DEFINE CLASS

```
DEFINE CLASS class-name
[ [WITH] ACTIVATION [POLICY] {          EM                        } ]
[                            {          ES                        } ]
[                            {          IM                        } ]
[ OBJECT                     { USING { local-data-area     }      } ] ...
[                            {       { parameter-data-area }      } ]
[                            { data-definition ...                } ]
[ LOCAL                      { USING { local-data-area     }      } ] ...
[                            {       { parameter-data-area }      } ]
[                            { data-definition                    } ]
[ID class-GUID]
[ INTERFACE USING copycode ] ...
[ INTERFACE                ]
[PROPERTY ] ...
[METHOD ] ...
END-CLASS
```

This chapter covers the following topics:

- Function
- Syntax Description

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

Related Statements: `CREATE OBJECT` | `INTERFACE` | `METHOD` | `PROPERTY` | `SEND METHOD`

Belongs to Function Group: *Component Based Programming*

Function

The `DEFINE CLASS` statement is used to specify a class from within a Natural class module. A Natural class module consists of one `DEFINE CLASS` statement followed by an `END` statement.

Syntax Description

Syntax Element	Description
class-name	**Class Name:** This is the name that is used by clients to create objects of this class. The name can be up to a maximum of 32 characters long. The name may contain periods: this can be used to construct class names such as *company-name.application-name.class-name* Each part between the periods (...) must conform to the *Naming Conventions for User-Defined Variables*. If the class is planned to be used by clients written in different programming languages, the class name should be chosen in a way that it does not conflict with the naming conventions that apply in these languages.
WITH ACTIVATION POLICY	**WITH ACTIVATION POLICY Clause:** This clause is used to define explicitly the activation policy which is registered for the current class. You can set the following parameters: EM Sets activation policy "ExternalMultiple". ES Sets activation policy "ExternalSingle". IM Sets activation policy "InternalMultiple". When the class is stowed and registered, the setting in the WITH ACTIVATION POLICY clause overrides the ACTPOLICY profile parameter, but is in turn overridden by manual registration using the REGISTER command with an explicit activation policy definition. For further information, see the section *Activation Policies* in the *Operations* documentation.
OBJECT	**OBJECT Clause:** This clause is used to define the object data. The syntax of the OBJECT clause is the same as for the LOCAL clause of the DEFINE DATA statement. For further information, see the description of the LOCAL clause of the DEFINE DATA statement.

Syntax Element	Description
`LOCAL`	**LOCAL Clause:** This clause is only used to include globally unique IDs (GUIDs) in the class definition. GUIDs need only be defined if a class is to be registered with DCOM. GUIDs are mostly defined in a local data area. For further information, see the section *Globally Unique Identifiers (GUIDs)* in the *Programming Guide*. The syntax of the `LOCAL` clause is the same as for the `LOCAL` clause of the `DEFINE DATA` statement. For further information, see the description of the `LOCAL` clause of the `DEFINE DATA` statement.
`ID`	**ID Clause:** This clause is used to assign a globally unique ID to the class. The class GUID is the name of a GUID defined in the data area that is included by the `LOCAL` clause. The class GUID is a (named) alphanumeric constant. A GUID must be assigned to a class if it is to be registered with DCOM.
`INTERFACE USING`	**INTERFACE Clause:** This clause is used to include copycode that contains `INTERFACE` statements.
`copycode`	**Copycode:** The copycode used by the `INTERFACE USING` clause may contain one or more `INTERFACE` statements.
`PROPERTY`	**PROPERTY Statement:** The `PROPERTY` statement is used to assign an object data variable operand as the implementation to a property, outside an interface definition.
`METHOD`	**METHOD Statement:** The `METHOD` statement is used to assign a subprogram as the implementation to a method, outside an interface definition.
`END-CLASS`	**End of DEFINE CLASS Statement:** The Natural reserved word `END-CLASS` must be used to end the `DEFINE CLASS` statement.