

XREF Data

In addition to documentation objects, XREF data is stored in the Predict system file. XREF data documents objects of an application that have already been implemented. Data of this type mirrors the 'real world' of the implementation, providing a comprehensive summary of information for all implemented members of an application.

XREF data can be used to retrieve information on individual members or to retrieve comprehensive information about applications as a whole:

- its invocation structure
- the resources it uses
- its consistency, completeness and correctness.

This section covers the following topics:

- Contents of XREF Data
- Benefits of XREF Data
- Creating XREF Data

Contents of XREF Data

For Natural and for Third Generation Languages

- program and the application it belongs to
- program type (such as: main program, subroutine, subprogram, help routine, map or data area)
- language
- date and time the program was cataloged/precompiled
- ID of the user who cataloged/precompiled the program
- ID of the terminal from which the program was cataloged/precompiled (in batch mode: the job name)
- external programs invoked from a program
- files used in the program and the type of file use (deleting, reading or updating)
- fields of files used in the program and the type of field use (as counter, or for reading, searching or updating).
For 3GL: precompiled by Adabas SQL Server, Adabas Native SQL
- Copy Code used in the program
- entry points or functions defined in the program which can be invoked by other programs.

For Natural only

The following XREF information data is provided for Natural programs only:

- invoked Natural programs and the methods used to invoke them (for example CALLNAT, FETCH, FETCH RETURN, PERFORM, RUN, STACK, SEND EVENT, SEND METHOD)
- maps and help routines used in the program
- data areas used in the program and their types (local, global, or parameter)
- externally defined variables used in the program
- logical names of printers used in the program
- numbers of Natural error messages used in the program and the types of error messages (system-wide or application-specific)
- active processing rules used in a map and their type of use (automatic, free, or inline processing rules)
- retained sets used in the program and the type of use (build, use, or release a retained set)
- Numbers of Natural workfiles used in the program and the type of use (whether read from, written to or close).
- Natural resources used by the program

Benefits of XREF Data

The application development process can be advanced significantly at almost all phases by using XREF data:

Phase	Benefits of XREF Data
Design	Identifying programs of other applications that might possibly be used in the planned application.
Implementation	Finding resources that are not (or no longer) used. Finding errors and inconsistencies that cause errors at runtime, for example calls to non-existent subroutines. Supervising the implementation state of an application.
Testing	Detecting incomplete documentation. Detecting differences between design and implementation. Detecting programming errors.
Production	Reporting all information about the programs that are in production.
Maintenance	Estimating the impact of changes. Discovering the invocation structure of an application.

Creating XREF Data

XREF data is created differently for Natural programs and for programs in third generation languages.

Creating XREF Data for Natural Members

XREF data for Natural members is generated during compilation. This does not affect the performance of the program; only the cataloging time is insignificantly extended. Creation of XREF data can be controlled by setting the Natural XREF parameter to YES, NO, DOC or FORCE.

- If XREF is set to YES, XREF data will be created.
- If XREF is set to NO, XREF data will not be created.
- If XREF is set to FORCE, Natural programs cannot be cataloged if no corresponding documentation object exists. XREF data will be created.
- If XREF is set to DOC, Natural programs cannot be cataloged if no corresponding documentation object exists. No XREF data will be created.

If Natural Security is installed, the Natural parameter XREF is taken from the library's security definition.

See the section LIST XREF for Natural in the **Predict Reference documentation** for a complete description of where and when XREF data for Natural members is created.

Creating XREF Data for 3GL Members

XREF data for members written in third generation languages is created differently, depending on the types of objects processed:

- XREF data for COBOL, FORTRAN, PL/I and ADA is created by Adabas Native SQL.
See the Adabas Native SQL Reference documentation.
- XREF data for Assembler, COBOL and PL/I is created by the Predict preprocessor.
See the section The Preprocessor in the **External Objects in Predict documentation**.
- XREF data for COBOL and language C is created by the Adabas SQL Server.
- XREF data for Natural programs that use static SQL is created by the function CREATE DBRM of Natural for DB2.

See the section LIST XREF for Third Generation Languages in the **Predict Reference documentation** for a complete description of where and when XREF data for 3GL members is created.