

Third Generation Languages

Predict provides functions for documentation, development and redocumentation of 3GL applications and programs. The following third generation languages are supported:

- BAL/Assembler
- C
- COBOL
- FORTRAN
- PL/I
- Ada

Not all facilities are available for all of these languages. On the other hand there are areas in Predict where additional user-specified languages are supported. In other areas, special classes of programs are used that are treated by Predict like languages. These restrictions and extensions are either mentioned here or in the relevant section of this documentation or the **External Objects in Predict documentation**.

This section covers the following topics:

- Documenting 3GL Applications
- Documenting 3GL Programs
- XRef Data for 3GL Applications and Programs
- Using Predict Functions When Developing 3GL Applications
- Redocumenting of 3GL Applications
- Redocumenting COBOL Record Structures

Documenting 3GL Applications

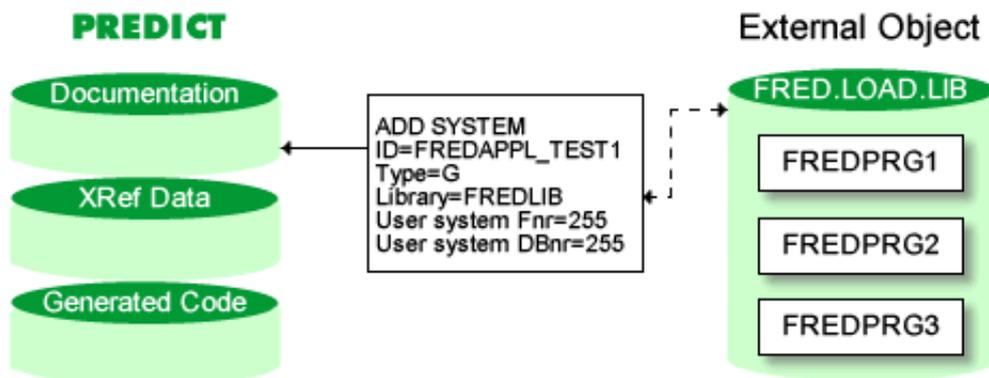
3GL applications are documented in Predict with system objects of type 3GL Application (code G).

The system object and the 3GL application it represents are connected by an implementation pointer.

Implementation Pointer for 3GL Application

Parameters	
Library	This name can be freely chosen when an object of type system is added or modified, and represents one or more source or load libraries/directories or parts thereof. Once defined here, this name can be used to document 3GL programs belonging to the application and for creating and retrieving XRef data. If XRef data related to this library exists, the name may no longer be changed.
User system Fnr, DBnr	These attributes are used to distinguish 3GL libraries from Natural libraries. Both must be set to 255 for 3GL applications.

Documenting a 3GL Application with a Predict Object of Type System



System implementation pointer Library represents for Predict one or more source or load libraries/directories or parts thereof.

Documenting 3GL Programs

3GL programs are documented in Predict with objects of type Program, with one of the languages listed on page 2 and one of the following subtypes, depending on the programming language.

- copy code
- documented
- program
- function
- subprogram.

Other languages can be defined in the user exit U-PGMLAN. See the section User Exits in the **Predict Administration documentation** for more information.

Predict also knows the pseudo-languages System program and Static SQL.

As with applications, the program object and the implemented 3GL member it represents are connected by an implementation pointer.

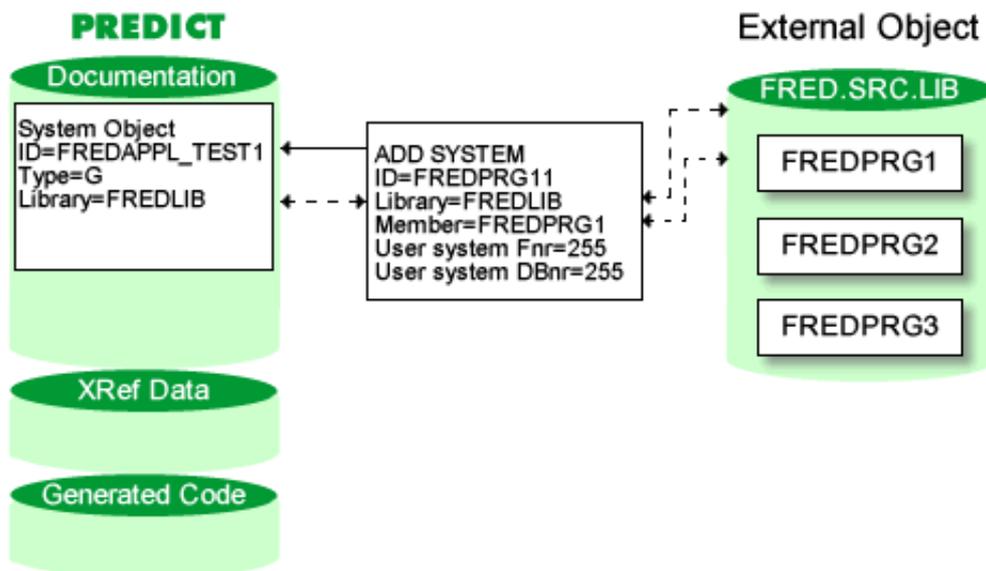
Implementation Pointer for 3GL Programs

If Member is not entered for the program implementation pointer, it is not possible to enter a value for Library. If a Member is specified, the possible values for Library depend on how the maintenance option Implementation library described below has been defined by your DDA in the General Defaults function.

Presetting	
Implementation library	<p>F Force. A library that is documented as a 3GL application must be entered. A default library - for example *SYSCOB* - may not be entered.</p> <p>A Allowed. Either a library documented as a 3GL application or a default library must be entered. See next table for a complete list of default libraries for 3GL programs.</p> <p>D Disallowed. Library concept is not used. Library *SYSALL* must be entered.</p>

Parameters																					
Member	Corresponds to the name of the implemented program as it is stored in a source or load library/directory.																				
Library	<p>The possible values for this parameter depend on the maintenance option Implementation Library. See Presetting above.</p> <p>Corresponds to the implemented application to which the program belongs. If a non-default library is specified, it must be defined in an object of type system if XRef data is to be created for the program.</p> <p>The program object does not have to be linked in Predict to the system object containing the library name.</p> <p>If Member is specified but Library is left blank, and if Implementation Library is set to A, the program object is connected automatically to the corresponding default library:</p> <table border="1" data-bbox="359 1146 750 1491"> <thead> <tr> <th>Language</th> <th>Default Library</th> </tr> </thead> <tbody> <tr> <td>COBOL</td> <td>*SYSCOB*</td> </tr> <tr> <td>BAL/Assembler</td> <td>*SYSBAL*</td> </tr> <tr> <td>PL/I</td> <td>*SYSPLI*</td> </tr> <tr> <td>FORTRAN</td> <td>*SYSFOR*</td> </tr> <tr> <td>C</td> <td>*SYSCCC*</td> </tr> <tr> <td>ADA</td> <td>*SYSADA*</td> </tr> </tbody> </table> <table border="1" data-bbox="359 1525 778 1675"> <thead> <tr> <th>Pseudo-Language</th> <th>Default Library</th> </tr> </thead> <tbody> <tr> <td>Static SQL</td> <td>*SYSSTA*</td> </tr> <tr> <td>System program</td> <td>*SYSSYS*</td> </tr> </tbody> </table> <p>Default libraries do not need to be defined explicitly in a system object.</p> <p>Note: Programs of language System program must be linked to library *SYSSYS*. Programs of all other languages in the above list can be linked either to their default library or to a user-defined library defined in a system object.</p>	Language	Default Library	COBOL	*SYSCOB*	BAL/Assembler	*SYSBAL*	PL/I	*SYSPLI*	FORTRAN	*SYSFOR*	C	*SYSCCC*	ADA	*SYSADA*	Pseudo-Language	Default Library	Static SQL	*SYSSTA*	System program	*SYSSYS*
Language	Default Library																				
COBOL	*SYSCOB*																				
BAL/Assembler	*SYSBAL*																				
PL/I	*SYSPLI*																				
FORTRAN	*SYSFOR*																				
C	*SYSCCC*																				
ADA	*SYSADA*																				
Pseudo-Language	Default Library																				
Static SQL	*SYSSTA*																				
System program	*SYSSYS*																				
User system Fnr, DBnr	These attributes are used to distinguish implemented 3GL programs from Natural programs. Both must be set to 255 for 3GL programs.																				

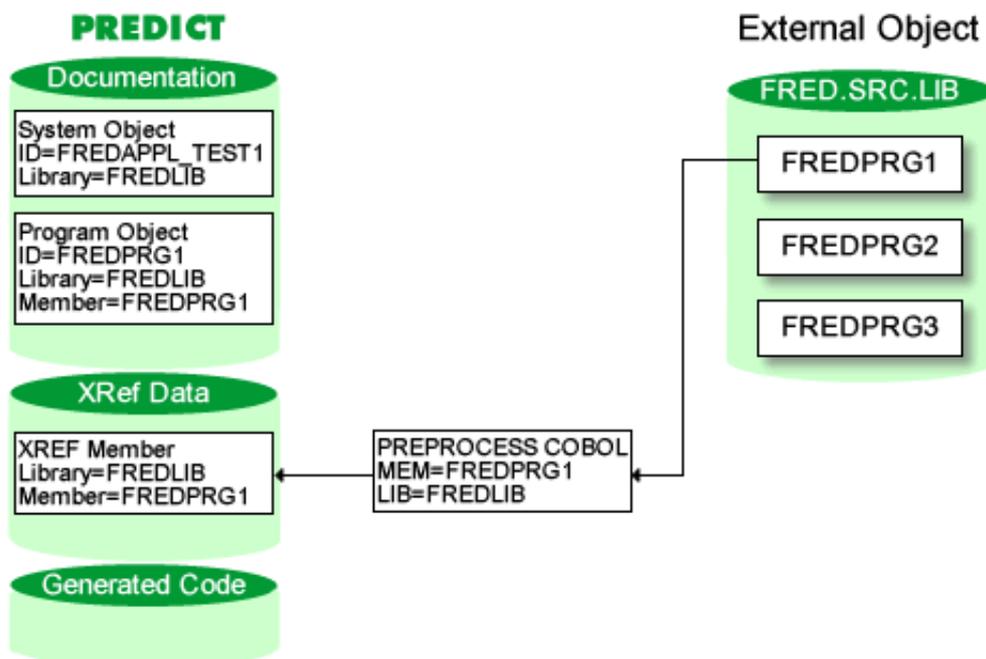
Documenting a 3GL Program with a Predict Object of Type Program



Program implementation pointer Member corresponds to the implemented program as it is stored in the source or load library/directory.

Library must be defined in an object of type System if XRef data is to be created for the program.

Creating XRef Data for Implemented Programs

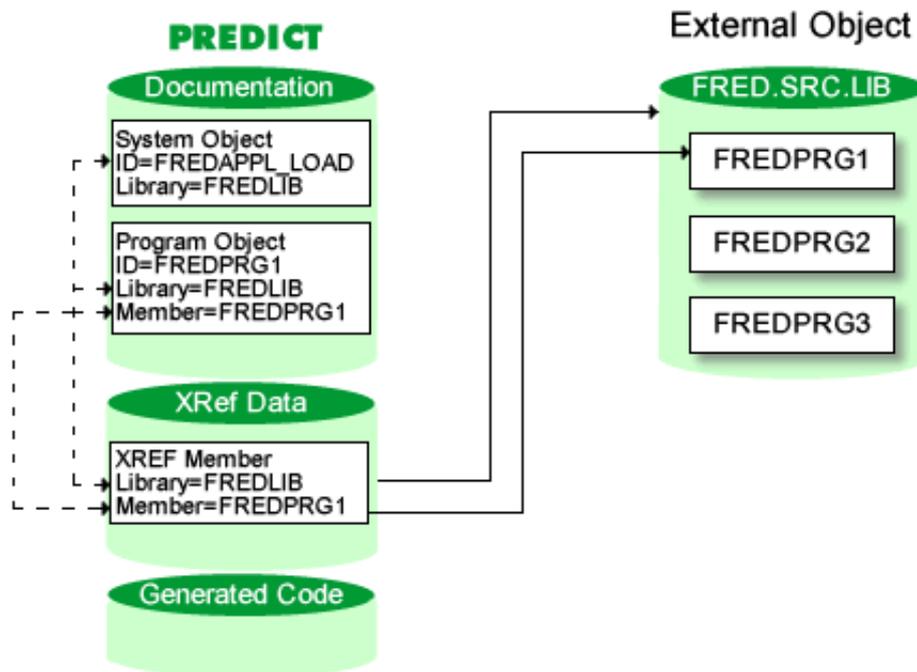


An implemented program is known to Predict only if XRef data exist.

In this example, the XRef data for COBOL program FREDPRG1 are created by assigning the program source as Workfile 1 and then processing it with the Predict Preprocessor.

The different methods of creating XRef data are listed in the section XRef Data for 3GL Applications and Programs.

Connecting External and Documentation Objects by Implementation Pointer



The Predict system object is now connected to external library, and the program object is now connected to the implemented program.

Documenting Entry Points for 3GL Programs

Entry points can be documented in Predict for the following languages:

- Assembler
- C
- COBOL
- FORTRAN
- PL/I
- Ada
- Other (language code O)

If a member name is entered in the implementation pointer of a program object, this name is automatically entered as an entry point. Other entry points can be entered using one of the following methods:

- **Documenting entry points manually**
 - with program maintenance function Edit entry-points (Code R)
 - or by setting "Attribute" to Y, and then selecting Entry points in the "Additional attributes" window.
- **Documenting entry points with editor commands**

If XRef data already exists for the implemented program, call the function Edit Entry Points with one of the

methods above and enter command ACTIVE or UPDATE in the editor command line.

- ACTIVE reads the entry names from the XRef data of the program into the editor workspace and marks them as < active. Entry names that have been entered manually but are not in the XRef data are marked < unused.
- UPDATE additionally deletes the entries marked < unused from the editor workspace.

See Program List Editor Commands in the section **Program** in the **Predefined Object Types in Predict documentation**.

- **Documenting entry points with function Redocument program**

See Redocumenting of 3GL Applications or Redocument Program section **Program** in the **Predefined Object Types in Predict documentation** for more information.

XRef Data for 3GL Applications and Programs

XRef data for 3GL programs plays two important roles in Predict:

- It contains information on the dependencies among implemented programs and between programs and other objects they use.
- It represents the implemented program in Predict. This means an XRef member corresponding to the implemented program must exist if the program is to be known to Predict.

How is XRef Data Created?

XRef data for applications is created by creating XRef data for one or more programs contained within the application. The method used for creating XRef data for programs depends on the program type:

For 3GL Programs

- By Adabas Native SQL (ADA, COBOL, FORTRAN and PL/I).
- By the Predict Preprocessor (Assembler, COBOL and PL/I). See the section Preprocessor in the **External Objects in Predict documentation**
- By Adabas SQL Server precompiler (C, COBOL, FORTRAN and PL/I).
See XRef Data for Adabas SQL Objects.

For Static SQL

If Natural for DB2 is installed, the function CREATE DBRM of Natural for DB2 creates XRef data for Static SQL access modules (DBRMs) and for Natural programs that use Static SQL. See Static SQL.

For System Programs

It is not always possible to create XRef data for a 3GL program using one of the above methods. This applies particularly to operating system routines, TP Monitor programming interfaces or other programs that are invoked from within a 3GL application but for which no source code is available.

However, these programs can be documented as program objects of type Documented or External program (subtypes D or E) with pseudo-language System program (language code Z).

For each program object of this type, Predict creates a minimal set of XRef data, containing directory information and a list of entry points.

What is Contained in 3GL XRef Data?

The following information is stored for 3GL programs:

- The name of the program and the application to which it belongs.
- The program type (only main program).
- The date and time the program was last cataloged.
- ID of the user who cataloged the program.
- ID of the terminal from which the program was cataloged. In batch mode the job name is given.
- The entry points defined in the program. The member name is always entered as one of the entry points.
- The entry points of invoked 3GL programs and the methods used to invoke them (only CALL, static SQL).
- The names of files used in the program and the type of file usage.
- Names of fields of files used in the program and the type of field usage.

How is XRef Data Used?

There are three main areas where XRef data is used:

- **Active Retrieval**
Predict active retrieval functions evaluate XRef data and Predict documentation data to determine
 - if objects documented in the dictionary are not yet implemented
 - if implemented programs are not yet documented or
 - if documentation data differs from the implementation.

XRef data also provides answers to questions such as

- which programs refer to file ABC*
- which programs call the entry point MAIN in program START in library FREDLIB.

For more information see the section Active Retrieval in the **Predict Reference documentation**.

- **LIST XREF**
XRef data for third generation languages is retrieved with functions of the Predict XRef menu. There are essentially three groups of functions: those which
 - retrieve information on specific types of objects in an application
 - retrieve information on the consistency of an application as a whole
 - manage sets.

For more information see the section LIST XREF for Third Generation Languages in the **Predict Reference documentation**.

- **Redocumenting of 3GL Applications**
3GL applications for which XRef data exists can be redocumented automatically in Predict. See Redocumenting of 3GL Applications.

Using Predict Functions When Developing 3GL Applications

Two major features are available for the development of 3GL applications:

- Generation of file layouts from Predict file objects in the syntax of several third generation languages. See appropriate sections in the section Generation in the **External Objects in Predict documentation**.
- Insertion of Predict generated file layouts and Adabas format buffers into 3GL source programs by the Predict Preprocessor. See the section Preprocessor in the **External Objects in Predict documentation** for more information.

Redocumenting of 3GL Applications

3GL applications for which XRef data exists can be automatically redocumented in Predict. The XRef data must have been created using one of the methods described in XRef Data for 3GL Applications and Programs.

The Redocument program function (see the section Program in the **Predefined Object Types in Predict documentation**) creates for each implemented program a new Predict Program object or updates an existing object and evaluates the XRef data to establish links to other program and file objects.

This results in a basic documentation of the application objects and their relationships, which can be extended by an abstract, extended description, keywords, owners etc.

Redocumenting COBOL Record Structures

Data definitions in the form of COBOL Copy Code members can be redocumented in Predict using the function Incorporate COBOL. A file object of type Sequential is created for each Copy Code member. See appropriate part of section Incorporation in the **External Objects in Predict documentation** for more information.