         else {  /* unknown element */
            rtxDListAppend (pctxt, &pvalue->extElem1, 0);
         }
      }

      rtxMemFreePtr (pctxt, poptbits);
   }
#endif
```



```
            }

            rtxMemFreePtr (pctxt, poptbits);
        }
#endif
    RTXCTXTPOPTYPENAME (pctxt);

    return (stat);
}

EXTERN int asn1Init_e2ap_E2nodeComponentConfigAddition_Item (e2ap_E2nodeComponentConfigAddition_Item* pvalue)
{
    if (0 == pvalue) return RTERR_NULLPTR;
//    asn1Init_e2ap_E2nodeComponentInterfaceType (&pvalue->e2nodeComponentInterfaceType);
    asn1Init_e2ap_E2nodeComponentID (&pvalue->e2nodeComponentID);
    asn1Init_e2ap_E2nodeComponentConfiguration (&pvalue->e2nodeComponentConfiguration);
//    rtxDListFastInit (&pvalue->extElem1);
    return 0;
}

EXTERN void asn1Free_e2ap_E2nodeComponentConfigAddition_Item (OSCTXT *pctxt,
    e2ap_E2nodeComponentConfigAddition_Item* pvalue)
{
    if (0 == pvalue) return;
//    asn1Free_e2ap_E2nodeComponentID (pctxt, &pvalue->e2nodeComponentID);
//    rtxMemFreeOpenSeqExt (pctxt, &pvalue->extElem1);
}
```