

Steplib Support

Natural as well as 3GL applications allow up to 8 steplibs for one main library. This structure can be documented in Predict with the object type library structure. This structure is evaluated by LIST XREF and active retrieval functions for programs and systems.

This section covers the following topics:

- General Information
 - Documenting Dynamic Structures
 - Steplib Support with Active Retrieval Functions
 - Steplib Support with LIST XREF for Natural
-

General Information

Predict supports the Natural steplib concept using the features listed below.

Object Type Library Structure

An object of type library structure documents a structure which describes a runtime or development environment (for example libraries for copy code). The system objects which document these libraries are linked as children to the library structure via *Contains SY*. The following rules apply:

- The first entry in the link list is the main library, the following entries are steplibs.
- The link list of a library structure can contain up to 10 systems of type A (Application):
 - the first system in the list is the main library
 - the default steplib *STEPLIB plus up to 8 additional steplibs can be defined.
- The link list can contain additional systems of type G (3GL Application), but the maximum number of linked systems is 15.
- Dummy objects and systems without an implementation pointer for Library are permitted in the link list, but these objects are ignored when the library structure is evaluated for active retrieval function Program using programs and all LIST XREF functions.

See the section Library Structure in the **Predefined Object Types in Predict documentation**.

Program Type Dynamic

Programs of type dynamic are used to document calls of programs of the same name from different steplibs depending on the library structure. The following rules apply:

- Because programs of type dynamic document any number of implemented members, no check is performed as to whether the members documented by the program are actually implemented.
- With the active retrieval function Programs using programs, programs of type dynamic are ignored as current objects.
- Programs of this type can only have children of type program (via *Uses PR concept*). The linked programs document the possible implementations. Therefore they all must use the same programtype and member name.

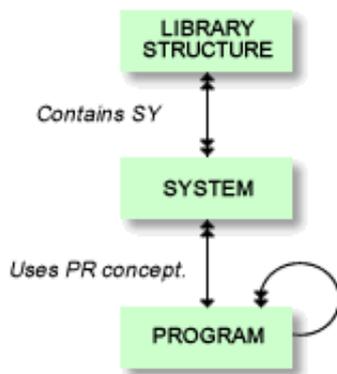
See Documenting Dynamic Structures.

Metadata Diagram

The diagram below is an extract of the metadata structure in Predict showing the object type library structure and its associations.

A library structure can have system children of type Application Library.

The first system of type Application Library is the main library, the other child systems are the steplib. The order of the children in the link list reflects the steplib hierarchy.



Active Retrieval Functions

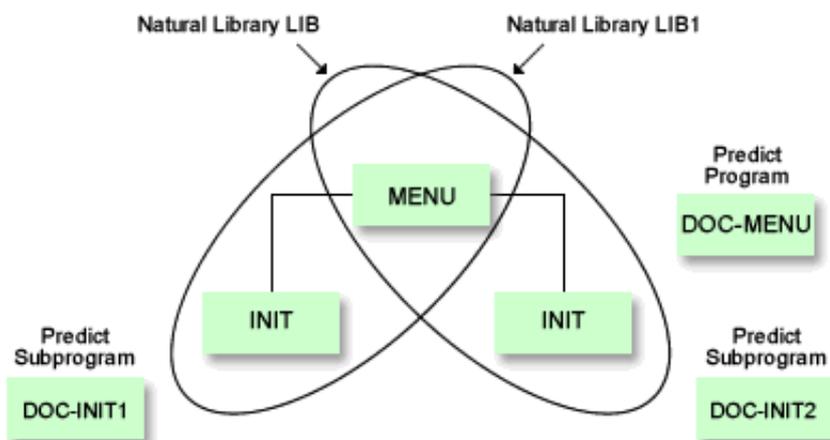
The following active retrieval functions evaluate library structures to retrieve documentation and XRef data according to a specified steplib structure:

- Programs using programs
- Systems containing programs

Documenting Dynamic Structures

Example

In the example below, member MENU calls one of two INIT members depending which library is active at runtime.



Documentation without Library Structure

The following table shows the objects needed to document the structure above without evaluating the steplib structure.

Object	Type	Subtype	Implementation Pointer			
			Member	Library	Fnr	DBnr
DOC-LIB-MAIN	System	Application Library	-	LIB-MAIN	54	180
DOC-LIB	System	Application Library	-	LIB	54	180
DOC-LIB1	System	Application Library	-	LIB1	64	180
DOC-MENU	Program	Program	MENU	LIB	54	180
DOC-INIT1	Program	Subprogram	INIT	LIB	54	180
DOC-INIT2	Program	Subprogram	INIT	LIB1	64	180

Enter DOC-INIT1 and DOC-INIT2 in the Program>Program link list of DOC-MENU. This has the disadvantage that you cannot tell which INIT member will be called by member MENU at runtime.

Documentation with Library Structure

The following table shows the objects needed to document the structure above in such a way that the steplib structure is evaluated.

Object	Type	Subtype	Implementation Pointer			
			Member	Library	Fnr	DBnr
DOC-LS1	Lib. Structure					
DOC-LS2	Lib. Structure					
DOC-LIB-MAIN	System	Application Library	-	LIB-MAIN	54	180
DOC-LIB	System	Application Library	-	LIB	54	180
DOC-LIB1	System	Application Library	-	LIB1	64	180
DOC-MENU	Program	Program	MENU	LIB	54	180
DOC-INIT	Program	Dynamic	INIT	See note below		
DOC-INIT1	Program	Subprogram	INIT	LIB	54	180
DOC-INIT2	Program	Subprogram	INIT	LIB1	64	180

Enter DOC-INIT (program of type Dynamic) in the link list of DOC-MENU. Member MENU will call up the member documented by DOC-INIT1 or DOC-INIT2 at runtime depending on the library structure.

Steplib Support with Active Retrieval Functions

The following active retrieval functions use library structures:

- Programs using programs
- Systems containing programs
- Programs using files

If the first two functions listed above are executed with the parameter Library structure, the steplib structure documented by the corresponding library structure object is evaluated.

Note:

These functions are also described in the section Active Retrieval in the **Predict Reference documentation**. The descriptions there apply **without** evaluating the steplib structure.

Function Program using Program

This section describes the active retrieval function Program using program where a library structure is evaluated. See the section Active Retrieval in the **Predict Reference documentation** for a description of this function without evaluating the library structure.

Specifying the Library Structure

- Enter fully qualified library structure ID. Asterisk notation can be used to select one library structure from a list.
- If the implementation pointer of the main library in the library structure is incomplete, a window appears in which you must enter the missing parameters Library, Fnr, or DBnr.

```

+----- Additional criteria -----+
Retrieval t ! Main Library/first Natural Library of the      !
Output mode ! Library Structure DOC-LS1                      !
Program ID  ! has no qualified Implementation Pointer.      !
in system  . ! Please enter following parameters:           !
Member     . . . . !                                         !
Library    . . . ! Library ..... ARH1                      !
Library str ! Fnr ..... 64                                  !
Entry      . . . . ! DBnr ..... 180                        !
Restriction !                                             !
Output opti ! Press ENTER to confirm                        !
+-----
```

If the main library in the library structure is a 3GL library, the first Natural library is taken as main library and must be given a fully qualified implementation pointer if required.

- The implementation pointer defined for the main library (or first Natural library) is used to append all other incomplete implementation pointers for all other libraries in the library structure.
- If you set the output option Cover page to Y, the library structure with complete implementation pointers for all libraries is displayed:

```

----- Cover page -----
Program ID ... DOC*

Library structure ID .. DOC-LS1

          Library   Fnr   DBnr
          ARH1      64    180
          ARH3      64    180
          ARH        54    180
          ARH3GL2   255   255
```

In the example above, the missing values have been appended with the DBnr 180 specified under Additional criteria above.

Determining the Current Objects to be Output

The following rules apply:

- Programs of type Dynamic are ignored as current objects.
- Only programs with identical implementation information to a system contained in the library structure are output:
 - if the implementation pointer of the program object is complete (Member, Library, Fnr, DBnr), this is evaluated;
 - if the implementation pointer of the program object is incomplete, the XRef data is evaluated.

Determining the Related Objects to be Output

Related objects are evaluated against documentation data (implementation pointer) and XRef data. The following rules apply:

- If a program of type Dynamic is linked to the current object via *Uses PR concept.*, this link must be resolved. The programs represented by the program of type Dynamic are checked, and the program with the implementation pointer that best matches the library structure replaces the program of type Dynamic.
- During checks as to whether a program is implemented, the entire library structure is evaluated. If the implemented member is found in a steplib, the program is marked as I and the comment >>>impl. in steplib XXXXXXXX DBnr 99999 Fnr 99999<<<.

Sample Output

The screen below shows sample output for function Programs using Programs.

```

13:27:36          ***** P R E D I C T 4.3.1 *****          2003-05-31
                    - List Program Using Programs -

Program ID ..... * DOC-MENU
Type ..... Program
-----

Implementation
Member .. MENU      Library .. ARH      Fnr .. 54  DBnr .. 180

Cnt  Program ID          Ty La Member      Library  Fnr DBnr L D I U
   1  DOC-INIT2          P  N  INIT      ARH1     64 180 L D
      >>> Dynamic call defined in DOC-INIT <<<

*** End of report ***

Command ==>>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
                    Quit          RFind Flip  -      +          Left  Right
    
```

Comments

Program DOC-INIT2 is linked via *Uses PR concept.* to a program of type Dynamic named DOC-INIT. The link list of the dynamic program DOC-INIT is checked and the program with the most complete implementation pointer is given.

Function Systems containing Programs

This section describes the active retrieval function Systems containing programs where a library structure is evaluated. See the section Active Retrieval in the **Predict Reference documentation** for a description of this function without evaluating the library structure.

Specifying the Library Structure

- Enter fully qualified library structure ID. Asterisk notation can be used to select one library structure from a list.
- If the implementation pointer of the main library in the library structure is incomplete, a window appears in which you must enter the missing parameters Library, Fnr, or DBnr.

```

+----- Additional criteria -----+
Retrieval t ! Main Library/first Natural Library of the !
Output mode ! Library Structure DOC-LS1 !
Program ID ! has no qualified Implementation Pointer. !
in system . ! Please enter following parameters: !
Member .... ! !
Library ... ! Library ..... ARH1 !
Library str ! Fnr ..... 64 !
Entry ..... ! DBnr ..... 180 !
Restriction ! !
Output opti ! Press ENTER to confirm !
+-----
```

If the main library in the library structure is a 3GL library, the first Natural library is taken as main library and must be given a fully qualified implementation pointer if required.

- The implementation pointer defined for the main library (or first Natural library) is used to append all other incomplete implementation pointers for all other libraries in the library structure.
- If you set the output option Cover page to Y, the library structure with complete implementation pointers for all libraries is displayed:

```

----- Cover page -----
System ID ... DOC*

Library structure ID .. DOC-LS1

Library   Fnr   DBnr
ARH1      64   180
ARH3      64   180
ARH       54   180
ARH3GL2   255  255
```

In the example above, the missing values have been appended with the DBnr 180 specified under Additional criteria above.

Determining the Current Objects for Output

XRef data is evaluated, and only systems with an identical implementation pointer to one of the systems in the library structure are given.

Determining the Related Objects for Output

A program is marked as implemented if it is contained in the current library. The current library is the library documented by the system specified.

A program is also marked as implemented if it is linked via *Uses PR concept*. to a system documented as a steplib. (The program would not be marked as implemented if you were working without a library structure.)

A note is given indicating in which other steplib(s) of the library structure the program is implemented.

Steplib Support with LIST XREF for Natural

The three possible methods of evaluating XRef data are listed below. Select the method you require in the LIST XREF menu before you call a function. This option is valid for the duration of your session or until you select another option in the LIST XREF menu.

See also section LIST XREF for Natural in the **Predict Reference documentation**.

Using the command INFO you can display at any time during your LIST XREF session all libraries that will be evaluated by LIST XREF functions. The current library is marked with an arrow.

The Library Structure Documented in Predict

The link list Library structure to System is evaluated via *Contains SY*. Each system in the list is checked as follows:

- If no information is present in the implementation pointer of the system, the system is ignored.
- If the implementation pointer is incomplete, the system searches for possible XRef data. This XRef data is used to supply the missing Fnr and DBnr information in the implementation pointer.
- If no XRef data is found, the values of the current FUSER file are used to supply the missing DBnr and Fnr information in the implementation pointer.
- If the current library is a Natural library, the structure is appended with --> *STEPLIB <--.

Runtime Structure

The runtime structure is determined as follows. The following rules apply:

- The current library always appears first in the list. If this library is documented in Predict, the corresponding system ID is also displayed.
- With Natural Security, up to 8 Libraries can be specified as steplibs with Library, DBnr and Fnr.
- The default steplib is declared in the Natural parameter module NATPARAM or allocated with the dynamic parameter STEPLIB when starting Natural (*STEPLIB).

Without any Structure

The LIST XREF functions evaluate XRef data without specification of steplibs. Only objects in the current library are displayed.

Steplib Support in Batch Mode

In batch mode, too, there are three possible methods of evaluating XRef data:

- **STRUCTURE <structure-name>**
With this command you can specify which library structure is to be used for evaluating XRef data.
- **STRUCTURE *R**
This command specifies that the runtime structure is to be used for evaluating XRef data.

- **No Structure specified**

If you do not specify any structure, LIST XREF functions work without evaluation of steplib specification.

Effects of Steplib Support on LIST XREF

Steplib support affects LIST XREF functions as follows. A distinction is made between Top-down and Bottom-up functions:

Top-down

Example: Function Program using program

```

09:50:43          ***** P R E D I C T 4.3.1 *****                2003-05-31
Library: PDLX          - Invoked Programs -                DBnr:   180 Fnr:   54
Command: PROG * (*) USING PROG * (*) WITH * VIA *        Page:    1

      T:Program          using          via
-----
  1 P:ZPDFIELD          1 N:N-BUFEDT <<- nfn             Callnat
                                via ZPDP&
  2 P:ZPDP0             1 P:ZPDP1             Fetch
                                2 S:ZPDS1 (NEWDICLX,180,54)
                                Function: SUB-IN-ZPDS
    
```

Comments

Only programs in the current library that call other programs are displayed.

to 1	The note <<- nfn means that the called program was not found within the structure specified. "Not found" in this context means that no XRef data is present.
to 2	If the called program is not contained in the current library, it is displayed in parentheses with DBnr and Fnr.

Bottom-up

Example: Function Programs referenced in programs

Comments

The note <--steplib means that the evaluation was stopped at this point because the called program is contained in another library.

```

09:44:21          ***** P R E D I C T 4.3.1 *****                2003-05-31
Library: NEWDICLX          - XRef Menu -                DBnr: 180 Fnr: 54

Structure .....: LS-NEWDICLX

      System Id                Library  Fnr  DBnr
      -----
      PD-COB                   *SYSCOB* 255  255
-->PD-NEWDICLX                NEWDICLX 180   54
      --> *STEPLIB <--        SYSTEM    180   54

```