Defining Local Data

General syntax of `DEFINE DATA LOCAL`:

```
[ LOCAL { USING { local-data-area     } } ] ...
[       {       { parameter-data-area } } ]
[       { data-definition ...           } ]
```

This chapter covers the following topics:

- Function
- Restriction
- Syntax Description

For an explanation of the symbols used in the syntax diagram, see *Syntax Symbols*.

Function

The `DEFINE DATA LOCAL` statement is used to define the data elements that are to be used exclusively by a single Natural module in an application. These elements or fields can be defined within the statement itself (see *data-definition*); or they can be defined outside the program in a separate local data area (LDA) or a parameter data area (PDA), with the statement referencing that data area.

Restriction

The LDA and the objects which reference it must be contained in the same library (or in a steplib).

Syntax Description

local-data-area	A local data area contains predefined data elements which can be included in the `DEFINE DATA LOCAL` statement. You may reference more than one data area; in that case you have to repeat the reserved words `LOCAL` and `USING`, for example:
	`DEFINE DATA LOCAL`
	`  LOCAL USING DATX_L`
	`  LOCAL USING DATX_P`
	`  ...`
	`END-DEFINE ;`
	For further information, see also *Defining Fields in a Separate Data Area* and *Local Data Area*, *Example 2* in the *Programming Guide*.

parameter-data-area	A data area referenced with `DEFINE DATA LOCAL` may also be a parameter data area (PDA). By using a PDA as an LDA you can avoid the extra effort of creating an LDA that has the same structure as the PDA.
direct-data-definition	See *Direct Data Definition* below.
END-DEFINE	The Natural reserved word `END-DEFINE` must be used to end the `DEFINE DATA` statement.

Direct Data Definition

Local data can be defined directly within a program or routine. For direct data definition, the following syntax applies:

$$\left\{ \textit{level} \left\{ \begin{array}{l} \textit{group-name}\ [(\textit{array-definition})] \\ \textit{variable-definition} \\ \textit{view-definition} \\ \textit{redefinition} \\ \textit{handle-definition} \end{array} \right\} \right\}$$

For further information, see

- *Example 1 - DEFINE DATA LOCAL* (Direct Data Definition)
- *Defining Fields within a DEFINE DATA Statement* in the *Programming Guide*
- *Local Data Area*, *Example 1* in the *Programming Guide*

Syntax Element Description for Direct Data Definition:

level	Level number is a 1- or 2-digit number in the range from 01 to 99 (the leading zero is optional) used in conjunction with field grouping. Fields assigned a level number of 02 or greater are considered to be a part of the immediately preceding group which has been assigned a lower level number. The definition of a group enables reference to a series of fields (may also be only 1 field) by using the group name. With certain statements (`CALL`, `CALLNAT`, `RESET`, `WRITE`, etc.), you may specify the group name as a shortcut to reference the fields contained in the group. A group may consist of other groups. When assigning the level numbers for a group, no level numbers may be skipped. A view-definition must always be defined at Level 1.
group-name	The name of a group. The name must adhere to the rules for defining a Natural variable name. See also the following sections: ● *Naming Conventions for User-Defined Variables* in the *Using Natural* documentation. ● *Qualifying Data Structures* in the *Programming Guide*.
array-definition	With an *array-definition*, you define the lower and upper bounds of dimensions in an array-definition. See *Array Dimension Definition*.
variable-definition	A *variable-definition* is used to define a single field/variable that may be single-valued (scalar) or multi-valued (array). See *Variable Definition*.
view-definition	A *view-definition* is used to define a view as derived from a data definition module (DDM). See *View Definition*.
redefinition	A *redefinition* may be used to redefine a group, a view, a DDM field or a single field/variable (that is a scalar or an array). See *Redefinition*.
handle-definition	A handle identifies a dialog element in code and is stored in handle variables. See *Handle Definition*.