ℓ software ^AG

Natural Business Services

Natural Business Services Natural Plug-in

Version 5.3.1

February 2010

Natural Business Services

Table of Contents

1 Natural Business Services Natural Plug-in

Natural Business Services Natural Plug-in describes the Natural plug-in supplied by Natural Business Services.

This documentation is intended for developers who are familiar with Natural for Windows and Natural Development Server (NDV) and want to use the Natural Business Services Natural plug-in to create and maintain business services.

Natural Business Services Natural Plug-in covers the following topics:

●	**Introduction**	Describes the Natural plug-in for Natural Business Services and how to install and activate the plug-in.
●	**Using the Business Service Repository**	Describes the common tasks you can perform when using the Business Services repository explorer. These tasks include: ■ Access the business services ■ Find a business service ■ Set trace options ■ Create a directly enabled Web service ■ Check environment information for a connection
●	**Developing Business Services**	Describes the procedures to create and maintain Natural Business Services. These procedures include: ■ Create a new business service ■ Regenerate a business service ■ Regenerate a business service proxy ■ Test a business service method ■ Test a business service ■ Delete a business service

		■ Edit a service definition
		■ Edit the service modules
		■ Transform a browse module
🔵	**Administering Business Services**	Describes the procedures to administer business services. These procedures include: ■ Configure the Natural plug-in ■ Define security for business service domains ■ Audit security exceptions for a connection ■ Audit domains ■ Audit business services ■ Deploy the business services in a domain to another repository ■ Deploy a business service to another repository or environment ■ Access the Communication Log report ■ Generate a Natural client

2　Introduction

The Natural Business Services Natural plug-in allows you to create, maintain, and administer business services. To create a business service, you can use any Natural subprogram that does not contain screen I/O (for example, subprograms generated using the Object-Maint-Subp, Object-Browse-Subp, Object-Browse-Select-Subp, or Object-Generic-Subp models).

The plug-in uses the Natural Business Services runtime via the Natural for Windows protocol to communicate with a Natural Business Services environment running on a mainframe, Unix, or Linux platform. Natural Development Server (NDV) provides the server environment for Natural for Windows.

> **Note:** Natural Business Services includes tools and utilities you can use when developing business services. For information, see Natural Business Services Tools and Utilities.

Once a business service has been created and tested, it can be used with other tools to generate classes or Web services. The generated classes and/or Web services are used to access business services at runtime. Typically, EntireX middleware is used to link the classes and/or Web services to the servers.

You can create various types of Web services to use with a business service, depending on which client technology you will be using. The Natural Business Services Natural plug-in uses a .NET runtime and IIS (Internet Information Server), the Visual Studio add-in uses the Visual Studio SDK and IIS, and the Eclipse plug-in uses non-Microsoft solutions.

Once a Web service has been created, it can be used in various ways. For example, it can be used:

- In a B2B scenario, where one computer talks to another computer with no user interface
- With a .NET or Java application that uses a sophisticated GUI front end
- With a web application generated by the workbench web application wizards

> **Tip:** One of the reasons for using a business service is to reduce redundant code. Before creating a new business service, search the existing business services for a service that does what you want.

Installation

The Natural plug-in is installed during the installation of Natural Business Services. Unlike other plug-ins, it is not installed by Natural. If you did not install the plug-in at that time, you can re-run the setup program and add the component now. Once installed, the plug-in is available through the Natural for Windows Plug-in Manager.

> **Note:** For information on installing the Natural Business Services Natural plug-in, see Natural Business Services Installation on Windows.
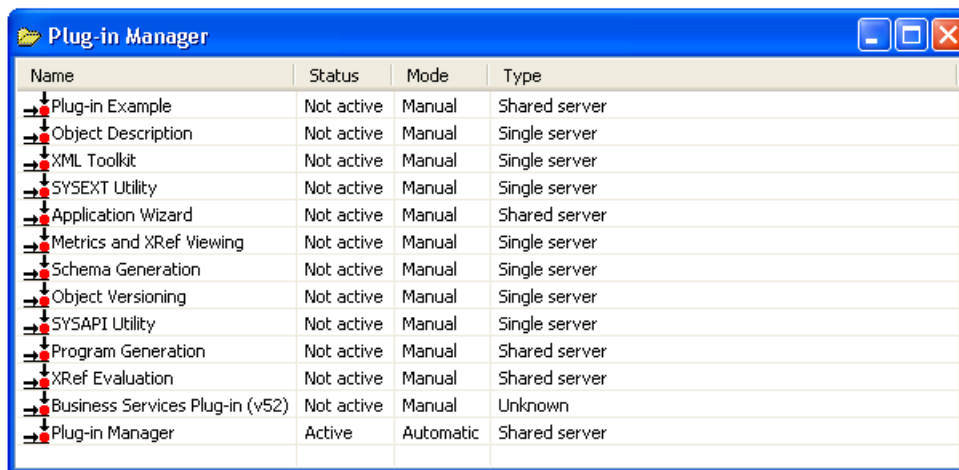
▶ **To activate the Natural plug-in:**

1   Invoke Natural for Windows.

   For information, see the Natural for Windows documentation.

2   Select **Configuration Tools > Plug-in Manager** on the **Tools** menu.

   The **Plug-in Manager** window is displayed. For example:

   This window displays the available plug-ins.

3   Open the context menu for the Business Service plug-in.

4   Select **Activate**.

   The **Status** is now Active.

   💡 **Tip:** If you want the Business Service plug-in to be available when you start Natural,
      open the context menu for the Business Service plug-in and set the activation mode to
      Automatic.

5   Close the **Plug-in Manager** window.

For information on accessing the business services, see **Access Business Services**.

3   Using the Business Service Repository

The **Business Service** repository provides access to business services on the server. It lists the connections for all business services in a tree view, which you can expand to display .

Use the **Business Service** repository to:

Access Business Services
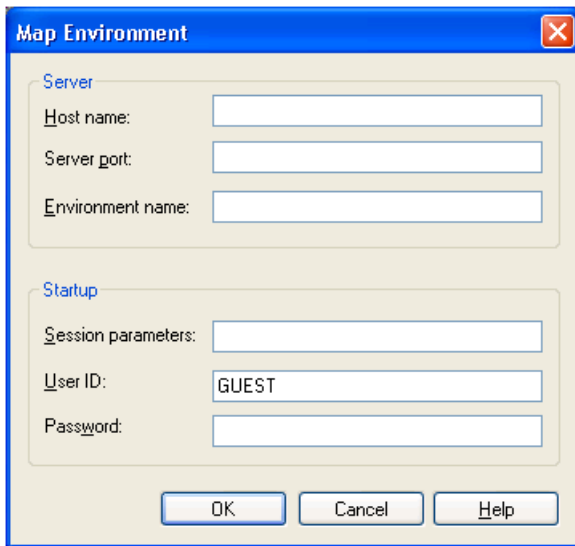
▶ **To access business services:**

1   Map to the remote Natural development environment where the **Business Service** repository is located.

For information on how to map to a remote Natural connection, see the Natural for Windows Single Point of Development (SPoD) documentation.

The **Map Environment** window is displayed. For example:

Use this window to define the name of the server defined for this connection, the number of the server port, and the name of the environment.

   Note: You can open multiple connection windows; each window includes the connection name in the window title.

2   Enter the security credentials to secure the connection.

📄   **Note:** The user ID used for this connection also determines whether access to the business services and methods is permitted.

If the plug-in is active, three new icons are displayed on the toolbar. These icons invoke:

- The business services repository explorer
- A wizard to create a business service (see **Create a Business Service**)
- A wizard to create a directly enabled Web service (see **Create a Directly Enabled Web Service**)

3   Select the repository explorer icon.

The **Business Services** repository explorer is displayed. For example:

4   Expand the **Business Services** node.

The **Domains** and **Configuration** nodes are displayed.

5   Expand the **Domains** node.

The server retrieves the list of business service domains for the specified connection.

6   Select a domain to view the services available within the domain.

The server retrieves the list of business services for the specified domain. For example:
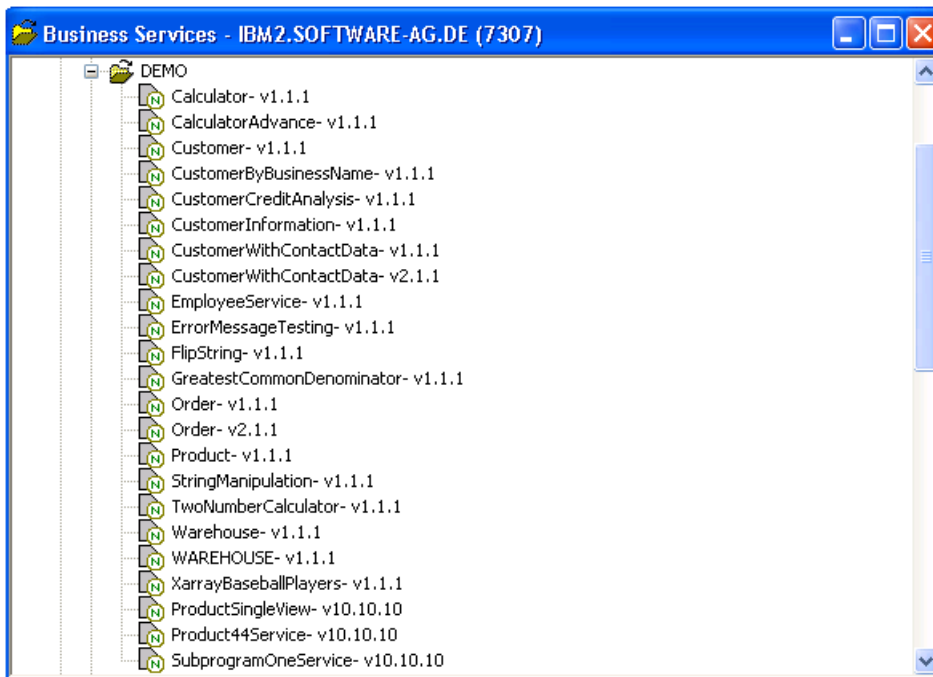
Find Business Services

You can use the repository explorer to find business services for a specific business function. For example, you can find all services in a domain.

▶ **To find all business services in a domain:**

1 Open the context menu for the **Business Services** node in the repository explorer.

2 Select **Search Services**.

The **Find Services** window is displayed. For example:
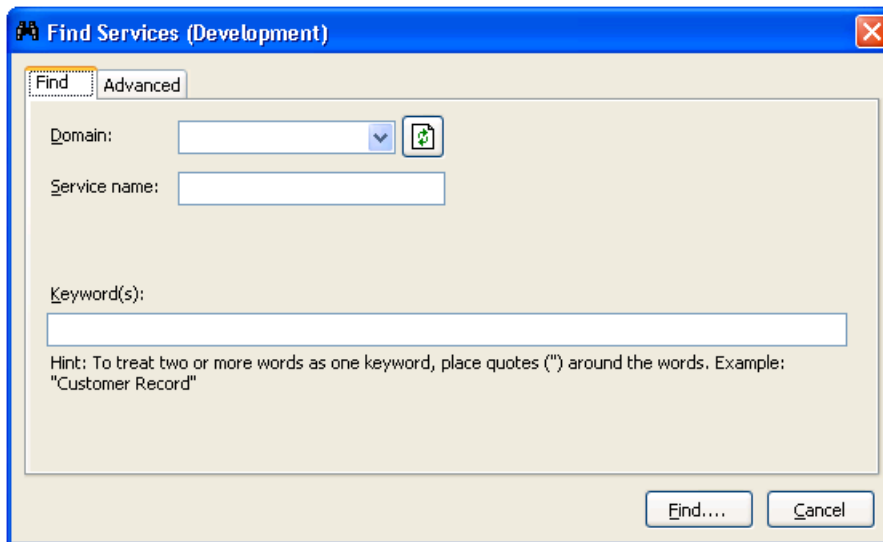
3　　Select the name of the domain in **Domain**.

4　　Select **Find**.

Natural Business Services returns a list of the business services in the specified domain.

You can refine the results by limiting the search in several ways, such as:

- Find Specific Services in All Domains
- Find Specific Services in One Domain
- Use the Advanced Tab

Find Specific Services in All Domains

▶ **To find specific business services in all domains:**

1　　Do one of the following:

- To return a list of all business services with a specific name in all domains, type the name in **Service name**.

- To return a list of all services with a specific keyword in their descriptions, type the keyword in **Keyword(s)**.

  Note: You can also use fields on the Advanced tab to limit the search. For information, see **Use the Advanced Tab**.

2   Select **Find**.

Natural Business Services returns a list of all specified business services in all domains. The following example shows the results of searching all domains for a service containing "Customer" in its description:
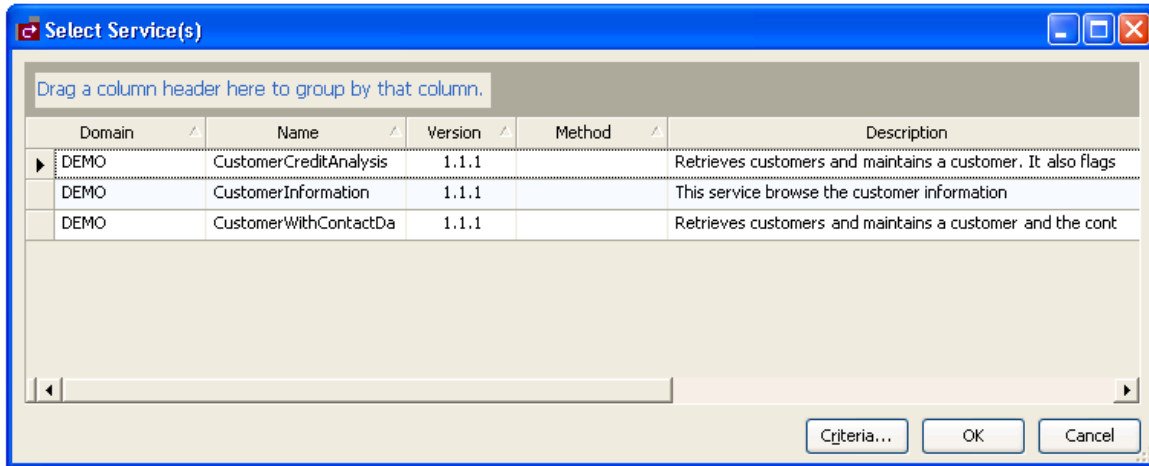
Optionally, you can also use this window to:

Task	Procedure
Group the results by column	Drag the column heading to the shaded area.
Rearrange the columns	Drag the column heading to the far left of this window (for example, to sort by version, drag the Version heading to the left of the Domain heading).

Find Specific Services in One Domain

▶ **To find specific business services in a specific domain:**

1   Select the name of the domain in **Domain**.

2   Do one of the following:

■ To return a list of all services with a specific name, type the name of the business service in **Service name**.

■ To return a list of all services with a specific keyword in their descriptions, type the keyword in **Keyword(s)**.

> **Note:** You can also use fields on the Advanced tab to limit the search. For information, see **Use the Advanced Tab**.

3   Select **Find**.

Natural Business Services returns a list of all specified business services in the specified domain.

Use the Advanced Tab

▶ **To limit the search of business services:**

1   Type one or more keywords in Keyword(s).

2   Select the Advanced tab in the **Find Services** window.

The Advanced tab is displayed. For example:
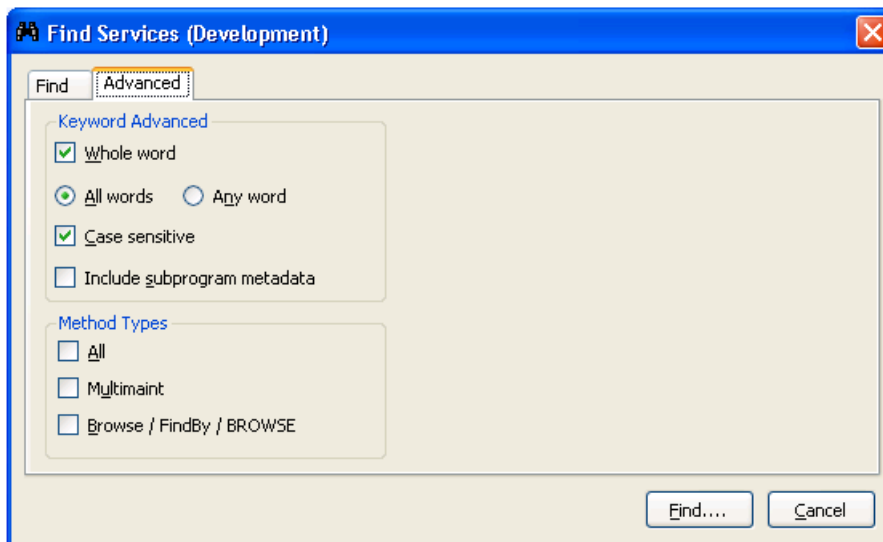
3   Select from the following search criteria:

- Search for whole words only (for example, if you type "Customers" in **Keyword(s)**, business services containing "Customer" will not be returned)

- Search for all words specified in **Keyword(s)** (for example, if you type "Customer record" in **Keyword(s)** and select All words on the Advanced tab, only business services containing both "Customer" and "record" will be returned)

▪ Search for any word specified in **Keyword(s)**(for example, if you type "Customer record" in **Keyword(s)** and select Any word on the Advanced tab, business services containing either "Customer" or "record" will be returned)

▪ Search for words using a particular case (for example, if you type "Customer" in **Keyword(s)** and select **Case sensitive**, business services containing "customer" will not be returned)

▪ Search business services and their subprogram metadata for services containing the keyword specified in **Keyword(s)**

▪ Search for business services containing one of the method types listed (for example, you can search for business services containing all method types, the MultiMaint method type, or the Browse, FindBy, or BROWSE method types)

4   Select **Find**.

Natural Business Services returns a list of all business services with the specified criteria.

Set Trace Options

To troubleshoot configuration and connection issues, you can set trace options on the wizard panels. This option records what is happening internally within the Natural plug-in and wizards.

▶ **To set trace options:**

1   Select **Trace** on a wizard panel.

The **Trace** window is displayed. For example:
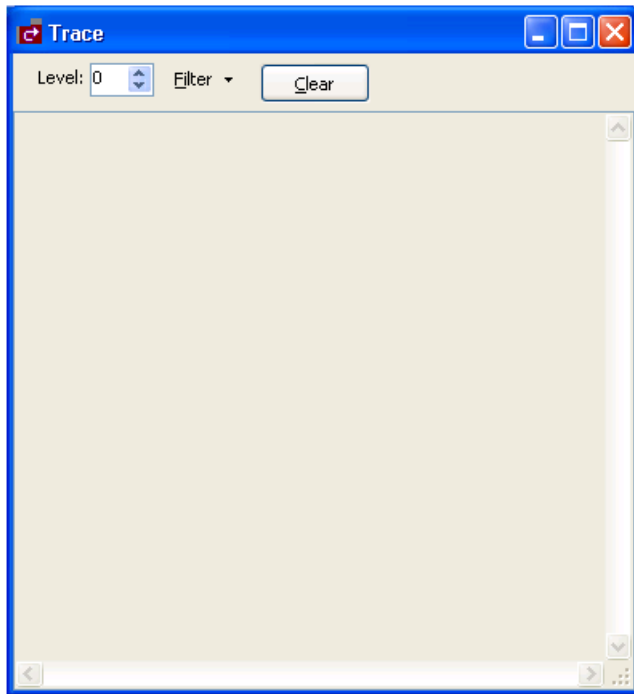
2   Select the desired trace level in **Level**.

3   Select one or more items in **Filter** to limit the items in the trace.

These items are:
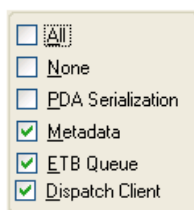
Create a Directly Enabled Web Service

You can create a single Web service from a selected Natural subprogram.

▶ **To create a directly enabled Web service:**

1   Open the context menu for **Business Services** in the repository explorer.

2   Select **Create direct Web service**.

The **Select a subprogram** panel is displayed. For example:
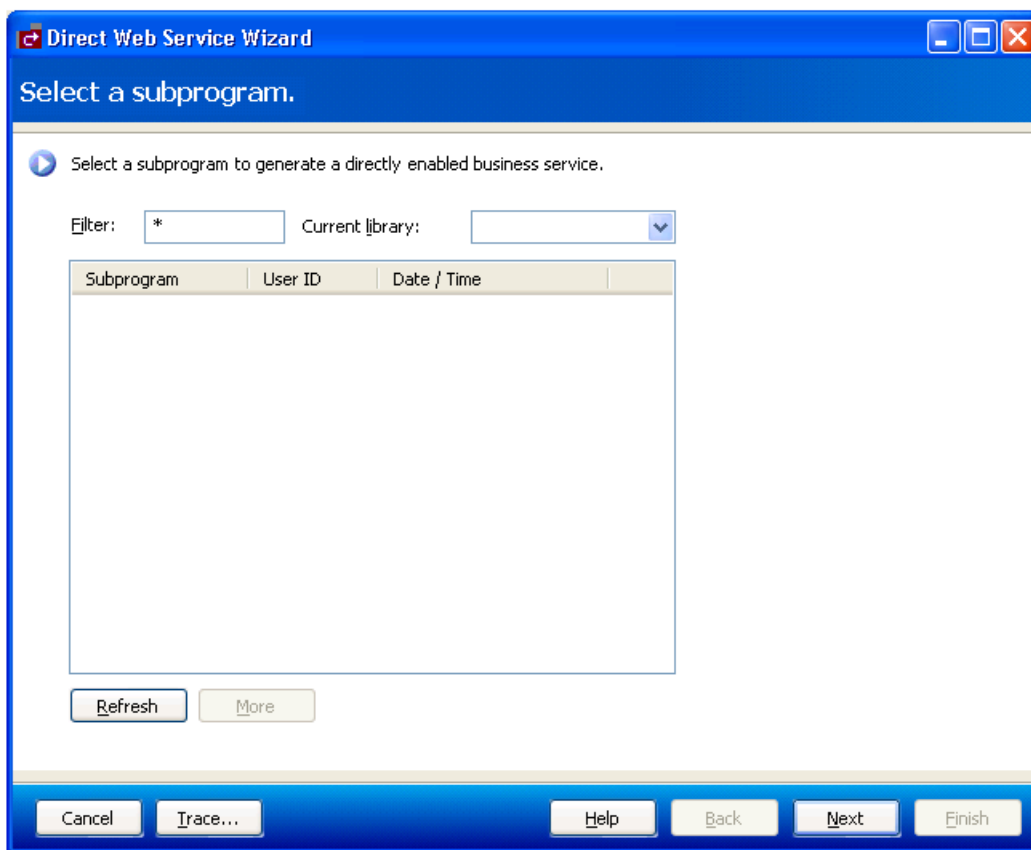
3   Select **Refresh**.

A list of subprograms in the current library is displayed. Optionally, you can:

Task	Procedure
Display additional subprograms	If all subprograms cannot be displayed in the space provided, use the scroll bar to display the additional subprograms.
Limit the list of subprograms displayed	Enter a starting value and/or wildcard characters in **Filter**. In the example above, all subprograms in C52DEMO will be displayed.
Change the name of the library	Select a different library in **Current library**.

4   Select the Natural subprogram you want to use for the service.

5   Select **Next**.

The **Configure client settings** panel is displayed. For example:

Use this panel to provide details about the Web service and server. By default, the following options are marked:

- Copy web application support files

- Show WSDL in browser after generation

- Use ASP.net development server

6   Enter the following information for the Web service:

Option	Description
Service name	Name of the Web service.
Domain	Name of the domain for the Web service.
Version	Number for this version of the Web service. For example, enter "1 1 1" for the first version.
Service description	Brief description of the Web service.
Port	Number of the port that hosts the web server.
Physical folder	Physical location of the web server.
Virtual root	Virtual root for the web server.

7   Select **Finish** to generate the Web service.

Check Environment Information for a Connection

▶ **To display environment information for a connection:**

1   Select **Business Services > Check Environment** on the **Tools** menu.

The **Environment Check** window is displayed. For example:
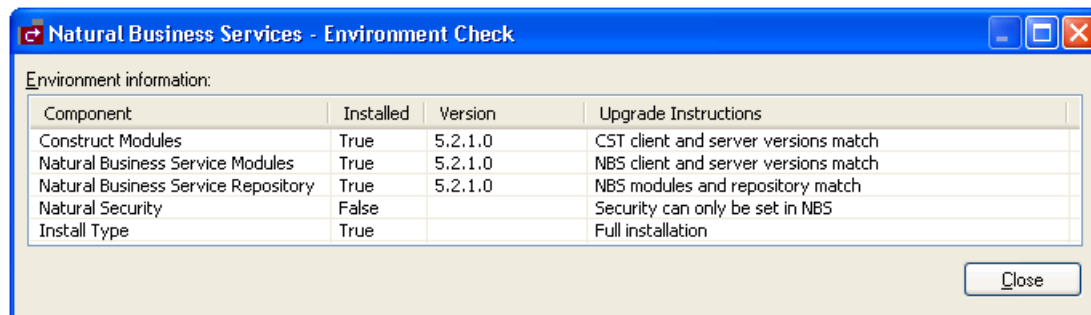
This window displays information about the environment for the current connection.

2   Select **Close**.

4 Developing Business Services

This section describes how to create and maintain business services. The following topics are covered:

Create a Business Service

A business service consists of a collection of methods related to a common business entity. Use the Business Service wizard to create and maintain a business service.

During generation, the Business Service wizard creates a subprogram proxy to translate data (including Unicode formats) and then adds an entry in the **Business Service** repository. It also generates additional subprograms as required.

▶ **To create a new business service:**

1   Select **Business Services > Business Service Wizard** on the **Tools** menu.

The **Create a Business Service** panel is displayed. For example:
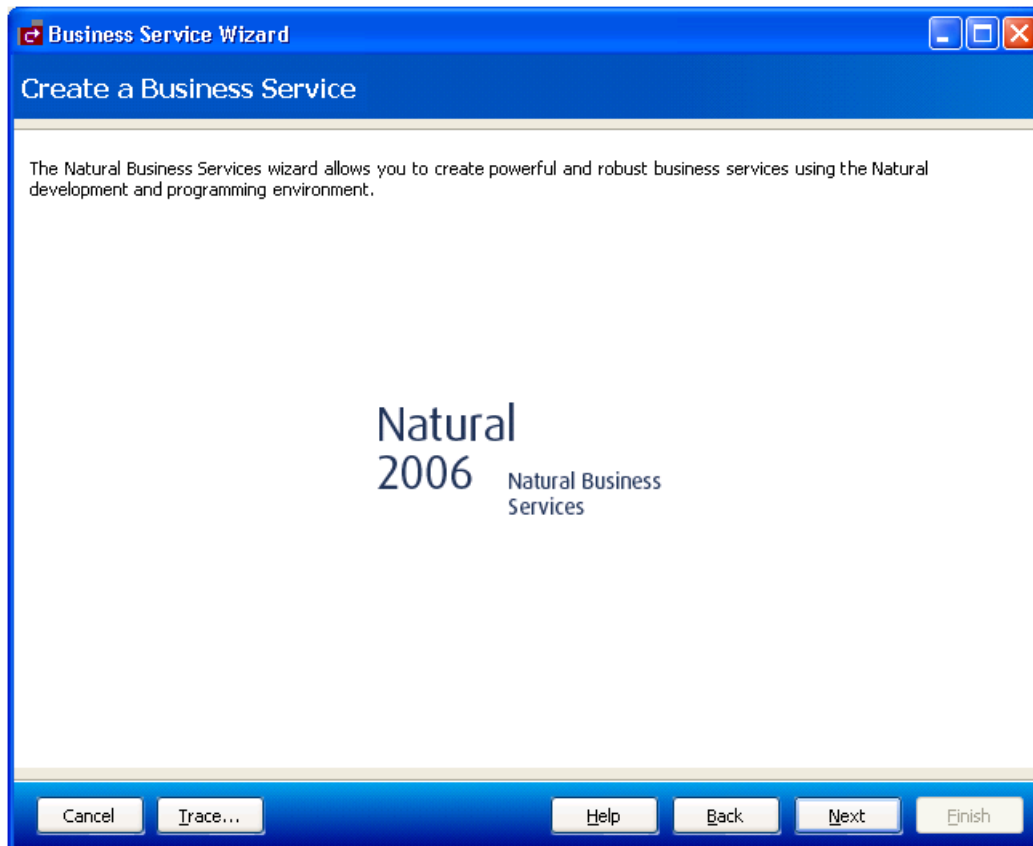
2   Select **Next**.

The **Choose Type** panel for the Business Service wizard is displayed:
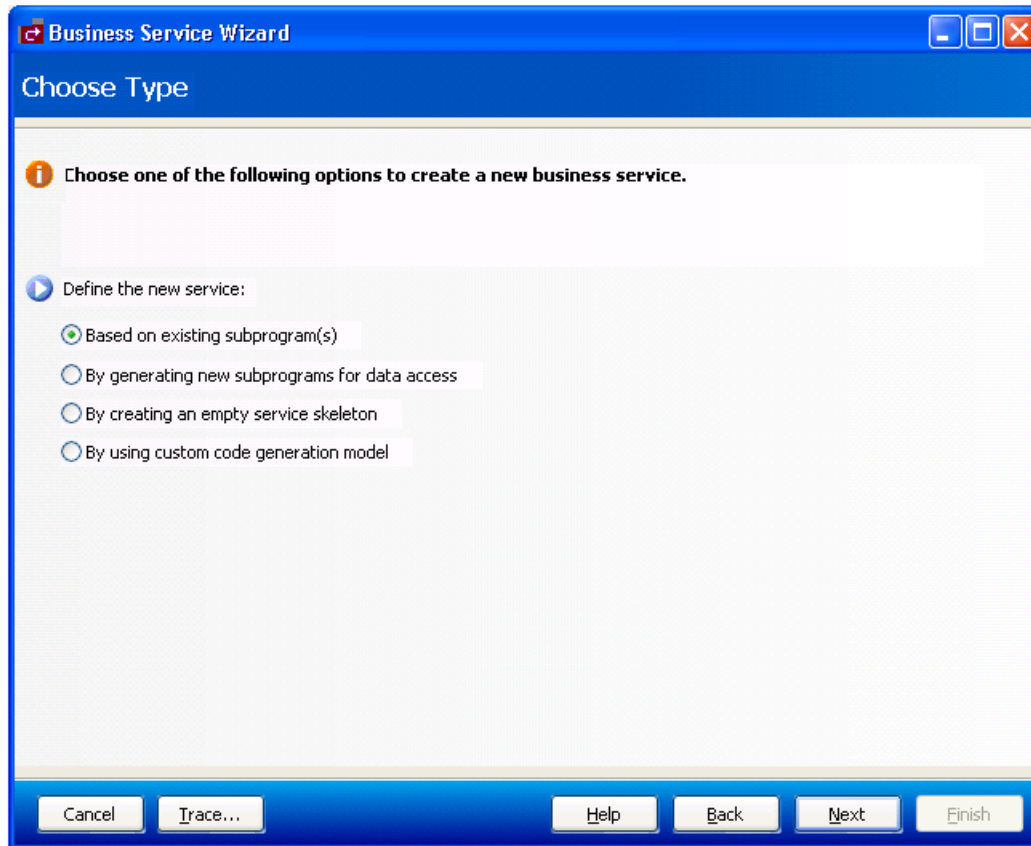
3   Select one of the following options:

- **Based on existing subprograms**
- **By generating new subprograms for data access**
- **By creating an empty service skeleton**
- **By using custom code generation model**

Depending on which option you select, one or more additional panels are displayed. After filling in the appropriate information, select **Next** on the last specification panel. The **Enter Service Name and Select Domain** panel is displayed. For example:
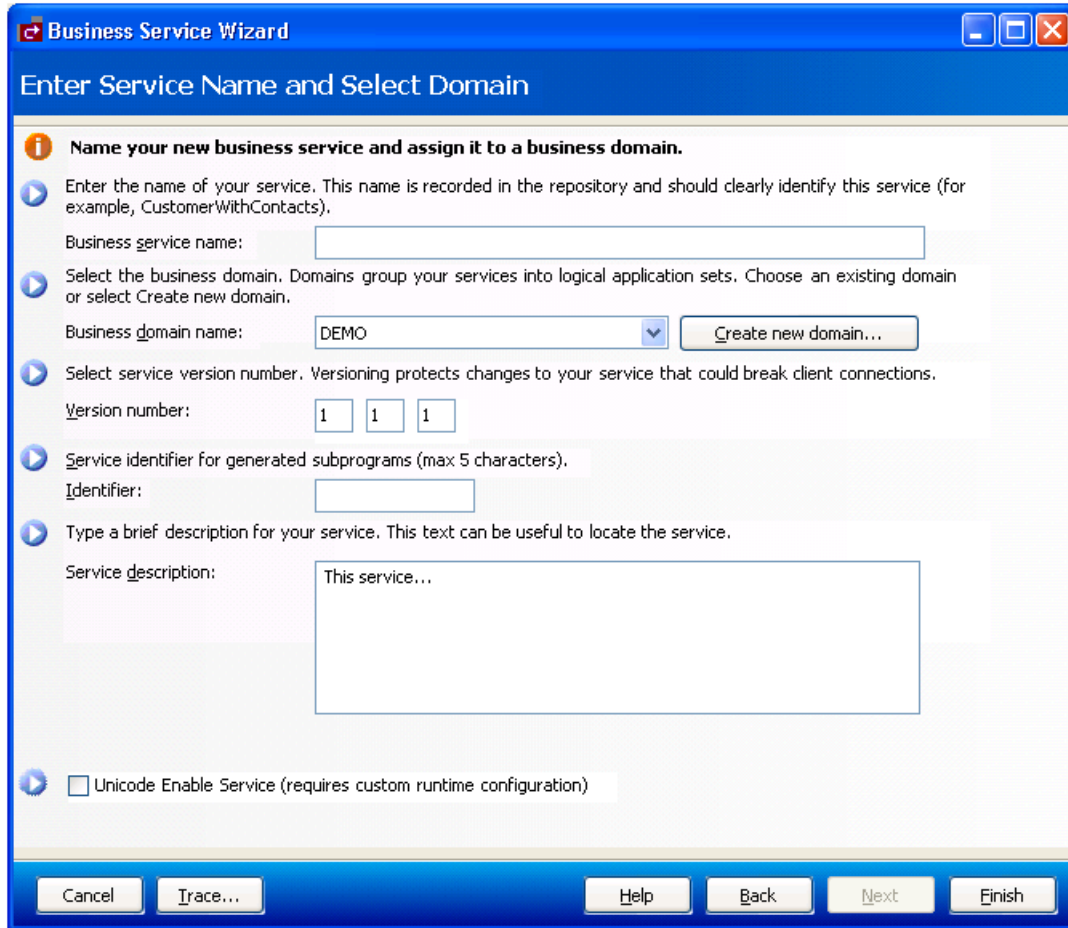
> Note: This panel contains an additional field (called Create FindBy* methods) when invoked from the program editor for an object browse subprogram. For information, see **Create FindBy* Methods**.

Use this panel to name your business service and assign the service to a domain. Optionally, you can:

Task	Procedure
Create a new domain	Select **Create new domain**. For information, see **Create a New Domain**.
Change the version number for the service	Type a new version number in **Version number**. Version numbers help protect changes to the service that can disrupt client communications. Use version 1.1.1 (default) for a new service.
Create the service interface using Unicode	Select **Unicode enable service**. Select this option if your service has Unicode fields (U data type). Services that use a Unicode dispatcher require a special EntireX runtime server definition. For information, see Use Unicode Parameters for Your Business Service.

4   Type the name of your business service.

This name should clearly identify the service (for example, CustomerWithContacts).

5   Select the business domain in **Business domain name**.

A domain groups business services into logical application sets. If a domain does not currently exist, select **Create new domain** to create one. For information, see **Create a New Domain**.

6   Verify the service version number.

7   Type the service identifier for generated subprograms in **Identifier**.

The service identifier can be up to five characters in length and will be used with a wizard-generated prefix and suffix to identify the generated subprograms used for this service.

8   Type a brief description of your service in **Service description**.

9   Select **Finish** to generate the business service.

The **Generation Status** window is displayed, showing the progress of the generation. When generation is completed, the window shows the names of the files generated for the business service.

The Results column indicates that the generated files have been generated for the first time (New).

> **Note:** To return to the wizard without saving the files, select **Cancel**.

10   Select **Save** to save the files.

You have successfully generated and saved a new business service. For information on testing the business service, see **Test a Business Service**.

Based on Existing Subprogram(s)

If you want to create a business service based on existing subprograms, you have various options:

- Directly enable a subprogram

  This option associates the DEFAULT method with an existing subprogram by directly enabling the subprogram. For information, see **Directly Enable Subprograms**.

  > **Note:** This option is only available when one subprogram is listed in **Selected Subprogram(s)** and the option applies to that subprogram alone.

- Categorize parameters

  If the subprogram(s) was generated by the Object-Browse or Object-Maint models, this option creates a business service that has associated methods. For information, see **Categorize Parameters**.

- Use traditional defaults and FindBy methods

  If the subprogram(s) was generated by Natural Construct, these options let you take advantage of the Business Service wizard's knowledge of Natural Construct models and allow it to generate default methods for your business service. For information, see **Select the Type of Methods Generated**.

When you select **Based on existing subprogram(s)** on the **Choose Type** panel, the **Select Subprograms** panel is displayed. For example:
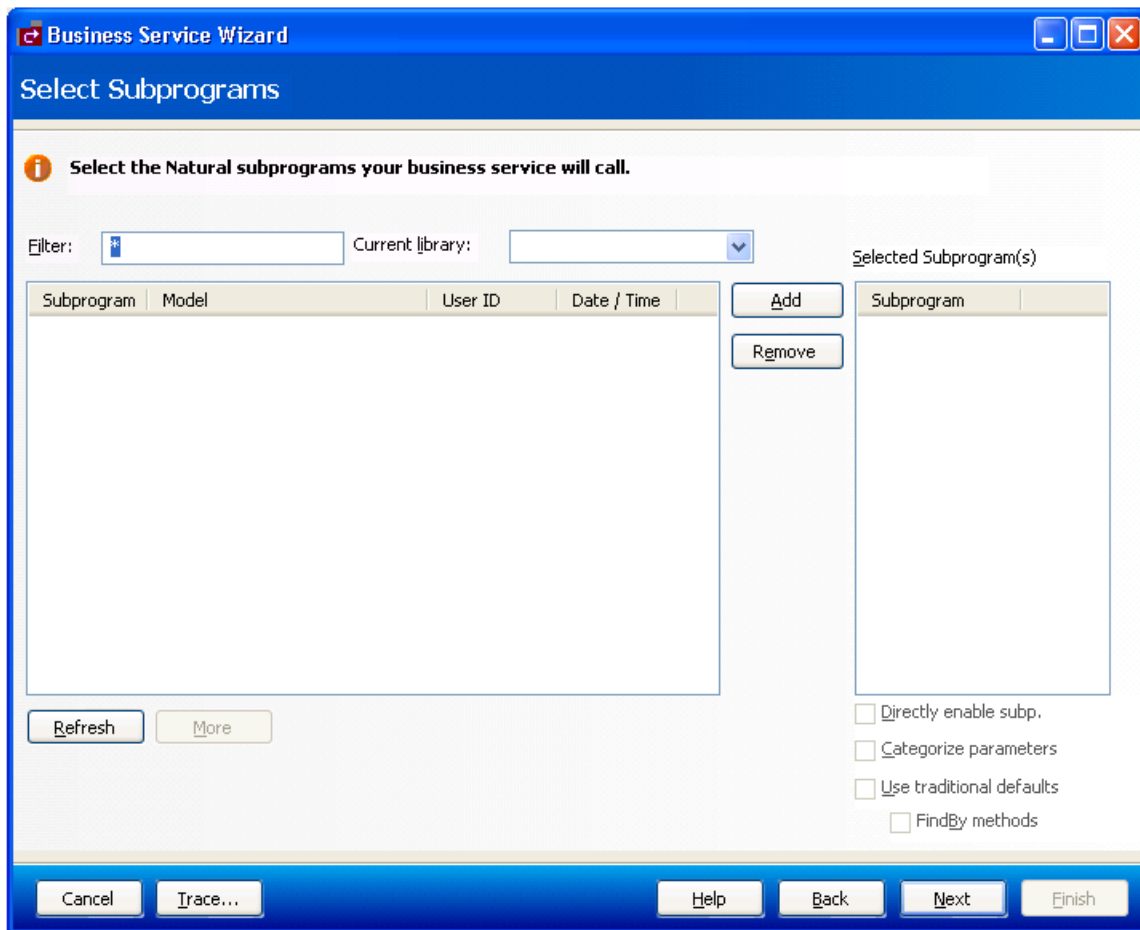
If a subprogram was not generated by Natural Construct, the Business Service wizard allows you to determine what the interface will look like (i.e., what data the service will expose to users) and which methods it will use. Use this panel to indicate which Natural subprograms the business service will call.

> **Note:** The list of available libraries is retrieved from either the FUSER or FNAT file. The wizard determines which libraries to display based on the library that is currently selected (i.e., the library in which the modules will be created). If the current library is an FNAT library, all libraries in the FNAT are available. If it is an FUSER library, all libraries in the FUSER are available.

1. Select **Refresh**.

   A list of available subprograms in the current library is displayed.

   > **Note:** The list of available libraries is retrieved from either the FUSER or FNAT file. The wizard determines which libraries to display based on the library that is currently selected in Natural for Windows (i.e., the library in which the modules will be created). If the

current library is an FNAT library, all libraries in the FNAT are available. If it is an FUSER library, all libraries in the FUSER are available.

- If all subprograms cannot be displayed in the space provided, select **More** to display the additional subprograms.

- To limit the list, enter a starting value and/or wildcard characters in **Filter**. In the example above, all subprograms in the current library will be displayed.

- If desired, select a different library in **Current library**.

2. Select the subprogram you want to include.

3. Select **Add**.

Each selected subprogram is listed in **Selected subprogram(s)**. To remove a subprogram from this list, select the subprogram and select **Remove**.

By default, the Business Service wizard will:

- Generate a subprogram proxy

- Populate the **Business Service** repository

- Generate a subprogram to call one or more existing subprograms

This occurs when the **Use traditional defaults** and **FindBy methods** options are NOT selected on the **Select Subprograms** panel. The wizard uses the Object-Generic-Subp model to generate the new subprogram. You can choose which methods the service will use and which combination of subprograms will be required to implement these methods. You can define customized methods for the business service, add user exit code, and choose and categorize attributes to be exposed to the business service user. Categorizing the attributes helps the business service user identify input, input/output, output, or state parameter structures.

> **Note:** State is similar to input/output, except that it identifies parameters that are not exposed to the business service user and are only required to maintain the state of the business object.

4. Select **Next**.

The **Define Methods** panel is displayed. For example:
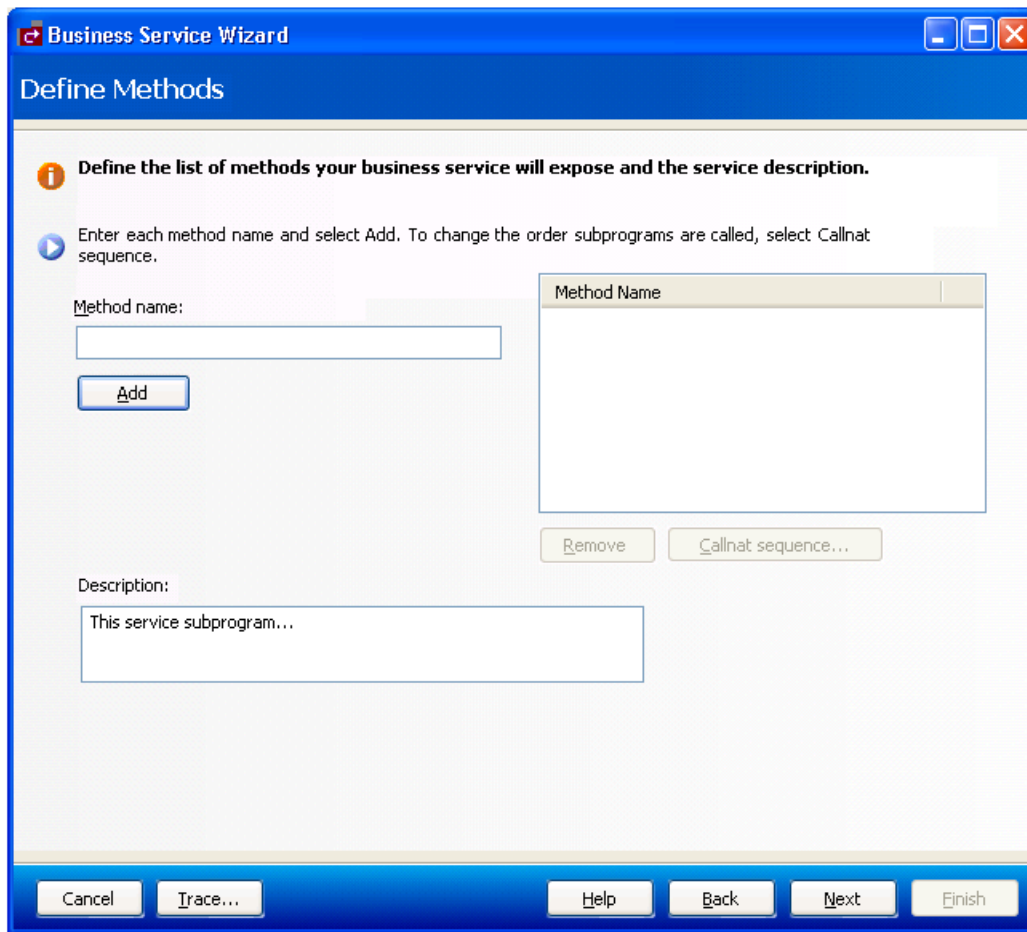
Use this panel to define the methods the business service will expose to the consumer and provide a brief description of the subprograms. Each time you enter a method name and select **Add**, the method is listed in **Method Name**.

You can change the functionality of each method by changing the Callnat sequence. For information, see **Change the Method Callnat Sequence**.

5. Select **Next**.

The last specification panel for the Business Service wizard is displayed. For a description of this panel, see **Enter Service Name and Select Domain**.

Directly Enable Subprograms

Whether a subprogram is directly enabled or not, the Business Service wizard:

- Generates a subprogram proxy
- Populates the service repository on the server

When a subprogram is directly enabled, the wizard does not generate a "wrapper" subprogram to call the specified subprograms. It does, however, create one method called `DEFAULT`. This method searches the subprogram from top to bottom and exposes each attribute in the subprogram's PDA when the subprogram is executed.

▶ **To directly enable a subprogram:**

1   Select a subprogram listed on the **Select Subprograms** panel.

2   Select **Add**.

     The subprogram is listed in **Selected Subprogram(s)**.

3   Select **Directly enable subp**.

4   Select **Next**.

     The **Enter Service Name and Select Domain** panel is displayed.

💡   **Tip:** You can also directly enable a subprogram from the program editor view by opening the context menu for the subprogram and selecting **Create Service**. The **Enter Service Name and Select Domain** panel is displayed.

📄   **Notes:**

1. This option is only available when there is one subprogram listed in **Selected Subprogram(s)** and the option applies to that subprogram alone.

2. Once the business service has been created, you can only change the method names; the basic functionality cannot be changed without modifying the existing subprogram or the input parameters for the subprogram.

Categorize Parameters

You can categorize parameters for your business service. Categorizing parameters allows users to easily identify input, input/output, state, and output requirements. This functionality allows the business service developer to understand the Natural subprogram without having to know Natural.

> **Notes:**

1. To reduce network traffic, we recommend that only user-required fields be exposed in the interface.

2. By default, the categorization is turned off and the parameters for existing subprograms are used at the same level for the service interface. Categorizing parameters moves these parameters under new level 1 structures for input, output, input/output, and state interface styles.

3. Whenever two subprograms interact, the exposed interface must be defined carefully. For information, see **Interface Considerations**.

When categorizing parameters, you must ensure that parameters containing the same name (but different values) are not placed in the same category. If this happens, the subprogram will generate but not compile and you will have to decide how to handle the duplicate names.

When two subprograms contain level 1 structures with the same field or variable name, they must have different interface styles (for example, one can be input/output and the other can be input only). For example:

- If the data for both parameters is the same, only one parameter must be exposed
- If the data is different, the parameters must have unique names so users can easily differentiate between the two

> **Note:** Parameters are not categorized by default, but this option is recommended for experienced users.

If you select **Categorize parameters** on the **Select Subprograms** panel, the **Define Interface** panel is displayed when you select **Next**. For example:
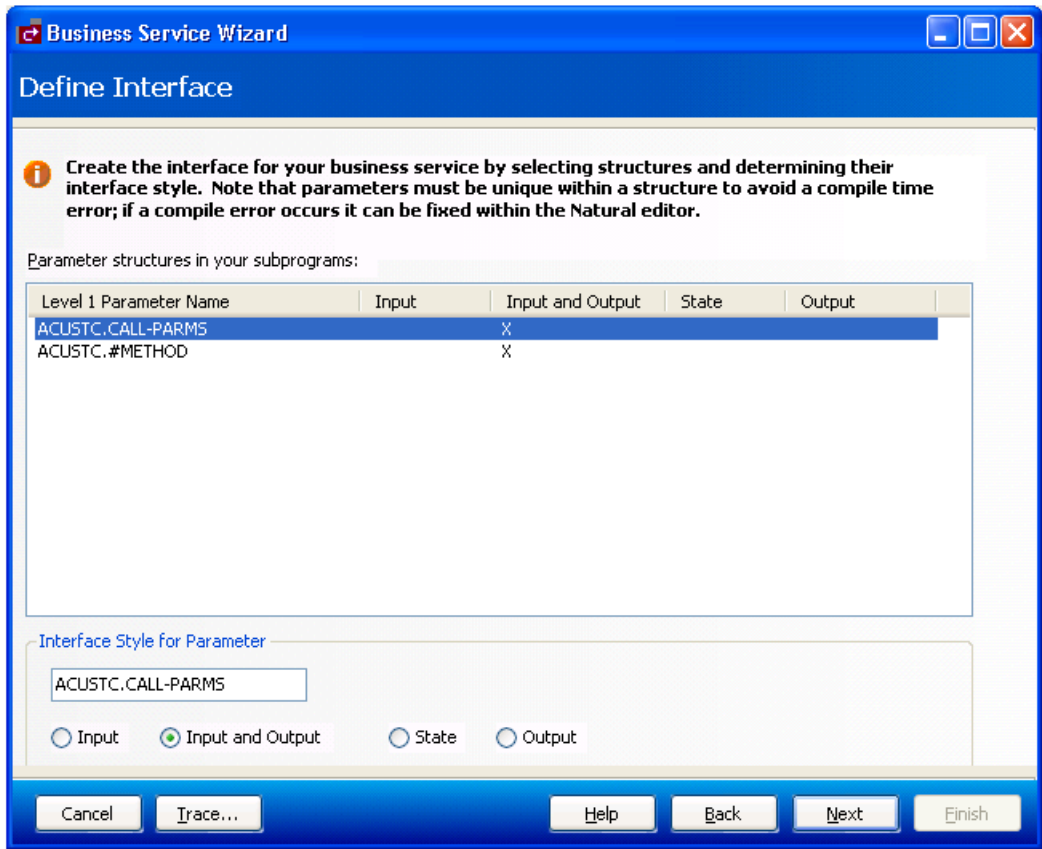
This panel lists each level 1 structure in the selected subprograms. By default, the parameters use the input and output interface style. To change the interface style for each parameter listed, select one of the following options:

■ Input

Data is moved from the exposed interface to the internal business service variables. These exposed fields will not be changed, even if the internal server fields change.

■ State

There is no coding difference between input and output and state interfaces styles. These structures just make it easier to identify which fields should be exposed to the user and which are only required to maintain a state with the server.

■ Output

Output parameters are reset. Even if the client copies data to the output structure, the data will be erased. Only data the server puts in these fields is sent back to the client.

After categorizing the parameters, select **Next** to display the **Define Methods** panel.

Interface Considerations

The following table lists several cases you can consider when defining the interface for your business service:

Case	Same Parameter Name	Example	Same Data	Example
1	No	Name, Make	No	Smith, Toyota
2	Yes	Personnel-ID, Personnel-ID	Yes	1111, 1111
3	No	#first, #num	Yes	4, 4
4	Yes	Name, Name	No	Smith, D&D Company

Case 1

This case is simple and can be handled by the wizard.

Case 2

This case is relatively simple. Consider the following example:

```
01 #BIZ-INPUTS
   02 VEH
       03 PERSONNEL-ID (A8)
       03 MAKE   (A30)
       03 MODEL (A30)
   02 EMP
       03 PERSONNEL-ID (A8)
       03 NAME
```

▶ **To solve this problem, do one of the following:**

■   Put one of the parameters under #BIZ-INPUTS and the other under #BIZ-INPUTS-OUTPUTS (although this will expose the same attribute and data twice, creating some confusion for the business service user)

Or:

Define the parameters as follows:

```
01 #BIZ-INPUTS
    02 VEH
*       03 PERSONNEL-ID (A8)
        03 MAKE   (A30)
        03 MODEL (A30)
    02 EMP
*       03 PERSONNEL-ID (A8)
        03 NAME
    02 PERSONNEL-ID (A8)
```

> **Note:** This solution will work as long as PERSONNEL-ID is not part of a redefined field and reserving a position in memory.

Case 3

In this case, you must decide which parameter should be exposed. Once this decision is made, you must ensure that the correct data is moved into the other parameter using the MOVE-TO and MOVE-BACK user exits. For example, if #NUM is exposed, add the following code to the MOVE-TO exit:

```
#FIRST := #NUM
```

Add the following code to the MOVE-BACK exit:

```
#NUM := #FIRST
```

Case 4

In this case, you must decide what "name" means and clarify the term for the business service user. Consider the following parameters:

```
01 #BIZ-INPUTS
  02 EMP
     03 NAME (A30)
     03 PHONE (N10)
  02 BUS
     03 NAME (A50)
     03 ADDRESS (A100/5)
```

For example, you can change the names to OWNER-NAME and BUSINESS-NAME:

```
01 #BIZ-INPUTS
  02 EMP
*     03 NAME (A30)
     03 OWNER-NAME (A30)
     03 PHONE (N10)
  02 BUS
*     03 NAME (A50)
     03 BUSINESS-NAME (A30)
     03 ADDRESS (A100/5)
```

If the parameter data area changes, you must define the names in the MOVE-TO and MOVE-BACK user exits as follows:

Add the following code to the MOVE-TO exit:

```
EMP.NAME := OWNER-NAME
BUS.NAME := BUSINESS-NAME
```

Add the following code to the MOVE-BACK exit:

```
OWNER-NAME := EMP.NAME
BUSINESS-NAME := BUS.NAME
```

⚠️ **Important:** If you change the business service interface (parameters), you must regenerate the subprogram proxy (or proxies). For information, see **Regenerate a Business Service Proxy**.

Modify a Subprogram That is Not Directly Enabled

If you do not select the **Directly enable a subprogram** option, the Business Service wizard generates another subprogram between the existing business service subprogram(s) and the proxy. This intermediate subprogram can contain multiple, named methods that call one or more subprograms.

📄 **Note:** For an example of the intermediate subprogram, refer to BNUM in the demo application.

When the Categorize Parameters option is selected (for information, see **Categorize Parameters**), the parameter data areas (PDAs) are generated into the PARAMETER-DATA user exit. This allows the programmer to decide which parameters to expose in the client code.

Generating the PDAs into a user exit creates a problem, however, if the subprogram being called has been changed. These changes will not be picked up. To solve this problem, use the Regeneration wizard. This wizard adds comment indicators to the existing PDA code and creates a "fresh" PDA. Unfortunately, this solution does not re-incorporate any manual changes. The programmer must re-evaluate the PARAMETER-DATA user exit to determine which portion of the old and "fresh" code to keep.

For example, if you regenerated the calculatorAdvance service in the demo application, BNUM appears as follows:

```
**SAG DEFINE EXIT PARAMETER-DATA
/*   01 #BIZ-INPUT-OUTPUTS
/*     02 E1-INPUT-DATA
/**      03 #FUNCTION (A30)
/*       03 #FIRST-NUM (N5.2)
/**      03 REDEFINE #FIRST-NUM
/**        04 #OPERAND-1 (I4)
/*       03 #SECOND-NUM (N5.2)
/**      03 REDEFINE #SECOND-NUM
/**        04 #OPERAND-2 (I4)
/*       03 #SUCCESS-CRITERIA (N5)
/**    02 E1-GCD-DATA
/**      03 #OPERAND-1 (I4)
/**      03 #OPERAND-2 (I4)
/**      03 #RESULT (I4) /* result goes into #GCD
/*   01 #BIZ-OUTPUTS
/*     02 E1-OUTPUT-DATA
```

```
/*      03 #RESULT (N11.2)
/*        /* Because result is used in both subprograms and because
/*        /* some methods will expose both the calculator result
/*        /* and the Greatest Common Denominator, a new exposed field
/*        /* has been created
/*      03 #GCD (I4)
/*      03 #TIME (T)
/*      03 #SUCCESS (L)
/*      03 #ERROR-MESSAGE (A79)
* Note: This EXIT creates MOVE-TO and MOVE-BACK exits.
*       To regenerate, delete all 3 exits
*
  01 #BIZ-INPUT-OUTPUTS
    02 E1-INPUT-DATA
      03 #FUNCTION (A30)
      03 #FIRST-NUM (N5.2)
      03 #SECOND-NUM (N5.2)
      03 #SUCCESS-CRITERIA (N5)
    02 E1-GCD-DATA
      03 #OPERAND-1 (I4)
      03 #OPERAND-2 (I4)
      03 #RESULT (I4)
  01 #BIZ-OUTPUTS
    02 E1-OUTPUT-DATA
      03 #RESULT (N11.2)
      03 #TIME (T)
      03 #SUCCESS (L)
```

In this example, the code added by the programmer before the regeneration has been commented out and may need to be re-incorporated.

Select the Type of Methods Generated

For Natural Construct-generated subprograms, you can use the **Select Subprograms** panel to select the type of methods generated for your business service. After selecting and adding one or more subprograms, the following options are available for the subprogram(s) listed in **Selected Subprogram(s)**:

- If an object browse OR object maintenance subprogram is listed, the **Use traditional defaults** field is selected and the wizard will:

  - Generate a subprogram proxy

  - Populate the repository with the default methods associated with either the Object-Browse or Object-Maint models

- If an object browse AND an object maintenance subprogram are listed, and they access the same file, the **Use traditional defaults** field is selected and the wizard will:

  - Generate a subprogram proxy for each subprogram

■ Populate the repository with the default methods associated with both the Object-Browse and Object-Maint models

■ If a single object browse subprogram is listed, or if an object browse and an object maintenance subprogram are listed that access the same file and the file has no intra-object relationships (i.e., relationships with other files that are maintained at the same time as the primary file), the **Use traditional defaults** and **FindBy methods** fields are selected and the wizard will:

■ Generate an object browse select subprogram and subprogram proxy

■ Populate the repository with the FindBy methods associated with the Object-Browse-Select-Subp model (and the default methods associated with the Object-Maint models, if an object maintenance subprogram is also selected)

For more information on the Object series of models, see Natural Construct Object Models.

Define the Methods

As with the interface considerations, decisions must also be made as to which subprograms are executed for each method and what order they are executed. In addition, data may need to be massaged before the subprograms are executed for the method to work effectively and accurately. For a better understanding of this, refer to the BNUM and BSTRING subprograms in the SYSBIZDE library.
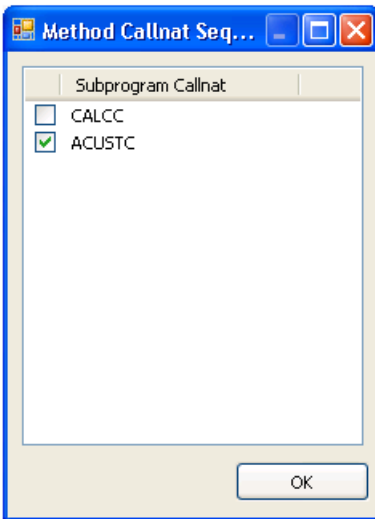
Change the Method Callnat Sequence

By default, the subprograms are executed in the order they were selected on the **Select Subprograms** panel.

▶ **To change the method Callnat sequence:**

1   Select a method in the **Method Name** column on the **Define Methods** panel.

2   Select **Callnat sequence**.

The **Method Callnat Sequence** window is displayed. For example:

This window displays the name of each subprogram that is executed for the selected method and in what order it is executed. Use this window to change the order of subprograms.

3   Select **OK** to save your changes.

In addition to these changes, you can further customize the functionality of methods within user exits generated into the Natural code by the Business Service wizard. The subprogram created by the wizard is called BserviceIdentifier and is located in the library containing Natural for Windows when the wizard is invoked. If you make any changes to the exposed interface in this subprogram (i.e., changes to the PDAs), you must regenerate the service proxy. For information, see **Regenerate a Business Service Proxy**.

By Generating New Subprograms for Data Access

This option will create the minimum components for a business service (a subprogram proxy and an entry in the **Business Service** repository), as well as at least one Natural Construct object subprogram. Depending on which data access type is selected on the **Choose Data Access Type** panel, one of the following will be generated:

■ An object maintenance subprogram

■ An object browse subprogram

■ An object browse-select subprogram
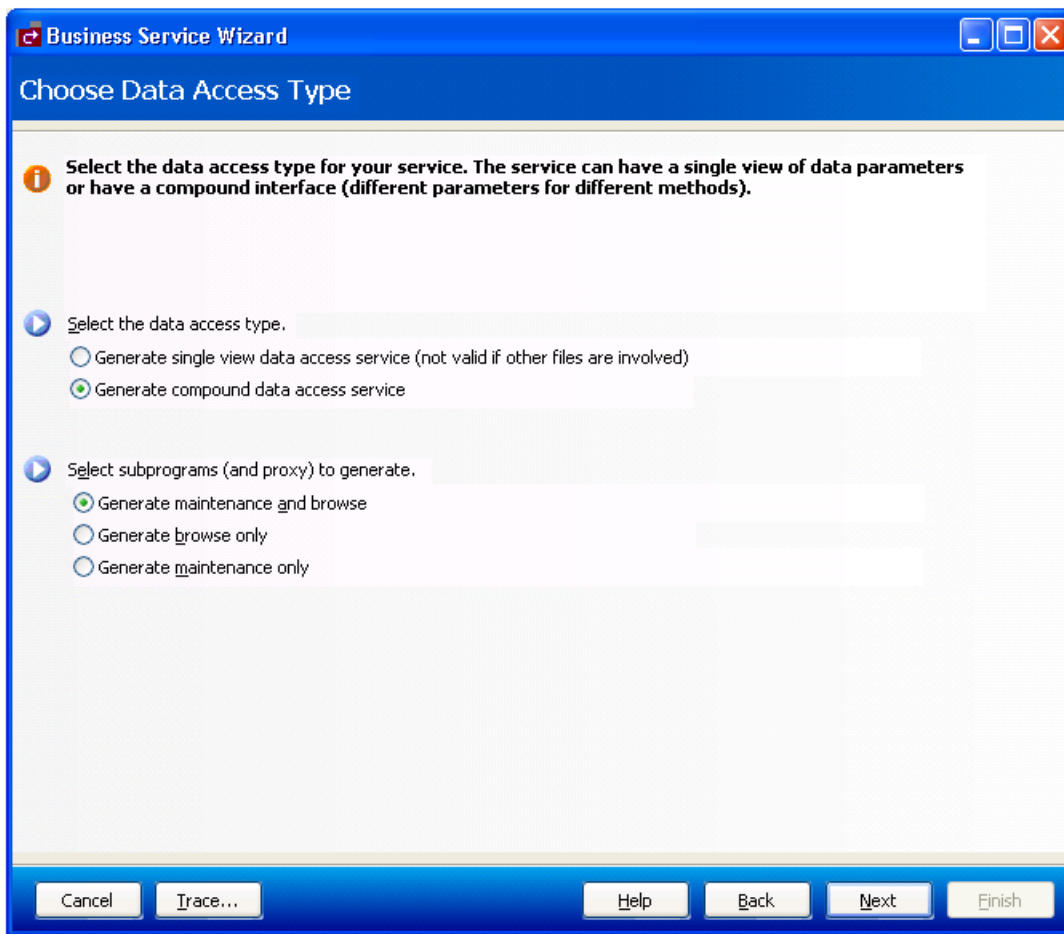
■ A combination of these subprograms

If you select **By generating new subprograms for data access** on the **Choose Type** panel, the **Data Parameters** panel is displayed. For example:
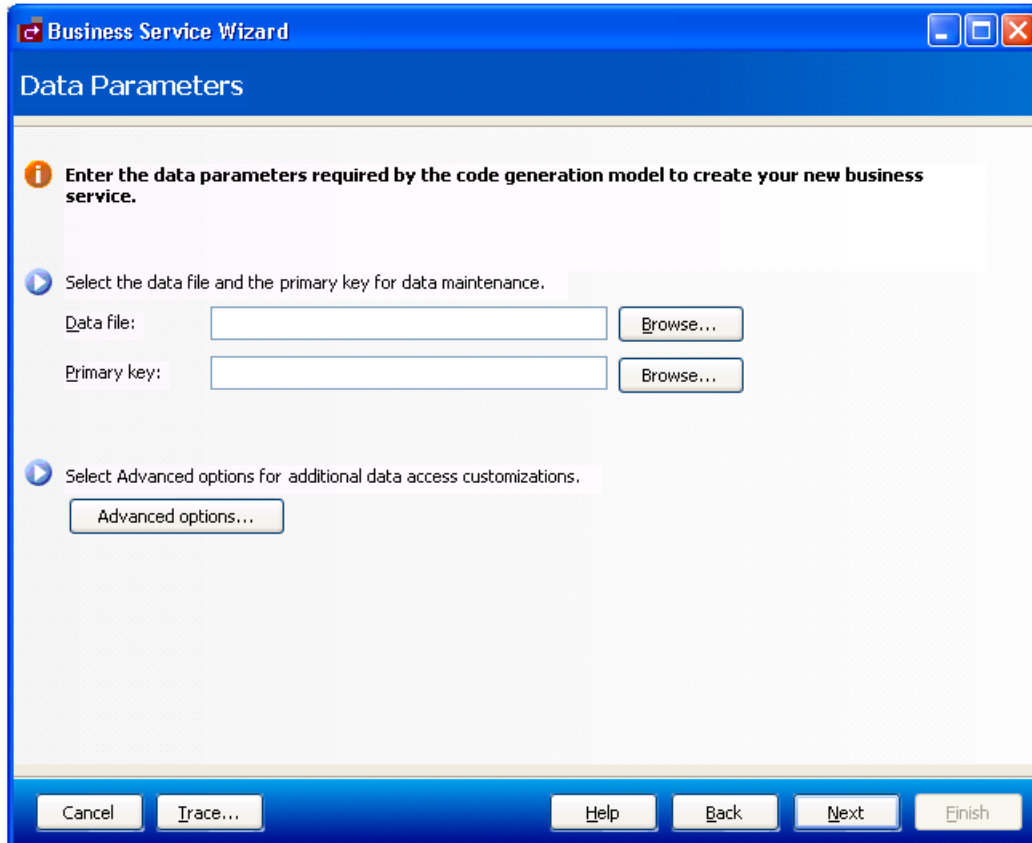
Use this panel to indicate the name of the data file and the primary key used for data maintenance. To select advanced options for additional data access customizations, select **Advanced options**. For information, see **Specify Advanced Options for Data Access**.

1. Select the name of the file used for data maintenance in **Data file**.

   This file must currently exist in Predict.

2. Select the primary key for the specified file in **Primary key**.

3. Select **Next**.

The **Choose Data Access Type** panel is displayed:

4. Select one of the following data access types:

Data Access Type	Description
Generate single view data access service	Generates an object browse, object maintenance, and object browse-select subprogram and the business service will have the following methods: Delete, MultiMaint, Store, Update, FindBy (one or more, such as the FindByDomainName method), and, optionally, Count (one or more, such as the ServiceCountByDomain method). This access type does not work with files that have intra-object relationships (for example, the Order header has an intra-object relationship with Order lines). But if only one physical file is involved, this type only requires one user interface to browse and maintain data. It is also designed for network efficiency, which means n rows of data can be processed at a time for browse or data maintenance activities.

Data Access Type	Description
Generate compound data access service	Generates an object browse and object maintenance subprogram and the business service will have the following methods: BROWSE, DELETE, FORMER, EXISTS, GET , INITIALIZE, NEXT, STORE, UPDATE. In addition, two subprogram proxies are created: one for the BROWSE method and one for all other methods. This access type handles complex data structures with intra-object relationships that must be maintained. It assumes that the data browse subprogram has a different interface than the data maintenance subprogram. A high-level browse interface can be exposed with multiple rows, but when an object must be maintained, all details can be exposed. For example, the maintenance subprogram can display all fields for Order and the browse subprogram can display the Order header for n rows. **Note:** The wizard assumes 20 rows, but if the rows are very large, the wizard will lower the number of rows until it reaches a reasonable message size.

For more information, see **More About the Object Browse-Select Subprogram**.

5. Select which subprograms (and proxies) to generate.

- Generate maintenance and browse (the default)

- Generate browse only

- Generate maintenance only

6. Select **Next**.

The last specification panel for the Business Service wizard is displayed. For a description of this panel, see **Enter Service Name and Select Domain**.
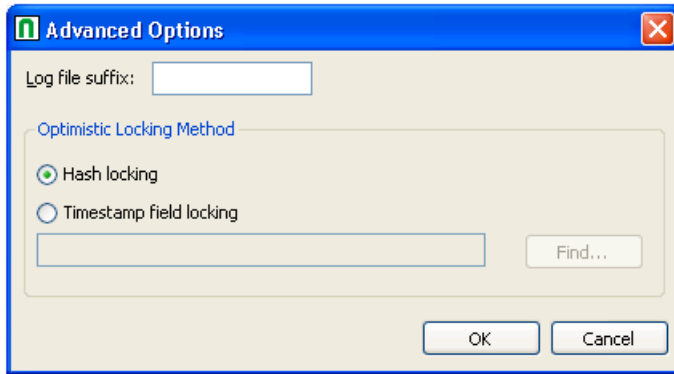
Specify Advanced Options for Data Access

This option is available if an object maintenance subprogram will be generated.

▶ **To specify advanced options for data access:**

1   Select **Advanced options** on the **Data Parameters** panel.

The **Advanced Options** window is displayed. For example:

> **Note:** To locate subprograms to use with your business service, select **Find**.

2   Type a suffix in **Log file suffix**.

The suffix identifies the log files for this business service. This option allows the wizard to generate code to maintain a log file whenever data is modified through this business service.

3   Select the record locking option.

The object maintenance subprogram has two methods to lock records: hash locking and timestamp field locking. The traditional method is using a timestamp. This method works well if the file is always maintained by Natural Construct-generated objects. If not, data may have changed and the timestamp field may not have been updated. The hash locking method checks all data to ensure that nothing has been changed between when the user saw the data and when the database locks the data.

4   Select **OK**.

More About the Object Browse-Select Subprogram

An object browse-select subprogram can:

- Determine the key fields for the object browse subprogram and separate them into different methods (for example, the various FindBy and Count methods)

- Allow a row-state attribute on each row to process methods at the row level

- Reduce network traffic by executing both the object browse and object maintenance subprograms from the same object browse-select subprogram

For example, if the row-state attribute determines that rows 3, 5, and 10 in the object browse subprogram must be modified, and the modified values are sent back to the server with all the rows, three calls to the object maintenance subprogram can be processed without going back to the client.

■ Be modified through user exit code.

You can write Natural code to massage the data and/or call other Natural subprograms.

■ Expose the data as a dataset

This allows the Natural Business Services .NET plug-in to take advantage of dataset processing and handle row processing beyond the *n* rows defined for the object browse-select subprogram. For example, the object browse-select subprogram can pass the 20 rows it receives from the object browse subprogram. But if the user adds four rows, special processing must be done internally because the object browse-select subprogram only handles a specified number of rows. In this case, two calls to the server must be made to process the 24 rows. Similar complexities arise when a user deletes rows.

The flexibility of allowing methods to be processed at the row level adds some complexity to security considerations. For example, assume that one user is allowed to add, update, and delete rows, and wants to do this to a group of rows at a time, but another user can only add and update rows. When an object browse-select subprogram is used, the FindBy* methods retrieve the data and the MultiMaint method processes row-level methods. As Delete, Store, and Update are row-level methods, both users can be granted access to the MultiMaint method, but only the first user will be granted access to the Delete method.

The Business Service wizard automatically generates the standard methods, but you can also add custom methods to the object browse-select subprogram. For information about this model, see Object-Browse-Select-Subp Model.

By Creating an Empty Service Skeleton

Use this option when you want full control of creating a new subprogram to be used as a business service. The wizard will generate the subprogram proxy and populate the **Business Service** repository. You can edit the new subprogram in Natural for Windows. For information on determining the module names that belong to a business service and loading the modules into the Natural for Windows editor, see **Edit the Service Modules**.

If you select **By creating an empty service skeleton** on the **Choose Type** panel, the **Define Service Parameters** panel is displayed. For example:
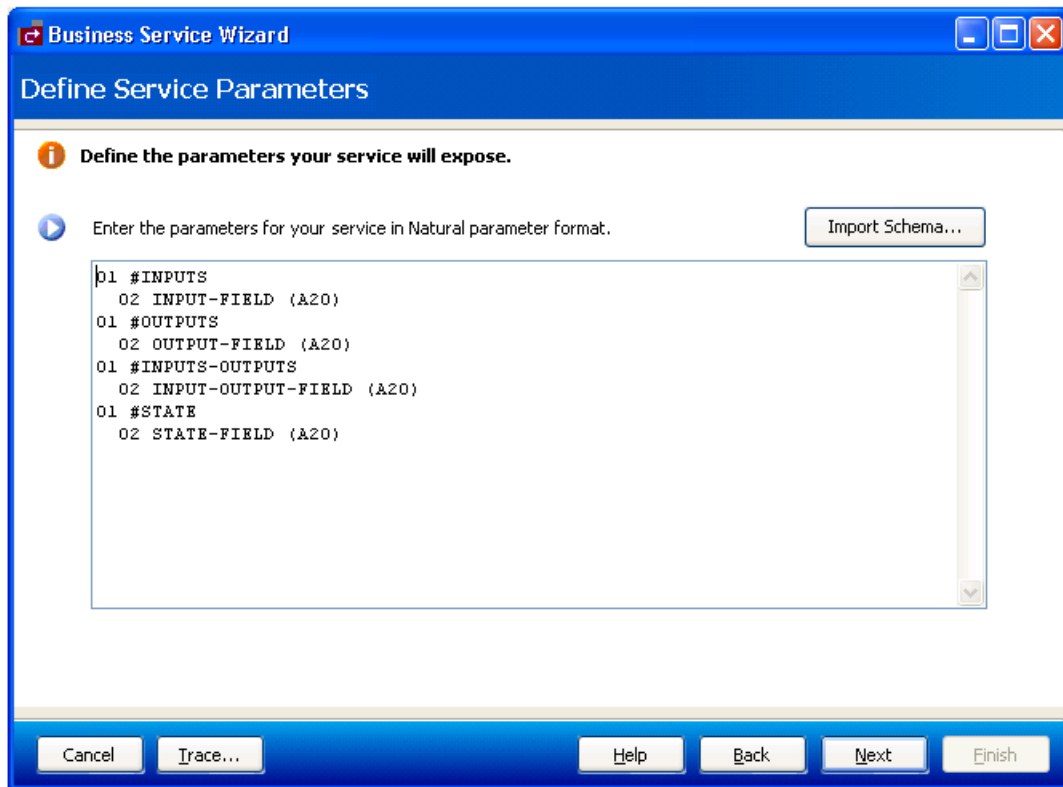
Use this panel to define parameters for the business service. Optionally, you can:

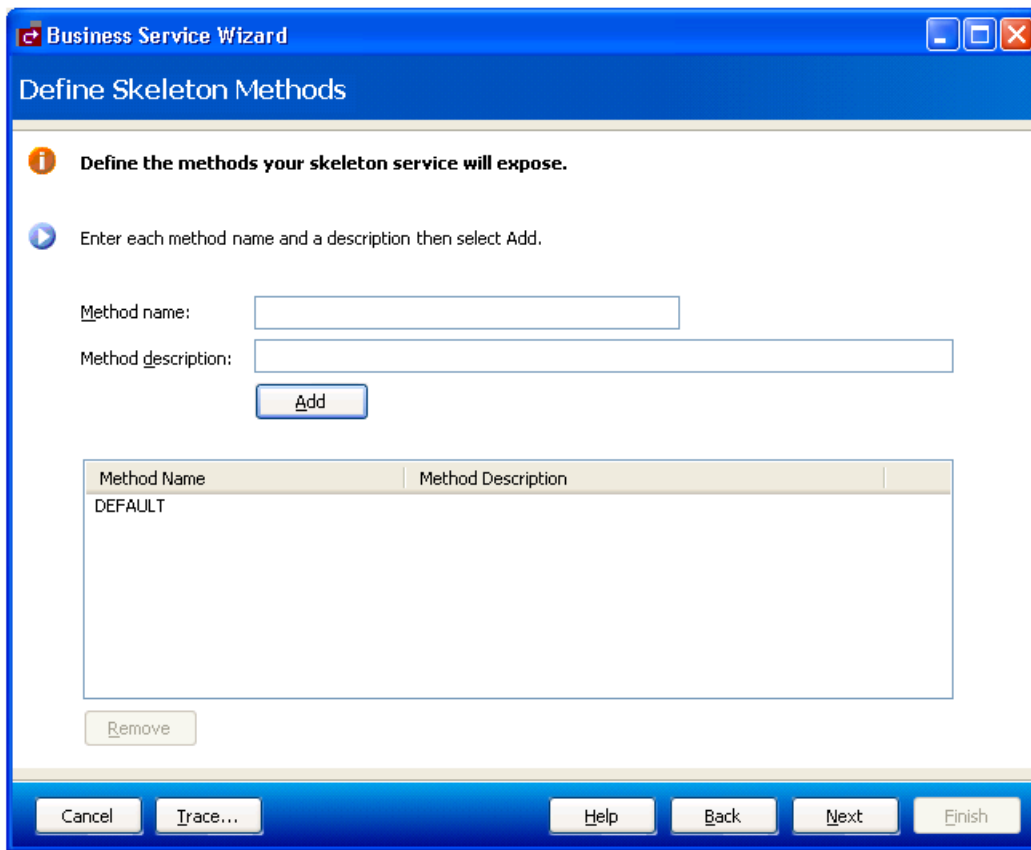| Task	Procedure
Import an external schema to use as the data parameters for the service skeleton	Select **Import schema**. For information, see **Import Schema**.

1. Enter the parameters for the new business service in the space provided.

   The parameters must be in standard Natural parameter format (see example above).

2. Select **Next**.

The **Define Skeleton Methods** panel is displayed. For example:

This panel lists the methods your skeleton service will expose to the user (DEFAULT in this example). Optionally, you can:

| Task	Procedure
Add a new method	Type the method name in **Method name**, a brief description of the method in **Method description**, and select **Add**.
Remove a method	Select the method name in **Method Name** and select **Remove**.

3. Select **Next**.

The last specification panel for the Business Service wizard is displayed.

Import Schema

▶ **To import an external schema to use as the data parameters for the service skeleton:**

1   Select **Import schema** on the **Define Service Parameters**.

The **Import XSD/WSDL** window is displayed. For example:
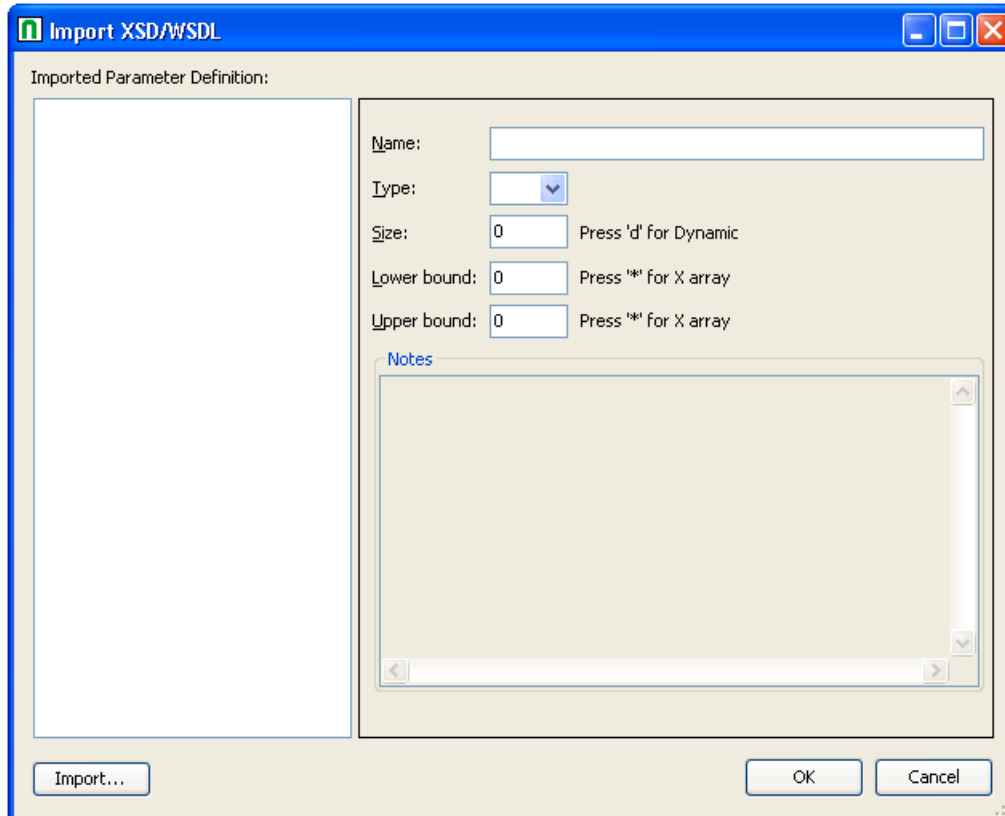
Use this panel to import a schema and, optionally, change the field names, data types, and array definitions (i.e., the use of X-array or dynamic variables). The schema can take be either an XSD (XML Schema document) or WSDL (Web Service Description Language) file. The selected XML schema will be converted into Natural data area format. Optionally, you can:

| Task	Procedure
Change the field name	Select the field in **Imported parameter definition** and type a new name in **Name**.
Change the data type	Select the field in **Imported parameter definition** and select a different data type in **Type**.
Change the use of dynamic variables	Select the field in **Imported parameter definition** and either type "D" in **Size** or remove "D" from **Size**.
Change the use of lower bound X-array variables	Select the field in **Imported parameter definition** and either type "*" in **Lower bound** or remove "*" from **Lower bound**.
Change the use of upper bound X-array variables	Select the field in **Imported parameter definition** and either type "*" in **Upper bound** or remove "*" from **Upper bound**.

2   Select **Import**.

A selection window is displayed to select the schema to import.

3   Select **OK** to save the changes to the parameter definition.

By Using Custom Code Generation Model

Use this option if the subprogram to be used as a business service was generated by a custom model. If you select **By using custom code generation model** on the **Choose Type** panel, the **Configure Custom Model** panel is displayed. For example:
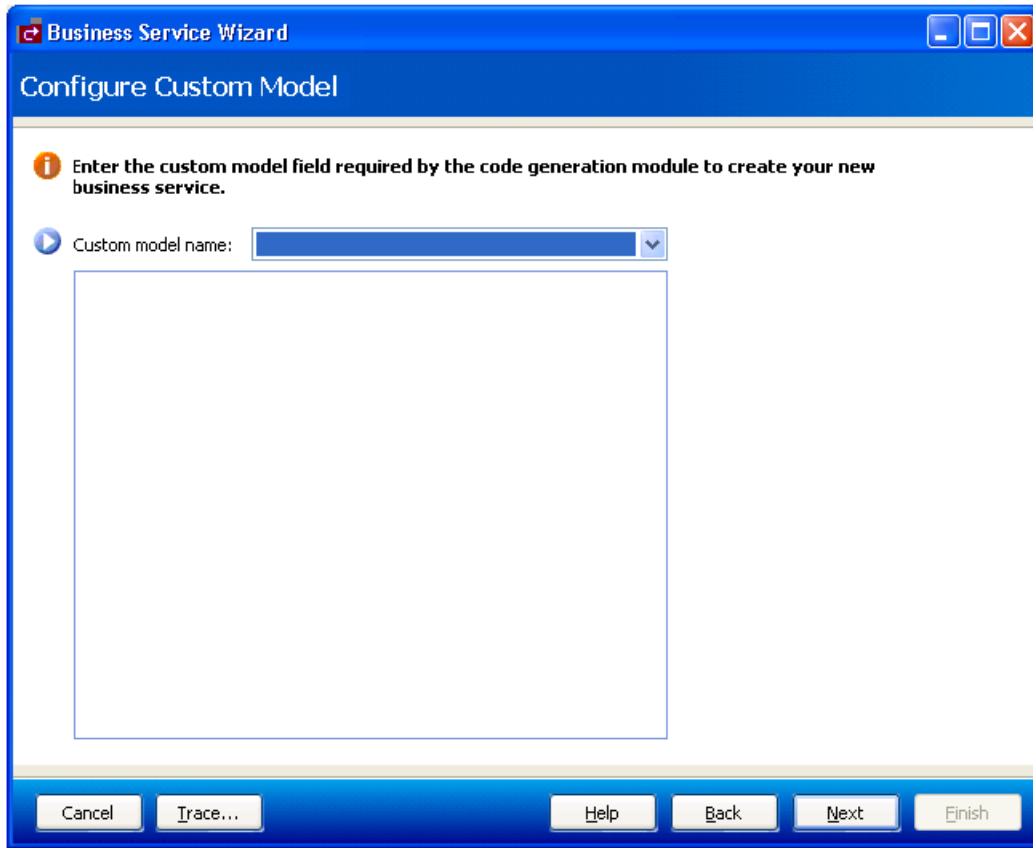
1. Select the custom model in **Custom model name**.

   The fields available for the model are displayed.

2. Enter the fields and values required to use the code generation model.
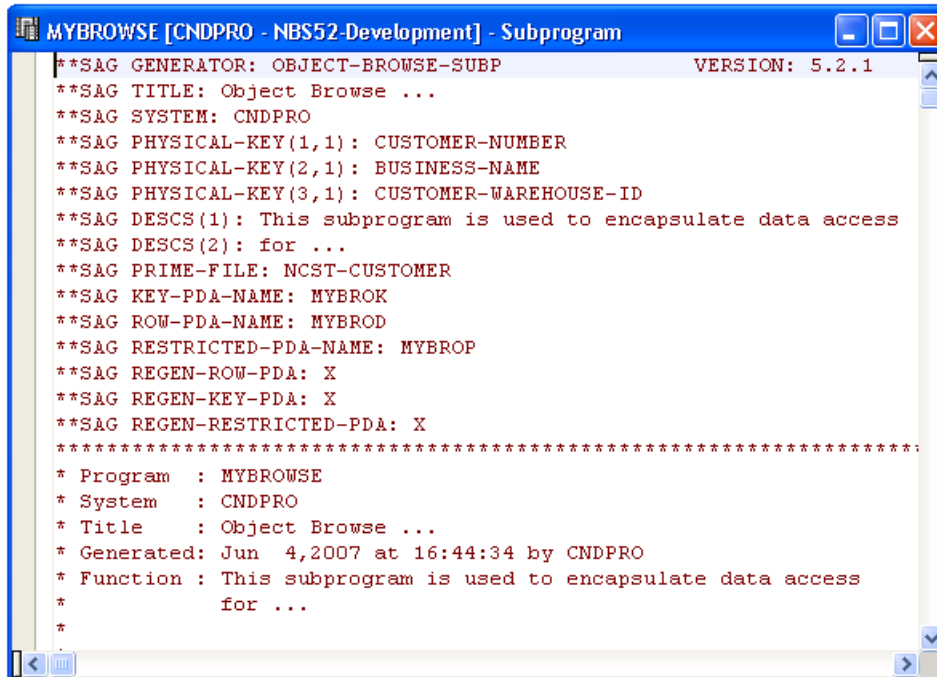
3. Select **Next**.

   The last specification panel for the Business Service wizard is displayed. For a description of this panel, see **Enter Service Name and Select Domain**.

> **Note:** The custom model must be set up in Natural Business Services Administration. For information, see Using Custom Models with Natural Business Services.

Create a New Service from the Program Editor

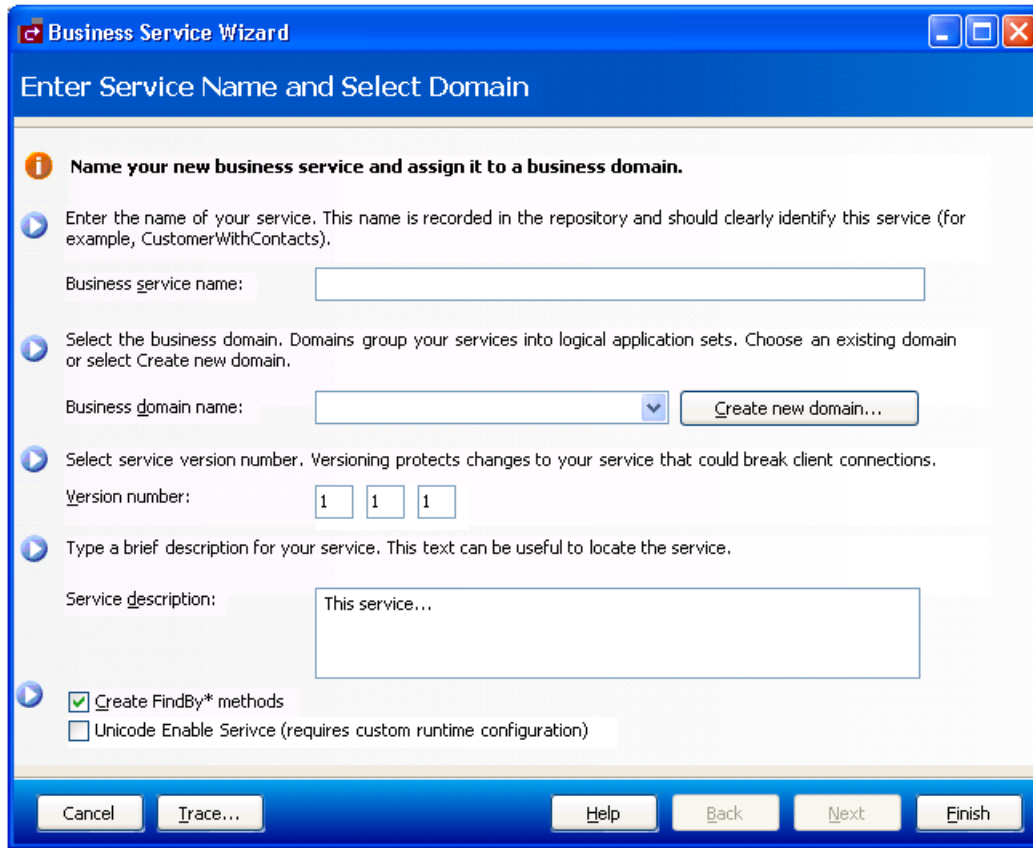You can also create a new business service from a subprogram in the program editor. For example:

▶ **To create a new service from the program editor:**

1   Open the context menu for the subprogram.

2   Select **Create Service**.

The **Enter Service Name and Select Domain** panel is displayed. For example:

Notice the **Create FindBy* methods** option on this panel. This option is available when the module in the program editor is an object browse subprogram.

- For information on the **Create FindBy* methods** option, see **Create FindBy* Methods**.
- For a description of the **Enter Service Name and Select Domain** panel, see **Enter Service Name and Select Domain**.

Create FindBy* Methods

If the subprogram in the program editor is an object browse subprogram, the **Enter Service Name and Select Domain** panel includes an additional field called **Create FindBy* methods**. Select this option to generate business-like method names for browse business services, as opposed to the default BROWSE method. The Business Service wizard:

- Generates a subprogram proxy
- Populates the **Business Service** repository
- Generates an object browse-select subprogram containing FindBy* methods, such as FindBy-BusinessName or FindByDomainName

Regenerate a Business Service

▶ **To regenerate a business service:**

1   Open the context menu for the business service in the **Business Service Repository**.

> **Note:** For information, see **Using the Business Service Repository**.

2   Select **Regenerate Service**.

The Regenerate Business Service wizard panels are displayed, showing the specifications used to generate the service.

3   Revise the specifications as desired.

For information about the options on the wizard panels, see **Create a Business Service**.

4   Select **Finish** on the last wizard panel to regenerate the service.

Regenerate a Business Service Proxy

When you change the parameters in a subprogram used for a business service, you must also change the subprogram proxy for the service to accommodate the new message size. This is done by regenerating the business service proxy.

> **Note:** If the parameters for a business service have changed and a business service consumer has already incorporated the service, the consumer code (i.e., the .NET class) must be regenerated as well.

▶ **To regenerate a business service proxy:**

1   Expand the domain node in the repository explorer for the business service you want to regenerate.

2   Open the context menu for the business service.

3   Select **Regenerate Service Proxy(s)**.

The Natural plug-in regenerates the service proxy (or proxies) without displaying the wizard panels. New metadata is downloaded from the server before regeneration.

> **Note:** The service proxy is also referred to as the subprogram proxy. It provides the link between a subprogram and the Natural Business Services dispatch server.

Test a Business Service Method

▶ **To test a method used by a business service:**

1   Expand the business service node in the repository explorer.

The methods and modules used for the business service are displayed as nodes. For example:
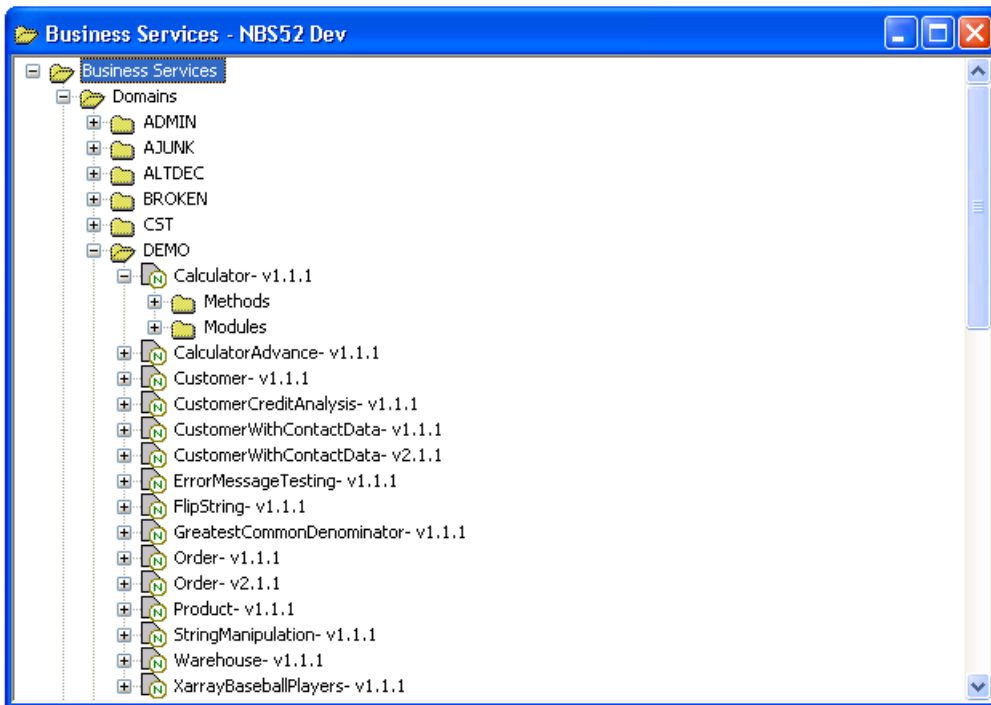
2   Expand the **Methods** node.

3   Open the context menu for the method you want to test.

4   Select **Test**.

The test window is displayed. For a description of this window, see **Test a Business Service**.

Parameters for the Standard Methods

Certain methods are standard to business services. These methods can be divided into the following categories:

Category	Standard Methods
Single-row access	DELETE, EXIST, FORMER, GET, INITIALIZE, NEXT, STORE, and UPDATE
Multiple-row access	BROWSE, MultiMaint, Update, Delete, Store, and the FindBy* series of methods

Understanding the parameters for these methods will simplify the testing process. This section covers the following topics:

- Single-Row Access
- Multiple-Row Access

Single-Row Access

The DELETE, EXIST, FORMER, GET, INITIALIZE, NEXT, STORE, and UPDATE methods are used on a single row of data. All methods that access a single row of data contain the same PDAs. These PDAs are:

PDA	Description
Data	Contains the -ID values (for example, MCUSTA-ID).
Restricted	Determines whether data has been modified between the time it was retrieved for the test and the time a data maintenance method was requested. This PDA should not be altered. **Tip:** The name of this PDA typically ends with an "R".
Method (CDAOBJ2)	Contains method data that is handled by Natural Business Services.
MSG-INFO	Contains messages from the server; it is used for output only.

This section covers the following topics:

- DELETE Method
- EXIST Method
- FORMER, GET, or NEXT Method
- INITIALIZE Method
- STORE Method

- UPDATE Method

DELETE Method

▶ **To test the DELETE method:**

1   Issue the GET, NEXT, or FORMER method to retrieve a record.

2   Issue the DELETE method to delete the record.

A confirmation message should be displayed.

Note: Ensure you are deleting the correct record.

EXIST Method

▶ **To test the EXIST method:**

1   Issue the INITIALIZE method (to delete all data).

2   Enter a customer number (for example, "5555").

3   Issue the EXIST method.

This request will be successful whether the customer exists or not. The result of the request is contained within the method PDA (CDAOBJ2) under OUTPUTS (the EXIST flag will be either True or False).

FORMER, GET, or NEXT Method

The easiest single-row access method to test is NEXT because it does not require any input parameters. This method simply gets the next record in the dataset.

The sequence of records within the dataset is determined by the PDA values that end with -ID. For example, when testing the NEXT method for the Customer business service in the Demo domain, MCUSTA and MCUSTA-ID will be displayed (MCUSTA will contain additional data). When you expand MCUSTA-ID, CUSTOMER-NUMBER is displayed. This indicates that the next highest customer number is displayed when the NEXT method is issued (and the previous customer number is displayed when the FORMER method is issued).

To ensure that the restricted PDA is populated correctly, a record must be retrieved before an UPDATE or DELETE method can be issued. The record can be retrieved by issuing the NEXT or FORMER method, or issuing the GET method when you know what the key value is and whether the data for the key value is supplied in the data PDA.

▶ **To test the GET method:**

1 Enter a valid customer number in CUSTOMER-NUMBER in the data PDA.

2 Issue the GET method.

   The record associated with specified customer number should be displayed.

INITIALIZE Method

▶ **To test the INITIALIZE method:**

■ Issue the INITIALIZE method to delete all data except the key values.

💡 **Tip:** You can also delete the data from the -ID section of the data PDA and from the restricted PDA (except for the first reference to the key value) and then reissue the GET method. This should retrieve all the values for a customer based on the customer number provided, assuming that number exists. If the record does not exist, a message is displayed.

STORE Method

▶ **To test the STORE method:**

1 Enter a unique value in the key field in the data PDA.

2 Issue the STORE method to store the record.

   A confirmation message should be displayed.

UPDATE Method

▶ **To test the UPDATE method:**

1 Issue the GET, NEXT, or FORMER method to retrieve a record.

2 Change the fields in the data PDA.

3 Issue the UPDATE method to update the record.

   A confirmation message should be displayed.

Multiple-Row Access

The BROWSE, MultiMaint, Update, Delete, and Store methods, as well as the FindBy* series of methods, are used on multiple rows of data.

> **Note:** The Update, Delete, and Store methods are handled internally by the MultiMaint method and should not be used as individual methods. They allow administrators to revoke access to these methods when the MultiMaint method is used.

All methods that access multiple rows of data contain the same PDAs. These PDAs are:

PDA	Description
Row	Contains the rows of data retrieved from the database (in an array of 1:20). **Tip:** The name of this PDA typically ends with a "D" or "E1".
Key	Contains the key fields and starting values for components of the key being used. For example, if you enter "M" in BUSINESS-NAME for the Customer business service in the Demo domain, the BROWSE method displays records beginning at "M". **Tip:** The name of this PDA typically ends with a "K".
Restricted	Contains state information, such as where to resume browsing, as well as fields like FIRST-TIME and KEY-DATA. This PDA should not be altered. **Tip:** The name of this PDA typically ends with a "P".
MSG-INFO	Contains messages from the server; it is used for output only.

This section covers the following topics:

- BROWSE Method
- MultiMaint and FindBy* Methods

BROWSE Method

In addition to the standard PDAs, the BROWSE method contains additional PDAs that provide specialized functionality. For example, the BROWSE method can sort data up to six different ways depending on the availability of server-side keys. In addition, the server-side keys can be derived (so that they make up more than one field).

Unfortunately, which keys are available, and which fields make up the keys, is unknown during testing. The names of the fields that make up the keys are contained in the key PDA.

▶ **To test the BROWSE method:**

1   Determine which keys are available and which fields make up the keys.

To do this, refer to the specification lines for the object browse subprogram. For example, the following keys are contained in the specifications for the ACUSTN subprogram used by the Customer business service in the Demo domain:

```
**SAG LOGICAL-KEY(2): NAME-BACKWARDS
**SAG PHYSICAL-KEY(2,1): BUSINESS-NAME
**SAG DESCENDING(2,1): X
```

where LOGICAL-KEY contains the field used to sort data in a particular order and PHYSICAL-KEY contains the fields that make up that key. For instance, back to our customer business service in the demo domain.

2   Enter "NAME-BACKWARDS" in the SORT-KEY field in the CDBRPDA PDA.

The results should be sorted by name in descending order.

You can also use other fields in CDBRPDA to test the BROWSE method. For example, if the row PDA contains the COUNT field, you can use the HISTOGRAM field to return the number of key values, as opposed to the entire record (such as "SMITH 20" to indicate there are 20 Smiths in the database).

You can also request that fewer rows of data be returned (than the standard 20 rows) by entering a number in the ROWS-REQUESTED field.

> **Note:** For obvious reasons, you cannot specify a value higher than the number of rows available.

To specify a range of values, enter a number in the RANGE-OPTION field based on the following information:

```
3 DEFAULT                            N    1 INIT<0> /* Input specifies a starting
value, LE or GE will be determined based the sort order.
 *                                                          /* Embedded
wildcard can be specified using >, < and characters for Alpha an
 *                                                          /* numeric
characters.
   3 LESS-THAN                       N    1 INIT<1>
   3 LESS-THAN-OR-EQUAL              N    1 INIT<2>
   3 EQUAL                           N    1 INIT<3>
   3 GREATER-THAN-OR-EQUAL           N    1 INIT<4> /* Default
   3 GREATER-THAN                    N    1 INIT<5>
   3 BEGINS-WITH                     N    1 INIT<6> /* Prefix of key mat
 *                                        /* the input key.
   3 NO-WILDCARD                     N    1 INIT<7>
```

If the RESTART field is False and the data has not changed, the BROWSE method will continue to get the next *n* rows of data until the end of data is reached.

> **Tip:** The end of data is reached when the END-OF-DATA field is True.

> **Note:** If the RESTART field is True, the browse action will restart.

The ACTUAL-ROWS-RETURNED field contains the number of rows returned.

MultiMaint and FindBy* Methods

In addition to the standard PDAs used for multiple-row access, the MultiMaint and FindBy* series of methods contain the CDBUPDA PDA, which is a subset of the CDBRPDA PDA used by the BROWSE method. All fields in CDBUPDA behave the same way as described above. The key differences between CDBUPDA and CDBRPDA include:

- The SORT-KEY field is not required because the FindBy* methods assign the SORT-KEY value on the server. You do not have to guess what the sort key should be.

- The BUSINESS-INFO field is a subset of MSG-INFO; this field contains messages that pertain to all rows, as opposed to messages for a specific row.

Except for the Add action, the MultiMaint method can only be issued after a FindBy* method has retrieved the rows for maintenance. The MultiMaint method does not retrieve new rows (as do the FindBy* and BROWSE methods); it only alters the current rows based on the value in the ROW-STATE field for each row in the data PDA.

▶ **To test the MultiMaint method:**

1   Enter "A" in the ROW-STATE field in the CDBUPDA PDA.

    The output response should be displayed in the state as "AS" add successful.

2   Enter "U" in ROW-STATE.

    The output response should be displayed in the state as "US" update successful.

3   Enter "D" in ROW-STATE.

    The output response should be displayed in the state as "DS" delete successful.

> **Note:** The existing states can be found in CDSTATE. For a list of valid values, see ROW-STATE Values.

Test a Business Service

This section describes how to test a business service. You can open multiple test windows as required. You can also edit and test code without closing the **Test** window.

▶ **To test your business service:**

1   Expand the domain node in the repository explorer for the business service you want to test.

2   Open the context menu for the business service.

3   Select **Test**.

The **Test Service** window is displayed for the selected service. The following example shows the GetLineup method for the **XarrayBaseballPlayers** business service in the DEMO domain:
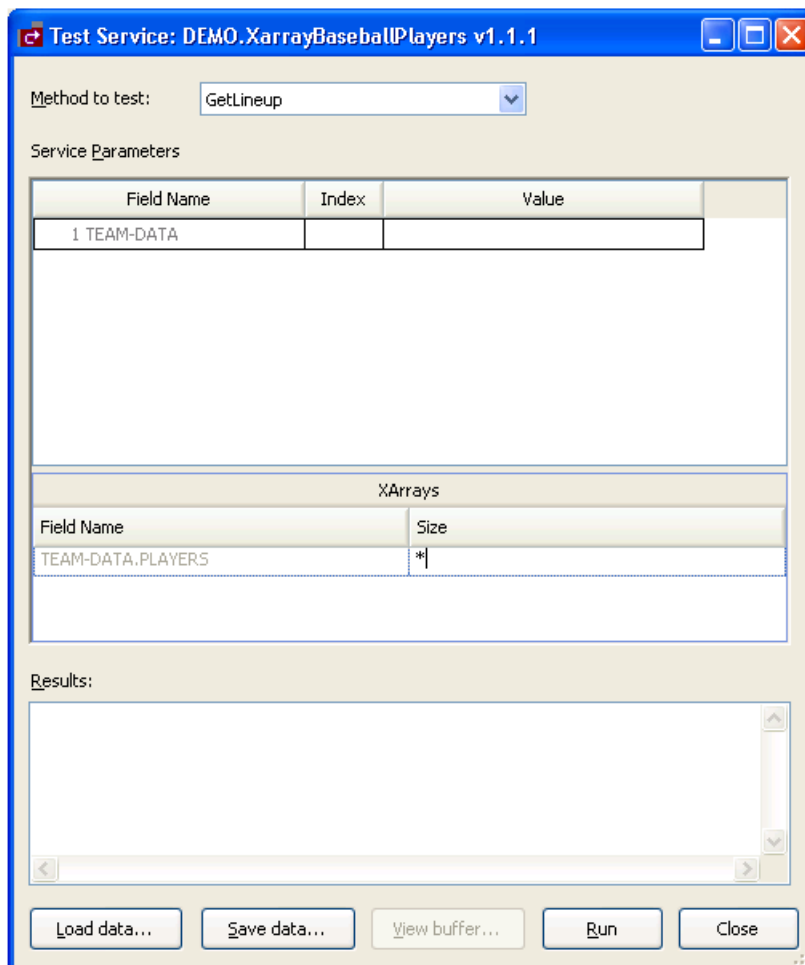
Use the panel to test the methods for the business service.

> **Note:** The **XArrays** section displays one row for each X-array.

Optionally, you can:

| Task	Procedure
Configure the cell alignment for the Field Name, Index, and Value columns	Open the context menu for the column and select **Cell alignment**. The **Cell Alignment** window is displayed. For information, see **Configure the Cell Alignment**.
Change the number of elements in an X-array	Type the new value in **Size**. For information, see **Change the Number of Elements in an X-Array**.
Save the test data	Select **Save data** and specify the location in which to save the test data.
Reload saved data for testing	Select **Load data** and specify the location from which to retrieve saved test data. For example, instead of entering "1111" each time to display customer 1111, you can save the data and reload it. This is useful when there is more than one data field, such as: domain (DEMO), service (Customer), and version (010101).
Display the parameter data in the editor	Select **View buffer**.

4   Select a method in **Method to test**.

5   Select **Run**.

The result of the test is displayed in **Results**. You can use this panel to test all methods defined for this business service.

6   Select **Close** to close the **Test Service** window.

Configure the Cell Alignment

You can configure the cell alignment for columns in the **Service Parameters** section in the **Test Service** window

▶ **To configure the cell alignment:**

1   Open the context menu for the column.

2   Select **Cell alignment**.

The **Cell Alignment** window is displayed. For example:
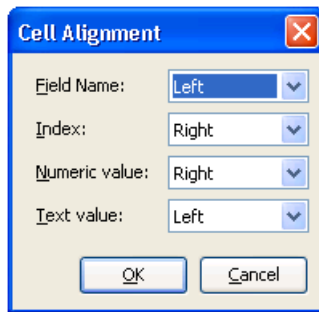
Select "Left", "Right", or "Center" as the cell alignment for each column. You can select a different alignment for numeric values than for text values.

Change the Number of Elements in an X-Array

▶ **To change the number of elements in an X-array:**

■   Enter the new number in **Size**.

The **Service Parameters** section reflects the change. For example:
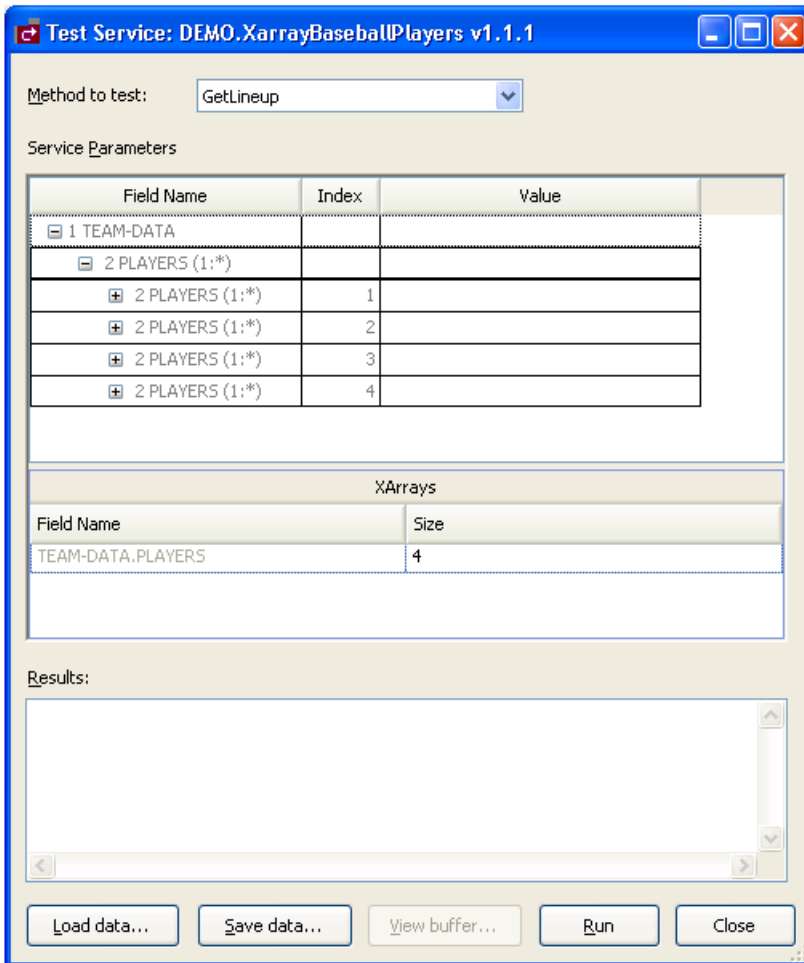
Delete a Business Service

> **Note:** Removing a business service will only delete the service from the repository, it will not delete the Natural modules.

▶ **To delete an existing business service:**

1 Expand the **Domains** node in the repository explorer.

2 Open the context menu for the business service you want to delete.

3 Select **Delete**.

A confirmation window is displayed.

4   Select **Yes** to delete the business service.

Edit a Service Definition
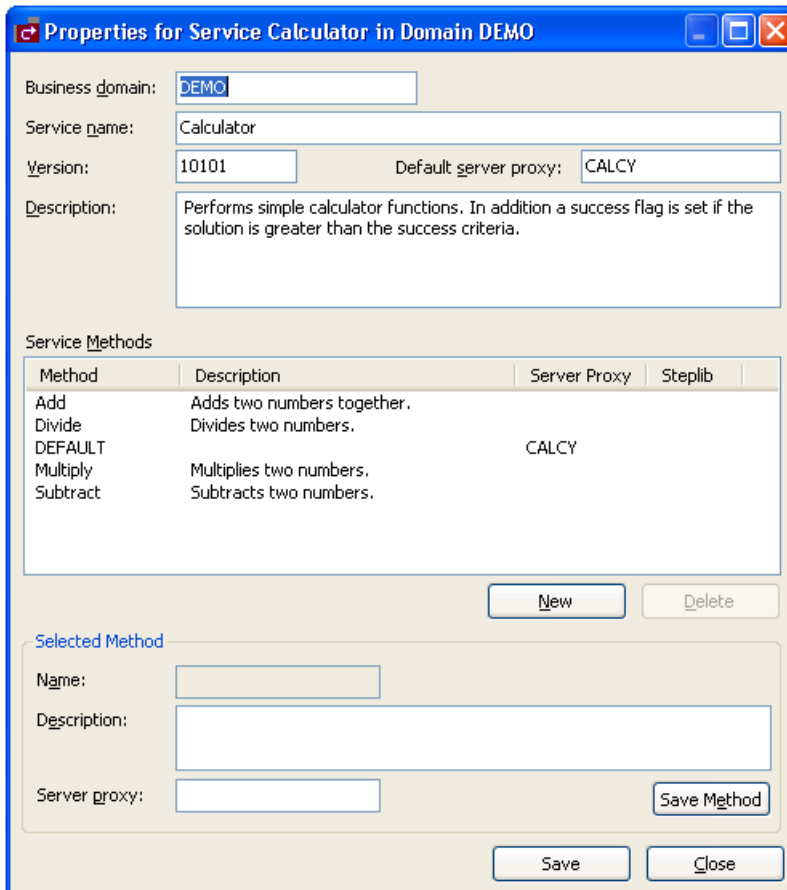
▶ **To edit a business service definition:**

1   Open the context menu for the business service in the repository explorer.

> 📄   **Note:** For information on the **Business Service** repository, see **Using the Business Service Repository**.

2   Select **Edit**.

The **Properties** window for the business service is displayed. For example:

Use this window to:

- Change the business service description.

- Edit an existing method.

  To edit a method, select the method in **Service Methods** and change the information in **Selected Method**. For example, you can the change the name, description, server proxy name, or step library chain.

- Add a new method.

  To add a new method, select **New** and enter information about the new method in **Selected Method**. Select **Save Method** to save the new method. The new method is displayed in **Service Methods**.

- Delete a method.

  To delete a method, select the method in **Service Methods** and select **Delete**. The method is deleted from **Service Methods**.

3   Select **Close** to close the **Properties** window.

Edit the Service Modules

▶ **To edit the business service subprograms:**

1   Expand the business service node in the repository explorer.

     The methods and modules used for the business service are displayed as nodes in the repository view.

2   Expand the **Modules** node.

     The Natural modules used for the business service are displayed. For example:
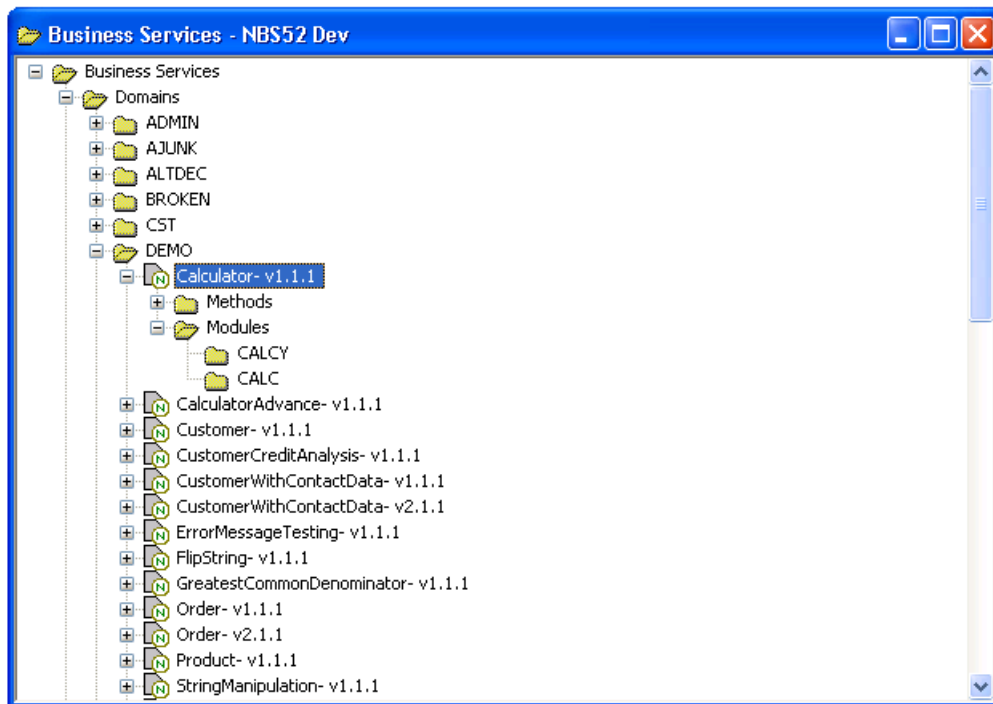
3   Do one of the following:

▪ To edit all modules, open the context menu for **Modules** and select **Edit all**.

▪ To edit one module, open the context menu for the module and select **Edit**.

The module(s) is displayed in the editor. For example:

You can edit the module in this editor and then save the changes. For information on using this editor, see the Natural documentation.

Transform a Browse Module

Use the Transform Browse wizard to transform a Natural Construct-generated browse module into object browse modules for use in a client/server environment.

▶ **To access the Transform Browse wizard:**

1   Display the browse module in the program editor (for information, see the Natural for Windows documentation).

2   Select **Transform Browse**.

The **Verify Transformation Specifications** panel is displayed. For more information, see *Natural Plug-in* in Verify Transformation Specifications Panel.

5 Administering Business Services

This section describes the procedures to administer business services. The following topics are covered:

Configure the Natural Plug-in

Before using the Natural plug-in for Natural Business Services the first time, you must set up appropriate domain and step library chains. During installation, some domains and steplib chains are automatically installed (for example, the DEMO domain and the DEMO steplib chain). We recommend that you also set up a project domain and steplib chain.

This section covers the following topics:

- Create and Maintain Steplib Configurations
- Create and Maintain Domain Definitions

Create and Maintain Steplib Configurations

This section describes how to create a new step library configuration (steplib chain) and edit an existing steplib chain. A steplib chain identifies where your business service libraries reside on the server. To locate and execute business service modules, you must set up a steplib chain and link it to your business service domain. For more information on steplibs, see Step 1: Define the Steplib Chain.

The following topics are covered:

- Create a New Steplib Configuration
- Modify an Existing Steplib Configuration

Create a New Steplib Configuration

▶ **To create a new steplib configuration:**

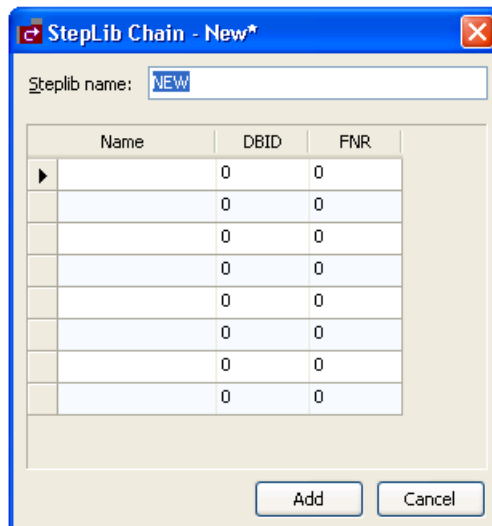1 Open the **Business Services** explorer.

   For information, see **Access Business Services**.

2 Expand the **Business Services** node.

3 Expand the **Configuration** node.

4 Open the context menu for **Steplibs**.

5 Select **Add New Steplib Configuration**.

The **StepLib Chain - New** window is displayed. For example:

6   Type the name of the steplib chain in Steplib name.

7   Type the names of each library in your steplib chain in the space provided.

   Optionally, you can type the database ID (DBID) and file number (FNR) for each library. Your steplib chain can contain up to eight libraries. By default, the FNAT SYSTEM library will be added to the list of libraries in the new steplib chain. If the DBID and FNR are blank, the default FUSER (or FNAT if a SYS* library is specified) will be used.

8   Select **Add** to add the new steplib configuration.

Modify an Existing Steplib Configuration

▶ **To modify an existing steplib configuration:**

1   Expand the **Steplibs** node in the **Business Services** explorer.

2   Open the context menu for the steplib configuration you want to edit.

3   Select **Edit**.

The **StepLib Chain -** *SteplibName* window is displayed. For example:
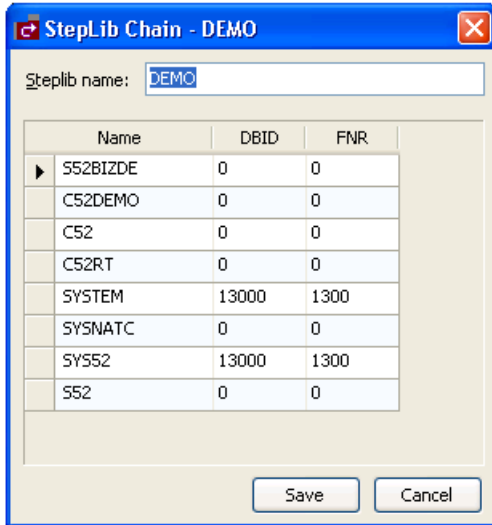
Use this window to:

Task	Procedure
Remove a library from the steplib configuration	Select the library name and select **Delete**.
Add a library to the steplib configuration	Type the library name and, optionally, DBID and FNR values. **Note:** You can only add a steplib configuration when there are less than eight libraries listed in the window.

4   Select **Save** to save changes to the steplib configuration.

> **Note:** If the steplib chains are not step up before using the Business Service wizard, the wizard may not be able to find required modules.

Create and Maintain Domain Definitions

This section describes how to create a new domain and edit an existing domain. Domains are used to group related business services. For more information on domains, see Step 2: Define the Domain.
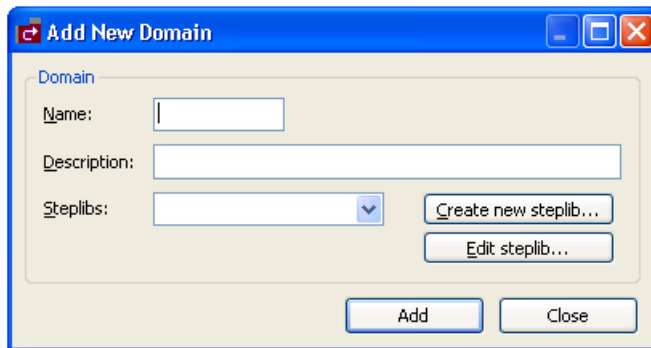
The following topics are covered:

- Create a New Domain
- Modify an Existing Domain

Create a New Domain

▶ **To create a new domain:**

1   Open the context menu for **Domains** in the **Business Services** explorer.

2   Select **Add New Domain**.

The **Add New Domain** window is displayed. For example:

Use this window to name the new domain, provide a description of the domain, and select a steplib chain. Optionally, you can:

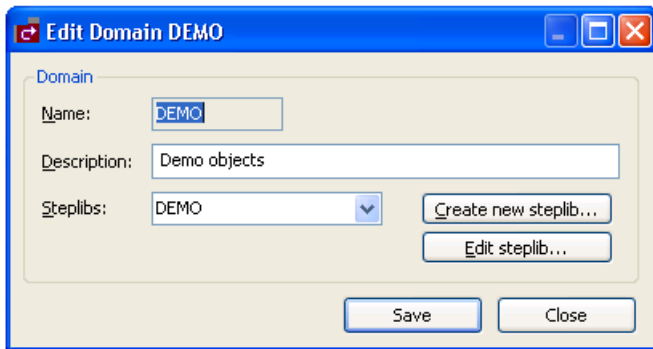| Task	Procedure
Create a new steplib configuration	Select **Create new steplib**. The **New Steplibrary** window is displayed. For information, see **Create a New Steplib Configuration**.
Modify an existing steplib configuration	Select **Edit steplib**. The **Edit Steplibrary** window is displayed. For information, see **Modify an Existing Steplib Configuration**.

3   Type the name of the domain in **Name**.

---

4   Type a brief description of the domain in **Description**.

5   Select at least one steplib chain from **Steplibs**.

6   Select **Add** to add the new domain.

Modify an Existing Domain

▶ **To modify an existing domain definition:**

1   Open the context menu for the domain you want to edit in the **Business Services** explorer.

2   Select **Edit**.

The **Edit Domain** *DomainName* window is displayed. For example:

Use this window to:

Task	Procedure
Change the description of the domain	Type a new description in **Description**.
Select another steplib chain to use for this domain	Select the new steplib chain in **Steplibs**.
Create a new steplib configuration	Select **Create new steplib**. The **New Steplibrary** window is displayed. For information, see **Create a New Steplib Configuration**.
Modify an existing steplib configuration	Select **Edit steplib**. The **Edit Steplibrary** window is displayed. For information, see **Modify an Existing Steplib Configuration**.

3   Select **OK** to save changes to the domain definition.

Define Security for Domains

Security is applied to a domain by associating the domain with a group. Access can then be granted or revoked at a domain, business service, and/or method level. In addition, the **Disable** option can be used to temporarily disable access to a business service without permanently affecting the security settings.

> **Note:** For more information on defining security, see Server Security Overview.

▶ **To define security for a domain:**

1   Open the context menu for the domain you want to secure in the **Business Services** explorer.

2   Select **Security**.

The **Security** window is displayed. For example:
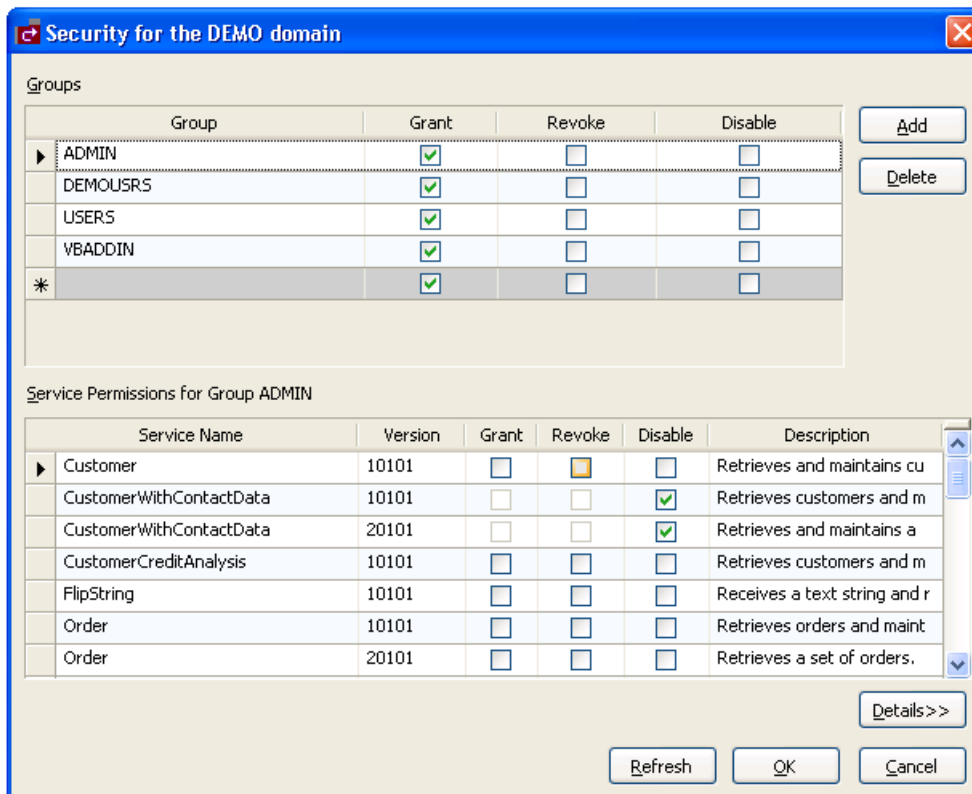
This window displays the groups, business services, and methods for the selected domain. (Use the scroll bar to display the methods.)

> **Tip:** The methods displayed in this window are used by the business service selected in Service Name (the Customer service in the example above). If you are applying method-level security, ensure that you have selected the correct service.

> **Note:** To return all settings to their original settings when the window was displayed, select **Refresh**.

3   Select **Grant**, **Revoke**, or **Disable** at the group, service and/or method level for the selected domain.

> **Note:** The **Disable** option temporarily revokes access to a method. When the method is enabled, the previous settings are restored.

If the methods are not displayed, select **Details**.

4   Select **OK** to save the security settings.

Add a Security Group

You can also use the **Security** window to add a security group to the domain.

▶ **To add a security group to the domain:**

1   Type the name of the group on an empty line in the Groups section.

The empty line is indicated with an asterisk (*).

2   Select **Add**.

The security group is added to the domain.

Delete a Security Group

You can also use the **Security** window to remove a security group from the domain.

▶ **To delete (remove) a security group from the domain:**

1   Select the group in the Groups section.

2   Select **Delete**.

A confirmation window is displayed.

3   Select **Yes**.
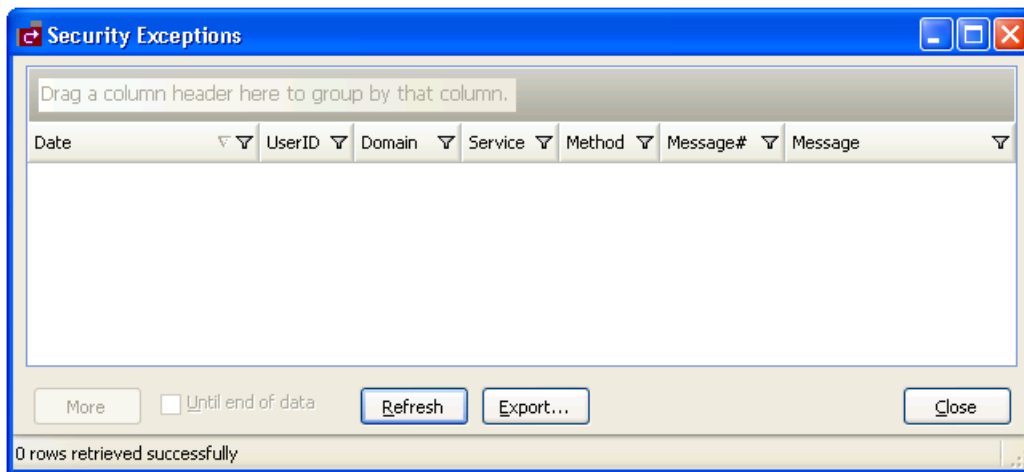
The security group is removed from the domain.

Audit Security Exceptions for a Connection

You can display information on security exceptions for a connection.

▶ **To audit security exceptions for a connection:**

1   Select **Business Services > Security Audit** on the **Tools** menu.

The **Security Exceptions** window is displayed. For example:

Use this window to display the date of each audit activity, the user ID of the user who performed the action, and the domain, service, and method names. You can also use it to display the message number and text for each exception. Optionally, you can use this window to:

Task	Procedure
Group the results by column	Drag the column heading to the shaded area.
Rearrange the columns	Drag the column heading to the far left of this window (for example, to sort by domain, drag the Domain heading to the left of the Date heading).
Filter the information based on a column heading	Select "Custom" in the column you want to filter and specify the filter criteria in the window that is displayed. For information, see **Filter Security Exceptions**.
Export the security exceptions to a work file	Select **Export** and specify the location of the work file.

2   Select **All** in **Date**.

3   Select **Refresh**.

All security exceptions for all dates are displayed.

> **Note:** If there is more information than can be displayed in one window, select **More** to display the additional activities.

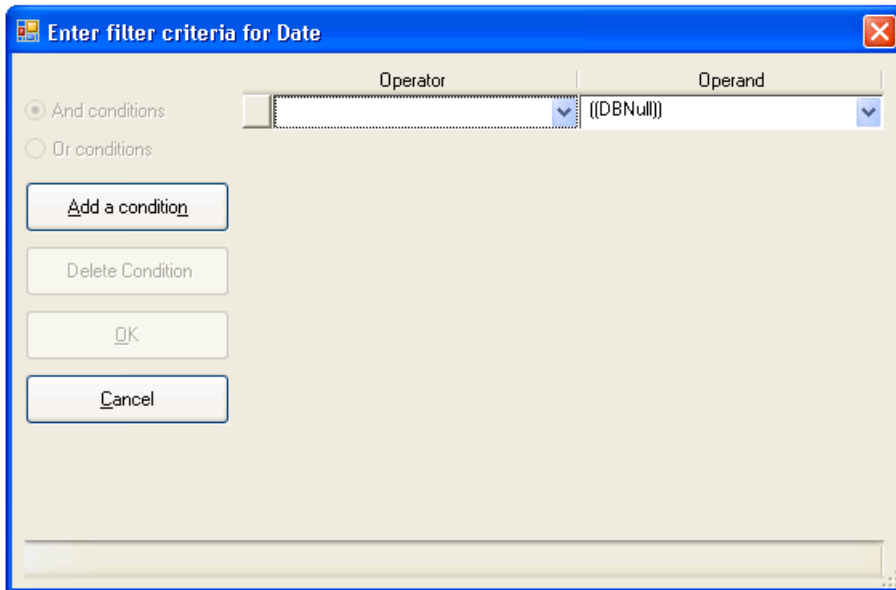4   Select **Close** to close the **Security Exceptions** window.

Filter Security Exceptions

You can filter the security exceptions based on any column heading. For this example, the inform-ation is filtered based on the Date heading.

▶ **To filter the security exceptions information by date:**
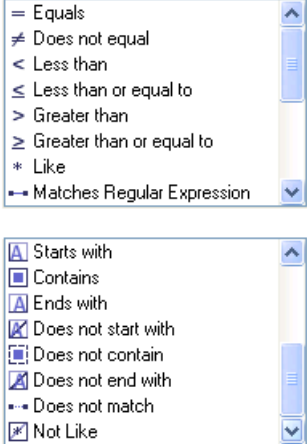
1   Select **Custom** in **Date**.

The **Enter Filter Criteria for Date** window is displayed. For example:

The options in this window are:

| Option	Description
Operator	 Search criteria for the operator. The search criteria are:
Operand	Search criteria for the operand. The search criteria are based on the operator. For example, if you select **< Less than** in Operator, the following operands are available: [Message#] [Message] [UserID] [Domain] [Service] [Method]
And conditions	If there is more than one condition, this option indicates whether to display audit information that matches all conditions. ■ For information on adding a new condition, see **Add a Condition**. ■ For information on deleting a condition, see **Delete a Condition**.
Or conditions	If there is more than one condition, this option indicates whether to display audit information that matches any condition.

2   Select **OK**.

The **Security Exceptions** window displays the audit information that matches the search criteria.

Add a Condition

▶ **To add a condition:**

1   Select **Add a condition**.

An input field is displayed for the operator and operand. For example:
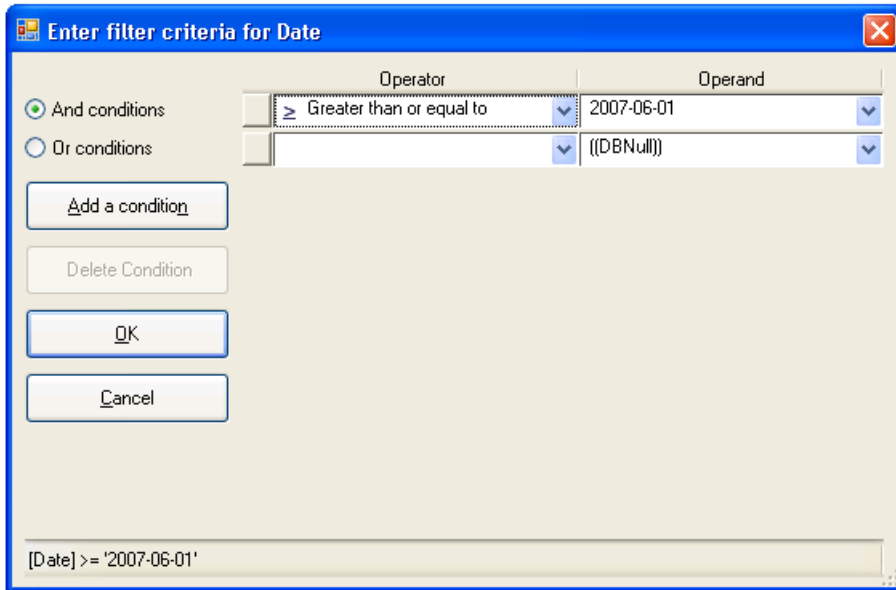
2   Select the operator for the condition in **Operator**.

3   Select the operand for the condition in **Operand**.

4   Select one of the following:

  ▪ **And conditions**

    This option displays the security exceptions that match all conditions.

  ▪ **Or conditions**

    This option displays the security exceptions that match any condition.

5   Select **OK**.

The **Security Exceptions** window is displayed, showing the security exceptions that match the search criteria.

Delete a Condition

▶ **To delete a condition:**

1   Select the operator/operand condition you want to remove.

2   Select **Delete a condition**.

    The condition is removed from the **Enter Filter Criteria for Date** window.
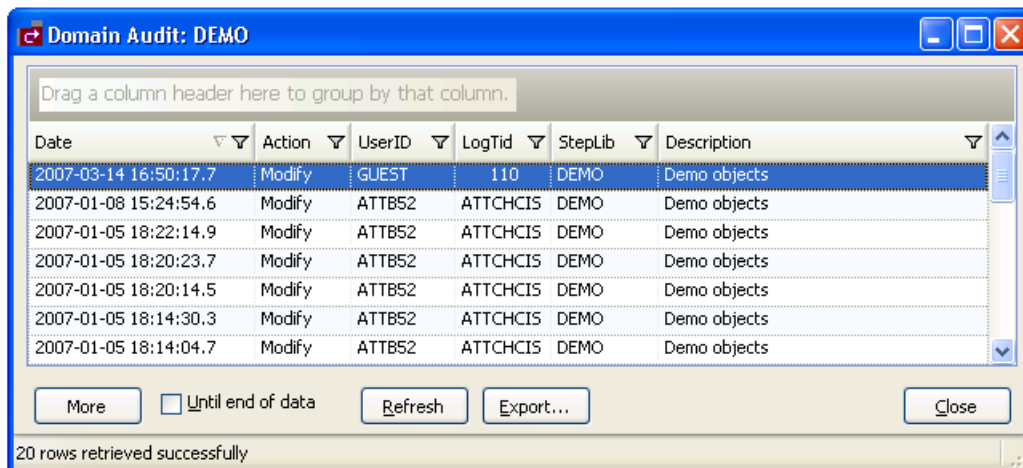
3   Select **OK** to close the window.

Audit Domains

You can display audit information on activities in a domain.

▶ **To audit a domain:**

1   Open the context menu for the domain you want to audit in the **Business Services** explorer.

2   Select **Audit**.

    The **Domain Audit** window is displayed. For example:

Use this window to display the date of each audit activity, the action performed, and the user ID of the user who performed the action. You can also use it to display the terminal ID for the log file, the names of the steplibs, and a brief description of the domain. Optionally, you can use this window to:

Task	Procedure
Group the results by column	Drag the column heading to the shaded area.
Rearrange the columns	Drag the column heading to the far left of this window (for example, to sort by domain, drag the Domain heading to the left of the Date heading).
Filter the information based on a column heading	Select "Custom" in the column you want to filter and specify the filter criteria in the window that is displayed. For information, see **Filter Security Exceptions**.
Export the security exceptions to a work file	Select **Export** and specify the location of the work file.

Note: If there is more information than can be displayed in one window, select **More** to display the additional activities.

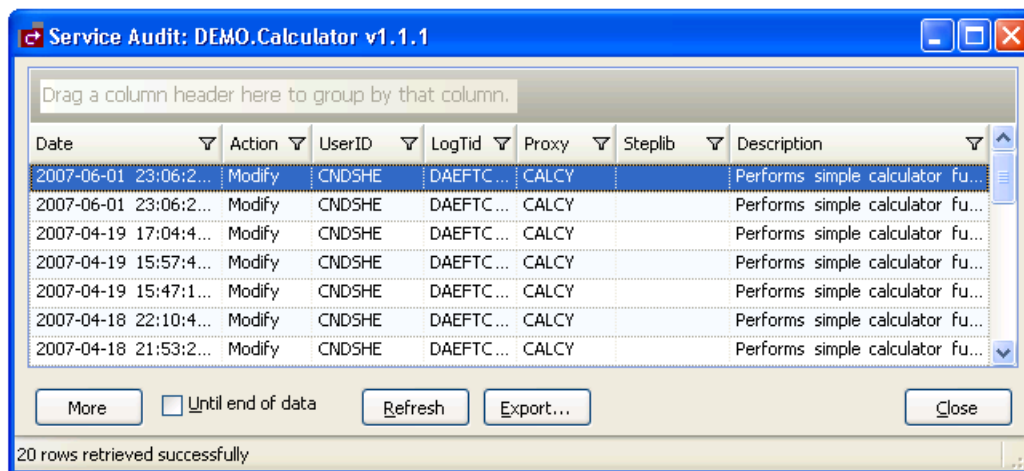3   Select **Close** to close the **Domain Audit** window.

Audit Business Services

You can display audit information on activities in a business service.

▶ **To audit business services:**

1   Open the context menu for the business service you want to audit in the **Business Services** explorer.

2   Select **Audit**.

The **Service Audit** window is displayed. For example:

Use this window to display the date of each audit activity, the action performed, and the user ID of the user who performed the action. You can also use it to display the terminal ID for the log file, the names of the subprogram proxies and steplibs, and a brief description of each business service. Optionally, you can use this window to:

Task	Procedure
Group the results by column	Drag the column heading to the shaded area.
Rearrange the columns	Drag the column heading to the far left of this window (for example, to sort by domain, drag the Domain heading to the left of the Date heading).
Filter the information based on a column heading	Select "Custom" in the column you want to filter and specify the filter criteria in the window that is displayed. For information, see **Filter Security Exceptions**.
Export the security exceptions to a work file	Select **Export** and specify the location of the work file.

3   Select **Close** to close the **Service Audit** window.

Deploy a Domain to Another Repository

You can deploy all business services in a specified domain to another repository.

▶ **To deploy all business services in a domain to another repository:**

1   Open the context menu for the domain you want to deploy in the **Business Services** explorer.

2   Select **Deploy**.
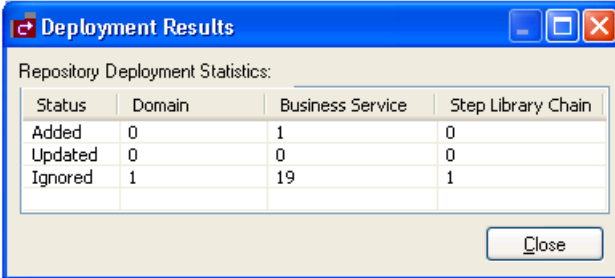
The **Deploy Services** window is displayed. For example:
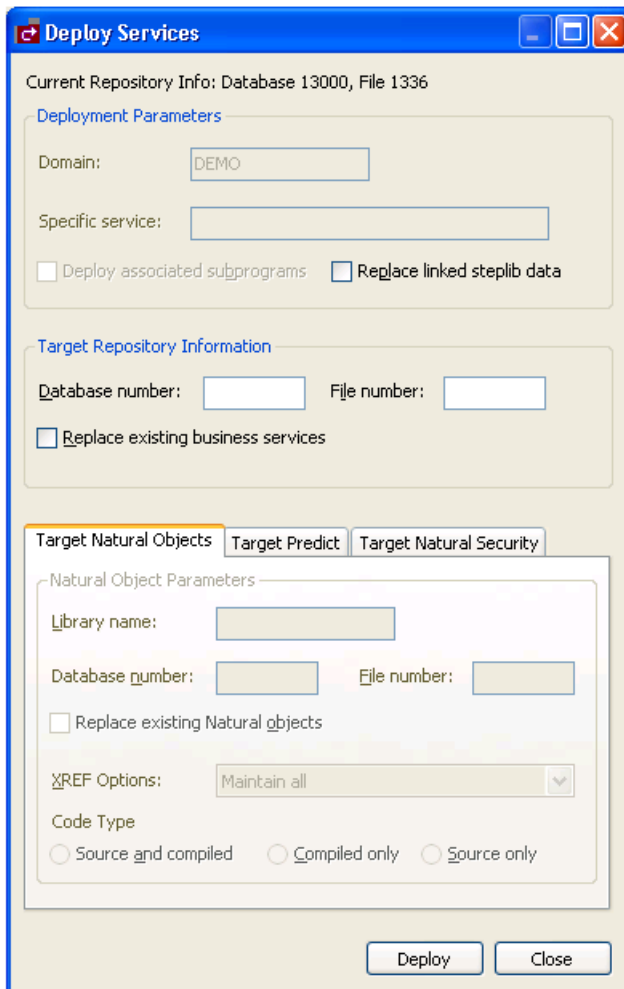
This window displays information about the current repository, as well as the name of the domain you selected. Optionally, you can use this window to:

Task	Procedure
Update the steplib data in the domain	See **Replace Linked Steplib Data**.
Replace the current business services in the domain with updated services	See **Replace Updated Business Services**.

> **Note:** If an option is left blank in this window, it is assumed that the option is the same in both the source and target environment. For example, if **Library name** is blank, the Natural modules in the source library will be deployed to the library with the same name in the target FUSER file.

3   Type the database and file number for the target repository in the appropriate fields.

Use the target repository database and file numbers for LFILE 136.

4   Select **Deploy**.

The **Deployment Results** window is displayed. For example:
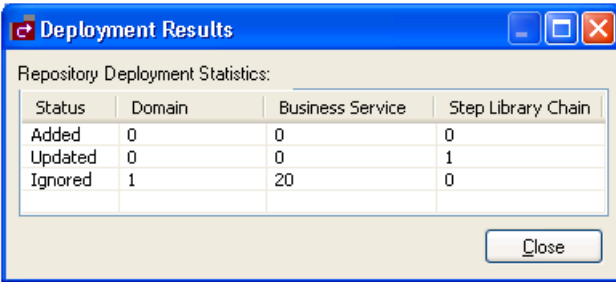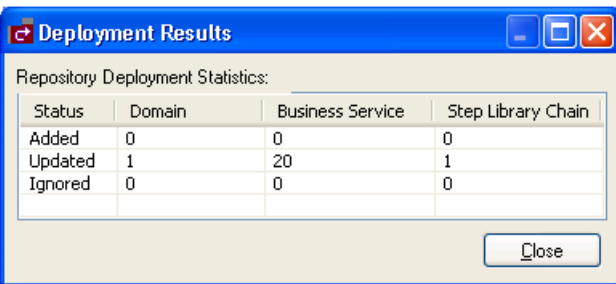

5   Select **Close**.

Replace Linked Steplib Data

▶ **To replace linked steplib data for a domain:**

1   Select **Replace linked steplib data** in the **Deploy Services** window.

2   Type the database and file number for the target repository in the appropriate fields.

Use the target repository database and file numbers for LFILE 136.

3   Select **Deploy**.

The **Deployment Results** window is displayed. For example:

4   Select **Close**.

Replace Updated Business Services

▶ **To replace updated business services in the domain:**

1   Select **Replace existing business services** in the **Deploy Services** window.

2   Type the database and file number for the target repository in the appropriate fields.

    Use the target repository database and file numbers for LFILE 136.

3   Select **Deploy**.

    The **Deployment Results** window is displayed. For example:

    Note: If the **Replace** option was not selected and the domain already exists in the target library, it will not be replaced.

4   Select **Close**.

Deploy a Business Service to Another Repository

You can deploy the business service information in the **Business Service** repository or, optionally, deploy the subprograms associated with the service to a Natural environment.

▶ **To deploy a business service to another repository:**

1   Open the context menu for the business service the **Business Services** explorer.

   For information, see **Access Business Services**.

2   Select **Deploy**.

   The **Deploy Services** window is displayed. For example:

This window displays the name of the domain and business service you selected, as well as information about the current repository. Optionally, you can use this window to:

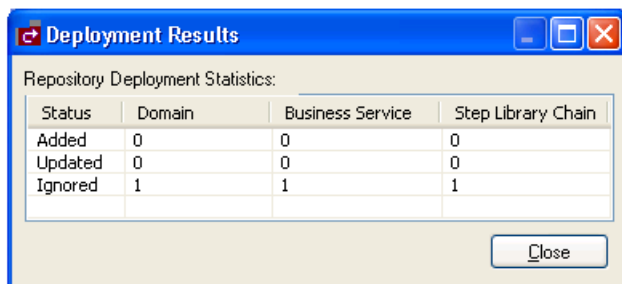| Task	Procedure
Deploy the subprograms associated with the business service to another environment	See **Deploy Associated Subprograms**.
Replace the linked steplib data in the domain	See **Replace Linked Steplib Data**.
Replace the current business service in the domain with the updated service	See **Replace Updated Business Service**. **Note:** If this option is not selected and the business service exists in the target library, it will not be replaced.

> **Note:** If an option is left blank in this window, it is assumed that the option is the same in both the source and target environment. For example, if **Library name** is blank, the Natural modules in the source library will be deployed to the library with the same name in the target FUSER file.

3   Type the database and file number for the target repository in the appropriate fields.

Use the target repository database and file numbers for LFILE 136.

4   Select **Deploy**.

The **Deployment Results** window is displayed. For example:

In this example, one domain, business service, and steplib chain were ignored. This indicates that the business service existed in the target repository and was not replaced.

> **Note:** You can deploy more than one service at a time if you do not select the **Deploy associated subprograms** button.

5   Select **Close**.

Deploy Associated Subprograms

You can deploy the subprograms associated with the business service to another environment.

Basically, a business service is defined by the methods it provides to users, the subprogram proxies that expose each of the methods, and the logical organization of libraries (steplibs) that host the Natural objects that implement the methods. This definition is stored in the NBS repository represented by LFILE 136. The repository also contains details about the name, version, and domain that uniquely identify each service, as well as a description of each method exposed by the service. Although the repository contains the service definitions, the Natural objects that implement the functionality are stored in steplibs within the FUSER or FNAT file.

In addition to the service definition, a service deployment may need to copy XREF information for each Natural object associated with the service. The XREF information is stored in the Predict FDIC file and the requirement to include the XREF information in the deployment is defined in the Natural Security FSEC file used in the target environment. Additionally, since the deployment function invokes the Natural SYSMAIN utility to copy the data to the target environment, the Password and Cipher options (for Target Predict and Target Natural Security information) were included to maintain consistency.

When you select **Deploy associated subprograms** in the **Deploy Services** window, the target setting tabs become active. For example:
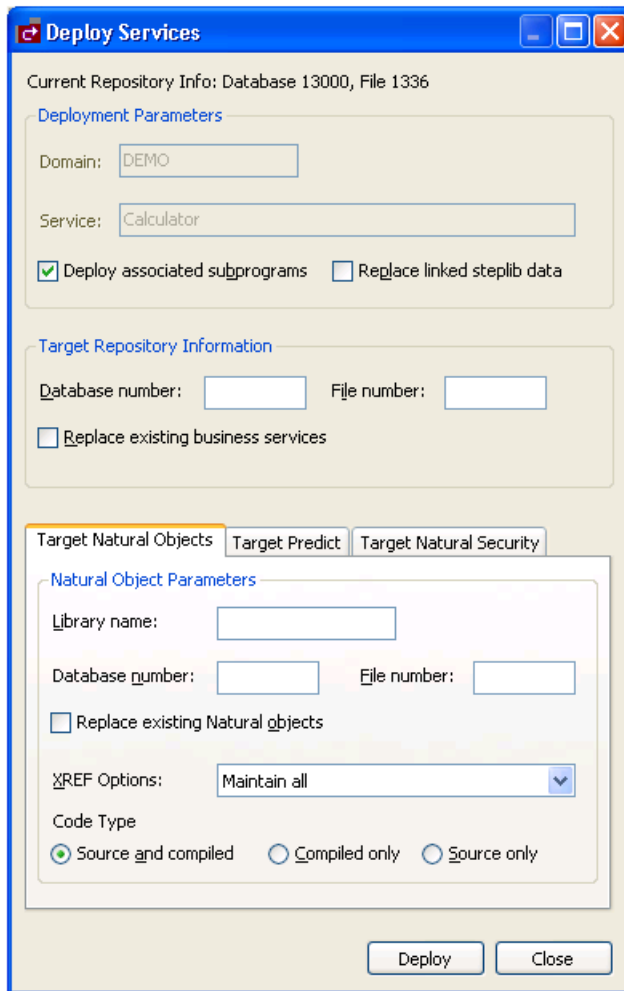
The tabs for the target settings are:

- Target Natural Objects
- Target Predict
- Target Natural Security

Target Natural Objects

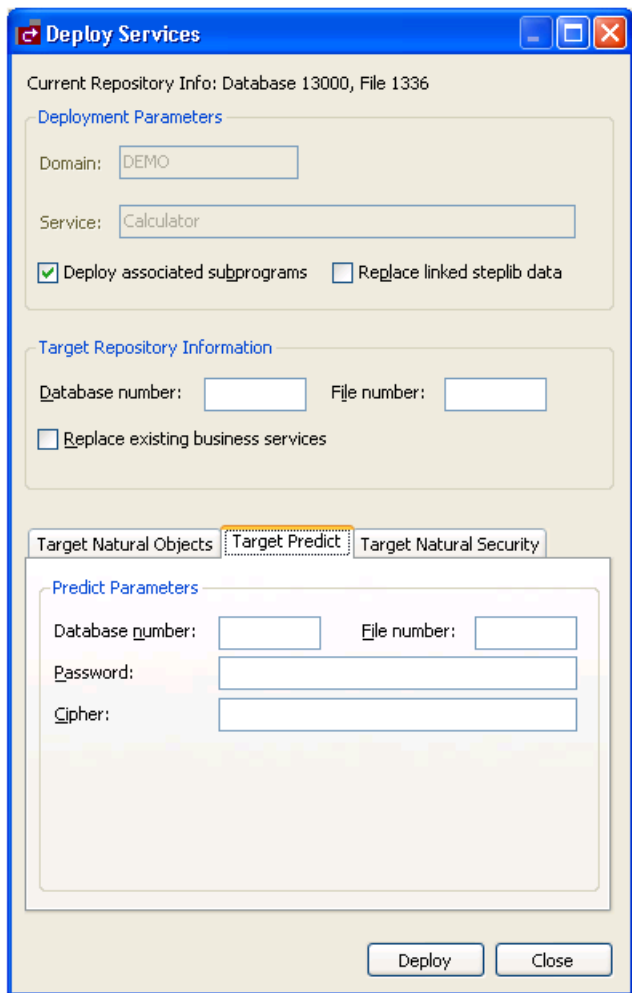Specify the following target settings for Natural objects:

Target Setting	Description
Library name	Name of the Natural library in which to copy all Natural objects that implement the business service being deployed.
Database number	Database ID for the FUSER (or FNAT) file that will store the Natural objects (subprograms, data areas, etc).
File number	File number for the FUSER (or FNAT) file that will store the Natural objects.

Optionally, you can use the **Target Natural Objects** tab to:

Task	Procedure
Replace existing Natural objects with the ones you are deploying	Select **Replace existing Natural objects**.
Maintain cross-reference data using an alternate method	Select another setting in **XREF options**. The cross-reference options are: ■ Maintain all ■ Do not maintain (except when deleting) ■ Maintain and document in Predict ■ Maintain whenever exists
Only deploy the object code for Natural objects	Select **Compiled only**.
Only deploy the source code for Natural objects	Select **Source only**.

Target Predict

The following example shows the settings on the **Target Predict** tab:

Specify the following target settings for Predict:

Target Setting	Description
Database number	Database ID for the FDIC file that will store the XREF data.
File number	File number for the FDIC file.
Password	Password for the FDIC file.
Cipher	Cipher key for the FDIC file.

Target Natural Security

The following example shows the settings on the **Target Natural Security** tab:

Specify the following target settings for Natural Security:

Target Setting	Description
Database number	Database ID for the FSEC file that contains the XREF requirements.
File number	File number for the FSEC file.
Password	Password for the FSEC file.
Cipher	Cipher key for the FSEC file.

Replace Linked Steplib Data

▶ **To replace the linked steplib data for the business service:**

1   Select **Replace linked steplib data** in the **Deploy Services** window.

2   Type the database and file number for the target repository in the appropriate fields.

Use the target repository database and file numbers for LFILE 136.

3   Select **Deploy**.

The **Deployment Results** window is displayed. For example:

In this example, the **Replace updated business service** option was not selected and the domain and business service were not replaced.
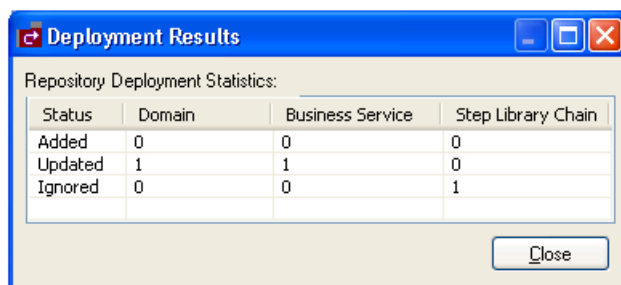
4   Select **Close**.

Replace Updated Business Service

▶ **To replace the existing business service with an updated version:**

1   Select **Replace existing business services** in the **Deploy Services** window.

2   Type the database and file number for the target repository in the appropriate fields.

Use the target repository database and file numbers for LFILE 136.

3   Select **Deploy**.

The **Deployment Results** window is displayed. For example:

In this example, the **Replace linked steplib data** option was not selected and the linked steplib data was not replaced.
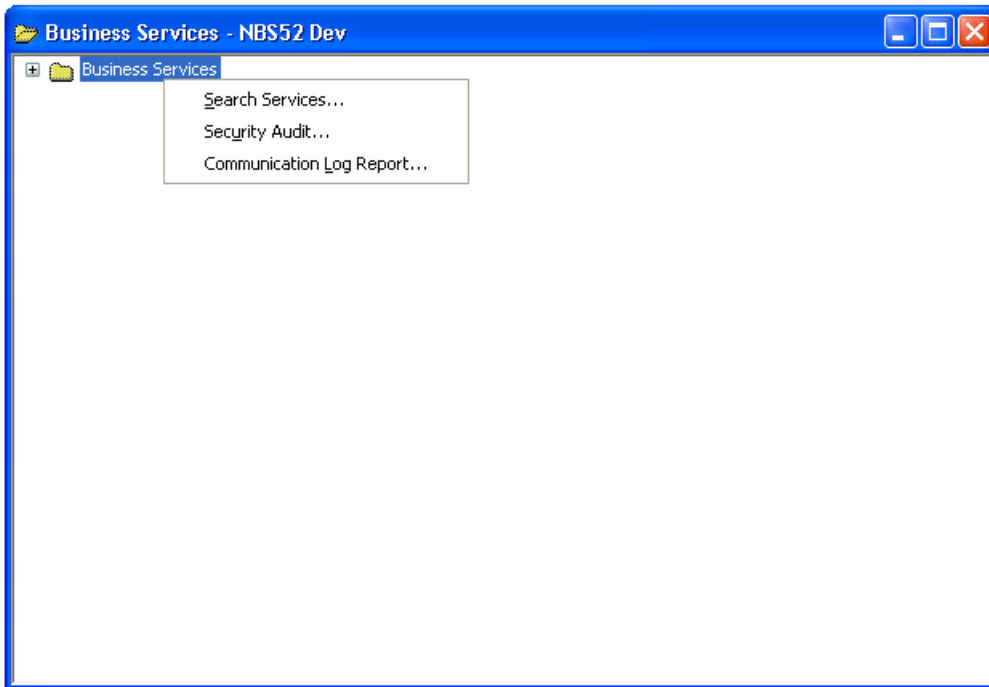
4   Select **Close**.

Access the Communication Log Report

A connection may have problems communicating with the server. If so, you can access the Communication Log report and view the status and error messages for the connection.

▶ **To access the Communication Log report:**
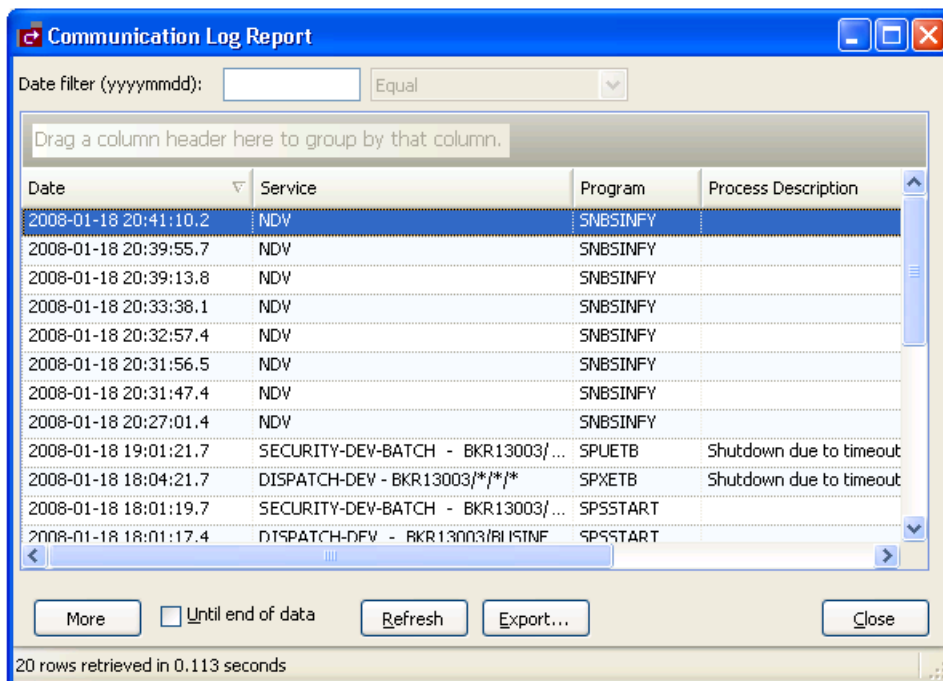
1   Open the context menu for the connection in the **Business Services** explorer.

For example:

2   Select **Communication log report**.

The **Communication Log Report** window is displayed. For example:

This window shows the first 20 communication status and error messages in descending order by date (the default). To sort by another column, drag that column header to the area indicated.

> **Note:** All services that do not use the NDV (Natural Development Server) use the EXX (EntireX) middleware.

Optionally, you can use this window to:

Task	Procedure
Restrict the list of messages displayed	Type the date and select the search criteria. For example, to display messages for January 11, 2008, enter "20080111" in **Date**, select **Equal**, and then select **Refresh**.
Display the next 20 messages	Select **More**.
Display all messages	Select **Until end of data** and then select **More**.
Export the log report to a spreadsheet (for example, Microsoft Excel)	Select **Export**. A window is displayed to select the file location and enter a file name.

3   Select **Close** to close the window.

> **Note:** This log is also available on the server by logging onto SYSBIZ, entering "MENU SA MS" on the command line, and then pressing PF4.

Generate a Natural Client

You can generate a Natural client for a business service. A Natural client is a subprogram proxy that allows a business service to be consumed in a Natural environment.
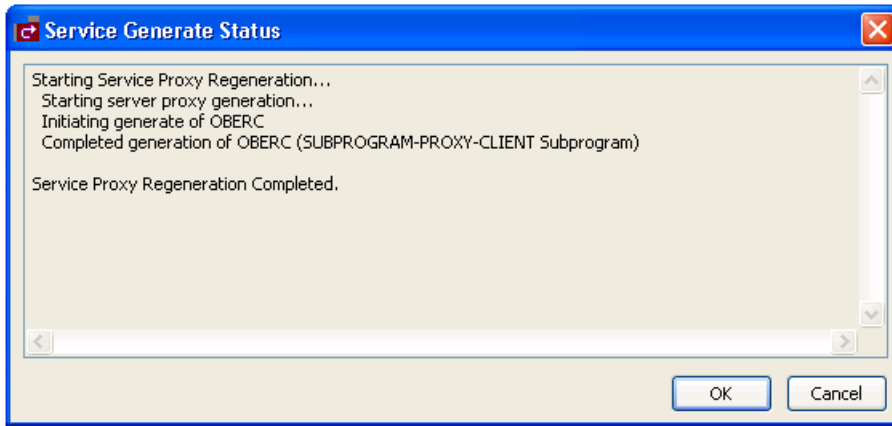
▶ **To generate a Natural client:**

1   Open the context menu for the business service in the **Business Service** repository.

> **Note:** For information on the **Business Service** repository, see **Using the Business Service Repository**.

2   Select **Generate Natural Client**.

The **Service Generate Status** window is displayed. For example:

3   Select **OK**.

To view the generated subprogram proxy client, open the subprogram in the program editor. For example: