NaturalX Servers

This chapter covers the following topics:

- COM Classes and Servers
- NaturalX Classes and Servers
- NaturalX Servers and Natural Sessions under Windows
- The Role of the Server ID
- Organizing Server IDs

COM Classes and Servers

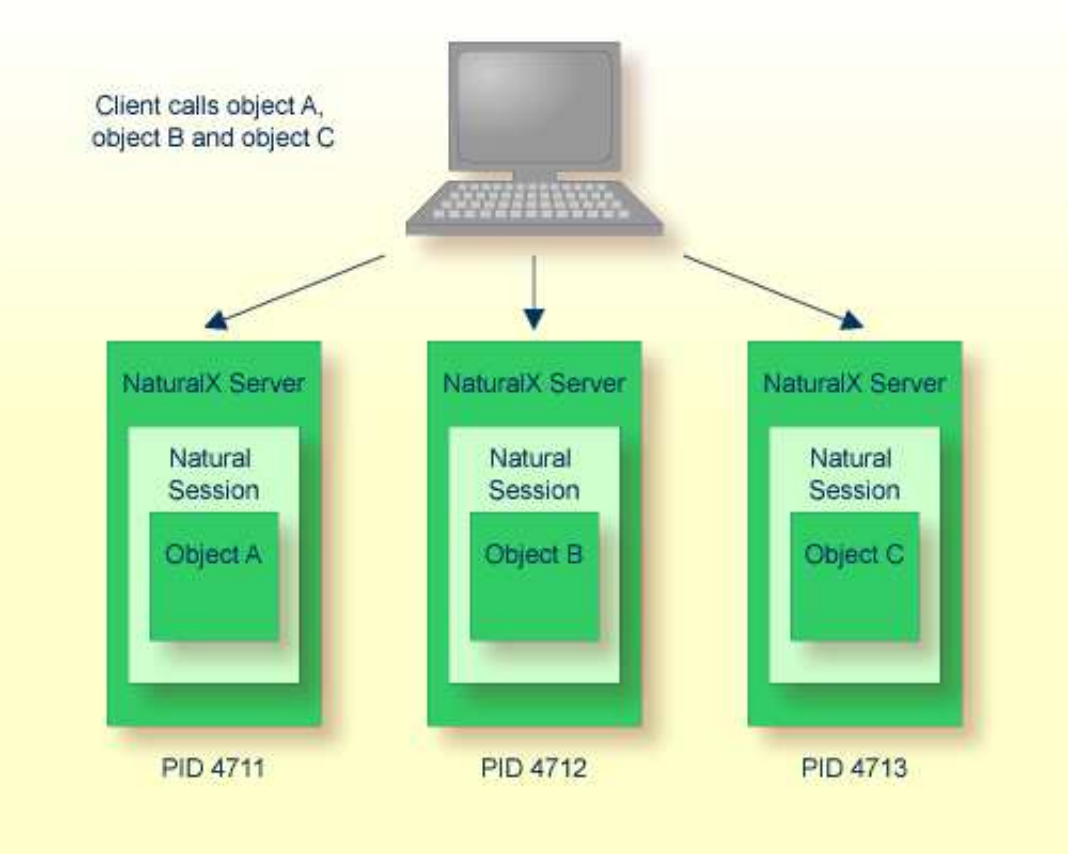
Each COM class must be hosted by a server process. The server process has a number of administrative and technical responsibilities, such as making the class and its interfaces available to DCOM and maintaining the memory occupied by the objects created. Whenever a client requests a new object of a certain class, DCOM checks whether the corresponding server process is already running. If this is not the case, DCOM launches it and passes the request to the server. When the server starts up, it makes its classes available to DCOM. While the server is running, it executes client requests for creation and deletion of objects and execution of methods. When the last object maintained by a server is deleted, the server shuts down automatically. For more detailed information about DCOM classes and servers, please refer to the Microsoft DCOM specification.

NaturalX Classes and Servers

Classes implemented with Natural can be made accessible as DCOM classes. But with Natural, it is not necessary to implement DCOM servers to host the classes. Instead, NaturalX itself performs the tasks of a DCOM server. NaturalX acts as a generic DCOM server for all classes written in Natural. The task that remains for a Natural class developer is just to implement the classes and to assign them to a NaturalX server.

NaturalX Servers and Natural Sessions under Windows

Under Windows, each Natural session runs in its own exclusive NaturalX server process.



The Role of the Server ID

One of the tasks of a DCOM server is to make its classes available to DCOM during startup. But since NaturalX acts as a generic DCOM server, it has no built-in knowledge about the classes it shall provide. Instead, it finds the list of these classes in the system registry under the key of its server ID. The server ID is a Natural-owned key in the system registry, keeping together all classes that belong to a given NaturalX server. It is an arbitrary alphanumeric string of 32 characters which does not contain blanks and which is not case sensitive.

How does a NaturalX server know under which server ID it is running? The server ID is defined with the Natural parameter `COMSERVERID`. This parameter is either passed to a NaturalX server as a dynamic parameter, or it is defined in the Natural parameter file.

How are classes assigned to server IDs? Assume Natural has been started with a certain server ID. Then every class that a user registers during this Natural session is entered into the system registry under the current server ID.

Server IDs provide a means of grouping classes created in Natural and assigning them to different NaturalX server processes. The use of server IDs is, however, not compulsory: if Natural is started without a server ID, all Natural classes are registered under the predefined server ID "Default".

Example

Consider the example Employees application consisting of the classes `DepartmentList`, `EmployeesList` and `Employee` (this application is contained in the example library `SYSEXCOM`). These three classes are to be hosted by a NaturalX server called Employees.

1. Start Natural with the desired server ID.
2. Logon to the library `SYSEXCOM`.

```
LOGON SYSEXCOM
```

3. Register the classes with the `REGISTER` command on the Natural command line.

```
REGISTER *
```

The three classes are now registered under the server ID "Employees".

Whenever an object of one of these classes is requested, DCOM will start a NaturalX server process with the server ID "Employees", which will then provide the classes.

Organizing Server IDs

The server ID represents the set of all classes that are made available to DCOM when the corresponding NaturalX server is started. It is recommended that you group under one server ID those classes that form an application from the business point of view, or that otherwise belong together logically. Similarly, classes that are never used in the same context should be registered under different server IDs. Another criterion for the assignment of classes to server IDs is security (see the section *Security with NaturalX*). From this aspect, it makes sense to group under the same server ID those classes for which common authorizations will be defined.