Data Areas

The following topics are covered:

- Use of Data Areas
- Local Data Area
- Global Data Area
- Parameter Data Area

Use of Data Areas

As explained in Defining Fields, all fields that are to be used in a program have to be defined in a DEFINE DATA statement.

The fields can be defined within the DEFINE DATA statement itself; or they can be defined outside the program in a separate data area, with the DEFINE DATA statement referencing that data area.

Natural supports three types of data areas:

- Local Data Area
  In a local data area, you define the data elements that are to be used by a single Natural module in an application.
- Global Data Area
  In a global data area, you define the data elements that are to be used by more than one Natural program, routine, etc. in an application.
- Parameter Data Area
  In a parameter data area, you define the fields that are passed as parameters to a subprogram, external subroutine or helproutine.

Local Data Area

Variables defined as local are used only within a single Natural module. There are two options for defining local data:

- You can define the data within the program.
- You can define the data in a local data area outside the program.

In the first example, the fields are defined within the DEFINE DATA statement of the program. In the second example, the same fields are defined in a local data area (LDA), and the DEFINE DATA statement only contains a reference to that data area.

Example 1 - Fields Defined within a DEFINE DATA Statement:

```
DEFINE DATA LOCAL
1 VIEWEMP VIEW OF EMPLOYEES
  2 NAME
  2 FIRST-NAME
  2 PERSONNEL-ID
1 #VARI-A (A20)
1 #VARI-B (N3.2)
1 #VARI-C (I4)
END-DEFINE
...
```

Example 2 - Fields Defined in a Separate Data Area:

Program:

```
DEFINE DATA LOCAL
    USING LDA39
END-DEFINE
  ...
```

Local Data Area "LDA39":

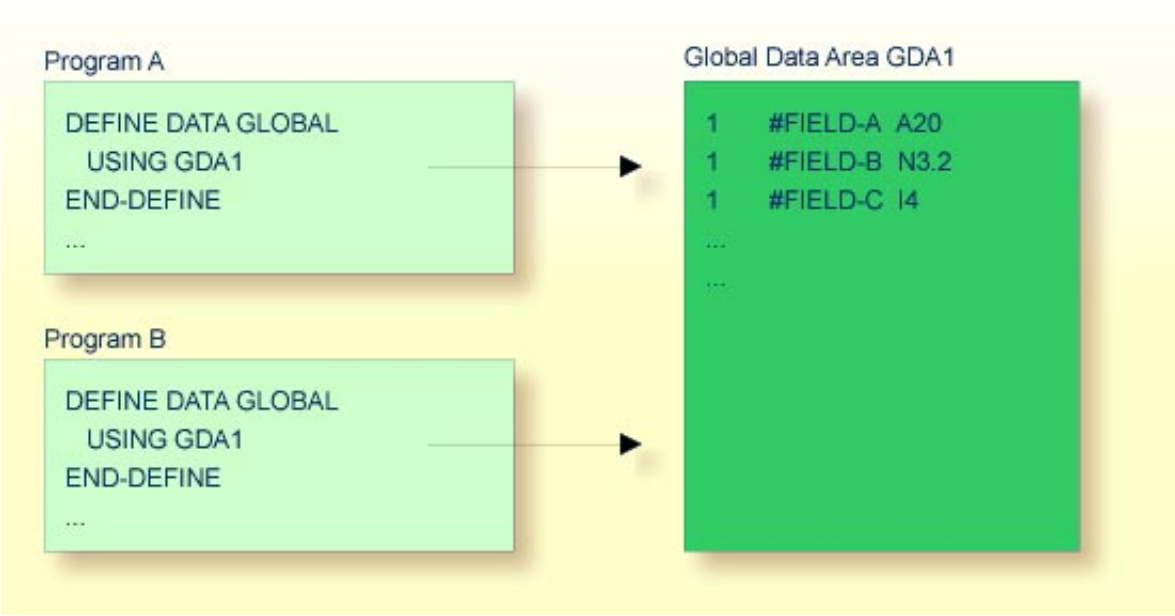
```
I T L Name                            F Leng Index/Init/EM/Name/Comment
- - - -------------------------------- - ---- ---------------------------------
  V 1 VIEWEMP                                 EMPLOYEES
    2 NAME                            A   20
    2 FIRST-NAME                      A   20
    2 PERSONNEL-ID                    A    8
    1 #VARI-A                         A   20
    1 #VARI-B                         N  3.2
    1 #VARI-C                         I    4
```

For a clear application structure, it is usually better to define fields in data areas outside the programs.

Global Data Area

In a global data area (GDA), you define the data elements that are to be used by more than one program, routine, etc. in an application.

Variables defined in a global data area may be referenced by several objects in an application.



The global data area and the objects which reference it must be in the same library (or a steplib).

Global data areas must be defined with the data area editor, and a program using that data area must reference it in the DEFINE DATA statement. Any number of main programs, external subroutines and helproutines can share the same global data area.

 Copyright Software AG 2003

Each object can reference only one global data area; that is, a DEFINE DATA statement must not contain more than one GLOBAL clause.

Note:
When you build an application where multiple objects share a global data area, remember that modifications to a global data area affect all programs or routines that reference that data area. Therefore these objects must be STOWed again after the global data area has been modified.

When are Global Data Areas Initialized?

A global data area is initialized when it is used for the first time. It remains active in the current Natural session (that is, the variables in the global data area retain their contents) until:

- the next LOGON, or
- another global data area is used on the same level (levels are described later in this section), or
- a RELEASE VARIABLES statement is executed. In this case, the variables in the global data area are reset when either the execution of the level 1 program is finished, or the program invokes another program via a FETCH or RUN statement.

Note:
If a GDA named "COMMON" exists in a library, the program named ACOMMON is invoked automatically when you LOGON to that library.

Parameter Data Area

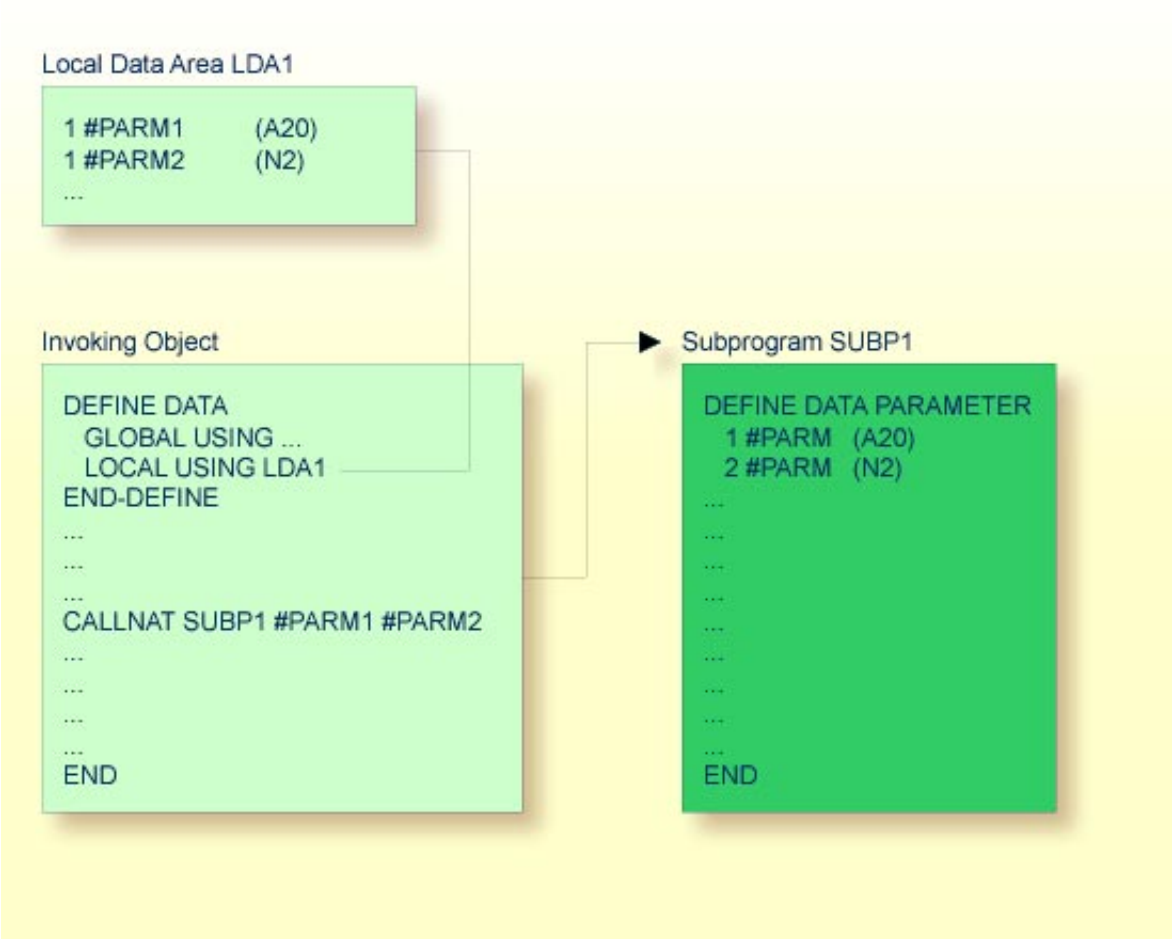
Parameter data areas (PDAs) are used by subprograms and external subroutines.

A subprogram is invoked with a CALLNAT statement. With the CALLNAT statement, parameters can be passed from the invoking object to the subprogram.

These parameters must be defined with a DEFINE DATA PARAMETER statement in the subprogram:

- they can be defined in the PARAMETER clause of the DEFINE DATA statement itself; or
- they can be defined in a separate parameter data area, with the DEFINE DATA PARAMETER statement referencing that parameter data area.

Parameter Defined within DEFINE DATA PARAMETER Statement



Parameter Defined in Parameter Data Area

In the same way, parameters that are passed to an external subroutine via a PERFORM statement must be defined with a DEFINE DATA PARAMETER statement in the external subroutine.

In the invoking object, the parameter variables passed to the subprogram/ subroutine need not be defined in a parameter data area; in the illustrations above, they are defined in the local data area used by the invoking object (but they could also be defined in a global data area).

The sequence, format and length of the parameters specified with the CALLNAT/PERFORM statement in the invoking object must exactly match the sequence, format and length of the fields specified in the DEFINE DATA PARAMETER statement of the invoked subprogram/subroutine. However, the names of the variables in the invoking object and the invoked subprogram/subroutine need not be the same (as the parameter data are transferred by address, not by name).

 Copyright Software AG 2003