

Examples

The following examples are included:

- Serialize Copycode
 - Generated Natural Data Area
 - Natural DTD Parser
 - Generated Type Definition
 - Parser CALLBACK Copycode
-

Serialize Copycode

Using the XML Toolkit, a copycode can be generated that can be used to convert a Natural group structure into an XML document.

The callback copycode takes the following operands:

Operand	Format/Length	Description	from PARSER-X
1	A	ex-XPATH to represent element structure	operand2
2	A1	Type of the XPATH content: ? Processing instruction D DOCTYPE ! Comment C CDATA section T Starting Tag @ Attribute / Close Tag	operand3
3	A	Parsed Data	operand4
4	L	Is TRUE if Parsed Data is empty	operand5
5	I4	Counter Variable 1st Dimension	
6	I4	Counter Variable 2nd Dimension	
7	I4	Counter Variable 3rd Dimension	

Copycode Example EMPL-C:

```

*
----- * Parameter
Definition * * &1& 'XML' /* XML Document * &2& '#CX' /* Counter
Variable 1st Dimension * &3& '#CY' /* Counter Variable 2nd Dimension *
&4& '#CZ' /* Counter Variable 3rd Dimension *
-----
* DTD E-\SAG\nat\NATAPPS\FUSER\XMLTK\RES\empl.dtd COMPRESS &1& '<EMPLOYEE'
' PERSONNEL-ID='EMPLOYEE.PERSONNEL-ID ' ' ' >' INTO &1& LEAVING NO
/* now the children COMPRESS &1& '<FULL-NAME' ' ' >' INTO &1&
LEAVING NO /* now the children COMPRESS &1& '<FIRST-NAME' ' ' >' EMPLOYEE.FIRST-NAME
'</FIRST-NAME>' INTO &1& LEAVING NO COMPRESS &1& '<NAME'
'>' EMPLOYEE.NAME '</NAME>' INTO &1& LEAVING NO /* COMPRESS &1&
'</FULL-NAME>' INTO &1& LEAVING NO COMPRESS &1& '<FULL-ADDRESS'
'>' INTO &1& LEAVING NO /* now the children FOR &2& = 1 TO
EMPLOYEE.C@ADDRESS-LINE COMPRESS &1& '<ADDRESS-LINE' ' ' >' EMPLOYEE.ADDRESS-LINE(&2&)
'</ADDRESS-LINE>' INTO &1& LEAVING NO END-FOR COMPRESS &1&
'<CITY' ' ' >' EMPLOYEE.CITY '</CITY>' INTO &1& LEAVING NO COMPRESS
&1& '<ZIP' ' ' >' EMPLOYEE.ZIP '</ZIP>' INTO &1& LEAVING
NO COMPRESS &1& '<COUNTRY' ' ' >' EMPLOYEE.COUNTRY '</COUNTRY>'
INTO &1& LEAVING NO /* COMPRESS &1& '<FULL-ADDRESS>' INTO
&1& LEAVING NO COMPRESS &1& '<TELEPHONE' ' ' >' INTO &1&
LEAVING NO /* now the children COMPRESS &1& '<PHONE' ' ' >' EMPLOYEE.PHONE
'</PHONE>' INTO &1& LEAVING NO COMPRESS &1& '<AREA-CODE'
'>' EMPLOYEE.AREA-CODE '</AREA-CODE>' INTO &1& LEAVING NO /*
COMPRESS &1& '<TELEPHONE>' INTO &1& LEAVING NO COMPRESS
&1& '<JOB-TITLE' ' ' >' EMPLOYEE.JOB-TITLE '</JOB-TITLE>' INTO
&1& LEAVING NO FOR &2& = 1 TO EMPLOYEE.C@INCOME COMPRESS &1&
'<INCOME' ' ' >' INTO &1& LEAVING NO /* now the children COMPRESS &1&
'<SALARY' ' ' >' EMPLOYEE.SALARY(&2&) '</SALARY>' INTO &1&
LEAVING NO FOR &3& = 1 TO EMPLOYEE.C@BONUS(&2&) COMPRESS &1&
'<BONUS' ' ' >' EMPLOYEE.BONUS(&2&,&3&) '</BONUS>' INTO
&1& LEAVING NO END-FOR /* COMPRESS &1& '</INCOME>' INTO
&1& LEAVING NO END-FOR /* COMPRESS &1& '</EMPLOYEE>' INTO
&1& LEAVING NO

```

Program Example:

```

*
----- * CLASS
NATURAL XML TOOLKIT * * * DESCRIPTION * Serialize a given Data structure. * *
* AUTHOR SAG 01.2006 * * VERSION 6.2. * * (c) Copyright Software AG 2006. All
rights reserved. * * -----
* DEFINE DATA
LOCAL USING EMPL /* add generated data structure LOCAL 1
XML (A) DYNAMIC * 1 OUT (A72) 1 II (I4) * 1 OUTDYN (A) DYNAMIC 1 OBJLEN (I4) 1
OBJEND (I4) 1 OBJSTART (I4) 1 OBJLINE (I4) * 1 #CX (I4) 1 #CY (I4) 1 #CZ (I4)
END-DEFINE * EMPLOYEE.PERSONNEL-ID := 4711 * EMPLOYEE.FIRST-NAME := "ADKINSON"
EMPLOYEE.NAME := "MARTHA" * EMPLOYEE.C@ADDRESS-LINE := 2 EMPLOYEE.ADDRESS-LINE(1)
:= "8603 GARLAND COURT" EMPLOYEE.ADDRESS-LINE(2) := "FRAMINGHAM" EMPLOYEE.ADDRESS-LINE(2)
:= "MA" EMPLOYEE.CITY := "FRAMINGHAM" EMPLOYEE.ZIP := "17010" EMPLOYEE.COUNTRY
:= "USA" * EMPLOYEE.AREA-CODE := "617" EMPLOYEE.PHONE := "210-4703" * EMPLOYEE.JOB-TITLE
:= "MANAGER" EMPLOYEE.C@INCOME := 2 EMPLOYEE.SALARY(1) := 47000 EMPLOYEE.C@BONUS(1)
:= 2 EMPLOYEE.BONUS(1,1) := 10500 EMPLOYEE.BONUS(1,2) := 7875 * EMPLOYEE.SALARY(2)
:= 47000 EMPLOYEE.C@BONUS(2) := 1 EMPLOYEE.BONUS(2,1) := 35700 *
INCLUDE EMPL-C
"XML" "#CX" "#CY" "#CZ" /* add generated Serialize * FOR II = 1 TO *LENGTH(XML)
STEP 72 OUT := SUBSTR(XML,II) WRITE OUT END-FOR * NEWPAGE * /* WRITE COMPLETE
(A) DYNAMIC VARIABLE IF POSSIBLE USE CR AND IGNORE LF OBJSTART := 1 * EXAMINE
xml FOR "><" REPLACE WITH ">" - H'0A' - "<" EXAMINE xml FOR H'0A' GIVING
POSITION OBJEND * REPEAT WHILE OBJEND NE 0 /* IF OBJSTART GT 0 THEN ADD OBJSTART
TO OBJEND END-IF /* OBJLEN := OBJEND - OBJSTART - 1 /* IF OBJLEN > 0 THEN OUTDYN
:= SUBSTRING(xml, OBJSTART, OBJLEN) /* FOR OBJLINE = 1 TO *LENGTH(OUTDYN) STEP
72 OUT := SUBSTR (OUTDYN,OBJLINE) WRITE OUT END-FOR ELSE WRITE " " END-IF /* OBJSTART
:= OBJEND IF OBJSTART GT *LENGTH(xml) ESCAPE BOTTOM END-IF /* EXAMINE SUBSTRING(xml,OBJSTART)
FOR H'0A' GIVING POSITION OBJEND END-REPEAT * END

```

Natural PDA EMPL Used:

```

DEFINE DATA PARAMETER 1 EMPLOYEE 2 ATTRIBUTES_OF_EMPLOYEE
3 PERSONNEL-ID(A8) * 2 FULL-NAME 3 FIRST-NAME(A20) 3 NAME(A20) * 2 FULL-ADDRESS
3 C@ADDRESS-LINE(I4) 3 ADDRESS-LINE(A20/1:6) 3 CITY(A20) 3 ZIP(A20) 3 COUNTRY(A3)
* 2 TELEPHONE 3 AREA-CODE(A6) 3 PHONE(A15) * 2 JOB-TITLE(A25) * 2 C@INCOME(I4)
2 INCOME(1:6) 3 SALARY(A9) 3 C@BONUS(I4) 3 BONUS(A9/1:4) END-DEFINE

```

Generated Natural Data Area

Using the XML Toolkit, a Natural Data Area, or more precisely a Local Data Area, Parameter Data Area or Global Data Area, can be generated that represents a given Document Type Definition.

Generation Rules:

- Each Empty Element without Attributes (<!ELEMENT br EMPTY>) is generated as a Natural variable of Type B1. This is necessary, because empty Natural groups are not allowed.
- Each Empty Element with Attributes (<!ELEMENT br EMPTY><!ATTLIST br width CDATA #IMPLIED>) is generated as a Natural group.
- Each Element with content (<!ELEMENT b (#PCDATA)>) is generated as a Natural variable of type A253.
- Each Sequence of Elements (<!ELEMENT spec (front, body*, back?)>) or Choice of Elements (<!ELEMENT div1 (p | list | note)>) is generated as a Natural group.
- Each clasped Sequence or Choice (<!ELEMENT address ((street, housenumber), (zip, city))>) is generated as a special group with the name prefix "##PSEUDO". This gives the possibility to represent the context or possible multiplicities.
- Each Attribute (<!ATTLIST br width CDATA #IMPLIED>) of an Element is generated as variable of Type A253 belonging to a group with the name prefix "ATTRIBUTES_OF_" followed by the name of the element.
- Multiple Elements are always generated as arrays of Dimension 1:v. The upper bound of the generated array has to be changed manually.
- If an Element is defined multiple (<!ELEMENT spec (front, body*)>), an additional counter field C@BODY, is generated to specify the number of available elements.
- All names used inside the DTD are converted into upper case, because Natural names are not case sensitive. Duplicate names inside a generated group will be extended with a suffix to make the names unique.
- Special Characters not valid for Natural names are converted into valid Natural names. For the conversion settings, see the option dialog of the XML Toolkit.

Restrictions:

- Elements with Mixed content data (<!ELEMENT p (#PCDATA | a | ul | b | i | em)*>) are not supported.

- DTDs that result in Natural data structures can not be used within Natural, because Natural only supports data structures with a maximum of three dimensions.

Example DTD:

```

<!ELEMENT EMPLOYEE (FULL-NAME , FULL-ADDRESS , TELEPHONE ,JOB-TITLE, INCOME* )>
<!ATTLIST EMPLOYEE PERSONNEL-ID CDATA #REQUIRED >

<!ELEMENT FULL-NAME (FIRST-NAME , NAME )>
<!ELEMENT FIRST-NAME (#PCDATA )>
<!ELEMENT NAME (#PCDATA )>

<!ELEMENT FULL-ADDRESS (ADDRESS-LINE* , CITY , ZIP , COUNTRY )>
<!ELEMENT ADDRESS-LINE (#PCDATA )>
<!ELEMENT CITY (#PCDATA )>
<!ELEMENT ZIP (#PCDATA )>
<!ELEMENT COUNTRY (#PCDATA )>

<!ELEMENT TELEPHONE (PHONE , AREA-CODE )>
<!ELEMENT PHONE (#PCDATA )>
<!ELEMENT AREA-CODE (#PCDATA )>

<!ELEMENT JOB-TITLE (#PCDATA )>

<!ELEMENT INCOME (SALARY , BONUS* )>
<!ELEMENT SALARY (#PCDATA )>
<!ELEMENT BONUS (#PCDATA )>

```

Generated Natural Data Area (*italic* written parts of the DTD, but necessary for Natural):

```

DEFINE DATA PARAMETER
1 EMPLOYEE
  2 ATTRIBUTES_OF_EMPLOYEE
    3 PERSONNEL-ID(A253)
  *
  2 FULL-NAME
    3 FIRST-NAME(A253)
    3 NAME(A253)
  *
  2 FULL-ADDRESS
    3 C@ADDRESS-LINE(I4)
    3 ADDRESS-LINE(A253/1:v)
    3 CITY(A253)
    3 ZIP(A253)
    3 COUNTRY(A253)
  *
  2 TELEPHONE
    3 AREA-CODE(A253)
    3 PHONE(A253)
  *
  2 JOB-TITLE(A253)
  *
  2 C@INCOME(I4)
  2 INCOME(1:v)
    3 SALARY(A253)
    3 C@BONUS(I4)
    3 BONUS(A253/1:v)
END-DEFINE

```

Natural DTD Parser

Translation Rules:

Natural	Document Type Definiton
1 G1 2 E1 (A–)	<!ELEMENT G1 (E1)> <!ELEMENT E1 (#PCDATA)>
1 G1 2 E1 (A–) 2 E2 (A–) 2 E3 (A–)	<!ELEMENT G1 (E1, E2, E3)> <!ELEMENT E1 (#PCDATA)> <!ELEMENT E2 (#PCDATA)> <!ELEMENT E3 (#PCDATA)>
1 C@E1_MAX (I4) CONST <10> 1 G1 2 C@E1 (I4) 2 E1 (A–/1:C@E1_MAX)	<!ELEMENT G1 (E1*)> <!ELEMENT E1 (#PCDATA)>
1 C@E1_MAX (I4) CONST <10> 1 G1 2 C@E1 (I4) 2 E1 (A–/1:C@E1_MAX)	<!ELEMENT G1 (E1+)> <!ELEMENT E1 (#PCDATA)>
1 G1 2 E1 (A–)	<!ELEMENT G1 (E1?)> <!ELEMENT E1 (#PCDATA)>
1 G1 2 E1 (A–) 2 E2 (A–) 2 E3 (A–)	<!ELEMENT G1 (E1 E2 E3)> <!ELEMENT E1 (#PCDATA)> <!ELEMENT E2 (#PCDATA)> <!ELEMENT E3 (#PCDATA)>
1 G1 2 E1 (A–) 2 E2 (A–) 2 G2 2 E1_2 (A–) 2 E3 (A–)	<!ELEMENT G1 (E1, E2, G2)> <!ELEMENT E1 (#PCDATA)> <!ELEMENT E2 (#PCDATA)> <!ELEMENT G2 (E1, E3)> <!ELEMENT E3 (#PCDATA)>
1 #G1 2 #E1 (A–)	<!ELEMENT G1 (E1)> <!ELEMENT E1 (#PCDATA)>
2 E1 (A–) 3 ATTRIBUTES_OF_E1 4 A1 (A–) CONST <'schema'> 4 A2 (A–) 4 A3 (A–)	<!ELEMENT E1 (#PCDATA)> <!ATTLIST E1 A1 #FIXED "schema" A2 NMTOKEN #IMPLIED A3 ID #REQUIRED>

Generated Type Definition

Using the XML Toolkit, a Natural Data Area, or more precisely a Local Data Area, Parameter Data Area or Global Data Area, can be used to generate a Document Type Definition.

Generation Rules:

- A Natural variable will result in an element with content.
- A Natural group will result in a sequence of elements.
- Multiple variables or groups will be generated with multiplicity "zero or more".
- Special characters not valid for XML names are converted into valid names. For the conversion settings, see the options screen of the XML Toolkit.

Example Natural Data Area:

```

DEFINE DATA LOCAL
1 NAT$EMPLOYEE
  2 ATTRIBUTES_OF_NAT$EMPLOYEE
    3 PERSONNEL/ID(A8)
  2 C@MAN@WORK(I4)
  2 MAN@WORK
    3 JOB(A10)
  2 A$TEST$MAKL(I4)
  2 AS/FA/SD(P7.5)
  2 #ASDFAS(F4)
  2 ASF#AS(N9)
  2 A-SF-D(A) Dynamic
  2 INC@OME(1:6)
    3 C@BONUS(I4)
    3 BONUS(A9/1:4)
END-DEFINE

```

Generated DTD:

```

<!-- DTD XMLTOOLS BEISP -->
<!ELEMENT NATdollarEMPLOYEE ( MANatWORK , AdollarTESTdollarMAKL ,
      ASslashFAslashSD , hashASDFAS , ASFhashAS , A-SF-D , INCatOME* ) >
<!ATTLIST NATdollarEMPLOYEE PERSONNELslashID CDATA #IMPLIED >
<!ELEMENT MANatWORK ( JOB ) >
<!ELEMENT JOB (#PCDATA) >
<!ELEMENT AdollarTESTdollarMAKL (#PCDATA) >
<!ELEMENT ASslashFAslashSD (#PCDATA) >
<!ELEMENT hashASDFAS (#PCDATA) >
<!ELEMENT ASFhashAS (#PCDATA) >
<!ELEMENT A-SF-D (#PCDATA) >
<!ELEMENT INCatOME ( BONUS* ) >
<!ELEMENT BONUS (#PCDATA) >

```

Parser CALLBACK Copycode

Using the XML Toolkit, a copycode can be generated that can be used with the Natural Simple XML Parser.

The callback copycode takes the following operands:

Operand	Format/Length	Description	from PARSEr-X
1	A	ex-XPATH to represent element structure	operand2
2	A1	Type of the XPATH content: ? Processing instruction D DOCTYPE ! Comment C CDATA section T Starting Tag @ Attribute / Close Tag	operand3
3	A	Content of found element	operand4
4	L	Is TRUE if Parsed Data is empty	operand5
5	I4	Counter Variable 1st Dimension	
6	I4	Counter Variable 2nd Dimension	
7	I4	Counter Variable 3rd Dimension	

Copycode Example EMPL-P:

```

* -----
* Parameter Definition
*
* &1& 'XML_PARSER_XPATH'           /* XPATH to represent element...
* &2& 'XML_PARSER_XPATH_TYPE'      /* Type of the XPATH:
*                                     ? Processing instruction
*                                     D DOCTYPE
*                                     ! Comment
*                                     C CDATA section
*                                     T Starting Tag
*                                     @ Attribute
*                                     / Close Tag
*                                     $ Parsed Data
* &3& 'XML_PARSER_CONTENT'         /* Content of found element
* &4& 'XML_PARSER_CONTENT_IS_EMPTY' /* Is TRUE if Content is empty
* &5& '#CX'                         /* Counter Variable 1st Dimension
* &6& '#CY'                         /* Counter Variable 2nd Dimension
* &7& '#CZ'                         /* Counter Variable 3rd Dimension
* -----
*
DECIDE ON FIRST &1&
VALUE 'EMPLOYEE'
RESET EMPLOYEE
VALUE 'EMPLOYEE/@PERSONNEL-ID'
/* #REQUIRED
EMPLOYEE.PERSONNEL-ID := &3&
VALUE 'EMPLOYEE/FULL-NAME'
IGNORE
VALUE 'EMPLOYEE/FULL-NAME/FIRST-NAME'
IGNORE

```

```

VALUE 'EMPLOYEE/FULL-NAME/FIRST-NAME/$'
  EMPLOYEE.FIRST-NAME := &3&
VALUE 'EMPLOYEE/FULL-NAME/NAME'
  IGNORE
VALUE 'EMPLOYEE/FULL-NAME/NAME/$'
  EMPLOYEE.NAME := &3&
VALUE 'EMPLOYEE/FULL-ADDRESS'
  IGNORE
VALUE 'EMPLOYEE/FULL-ADDRESS/ADDRESS-LINE'
  /* OPTIONAL MULTIPLE IST: 18 PARENT: FULL-ADDRESS
  ADD 1 TO EMPLOYEE.C@ADDRESS-LINE
VALUE 'EMPLOYEE/FULL-ADDRESS/ADDRESS-LINE/$'
  &5& := EMPLOYEE.C@ADDRESS-LINE
  EMPLOYEE.ADDRESS-LINE(&5&) := &3&
VALUE 'EMPLOYEE/FULL-ADDRESS/CITY'
  IGNORE
VALUE 'EMPLOYEE/FULL-ADDRESS/CITY/$'
  EMPLOYEE.CITY := &3&
VALUE 'EMPLOYEE/FULL-ADDRESS/ZIP'
  IGNORE
VALUE 'EMPLOYEE/FULL-ADDRESS/ZIP/$'
  EMPLOYEE.ZIP := &3&
VALUE 'EMPLOYEE/FULL-ADDRESS/COUNTRY'
  IGNORE
VALUE 'EMPLOYEE/FULL-ADDRESS/COUNTRY/$'
  EMPLOYEE.COUNTRY := &3&
VALUE 'EMPLOYEE/TELEPHONE'
  IGNORE
VALUE 'EMPLOYEE/TELEPHONE/PHONE'
  IGNORE
VALUE 'EMPLOYEE/TELEPHONE/PHONE/$'
  EMPLOYEE.PHONE := &3&
VALUE 'EMPLOYEE/TELEPHONE/AREA-CODE'
  IGNORE
VALUE 'EMPLOYEE/TELEPHONE/AREA-CODE/$'
  EMPLOYEE.AREA-CODE := &3&
VALUE 'EMPLOYEE/JOB-TITLE'
  IGNORE
VALUE 'EMPLOYEE/JOB-TITLE/$'
  EMPLOYEE.JOB-TITLE := &3&
VALUE 'EMPLOYEE/INCOME'
  /* OPTIONAL MULTIPLE IST: 18 PARENT: EMPLOYEE
  ADD 1 TO EMPLOYEE.C@INCOME
VALUE 'EMPLOYEE/INCOME/SALARY'
  IGNORE
VALUE 'EMPLOYEE/INCOME/SALARY/$'
  &5& := EMPLOYEE.C@INCOME
  EMPLOYEE.SALARY(&5&) := &3&
VALUE 'EMPLOYEE/INCOME/BONUS'
  /* OPTIONAL MULTIPLE IST: 18 PARENT: INCOME
  &5& := EMPLOYEE.C@INCOME
  ADD 1 TO EMPLOYEE.C@BONUS(&5&)
VALUE 'EMPLOYEE/INCOME/BONUS/$'
  &5& := EMPLOYEE.C@INCOME
  &6& := EMPLOYEE.C@BONUS(&5&)
  EMPLOYEE.BONUS(&5&,&6&) := &3&
NONE
IGNORE
END-DECIDE

```


Subprogram Example:

```

* -----
* CLASS NATURAL XML TOOLKIT - UTILITIES
*
*
* DESCRIPTION
*         Parse a given XML document.
*
*
* AUTHOR      SAG    01.2006
*
* VERSION     6.2.
*
* (c) Copyright Software AG 2006. All rights reserved.
* -----
*
DEFINE DATA PARAMETER
1 XML_PARSER_INPUT           (A) DYNAMIC
PARAMETER USING EMPL
PARAMETER
1 XML_PARSER_ERROR_TEXT     (A253)
1 XML_PARSER_RESPONSE       (I2)
*
LOCAL USING PARSE-X
LOCAL
1 XML_PARSER_XPATH          (A) DYNAMIC
1 XML_PARSER_XPATH_TYPE     (A1)
1 XML_PARSER_CONTENT        (A) DYNAMIC
1 XML_PARSER_CONTENT_IS_EMPTY (L)
*
LOCAL
1 #CX                       (I4)
1 #CY                       (I4)
1 #CZ                       (I4)
END-DEFINE
*
* ----- INCLUDE THE PARSE
INCLUDE PARSE_X 'XML_PARSER_INPUT' /* XML file to be parsed
'XML_PARSER_XPATH' /* XPATH to represent element...
'XML_PARSER_XPATH_TYPE' /* Type of callback
'XML_PARSER_CONTENT' /* Content of found element
'XML_PARSER_CONTENT_IS_EMPTY' /* Is TRUE if element is empty
'XML_PARSER_ERROR_TEXT' /* error Message
'XML_PARSER_RESPONSE' /* Error NR; 0 = OK
*
* ----- CALLBACK HANDLER
DEFINE SUBROUTINE CALLBACK
*
INCLUDE EMPL-P 'XML_PARSER_XPATH' /* XPATH to represent element...
'XML_PARSER_XPATH_TYPE' /* Type of callback
'XML_PARSER_CONTENT' /* Content of found element
'XML_PARSER_CONTENT_IS_EMPTY' /* Is TRUE if element is empty
'#CX'
'#CY'
'#CZ'
*
END-SUBROUTINE
/*

```

```
DEFINE SUBROUTINE PARSER_ERROR
IGNORE
END-SUBROUTINE
END
```

Natural PDA EMPL Used:

```
DEFINE DATA PARAMETER
1 EMPLOYEE
  2 ATTRIBUTES_OF_EMPLOYEE
    3 PERSONNEL-ID(A8)
  *
  2 FULL-NAME
    3 FIRST-NAME(A20)
    3 NAME(A20)
  *
  2 FULL-ADDRESS
    3 C@ADDRESS-LINE(I4)
    3 ADDRESS-LINE(A20/1:6)
    3 CITY(A20)
    3 ZIP(A20)
    3 COUNTRY(A3)
  *
  2 TELEPHONE
    3 AREA-CODE(A6)
    3 PHONE(A15)
  *
  2 JOB-TITLE(A25)
  *
  2 C@INCOME(I4)
  2 INCOME(1:6)
    3 SALARY(A9)
    3 C@BONUS(I4)
    3 BONUS(A9/1:4)
END-DEFINE
```