GLOBAL PARAMETERS

Adabas Native SQL provides a range of global parameters that can be used to define processing options and adapt them to your particular requirements. The options are specified in a parameter file, which is typically included in the job control stream and read by an '//ADAGLOB DD *' JCL card or equivalent.

This chapter lists the global parameters and describes their syntax and their effect.

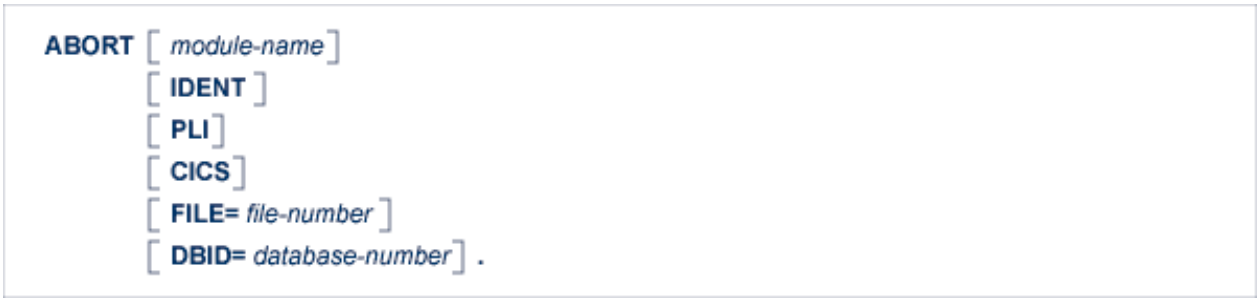
Note that each of these parameters is terminated by a period ('.').

Global parameters can now contain comment lines. Comment lines are signified by two asterisks ('**') starting in column 1.

This chapter covers the following topics:

- The ABORT Parameter
- The ADACALL Parameter
- The APOS Parameter
- The CICS STUB Parameter
- The LANG Parameter
- The LIBRARY Parameter
- The MODE Parameter
- The MONITOR Parameter
- The NAME Parameter
- The NETWORK Parameter
- The OPTIONS Parameter
- The SYSFILE Parameter
- The TELE Parameter
- The USER Parameter
- The VIRTUAL-MACHINE Parameter
- The XREF Parameter

The ABORT Parameter



The ABORT parameter is used to modify Adabas Native SQL's action when an Adabas command returns a response code other than 0 or 3.

See also *Response Code Interpretation* and the *Adabas Messages and Codes Manual*.

In the absence of an ABORT parameter, the abort module RESPINT (Ada, COBOL or PL/I) or RESPF (FORTRAN) is called. This module interprets the response code and prints the appropriate text from the ABEND error message file, the content of the CONTROL-BLOCK and the line sequence number of the erroneous source statement in the SYSOUT file, calls the appropriate trace module, issues an Adabas BT command, and closes the database. Finally, it ABENDs the run.

In particular, the following fields are passed to the error-handling routine:

```
CONTROL-BLOCK
DDFILE
CSEQ
FORMAT-BUF
RECORD-BUF
SEARCH-BUF
VALUE-BUF
CLN1
CLN2
TRCE
CLNNUM
DDDBID
```

CLN1 and CLN2 are arrays that contain the Adabas Native SQL statement. CLN1 contains characters 1..40 of each statement and CLN2 contains characters 41..80. CLNNUM is a variable that indicates the number of elements used in each of these two arrays.

If you want to trap certain error conditions and handle them differently, you must write your own error handling routine. Adabas Native SQL will generate calls to that module instead of to RESPINT if an ABORT parameter with the appropriate module-name is executed. The fields listed above are passed to the module.

If 'ABORT FILE=0.' is coded, Adabas Native SQL does not generate an OPEN for the Natural system.

If ABORT is coded with no module-name, i.e., 'ABORT .', Adabas Native SQL will not check the response code after executing Adabas commands and no exception handling routine will be called. You must write inline code following each Adabas Native SQL statement to handle exception conditions, or use the WHENEVER statement. In addition, if ABORT is coded with no module-name, Adabas Native SQL generates three global fields with the names SQLISN, SQLQTY and SQLRSP instead of generating the three fields ISN, QUANTITY and RESPONSE_CODE for each record buffer.

See also *Additional Fields in the Record Buffers (Ada, COBOL, PL/I)*. (In FORTRAN programs, since there are no record buffers, Adabas Native SQL always generates the above-mentioned three global fields.)

IDENT Clause

If the ABORT parameter is used with the IDENT keyword, Adabas Native SQL generates a statement of the form:

```
CALL identifier ...
```

where the variable identified by the identifier contains the name of the error-handling routine. Otherwise, a statement of the form:

```
CALL 'module-name' ...
```

is generated, where the name of the error-handling routine appears as a literal constant in the CALL statement.

The first form is used for dynamic calls, the second for static calls.

This option is only available in COBOL programs, and it is not supported by all COBOL compilers.

PLI Clause

If the user-written module is in PL/I, the keyword 'PLI' must be coded.

CICS Clause

This clause specifies that the calling mechanism to the response code analysis routine should be generated for CICS. Hence Adabas Native SQL will generate the following:

```
CALL 'ADASTWA' USING ADASQL-LINK-ADDRESSES CONTROL-BLOCKxxxx
DDFILE CSEQ FORMAT-BUFxxxx RECORD-BUFxxxx SEARCH-BUFxxxx VALUE-BUFxxxx
CLN1 CLN2 TRCE CLNNUM DDDBID
CALL 'ADASTWA' USING TWA ADASQL-LINK-ADDRESSES
EXEC CICS LINK PROGRAM ('RESPCICS') END-EXEC
```

Note:
The definition of ADASQL-LINK-ADDRESSES is generated by Adabas Native SQL, and the user should define the TWA as a 24-byte string.

FILE Clause

If the error texts reside in a file other than the standard Natural system file (FNAT), the FILE clause should be used to specify the file number. This number will be passed to the response code interpretation routine as the second parameter, DDFILE. If FILE=0 is coded, no OPEN command will be issued.

The error texts are commonly stored in the Natural system file (parameter FNAT in the SYSFILE statement).

DBID Clause

This clause may be used to specify a database where the FNAT exists in another environment. RESPINT now accepts another parameter DDDBID which has the database number of the FNAT or zero (if the DBID clause is not specified).

The ADACALL Parameter

ADACALL *module-name* [**IDENT**] [**USING** *id1*] [**LAST** *id2*] .

The ADACALL parameter is used to instruct Adabas Native SQL to generate non-standard Adabas calls. Instead of the standard call:

```
CALL 'ADABAS' USING CONTROL-BLOCK... etc.
```

Adabas Native SQL will generate a call as follows:

```
CALL 'module-name' USING id1... CONTROL-BLOCK... etc.
```

The subroutine name 'ADABAS' is replaced by the specified module name in each executable command generated by Adabas Native SQL.

This parameter is used mainly in teleprocessing (TP) applications programs, where the user must call the ADASTWA module. The first parameter of the 'CALL' statement is the terminal work area (TWA); this is followed by the Adabas buffers.

A TP program should therefore specify the following ADACALL parameter:

```
ADACALL ADASTWA USING TWA.
```

This will cause Adabas Native SQL to generate the following call instead of 'CALL 'ADABAS'':

```
CALL 'ADASTWA' USING TWA CONTROL-BLOCK... etc.
```

The ADACALL parameter may also be useful in installations that maintain an I/O interface between the application and Adabas. The ADACALL parameter can be used to direct the calls to the I/O interface, instead of to Adabas.

In CICS environments, the ADALNK module must be replaced by the ADALNC module, which fetches the Adabas parameters (control block, record buffer, etc.) from the TWA.

See also the description of the 'MONITOR' and 'TELE' parameters.

CICS users should also refer to *Using Adabas Native SQL Statements in TP Programs*.

IDENT Clause

If the ADACALL parameter is used with the IDENT keyword, Adabas Native SQL generates a statement of the form:

```
CALL identifier ...
```

where the variable identified by the identifier contains the name of the Adabas link routine. Otherwise, a statement of the form:

```
CALL 'module-name' ...
```

is generated, where the name of the Adabas link routine appears as a literal constant in the CALL statement. The first form can be desirable in certain circumstances. This option is only available in COBOL programs, and it is not supported by all COBOL compilers.

LAST Clause

The LAST clause is used to specify the seventh parameter generated for the Adabas call. id2 is a structure generated by Adabas Native SQL. It can be modified by the user as desired.

The seventh parameter is only an option of Adabas. It contains information that can be evaluated by an Adabas user exit. Adabas Review uses the seventh parameter to receive information on the program name and library name. Adabas Native SQL put the value of the program name within the structure. The user should plug in the library name, using a simple MOVE statement into the L-variable field. The last clause also causes Adabas Native SQL to generate code that may enable Review to identify the use of the seventh parameter.

The LAST clause of the ADACALL parameter generates a structure with the following names (not applicable to VMS or UNIX):

```
01 Variable
        02      FILLER PIC x(276).
        02      PR-variable PIC x(8)VALUE
                'program'.
        02      L-variable PIC x(8) VALUE
                'library'.
        02      RE-variable PIC x(76).
```

The APOS Parameter

```
APOS NO .
```

If the APOS parameter is set to 'NO', character strings generated by Adabas Native SQL will be enclosed in double quotes ("). If the APOS parameter is not coded, character strings will be enclosed in single quotes, sometimes known as apostrophes (').

The CICS STUB Parameter

```
CICS STUB .
```

This global parameter is used to improve performance of interpartition commands when using CICS.

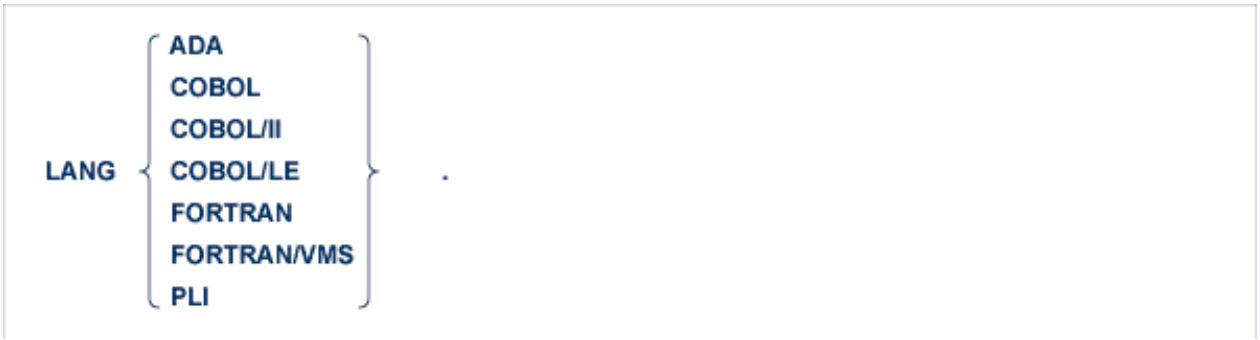
In this case, the call is made to the modulle "Adabas", supplying as the first parameter the stub pointer. Adabas Native SQL generates the definition of the stub pointer, and the user should supply the name by using the ADACALL parameter:

```
ADACALL ADABAS USING pointer
```

With the CICS stub, the user should also use the ABORT CICS parameter if the response code analysis routine should be invoked for CICS.

Note:
The user should define the TWA for this purpose.

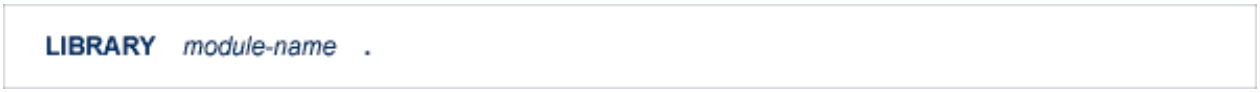
The LANG Parameter



Adabas Native SQL generates declarations and code in the language specified by this parameter. The code generated with the setting ’COBOL’ is also compatible with the COBOL/II compiler, but the code generated with the ’COBOL/II’ setting makes use of the structured ’END-IF’, ’END-PERFORM’, etc., clauses.

If this parameter is omitted, Adabas Native SQL attempts to determine the language in which the program is written by examining its first line. However, this technique is not completely reliable, so we strongly recommend you include this parameter in every Adabas Native SQL run.

The LIBRARY Parameter



This new parameter is used to support a library concept for 3GL applications. name represents a logical library name (max. 8 characters). If the library is not defined in Predict, an error message is displayed.

The MODE Parameter

This parameter controls debugging facilities that are built in to Adabas Native SQL.

MODE FLOW

If the parameter ’MODE FLOW’ is specified, all Adabas Native SQL statements will be printed out at runtime as they are executed.

MODE NOFLOW

If the parameter ’MODE NOFLOW’ is specified, the code that copies Adabas Native SQL source statements into a buffer is not generated. This reduces the size of the generated Ada, COBOL, FORTRAN or PL/I code, but the FLOW and TRACE facilities are not available and Adabas Native SQL cannot print out the source statement if a runtime error is detected. This could make debugging more difficult.

MODE NOUPDATE

If MODE NOUPDATE is coded, statements that would modify the database (DELETE, INSERT, UPDATE) have no effect.

MODE TRACE

This parameter must be coded if diagnostic output is required. Conversely, when a program has been debugged and diagnostic output is no longer required, you can delete this parameter and recompile the program. The resulting object module will be smaller and will run faster.

Diagnostic output is controlled by the following:

- the global parameter ’MODE TRACE.’
- the Adabas Native SQL statements ’TRACE ON’ and ’TRACE OFF’, and
- the value contained in the variable TRCE (Ada, COBOL, PL/I) or SQDE00 (FORTRAN).

The action of the global parameter ’MODE TRACE’ is described above.

When processing an Ada, COBOL, FORTRAN or PL/I program, and assuming that the global parameter ’MODE TRACE’ has been coded, Adabas Native SQL only generates the code for producing diagnostic output when it encounters a ’TRACE ON’ statement, it stops generating this code when it encounters a ’TRACE OFF’ statement. These two statements provide static control of the diagnostic output, that is, they control the section or sections of the program in which diagnostic code is generated.

When an Adabas Native SQL statement is executed, the first action of the diagnostic code is to test the value contained in the variable TRCE (Ada, COBOL, PL/I) or SQDE00 (FORTRAN). If this value is ’OFF’, then no further action is performed. Otherwise, the statement is printed out together with the contents of the buffers. This variable provides dynamic control of the diagnostic output. By assigning values to this variable at runtime, you have greater control over the diagnostic output. For example, you could limit output to the first five executions of a loop that may be executed several hundred times.

The MONITOR Parameter

MONITOR CICS [*twa*] [**COMMAREA** *commarea-name* **POINTERS** *addr-name*] .

This parameter makes it unnecessary to code the ADACALL and TELE global parameters.

If the optional *twa* clause is coded, this name is used instead of the default name 'TWA'.

For COBOL programs, coding 'MONITOR CICS.' is equivalent to coding the following three global parameters:

```
ADACALL ADASTWA USING TWA.
TELE "EXEC CICS LINK PROGRAM ('ADABAS') END-EXEC".
ABORT RESPCICS CICS
```

The MONITOR CICS parameter is not valid in FORTRAN programs.

For PL/I programs, coding 'MONITOR CICS.' is equivalent to coding the following three global parameters:

```
ADACALL ADASTWA USING TWA.
TELE "EXEC CICS LINK PROGRAM ('ADABAS') ;".
ABORT RESPCICS CICS
```

These defaults may be overridden by coding one or both of the ADACALL or TELE parameters.

Prior to all calls created by the ADACALL parameter, the following code will be generated for COBOL if 'MONITOR CICS.' has been coded:

```
MOVE xxxxxxxxxx to ADASQL-SAVE-TWA
```

The corresponding code for PL/I programs is:

```
ADASQL_SAVE_TWA = xxxxxxxxxx
```

xxxxxxxxxx is the field name supplied after USING in the global parameter ADACALL. (TWA is the default ADACALL used.)

After each TELE line is generated, the following COBOL code is inserted if 'MONITOR CICS.' has been coded:

```
MOVE ADASQL-SAVE-TWA TO xxxxxxxxxx
```

The code for PL/I programs is:

```
              xxxxxxxxxx = ADASQL_SAVE_TWA
```

If you are using more than 28 bytes in the TWA, code the following:

```
01 TWA.
             02 ADABAS-TWA          PIC X(28)
             02 REST-OF-TWA         PIC X(nnnn).
```

Then code the following ADACALL parameter for COBOL:

```
ADACALL ADASTWA USING ADABAS-TWA
```

We recommend defining the layout of the TWA in COBOL copy books which can be accessed by all Adabas Native SQL programs.

The code for PL/I is:

```
DCL 01 TWA,
           02 ADABAS_TWA        CHAR(28)
           02 REST_OF_TWA       CHAR(nnnnn);
```

This parameter also controls the generation of EXEC CICS LINK to RESPCICS and PRTRCICS instead of RESPINT and PRTRACE.

The COMMAREA parameter is for using the COMMAREA instead of the TWA.

The user then will have to define a structure for the Commarea usage as follows:

```
01 COMMAREA-NAME.
02 FILLER PIC X(8) VALUE 'ADABAS52'.
02 ADDR-NAME PIC X(4) OCCURS 6.
```

Adabas Native SQL will then generate

- a call to ADASTWA with addr-name to move the addresses of the Adabas buffers into it;
- and then an EXEC CICS command with COMMAREA(*commarea-name*) instead of TWA.

The constant "ADABAS52" is the indicator for the Adabas CICS interface to detect the COMMAREA parameter list. For the syntax of the Adabas parameter list, see also the Adabas CICS command level interface description for CICS Version 3.2 and higher.

Note:
The RESPCICS and PRTRCICS will continue to use the TWA.

The NAME Parameter

NAME *program-name* .

The program-name specified in the 'NAME' parameter is used by Adabas Native SQL in conjunction with the programming language (Ada, COBOL, FORTRAN or PL/I) when Adabas Native SQL writes Xref data to the data dictionary. The *program-name* is referred to in Predict as *Member*.

Adabas Native SQL provides cross-reference reports of programs, modules and fields using the Xref facilities of Predict. This information is automatically created during the preprocessor pass. The names of the files and fields that are used are taken from the FROM, SELECT, WHERE, SET, etc., clauses of the Adabas Native SQL statements; the name of the program that uses them is taken from the 'NAME' parameter.

If the NAME parameter is omitted, Adabas Native SQL takes the program name from the following sources:

Language	Program name taken from
ADA	the procedure
COBOL	PROGRAM-ID paragraph in the environment division
FORTRAN	the first line of the program, which must be PROGRAM *progname*
PL/I	the label preceding 'PROC OPTIONS(MAIN)'

The NETWORK Parameter

```
NETWORK network-name .
```

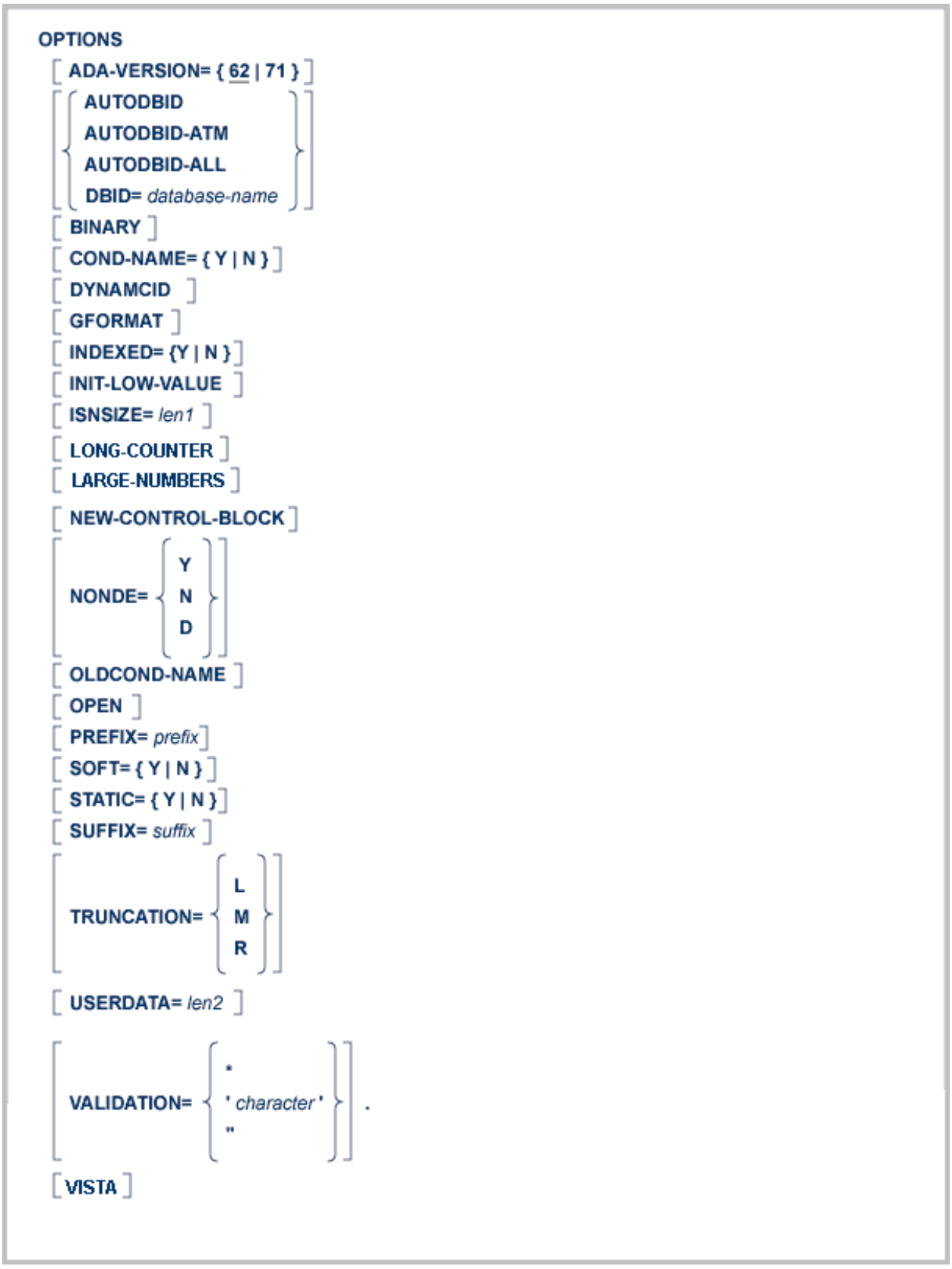
This global parameter defines the network in which the program is to run. *network-name* must be defined in Predict, and must be linked to the virtual machine specified with the parameter VIRTUAL-MACHINE.

This parameter is mandatory, if one or more networks other than HOME are defined in Predict.

A network contains all virtual machines and databases that are to be accessed. In fact, all databases that are used in the program should belong to the network specified here.

For every database used (DBID, AUTODBID, AUTODBID-ATM or AUTODBID-ALL clauses), Adabas Native SQL checks that if the database is defined as local it belongs to the current virtual machine, and if the database is isolated that it belongs to the current network.

The OPTIONS Parameter


OPTIONS
ADA-VERSION= { 62 | 71 }
AUTODBID
AUTODBID-ATM
AUTODBID-ALL
DBID= database-name
BINARY
COND-NAME= { Y | N }
DYNAMCID
GFORMAT
INDEXED= {Y | N }
INIT-LOW-VALUE
ISNSIZE= len1
LONG-COUNTER
LARGE-NUMBERS
NEW-CONTROL-BLOCK
NONDE=
Y
N
D
OLDCOND-NAME
OPEN
PREFIX= prefix
SOFT= { Y | N }
STATIC= { Y | N }
SUFFIX= suffix
TRUNCATION=
L
M
R
USERDATA= len2
VALIDATION=
*
' character '
"
.
VISTA

The OPTIONS parameter enables the user to specify various processing options that will take effect for the whole of the program unless they are overridden by declarations made at the individual statement level.

The OPTIONS parameter should not be confused with the OPTIONS clause of individual Adabas Native SQL statements.

ADA-VERSION Clause

The ADA-VERSION clause indicstes to Adabas Native SQL in which Adabas version the precompiled program is to be executed. The default is 62, and this will generate code that can be executed in all Adabas versions. The value 71 will enable using new features introduced in Adabas Version 7.1 in the READ LOGICAL and HISTOGRAM statements.

Note:
A precompiled program with ADA-VERSION=71 may fail or give unpredicted results if executed in an Adabas version lower than 7.1.

AUTODBID Clause

The AUTODBID option causes every access statement to use the database identified in Predict for that file. If the file is linked to a database and no specific DBID is specified in the statement, an error message is given.

AUTODBID-ALL Clause

The AUTODBID-ALL option causes all statements (both access and update) to use the database identified in Predict for that file. If the file is linked to a database and no specific DBID is specified in the statement, an error message is issued.

If AUTODBID-ALL is specified, neither the DBID nor the AUTODBID clause may be used.

The following rules apply to the various statements when using the AUTODBID-ALL clause:

Statement	Remarks
All statements	One file must be documented in exactly one database. If separate test and production environments are used, separate dictionary files (FDIC) are necessary. No source changes are necessary if only a recompile with the production dictionary is required.
CONNECT, DBCLOSE	These statements are mandatory and must be used within one program. The files specified in the UPDATE clause define the database to be updated. All update files must be within one database otherwise an error message is displayed. If more than one database is accessed, several OPEN commands must be generated, but only one OPEN command is generated for update. To generate a CLOSE command to the same databases which are opened using CONNECT, the DBCLOSE must occur within the same program, otherwise an error message is given.
COMMIT	The COMMIT statement is always sent to exactly one database. This database is identified by the CONNECT statement (updatefiles => database) or by the UPDATE statements available in the program. An UPDATE or CONNECT statement must be coded, otherwise an error message is given.
UPDATE, DELETE, INSERT	If the program does not contain a CONNECT statement, a warning is issued that CONNECT must be executed before the updates are performed, otherwise consistency cannot be guaranteed. Adabas Native SQL must be able to check that within one program updates are performed only on files that belong to the same database. The update of only one database is supported.

AUTODBID–ATM Clause

The AUTODBID–ATM option causes all statements (both accesss and update) to use the database identified in Predict for that file. If the file is linked to more than one database and no specific DBID is specified in the statement, an error message is issued.

If AUTODBID–ATM is specified, neither the AUTODBID-ALL nor the AUTODBID clause may be used.

With this option we do not restrict the number of updated databases (unlike the AUTODBID-ALL parameter) and the user does not have to specify any DBID in any of the statements.

Note:
Please note that if such a program will be run without the supervision of the Adabas Transaction Manager, we cannot guaranty the data integrity and this will be the user’s responsibility to ensure the use of the Adabas Transaction Manager.

The Commit statement will be generated with the default DBID and the Adabas Transaction Manager will take care of the two phase commit.

If within this program the user would like to use the CONNECT statement, then he should specify in this statement the DBID to which this CONNECT should run. The same would apply to the DBCLOSE statement.

BINARY Clause

This clause applies to COBOL programs only.

It will cause all binary fields to be generated as BINARY instead of COMP.

COND-NAME Clause

This clause applies to COBOL programs only.

If the clause ’COND-NAME = Y’ is coded, the record buffer generated by Adabas Native SQL includes the condition names defined in Predict as level-88 entries.

This global value will be overridden by any value specified in a clause of an individual Adabas Native SQL statement.

The field With Cond. names in the Predict Modify COBOL Defaults screen must be marked with an "X" if you want to specify this option. See also *Generate COBOL Copy Code* in the *Predict Administration Manual*.

DBID Clause

This clause should be used if the program accesses more than one database. The database-name must be defined in the data dictionary, and the data dictionary description of the database must include the file or files to be accessed. All statements including UPDATE, DELETE and STORE are affected by this clause.

DYNAMCID Clause

If the DYNAMCID keyword is coded, Adabas Native SQL generates the command IDs of the Adabas control blocks dynamically during program execution.

The automatic Adabas routine for generating the command ID is used. Using DYNAMCID increases performance significantly.

If this clause is not specified, the command ID used for each Adabas command is generated from the cursor-name of the corresponding Adabas Native SQL statement. If the DYNAMCID keyword is not coded in the global OPTIONS parameter and a cursor-name is not defined for a particular Adabas Native SQL statement, because the DECLARE option was not used, Adabas Native SQL will generate command IDs in the form -m-n, where mn is a sequence number starting from 01. The first statement without a DECLARE clause will have command ID -0-1, the second statement will have -0-2, etc.

The command ID is used by Adabas for the following purposes:

- As an identifier for the internal, decoded version of the format buffer. Efficiency is improved if Adabas statements that use the same format buffer use the same command ID, otherwise Adabas is compelled to re-interpret the format buffer each time.

- When executing HISTOGRAM, READ LOGICAL and READ PHYSICAL SEQUENCE statements. If the command ID is not given when the statement is executed, Adabas 'loses its place' in the file and gives inconsistent results.
- To identify ISN lists. The command ID links the Adabas command (COMPARE, FIND, FIND COUPLED, or SORT) that creates the ISN list with subsequent commands that retrieve the records whose ISNs are stored in the list.

If several programs that use Adabas Native SQL statements are linked together, all command IDs must be unique. This can be achieved explicitly, that is, by coding a unique cursor-name for each statement, or by allowing Adabas Native SQL to allocate the command IDs dynamically by means of the DYNAMCID global option. Coding unique cursor-names has the advantage that the Adabas command log is easier to interpret.

See chapter *Command ID Usage* in the *Adabas Command Reference Manual* for more information.

GFORMAT Clause

This clause indicates that a global format is to be generated for this program. Adabas Native SQL generates a unique global format ID for every declaration generated (with the exception of variable index used for periodic groups or multiple-value fields). The global format ID is unique and will not exist in other programs. This clause can help to improve application performance, particularly in on-line environments, by reducing the number of format buffer translations that Adabas has to perform.

If this option is used, the global format ID is generated from the following information:

```
GFID = abcdeeef
```

Where for FDIC file number and DBID < 255,

a = x'C0'
b = x'83'
c = FDIC DBID
d = FDIC FNR
eee = Adabas Native SQL sequence number from Predict defaults
f = An internal sequence number within the program

Where for FDIC file number or DBID > 255

a = x'C1'
b = Possible value x'00' to x'FF'
c = Right byte of FDIC DBID
d = Right byte of FDIC FNR
eee = Adabas Native SQL sequence number from Predict defaults
f = An internal sequence number within the program

The GFORMAT clause is not available in Ada programs.

INDEXED Clause

This clause applies to COBOL programs only.

If the INDEXED clause is specified, all multiple-value fields and periodic groups are generated with the ’INDEXED BY’ keywords. The name of the index is taken from Predict. If no index name is defined in the data dictionary, the name of the multiple-value field or periodic group is used, prefixed with ’I-’.

This global value will be overridden by any value specified in a clause of an individual Adabas Native SQL statement.

Indexed by in the Predict Modify COBOL Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this clause and section *Generate COBOL Copy Code* in the *Predict Administration Manual* for more information.

INIT-LOW-VALUE Clause

By default, Adabas Native SQL is generating the Value buffer fields with an initial value of blanks for alphanumeric fields and zeroes for numeric fields. In this way a Read logical command without a WHERE clause will start the sequential read from those starting values.

With this option the generated alphanumeric fields in the Value buffer will have an initial value of X’00’ which has a collating sequence lower than blanks. In this way if the descriptor which we use for the Read logical has values lower than blanks and we don’t specify the WHERE clause we will start the sequential read from the lowest value available for this field.

Please note that this feature is available for Cobol and PL/1 languages only.

ISNSIZE Clause

If the ISNSIZE clause is specified, the default size of the ISN buffer is defined and all Adabas Native SQL statements run in ’ISN buffer’ mode; however, the buffer size can be modified for individual retrieval statements by local ISNSIZE specifications.

If a global ISNSIZE value is not specified, ISN buffers are allocated for individual statements as determined by the presence or absence of the ISNSIZE parameter in the OPTIONS clause of each individual statement.

ISN buffer mode must not be used when accessing files that use the ’security by value’ facility.

See also *ISN Lists and the ISN Buffer*.

LARGE-NUMBERS Clause

This option will cause Adabas Native SQL to generate numeric (Unpacked)and Packed fields in Cobol for numeric fields with up to 31 digits. Without this option numeric fields with more than 18 digits will be generated as a character string. Users should note that in case of using the LARGE-NUMBER clause, they should make sure that the Cobol compiler option that allows for up to 31 digits numeric fields is set.

LONG-COUNTER Clause

This option will cause Adabas native SQL to generate the Multiple Value and Periodic Group counters as 4 binary bytes instead of the default of 2 binary bytes. This option should be used if the total occurrences of the Multiple Value or the Periodic group may exceed 32767 occurrences.

NEW-CONTROL-BLOCK Clause

This option will cause Adabas Native SQL to generate the new control block structure introduced in Adabas 6.1 on mainframe platforms and Adabas 4.1 in OpenVMS.

The new control block will allow file numbers and dbid's to be greater than 255.

NONDE Clause

This clause is available with Adabas 5 only. It is used to allow (NONDE=Y) or inhibit (NONDE=N) the use of non-descriptors within database search criteria. The option NONDE=D specifies that each search criterion must include at least one descriptor (and possibly some non-descriptors).

The field Non-descriptor search allowed in the Predict Modify Adabas Native SQL Defaults screen must be set to "Y" if you want to use this option.

OLDCOND-NAME Clause

Adabas Native SQL allows for the condition values to contain blanks. In versions prior to V231, a blank in the value of the condition name definition in Predict was considered as a delimiter and Adabas Native SQL generated several values for the same condition name value line.

With version 231 and up, only one value will be generated per value line definition in Predict and this value may contain blanks.

Users that would like to keep the old functionality may use this keyword.

With this option, Adabas Native SQL will generate the Condition names as in versions prior to V231 and consider a blank in the value to be a delimiter.

OPEN Clause

If this clause is coded, Adabas Native SQL performs an explicit 'open' on Predict file as it preprocesses your application program.

PREFIX Clause

If the option 'PREFIX = *prefix*' is coded, the field names generated for the record buffer will include the specified prefix. This global value will be overridden by any value specified in a clause of an individual Adabas Native SQL statement or taken from the data dictionary.

Field name prefix in the Predict Modify...Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this clause for more information.

SOFT Clause

This clause is used to enable (SOFT=Y) or inhibit (SOFT=N) the soft-coupling option.

The field Use of soft-coupling allowed in the Predict Modify Adabas Native SQL Defaults screen must be set to "Y" if you want to specify this option.

STATIC Clause

This option applies to PL/I programs only.

If the option ’STATIC = Y’ is coded, all buffers generated by Adabas Native SQL will be defined as static. This global value will be overridden by any value specified in a clause of an individual Adabas Native SQL statement.

The field Static in the Predict Modify PL/I Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this clause for more information.

SUFFIX Clause

If the option ’SUFFIX = *suffix*’ is coded, the field names generated for the record buffer will include the specified suffix. This global value will be overridden by any value specified in a clause of an individual Adabas Native SQL statement or taken from the data dictionary.

Field name suffix in the Predict Modify...Defaults screen must be marked with an "X" if you want to specify this option. See the previous discussion on this clause for more information.

TRUNCATION Clause

A field name may exceed the maximum number of characters permitted by the language, particularly if a prefix and/or suffix has been added. Adabas Native SQL uses the TRUNCATION clause to delete excess characters:

TRUNCATION = L	truncate from the left
TRUNCATION = M	truncate from the middle
TRUNCATION = R	truncate from the right

This global value will override the value in the data dictionary.

The field Truncation in the Predict Modify...Defaults screen must be marked with an "X" if you want to specify this option.

USERDATA Clause

The USERDATA clause may be used to specify the size of the ET-data buffer, i.e., RECORD-BUFOPN. The default size is 500 bytes. This buffer is used in COMMIT WORK, DBCLOSE and CHECKPOINT statements.

VALIDATION Clause

This option determines how invalid characters in field names - including prefix and suffix, if specified - are handled by Adabas Native SQL.

Validation Character	Result
Null string (two consecutive apostrophes)	Invalid characters in a field name will result in an error message but will not be modified.
Replace character (letters A-Z, digits 0-9 or special character depending on language)	Invalid characters in a field name are replaced by this character.
Asterisk	Invalid characters in the field name are deleted.

This global value will override the value in the data dictionary.

Validation in the Predict Modify...Defaults screen must be marked with an "X" if you want to use this option.

VISTA Clause

In case that the Predict file is defined under Adabas Vista configuration, the DBID and file number of the physical Predict file may differ from the Adabas Native SQL `SYSFILE` parameter. In this case Adabas Native SQL will issue an error that these numbers do not match the Predict Control record.

The VISTA clause will cause Adabas Native SQL to ignore the different values and use the numbers from the Predict Control record for generating the global format id.

The SYSFILE Parameter

```
SYSFILE   [ FNAT=( dbid1,fnr1 [,password1 [,cipher1 ] ] ) ]
          [ FDIC= ( dbid2,fnr2 [,password2 [,cipher2 ] ] ) ]
```

The SYSFILE parameter specifies to Adabas Native SQL the number of the Natural system file and the number of Predict file. The Adabas Native SQL error messages are normally stored in the Natural system file. The SYSFILE parameter is mandatory, and the *dbid* and *fnr* must be specified. These numbers are checked against the DDA default record in Predict and, in case of incompatibility, execution stops.

PASSWORD Clause

If the Predict file or Natural system file is password-protected, the correct password must be specified using this clause.

CIPHER Clause

If the Predict file or Natural system file is enciphered, the correct cipher key (cipher code) must be specified using this clause.

Example:

```
SYSFILE  FNAT = (3,9)  FDIC = (3,8)
```

The database ID (DBID) of the Natural system file is 3, and its filenumber (FNR) is 9. The DBID of the Predict file is 3, and its FNR is 8.

The TELE Parameter

```
TELE "text" .
```

The TELE parameter specifies a source statement to be inserted after each CALL command in the generated executable statements. The text may, for example, be a command required by a teleprocessing monitor.

Example:

```
TELE "EXEC CICS LINK PROGRAM ('ADABAS') END-EXEC".    (COBOL)
TELE "EXEC CICS LINK PROGRAM ('ADABAS');".            (PL/I)
TELE "EXEC CICS LINK PROGRAM ('ADABAS') END-EXEC".    (FORTRAN)
```

The above example inserts the CICS command level instruction after every CALL. This parameter should be used in conjunction with the ADACALL parameter. See the example in the ADACALL parameter.

There may be up to five TELE statements, so that up to five additional lines of text may be generated after the call.

Note that in COBOL programs the *text* must not include a period.

See also the MONITOR parameter.

Ada is still not supported by the CICS translator.

The USER Parameter

```
USER userid .
```

This parameter is used to identify the user responsible for the program. This userid will be documented in XREF. If no USER parameter is specified, Adabas Native SQL takes the first 3 characters of the program as the userid.

The VIRTUAL-MACHINE Parameter

```
VIRTUAL-MACHINE virtual-machine .
```

This statement defines the virtual machine, that is the real computer or node, in which the program is to run.

virtual-machine must be defined in Predict, and must be linked as a child object to the network specified with the NETWORK parameter.

This parameter is mandatory if one or more databasesVirtual Machines other than HOME are defined in Predict.

For every database used (DBID, AUTODBID, AUTODBID-ATM or AUTODBID-All clauses), Adabas Native SQL checks that if the database is defined as *local* it belongs to the current virtual machine, and if the database is *isolated* that it belongs to the current network.

The XREF Parameter

```
XREF { ON  }
     { OFF }
     { FORCE } .
```

This parameter controls the writing of cross-reference information (Xref data) to the data dictionary.

Note that the XREF global Adabas Native SQL parameter interacts with Predict's Preprocessor force option. If the Predict option is set to 'Y', then Adabas Native SQL ignores the XREF parameter in the Adabas Native SQL global parameter file (if present) and proceeds as though 'XREF FORCE' had been coded.

For details of the Predict `Preprocessor Force` option, see section *Common Parameters* in chapter *Generation* of the *Predict Administration Manual*.

XREF ON.	Adabas Native SQL makes entries in the data dictionary indicating which files and fields are used by the current program.
XREF OFF.	No entries are made in the data dictionary.
XREF FORCE.	Adabas Native SQL makes entries as for 'XREF ON' and additionally checks that the data dictionary contains a program description bearing the name of the current program. An error message is output if this condition is not satisfied.

If the data dictionary is opened for access only (global parameter DDFILE ACC.), 'XREF OFF.' must be coded.

The program is identified by its name (referred to in Predict as *Member*) together with the language in which it is written (Ada, COBOL, FORTRAN or PL/I). See the NAME parameter.