Introduction

Basically, business services are comprised of the following components:

- Natural subprograms that contain no screen Input/Output
- Subprogram proxies, which are specialized Natural subprograms that retrieve data off the wire and prepare it for the subprogram parameter data area (PDA)
- A repository that stores the business service metadata, such as the service name, a description of the service, and which methods the service will use.

  The repository metadata can be used during development to determine which coding options are available and at runtime to lock users out of certain business services or methods. To provide a logical grouping for security purposes, business services are stored in domains within each repository. The repository also isolates client developers (Java or .NET) from the Natural code. They simply select a business service in the repository and use the Business Services wizard to generate the corresponding Java or .NET class.

After generating a business service, you can use the repository to test the service and then search the repository to see the modules that were added for the service. The search option also provides an ideal method of determining which business services currently exist.

Naming Conventions for Modules

For information on the naming conventions used by the wizards for the Natural modules and how to change these conventions for your requirements, see Modify/Test the Naming Conventions for Natural Objects.

Subprograms Listed in the Repository

The repository lists all subprograms specified in the Natural Construct specifications. Although we've implied a one-to-one relationship between a subprogram proxy and a subprogram, more than one subprogram can be listed for each business service. This happens when the Business Service wizard uses the Natural Construct Object models to generate the subprograms. For example, the wizard can generate a subprogram that "wraps" multiple subprograms into one wrapper subprogram, which calls the other subprograms as required. Because the wrapper subprogram is generated by Natural Construct, NBS can determine which subprograms it is calling and list them in the repository.

Note:
If a subprogram that is called by the wrapper subprogram calls another subprogram, the other subprogram will not be listed in the repository.

Another example of more than one subprogram listed in the repository is the single view data access service, which contains:

- A subprogram that represents the single view
- An object browse subprogram that retrieves the rows of data

- Optionally, an object maintenance subprogram that maintains the data

Again, NBS will list these subprograms in the repository.

There can also be more than one subprogram proxy associated with each business service. An example of this is the compound data access service. The generated object browse subprogram and object maintenance subprogram for this type of service each require a subprogram proxy.