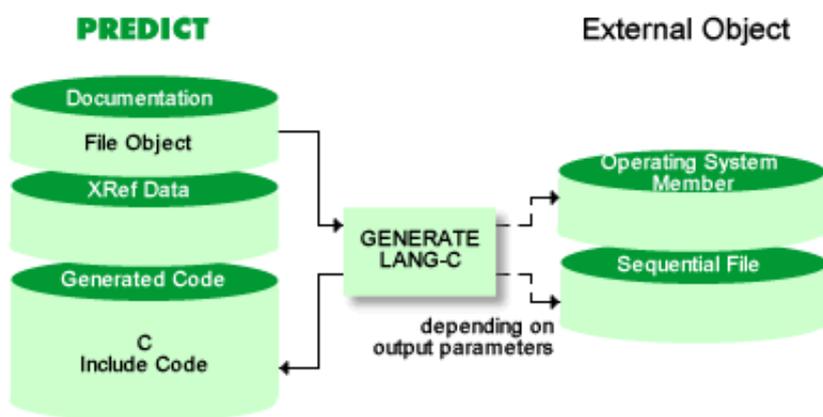


C Include Code

The function Generate C Include Code generates a record buffer for use in C programs based on a Predict file object.

In addition, an Adabas format buffer can also be generated if required for files of the following types:

- Adabas C file (file type A) with parameter Adabas C SQL usage = N
- Adabas C userview (file type U)



Calling the Function

The Generate C Copy Code screen is displayed with function code G and object code CC in a Predict main menu, or with the command GENERATE LANG-C.

```

10:00:41          ***** P R E D I C T  4.2.1  *****          2001-07-12
Plan    0          - Generate C Copy Code -

File ID ..... CHD-A-FILE

Save as member ..... Save in library .... CCCLIB
Overwrite option ..... Y (Y,N) Op. system member ..
Punch / output .....* N

List generated code ..... Y (Y,N) List offsets .....* N
Generate format buffer ....* N ADABAS version ....* I3
Check field name .....* A

Shift increment ..... 3 (0-9) Field name prefix ..
Nr. of abstract lines ..... (0-16) Field name suffix ..
Storage class .....* A Validate ..... _
Upper or lower case .....* L Truncation .....* R
Record buffer name .....
Format buffer name .....

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help Next Stop Last LnKEl Flip Print Impl AdmFi SelFi Prof Main

```

Values for input fields which have been locked by your data dictionary administrator cannot be overwritten. These fields are skipped when positioning the cursor with the TAB key. See Generation Defaults.

Presettings	
The parameters below can be changed in the second Modify C Defaults screen. See Generation Defaults.	
Library system	Library system for which the generated code is punched. Determines which additional cards need to be punched. An operating system member must be entered for the additional cards to be generated. See Parameters Specifying the Form of Output for more information.
Max. name length	Maximum length of name in C.
Signific. length	Number of characters that are used for uniqueness check. May not be greater than Max. name length.
Represent. of I2	Determines how 2-byte integers are represented in the generated code. C unsigned character, 2 bytes S short I int
Represent. of I4	Determines how 4-byte integers are represented in the generated code. C unsigned character, 4 bytes I int L long Note: It is not possible for both I2 and I4 to be represented by int.
Open square bracket	Up to 5 characters used to represent the character [.
Close square bracket	Up to 5 characters used to represent the character].
Open brace	Up to 5 characters used to represent the character {.
Close brace	Up to 5 characters used to represent the character }. Note: All characters are valid apart from the plus sign (+). Characters can be specified in hexadecimal notation. If either the normal or hexadecimal notation for any of these four parameters is changed, the alternative notation must be deleted. Predict then inserts the correct value in either normal or hexadecimal format.
Parameters	

The parameters Save as member, Save in library, Overwrite option, and Op. system member, Punch/output and List generated code are described in section Parameters Specifying the Form of Output.	
File ID	ID of the Predict file object from which the definitions are to be generated.
List offsets	<p>Y Include the offset of each item in the record buffer structure (relative to the beginning of the structure) in decimal and hexadecimal formats as a comment. The total length of each buffer is also included.</p> <p>P Include the absolute position (offset+1) as a comment.</p> <p>L Include the total lengths of the record buffer and the format buffer as a comment.</p> <p>V The file number and the calculated lengths of the record buffer and the format buffer are to be generated as constants in the include code. The name of the file number constant is the record buffer name prefixed by N_. The name of each length constant is the appropriate buffer name prefixed by L_. Each name is prefixed, suffixed, validated and truncated in the same way as any other field name.</p> <p>N No offset.</p>
Generate format buffer	<p>The contents of the format buffer will correspond exactly to the contents of the record buffer. Only valid for files of type A (with parameter Adabas C SQL usage set to N) or for files of type U.</p> <p>Valid values:</p> <p>Y Adabas format buffer is to be generated. Adabas groups, standard formats and lengths are used whenever possible. The resulting format buffers are then as short as possible.</p> <p>F Full format buffer is to be generated. Length and format of Adabas fields are included.</p> <p>N No format buffer is to be generated.</p> <p>Note: If you are generating for a WANG environment, set this parameter to F or N.</p>
Adabas version	The version of Adabas C for which the include code is to be generated. Enter an asterisk for valid values or see table in section Adabas Version for more information.
Check field name	<p>A Type names are checked for uniqueness (against other type names) within the whole structure. Field names must be unique as field names within the entire structure.</p> <p>Y As above, but duplicate field names are only identified as errors if this would result in compiler errors.</p> <p>N No check for duplicate names is performed.</p>
Shift increment	The number of positions to be shifted right when an open brace (or substitute character string, see Presetting above) is encountered.

Field name prefix	The prefix to be used for each name generated.
Nr. of abstract lines	Number of Predict abstract lines per field to be included in the generated code.
Field name suffix	The suffix to be used for each name generated.
Storage class	A Automatic (default) S Static.
Validate	Determines how invalid characters are handled. blank Invalid characters in a field name will result in an error message but will not be modified. rep.char Invalid characters in a field name are replaced by this character. Valid values: A-Z, a-z, 0-9 and _ (underscore). * Invalid characters in a field name are deleted.
Upper or lower case	Case of names in generated code. U upper-case names L lower-case names.
Truncation	Specifies which characters are deleted if a generated field name is too long: L Truncate from the left R Truncate from the right M Truncate from the middle A warning is given if field names are truncated.
Record buffer name	Specifies the name of the record buffer in the generated structure. If omitted, the file ID is used.
Format buffer name	Specifies the name of the format buffer in the generated structure. If omitted, the file ID prefixed by F_ is used.

Generate C Include Code in Batch Mode

Command: GENERATE LANG-C

Enter parameters on next line in positional or keyword form. File ID is obligatory, all other parameters are optional.
If a parameter is not specified, the default value is taken.

Field	Keyword	Position
File ID	FILE-ID	1
Save as member	MEM	2
Save in library	LIB	3
Overwrite option	REPLACE	4
Op. system member	OS-MEMBER	5
Punch / output	PUNCH	6
List generated code	LIST	7
List offsets	OFFSET	8
Generate format buffer	FORMAT-BUFFER	9
Adabas version	ADA-VER	10
Check field name	CHECK-NAME	11
Shift increment	SHIFT-INC	12
Field name prefix	PREFIX	13
Nr. of abstract lines	NR	14
Field name suffix	SUFFIX	15
Storage class	STO-CLASS	16
Validate	VALIDATION	17
Upper or lower case	UPPER-LOWER	18
Truncation	TRUNCATION	19
Record buffer name	RECORD-BUFFER-NAME	20
Format buffer name	FORMAT-BUFFER-NAME	21
Workfile name (see note below)	WORKFILE-NAME	22
If Entire System Server is used		
- Database ID	NP-DBID	23
- Dataset	NP-DSNAME	24
- Volume	NP-VOLSER	25
- Library	NP-LIB	26
- Sublibrary	NP-SUBLIB	27
- Member type	NP-MEMTYPE	28
- VSAM catalog name	NP-VSAMCAT	29

Note:

Parameter Workfile name is obsolete in this version of Predict.

Parameters NP-LIB, NP-SUBLIB and NP-MEMTYPE must be specified if the generated code is written to workfile 1 (Punch/output=Y) and Library system=3.

Names in C Include Code

The following rules apply to the generation of names for C include code.

- Field names are derived from the name of the corresponding Predict field object if no C field name synonym has been specified.
- The case of alpha characters in field and file names is determined by the parameter Upper or lower case.
- The parameters Field name prefix, Field name suffix, Validate and Truncate apply to the generation of field names (see description of parameters above).
- If supplementary fields have to be generated the names of these extra fields are derived from the Predict field names by generating prefixes. The rules for generating names for extra fields are described in the sections below.
- If a generated field name or type name is not unique, an error message may be displayed if Check field name is either set to A or Y.

C Names for Redefinition Fields - RE

When a field is redefined, a data type with structure union is generated. The name of the data type is tu_fieldname. The name of the corresponding variable is u_fieldname. The union consists of

- the redefined field
- one type tn for each redefinition with corresponding variable rn_fieldname.

C Names for Counter Fields (MC or PC)

When generating C fields for Predict fields of type MC or PC, a counter field will automatically be generated in the copy code. The name of this field is derived from the ID of the MC or PC field by adding the prefix C-.

With the following parameter settings

- Field name prefix = Adabas
- Field name suffix = *PERSONNELOFFICE
- Validate = _
- Truncation = R
- Max. name length = 30

the following field names are generated in C include code for field HOURS_DAY of type MC:

<pre>ADABAS_C_HOURS_DAY_PERSONNELOF ADABAS_HOURS_DAY_PERSONNELOFFI</pre>
--

C Names for Additionally Generated Indicator Fields

An additional indicator field prefixed with s_ is generated in the following cases:

- a field is defined with Suppression option set to U (null allowed)
- a field of a file of type A or U is defined with Suppression option set to R (not null)

Note:

For Adabas C fields, the additional indicator field is only generated for the following Adabas versions:

- I3 or above
- U1 or above
- V4 or above.

Field Format and C Include Code

Fields in the C include code have a C clause determined by the length and format of the corresponding Predict field object, as shown in the table below:

Predict Format	File Type	Predict Length	C Clause	Note
B/I	YT, YV	1	CS_TINYINT	
I	JT, JV	1	short	
	BT, BV, XT, XV	2	short int	Irrespective of default value for Represent. of I2
	JT, JV, B, AT A(SQL)	2	short	A(SQL) means file type A with Adabas C SQL usage=Y
	YT, YV	2	CS_SMALLINT	
B/I	other	2	short, int or unsigned char [2]	Depending on default value Represent. of I2.
I	BT, BV, XT, XV	4	long int	Irrespective of default value for Represent. of I4
	JT, JV	4	long	If in defaults Represent. of I4=L
			int	If in defaults Represent. of I4≠L
	B, AT A(SQL)	4	long	Irrespective of default value for Represent. of I4
YT, YV	4	CS_INT		
B/I	other	4	int, long or unsigned char [4]	Depending on default value Represent. of I4.
		8	unsigned char [8]	
B	A(SQL), AT,B		ESQ-BINARY[l]	
	other	l=3,5,6,7	unsigned char [l]	
		l=>9	unsigned char [l]	
F	YT, YV	4	CS_REAL	
	other	4	float	
	YT, YV	8	CS_FLOAT	
	other	8	double	

NS/US	XT, XV	nn.m	dec_t	
	B, AT A(SQL)	nn.m	long	nn < 10, m=0
			double	all other values of n and m
	D, E	nn.m	DECIMAL (l,m)	l=nn+m
BT, BV	nn.m	DECIMAL {l,m}	if nn+m<16: l=nn+m	
		char [l]	if nn+m>15: l=nn+m	
NS	JT, JV	nn.m	double	
N/U NS/US	other	nn.m	unsigned char [l]	l=nn+m
	D, E		DECIMAL (l,m)	
PS	JT, JV	nn.m	double	
	XT, XV	nn.m	dec_t	
	B, AT A(SQL)	nn.m	long	nn < 10, m=0
			double	all other values of nn and m
BT, BV	nn.m	DECIMAL {l,m}	if nn+m<16: l=nn+m	
		char [l]	if nn+m>15: l=(nn+m+2)/2	
P/PS	other	nn.m	unsigned char [l]	l=(nn+m+2)/2
	D, E		DECIMAL (l,m)	l=nn+m
D	D, E		unsigned char [10]	* See note at the end of this table
	BT, BV		char [8]	
	XT, XV		long int	
	other		unsigned char [4]	
T	BT, BV		char [8]	* See note at the end of this table
	D,E		unsigned char [8]	* See note at the end of this table
	other		unsigned char [7]	
L	BT, BV		short int	
	other		unsigned char [1]	
BT	YT, YV		CS_BIT	
DT	JT, JV		unsigned char [25]	* See note at the end of this table
	OT, OV		unsigned char [9]	
	XT, XV		dtime_t	
	YT, YV		CS_DATETIME	
DS	YT, YV		CS_DATETIME4	
G	D, E	n	unsigned char [m]	m=2*n
GV	D, E	n	unsigned char [m]	
GL	D, E	n	unsigned char [m]	
IV	XT, XV	7 17	intrvl_t	

S	XT, XV		long int	
TS	BT, BV		char [20]	* See note at the end of this table
	D, E		unsigned char [26]	* See note at the end of this table
	YT, YV		CS_VARBINARY	
MO	JT, JV		double	
	XT, XV	n.m	dec_t	
	YT, YV		CS_MONEY	
MS	YT, YV		CS_MONEY4	
TK	JT, JV			Is generated like format A with Character Set B and n=8. See next table below.
OK	JT, JV			As above, but n=16.

Note:

For fields marked with an asterisk (*), the length given in the table is applicable if the field is within a redefinition. If the field is not within a redefinition, an additional byte is used to denote the end of the string.

File Type in Predict	Predict Format	Predict Character Set	Length	C Clause	Note
BT, BV	A	any	n	char [n]	* See note above this table
	AL	any	n	VARCHAR [n]	
				VARCHAR *	
	AV	any	n	VARCHAR [n]	

JT, JV	A, AV	blank	n	unsigned char [n]	* See note above this table
		B	n	Two-level group is generated: varchar struct { short Fieldname_len; char Fieldname_txt [n]; } Fieldname;	
	AL	B	n	Two-level group is generated: varchar struct { short Fieldname_len; char Fieldname_txt [n]; } Fieldname;	Field is skipped if no length is specified
	B, BL, BV		n	Two-level group is generated: varbyte struct { short Fieldname_len; char Fieldname_txt [n]; } Fieldname;	Field is skipped if no length is specified
OT, OV	A, AL	blank	n	unsigned char [n]	
	AV	blank	n	VARCHAR [n]	
	A	B	n	RAW	
	AL	B	n	LONGRAW	
	LO	blank, M	n	OCICloblocator *	Indicator field is generated as OCIInd
		B	n	OCIBloblocator *	Indicator field is generated as OCIInd

XT, XV	A	blank	n	unsigned char [n]	* See note above this table
		M	n	Two-level group is generated: struct t_Fieldname { short int Fieldname_len; unsigned char Fieldname_txt [n]; } Fieldname;	
	AL	any	n	loc_t	
	AV	any	n	Two-level group is generated: struct t_Fieldname { short int Fieldname_len; unsigned char Fieldname_txt [n]; } Fieldname;	
YT, YV	A	S	n	CS_CHAR [n]	
	A	D	n	CS_CHAR [2*n]	
	A	B	n	CS_BINARY [2*n]	
	AL	blank		CS_TEXT	
	AL	B		CS_IMAGE	
	AV	S	n	CS_VARCHAR	
	AV	D	n	CS_VARCHAR	
	AV	B	n	CS_VARBINARY	
D, E	A, AL	any	n	unsigned char[n]	* See note above this table
	AV	any	n	Two-level group is generated: struct t_Fieldname { i2form Fieldname_len; unsigned char Fieldname_txt [n]; } Fieldname;	i2form corresponds to the code generated for an I2 field: short, int or unsigned char [2]
	LO	any	n	int, long or unsigned char[4]	Depending on default value Represent. of I4. (Field is generated as locator)
A(SQL), B, AT	A, AV	only 1 Char. Set	n	unsigned char [n]	* See note above this table

A, U	A	anyt	n	unsigned char [n]	
	AV		n	Two-level group is generated. See file type D.	

An automatically generated counter field has the same clause as an I2 field.

A numeric or binary format field with a length not included in the table above is treated in C include code as an alphanumeric format field. A warning message is given.

Note:

If C include code for DB2 tables/views is generated, any redefinition of a field with format NS or US is skipped.

Sample Output

```
14:27:28          ***** P R E D I C T 4.2.1 *****          2001-08-20
                    - Generate C Copy Code -                      Page: 1
```

```
File ID .. GENERATION-EXAMPLE
```

```

/*****
/*      THIS RECORD-BUFFER LAYOUT WAS GENERATED BY PREDICT      */
/*      FOR FILE: GENERATION-EXAMPLE                            */
/*      ON: 2001-08-20 STARTING AT 14:27:28                    */
/*      FILE-COMMENTS: Example file for the                    */
/*      PREDICT generation subsystem                            */
/*      ..                                                       */
/*****
struct t_generation_example
{
    struct t_group_1
    {
        unsigned char ele_n_9v5[14];
/*      * >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' N ' --> ' A ' ) */
        struct t_gr_in_group
        {
            long ele_b_4;
            unsigned char ele_ps_5v2[4];
/*      * >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' PS ' --> ' A ' ) */
            union tu_mu_b_4
            {
                long mu_b_4[5];
                struct t1_mu_b_4
                {
                    unsigned char mu_red_b_7[7];
/*      * >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' B ' --> ' A ' ) */
                    unsigned char mu_red_us_13[13];
/*      * >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' NS ' --> ' A ' ) */
                } r1_mu_b_4;
                struct t2_mu_b_4
                {
                    unsigned char mu_red_b_8[8];
/*      * >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' B ' --> ' A ' ) */
                } r2_mu_b_4;
            } u_mu_b_4;
        } gr_in_group;
        unsigned char ele_a_42[42];
        double ele_f_8;
        unsigned char ele_b_3[3];
    }
};

```

```

14:27:28          ***** P R E D I C T 4.2.1 *****          2001-08-20
                      - Generate C Copy Code -                      Page: 2

File ID .. GENERATION-EXAMPLE

/*      * >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' B ' --> ' A ' ) */
    } group_1;
    short c_pc_occ_7;
    struct t_pc_occ_7
    {
        unsigned char pc_ele_de_ns_7v3[10];
/*      * >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' NS ' --> ' A ' ) */
        short c_pc_mc_ps_6v1;
        unsigned char pc_mc_ps_6v1[11] [4];
/*      * >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' PS ' --> ' A ' ) */
        struct t_pc_gr
        {
            short pc_ele_i_2;
            unsigned char pc_ps_20v7[14];
/*      * >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' PS ' --> ' A ' ) */
        } pc_gr;
        float pc_ele_f_4;
        } pc_occ_7[7];
        unsigned char ele_d[4];
/*      * >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' D ' --> ' A ' ) */
        unsigned char ele_t[7];
/*      * >>> DIC1818 WARNING: FORMAT(S) CHANGED ( ' T ' --> ' A ' ) */
        unsigned char ele_l;
        } generation_example;
/*****
/*      THIS FORMAT-BUFFER WAS GENERATED BY PREDICT
/*      FOR FILE: GENERATION-EXAMPLE
/*      ON: 2001-08-20 STARTING AT 14:27:28
/*****
unsigned char f_generation_example [ 293 ] =
    "AB,2X,AD,AE,AF1-5,AG,2X,AH,AI,1X,ALC,2,AM1,AN1C,2,AN"
    "1(1-11),AO1,2X,AR1,2X,AM2,AN2C,2,AN2(1-11),AO2,2X,AR"
    "2,2X,AM3,AN3C,2,AN3(1-11),AO3,2X,AR3,2X,AM4,AN4C,2,A"
    "N4(1-11),AO4,2X,AR4,2X,AM5,AN5C,2,AN5(1-11),AO5,2X,A"
    "R5,2X,AM6,AN6C,2,AN6(1-11),AO6,2X,AR6,2X,AM7,AN7C,2,"
    "AN7(1-11),AO7,2X,AR7,2X,AS,AT,AU."
/*****

DIC1800 SUMMARY:      25 FIELD(S) PROCESSED
DIC1818 WARNING:      11 FORMAT(S) CHANGED
DIC1819 MESSAGE:      33 SLACK BYTE(S) GENERATED
DIC1847 MESSAGE:       3 FIELD(S) SKIPPED FOR RECORDBUFFER STRUCTURE

```