Natural Business Services

Natural Business Services Eclipse Plug-in

Version 8.2.1

November 2013

Natural Business Services

This document applies to Natural Business Services Version 8.2.1.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: NBS-ECLIPSEPLUGIN-821-20131119

Table of Contents

Preface

Natural Business Services Eclipse Plug-in describes the Natural Business Services plug-in for Eclipse.

This documentation is intended for developers who are familiar with Eclipse and want to use the Natural Business Services Eclipse plug-in to create Web services and/or client proxy classes to access business services created in Natural.

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

Introduction

Describes the Natural Business Services Eclipse plug-in, how to install the plug-in, and how to verify the installation.

Using the Eclipse Plug-in

Describes the tasks you can perform when using the Eclipse plug-in. These tasks include:

- Configure connections
- Use the **NBS Repositories** view
- Transform a browse module into object-browse modules
- Create a directly enabled Web service
- Create a REQUEST DOCUMENT client for Natural
- Check environment information for a connection
- Add repository metadata to CentraSite
- Set Natural Business Services preferences, such as which Java Virtual Machine (VM) version to use
- Set Web service stack preferences for the WS-Stack plug-in, such as which deployment connection to use

Developing Business Services

Describes how to use the Eclipse plug-in to create and maintain business services. These procedures include:

- Create a business service
- Regenerate a business service
- Regenerate the service proxies
- Test a business service method
- Test a business service
- Delete a business service
- Edit a service definition
- Edit service modules

Using the Custom Ant Tasks	Describes the Ant tasks supplied with Natural Business Services, the prerequisites for using the tasks, and details about the attributes and child nodes. It includes an example of using the tasks.
Administering Business Services	Describes how to use the Eclipse plug-in to administer business services. These procedures include: ■ Configure the Eclipse plug-in (create and maintain domain definitions and steplib configurations) ■ Define security for domains ■ Audit business services ■ Deploy business services to another repository or environment ■ Access the Communication Log report ■ Generate a Natural client
Developing Client Proxy Classes and Web Services	Describes how to use the Eclipse plug-in to create a client proxy class and Web service. Client proxy classes provide access to business services running in Natural. These procedures include: ■ Generate a client proxy class ■ Customize a generated client proxy class ■ Use the generated client proxy classes ■ Generate log4J log statements

1 Introduction

Use the Eclipse plug-in to:

Task	Description
Create and maintain business services.	See *Developing Business Services*.
Access the service repository.	See *Use the NBS Repositories View*.
Generate Web services to access Natural business services.	See *Deploy the Client Proxy Class as a Web Service Class*.
Generate client proxy classes to access Natural business services.	See *Developing Client Proxy Classes and Web Services*.
View and edit all required files.	See *Use the NBS Repositories View*.

Installation

During the Natural Business Services (NBS) Windows installation procedure, the Eclipse plug-in is copied to the default NBS installation folder. If desired, you can change the default location and install the plug-in in your Eclipse installation folder. This allows the plug-in to be automatically available in Eclipse.

Note: For information on installing the Eclipse plug-in on Windows, see *Natural Business Services Installation on Windows*.

This section covers the following topics:

- Install Plug-in for Eclipse Europa (3.3)
- Install Plug-in for Eclipse Ganymede (3.4)
- Activate the Eclipse Plug-in

Install Plug-in for Eclipse Europa (3.3)

After the Eclipse plug-in has been copied to the default NBS installation folder, you can install the plug-in on Eclipse.

To install the plug-in on Eclipse Europa:

1 Start Eclipse.

2 Select **Software Updates > Find and Install** on the **Help** menu.

3 Select **Search for New Features to install**.

4 Select **Next**.

5 Select **New Archived Site**.

6 Select the path for the NBS 82 archive site.

By default, this path is *C:\Program Files\Software AG\Natural Business Services\V8.2\EclipsePlugin\Updates\com.softwareag.nbs-82.zip*.

7 Provide a site name.

8 Select **OK**.

9 Ensure that the new site is selected.

10 Select **Finish**.

The installation wizard is displayed.

11 Install the plug-in.

12 Restart Eclipse.

Install Plug-in for Eclipse Ganymede (3.4)

After the Eclipse plug-in has been copied to the default NBS installation folder, you can install the plug-in on Eclipse.

▶ To install the plug-in on Eclipse Ganymede:

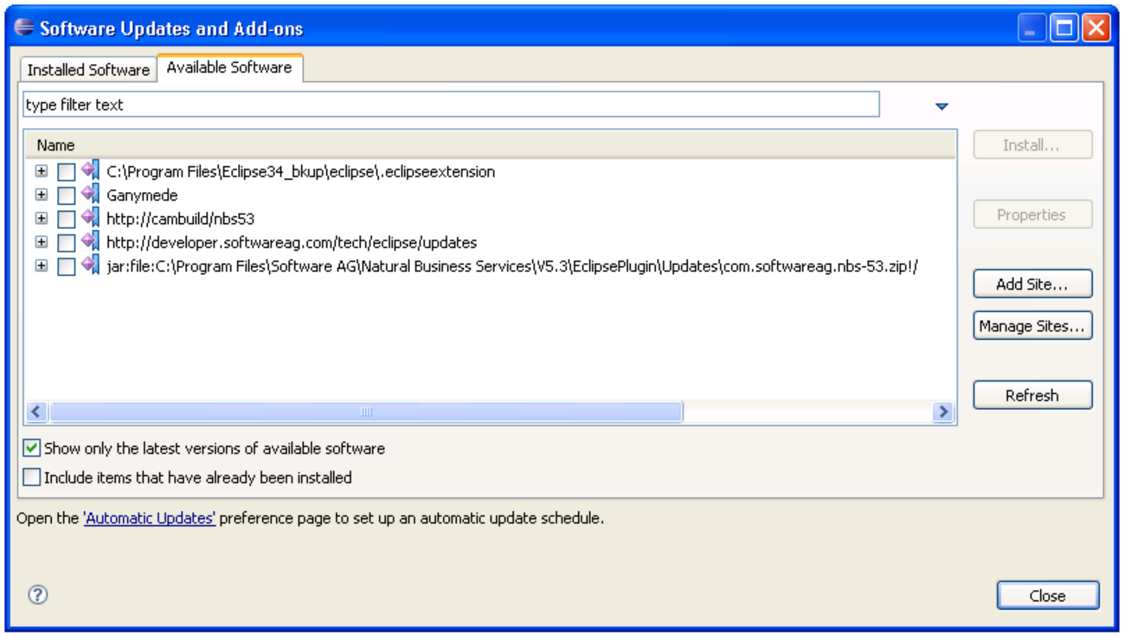
1 Start Eclipse.

2 Select **Software Updates** on the **Help** menu.

The **Software Updates and Add-ons** panel is displayed.

3 Select the **Available Software** tab.

The available software is listed. For example:



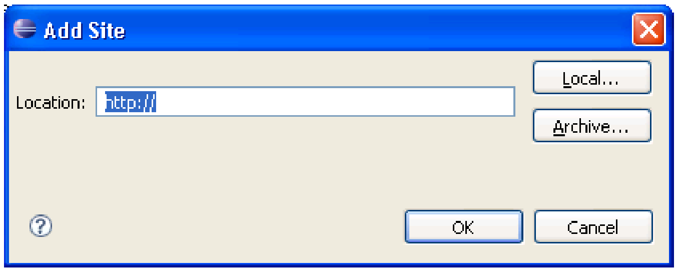
C:\Program Files\Eclipse34_bkup\eclipse\.eclipseextension
Ganymede
http://cambuild/nbs53
http://developer.softwareag.com/tech/eclipse/updates
jar:file:C:\Program Files\Software AG\Natural Business Services\V5.3\EclipsePlugin\Updates\com.softwareag.nbs-53.zip!/

Show only the latest versions of available software [✓]
Include items that have already been installed

Open the 'Automatic Updates' preference page to set up an automatic update schedule.

4 Select **Add Site**.

The **Add Site** window is displayed. For example:



Location: http://

5 Select **Archive**.

The **Repository Archive** window is displayed.

6 Select the path for the NBS 82 archive site.

By default, this path is *C:\Program Files\Software AG\Natural Business Services\V8.2\EclipsePlugin\Updates\com.softwareag.nbs-82.zip*.

7 Provide a Site name.

8 Select **OK**.

The **Software Updates and Add-ons** panel is redisplayed.

9 Select the features you want to install.

We recommend that you select all features.

10 Select **Install**.

The installation wizard is displayed.

11 Install the plug-in.

12 Restart Eclipse.

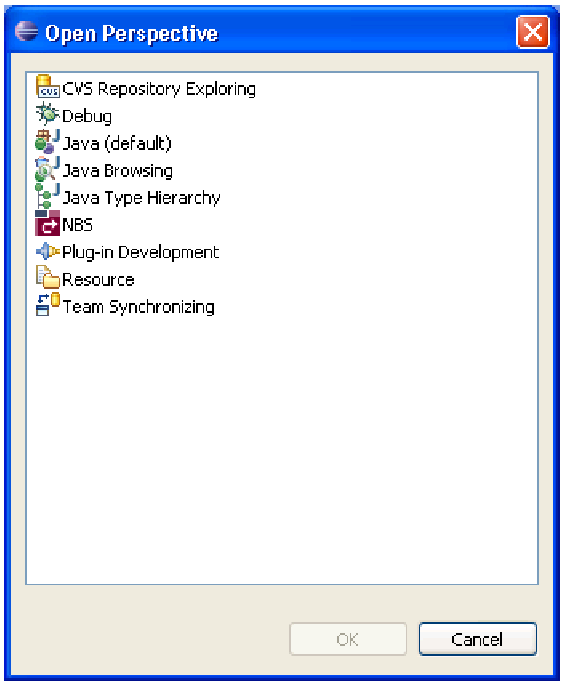
Activate the Eclipse Plug-in

This section describes how to activate the Eclipse plug-in for Natural Business Services (NBS). To help organize the development environment, an NBS perspective is provided.

To activate the Eclipse plug-in:

1 Select **Open Perspective > Other** on the **Window** menu.

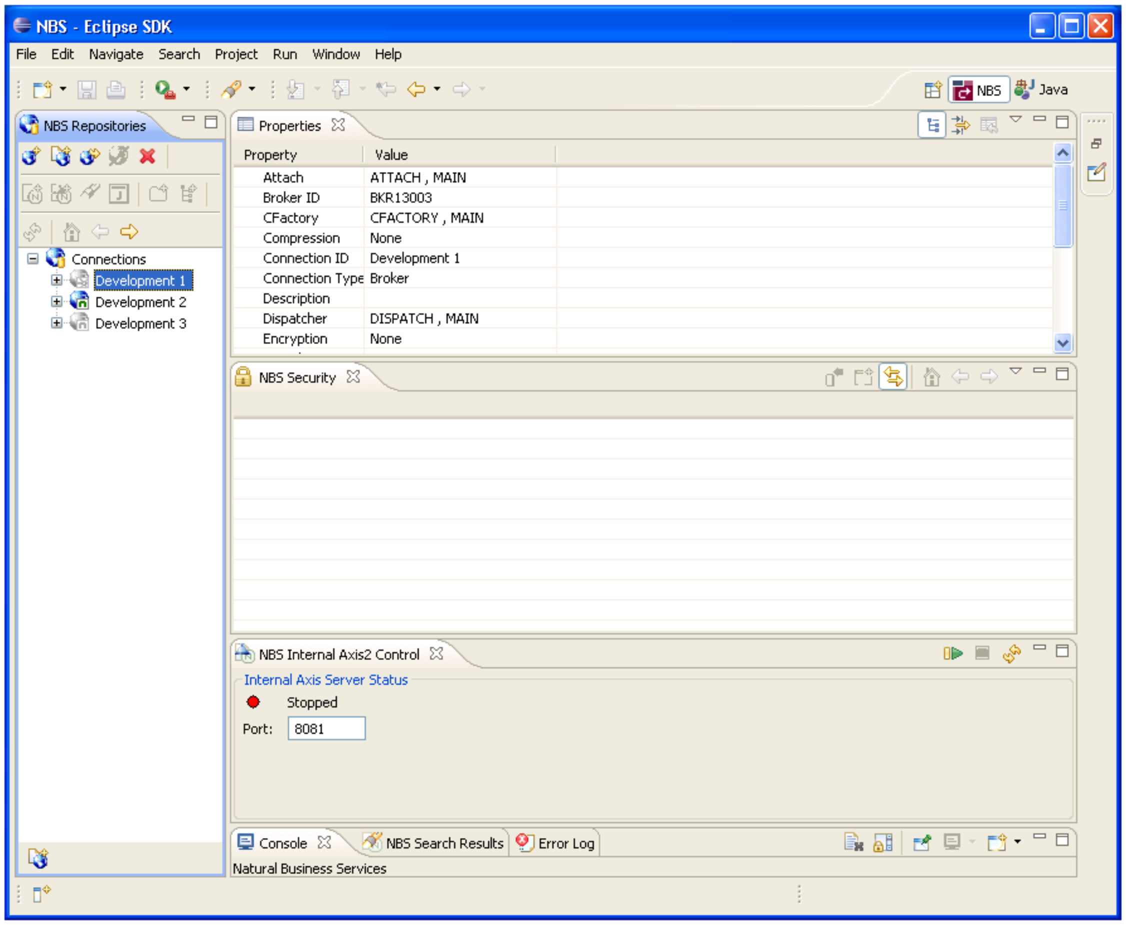
The **Open Perspective** window is displayed. For example:



2 Select **NBS**.

3 Select **OK**.

The **NBS - Eclipse SDK** is displayed, showing the standard views defined for the NBS perspective. The **NBS Repositories** view lists the available connections. To view the settings defined for a connection, select the connection in the **NBS Repositories** view. The settings are displayed in the **Properties** view. For example:



Property	Value
Attach	ATTACH , MAIN
Broker ID	BKR13003
CFactory	CFACTORY , MAIN
Compression	None
Connection ID	Development 1
Connection Type	Broker
Description	
Dispatcher	DISPATCH , MAIN
Encryption	None

Internal Axis Server Status
Stopped
Port: 8081

Natural Business Services

You can customize the NBS perspective as desired. The next time you start Eclipse, your NBS perspective will be loaded.

Note: For information on configuring connections, see *Configure Connections*.

2 Using the Eclipse Plug-in

Configure Connections

Before you can use the Natural Business Services Eclipse plug-in, you must configure connections to a remote repository. Since the Natural code (subprograms) used during development may be located in a different Natural environment than what is used during production (runtime), you will define more than one connection. The location of the Natural code is determined by the configuration settings for the connection.

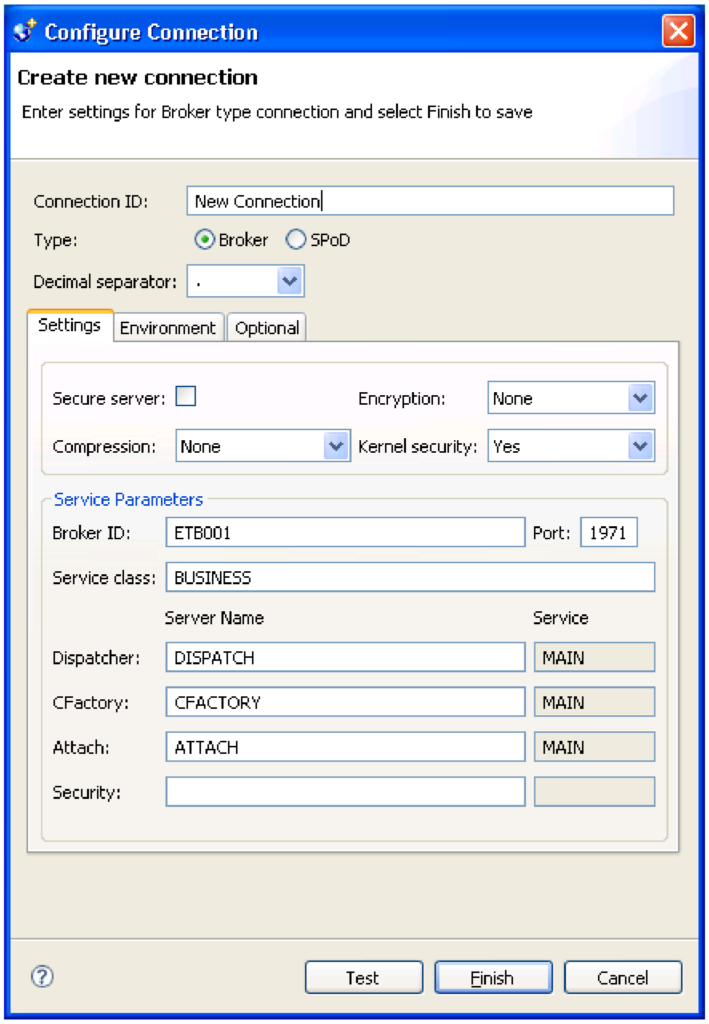
Tip: To make future Eclipse plug-in upgrades easier, create your configuration file in a neutral location. To do this, select **Create new configuration** in the **Preferences** window. For information, see *Set Natural Business Service Preferences*.

To create a new connection:

1 Open the context menu for Connections in the **NBS Repositories** view.

2 Select **New**.

The **Create new connection** panel is displayed. For example:


Configure Connection

Create new connection

Enter settings for Broker type connection and select Finish to save

Connection ID: New Connection

Type: [✓ Broker] SPoD

Decimal separator: .

Settings | Environment | Optional

Secure server: ☐ Encryption: None

Compression: None Kernel security: Yes

Service Parameters

Broker ID: ETB001 Port: 1971

Service class: BUSINESS

	Server Name	Service
Dispatcher:	DISPATCH	MAIN
CFactory:	CFACTORY	MAIN
Attach:	ATTACH	MAIN
Security:		

Use this panel to configure connections to the remote repository containing your business services. You can create a new Broker or SPoD connection, as well as specify the separator to use for decimal numbers. This example shows a Broker connection.

This section covers the following topics:

- Add a Broker Connection
- Add a SPoD Connection
- Copy an Existing Connection
- Modify a Connection
- Test a Connection

- Delete a Connection

Add a Broker Connection

A Broker connection uses the EntireX communication protocol to connect to Natural Business Services servers running in Natural. This type of connection is suitable for administering and running business service applications.

Note: You must use a SPoD connection to create, test, or deploy business services.

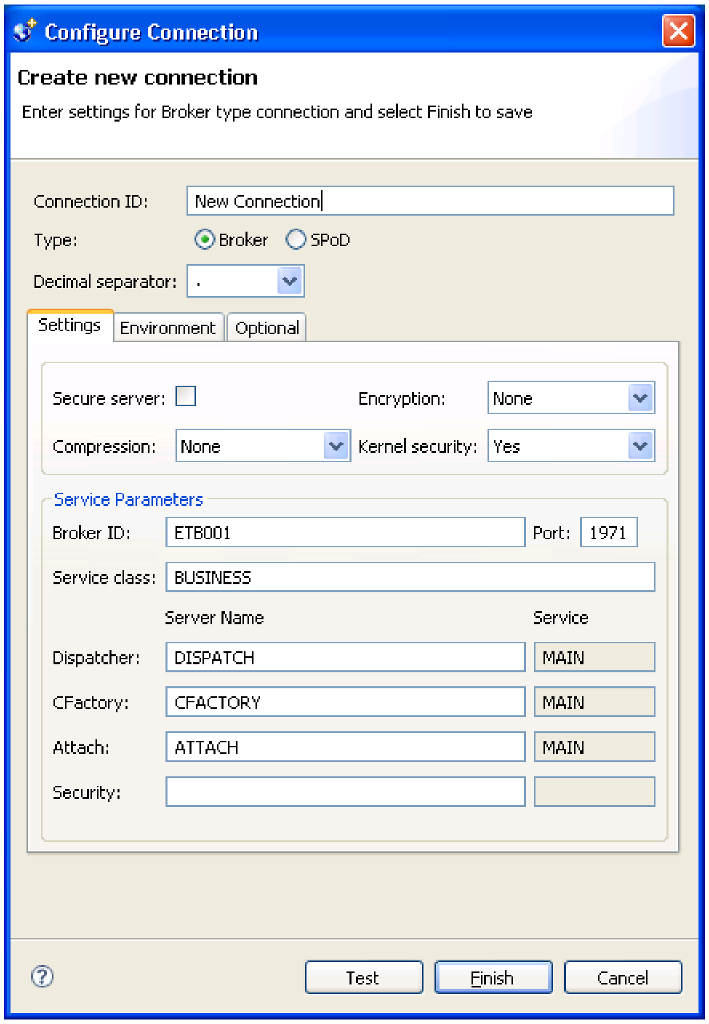
Tip: If you have an existing connection that is similar to the one you want to add, you can copy the connection and then modify it to suit your requirements. For information, see *Copy an Existing Connection*.

To add a Broker connection:

1 Open the context menu for **Connections** in the **NBS Repositories** view.

2 Select **New**.

The **Create new connection** panel is displayed. For example:



Create new connection

Enter settings for Broker type connection and select Finish to save

Connection ID: New Connection

Type: [✓ Broker] SPoD

Decimal separator: .

Secure server: ☐

Encryption: None

Compression: None

Kernel security: Yes

Service Parameters

Broker ID: ETB001

Port: 1971

Service class: BUSINESS

	Server Name	Service
Dispatcher:	DISPATCH	MAIN
CFactory:	CFACTORY	MAIN
Attach:	ATTACH	MAIN
Security:		

The **Create new connection** panel for a Broker connection contains the following tabs:

Panel Tab	Description
Settings	Defines connections to remote repositories (via the EntireX Communicator or Natural for Windows protocols). Use the **Settings** tab to view, edit, delete, or add connections. These connections define communication paths and options to remote business service repositories running on a Natural server.
Environment	Defines configuration settings for the environment. This tab is only available for a Broker connection. For information, see *Environment Tab*.
Optional	Defines optional configuration settings. This tab is only available for a Broker connection. For information, see *Optional Tab*.

3 Type the name of the connection in **Connection ID**.

This name will be listed in the **NBS Repositories** view and used in code when creating connections to the business services.

4 Select **Broker** in **Type**.

5 Type the broker ID in **Broker ID**.

6 Type the TCP/IP port number the EntireX broker is running on in **Port**.

Note: If you intend to define the port in your Windows services file, leave this option blank.

Optionally, you can specify the following information:

Option	Description
Secure server	Select this option if the server is running Natural or Natural Business Services security and a password must be transported in the communication buffer. **Note:** When you select this option, also select **Encryption**.
Encryption	Select this option to encrypt messages transported to and from the server. This value is based on the EntireX ACI settings for encryption.
Compression	Select this option to compress messages at a specified level. This value is based on the EntireX ACI settings for compression. **Note:** The higher the level, the more CPU processing is required to compress and decompress the message. As this may end up saving a lot of network bandwidth, it is worth the CPU trade off in most instances.
Kernel security	Select this option to use Broker security for the EntireX server. If you are not sure which security is used, you can leave this setting on **Auto**. To achieve the best network performance, however, you should select **Yes** or **No**.
Server class	EntireX server class defined for your business services. If you do not know this value, ask your Natural Business Services administrator.
Dispatcher, CFactory, Attach, and Security	Timestamp server names of the four Natural Business Services runtime servers. Leave these values as the defaults; only your Natural Business Services administrator should change these values. **Note:** The Security server name is optional and is defaulted to blank.

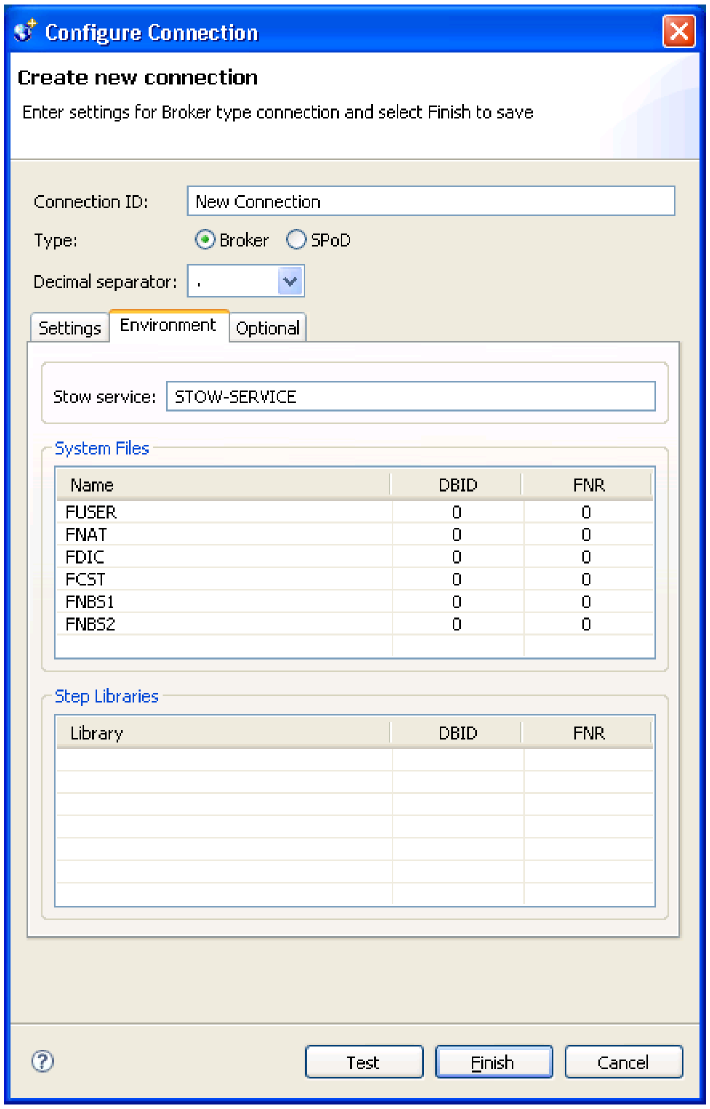
You can also define settings on the **Environment** and **Optional** tabs. For information, see *Environment Tab* and *Optional Tab*.

Note: To test the new connection, select **Test**. For more information, see *Test a Connection*.

7 Select **Finish** to create the new Broker connection.

Environment Tab

This tab contains the System Files and Step Libraries tables. For example:



Create new connection

Enter settings for Broker type connection and select Finish to save

Connection ID: New Connection

Type: [✓ Broker] SPoD

Decimal separator: .

Stow service: STOW-SERVICE

System Files

Name	DBID	FNR
FUSER	0	0
FNAT	0	0
FDIC	0	0
FCST	0	0
FNBS1	0	0
FNBS2	0	0

Step Libraries

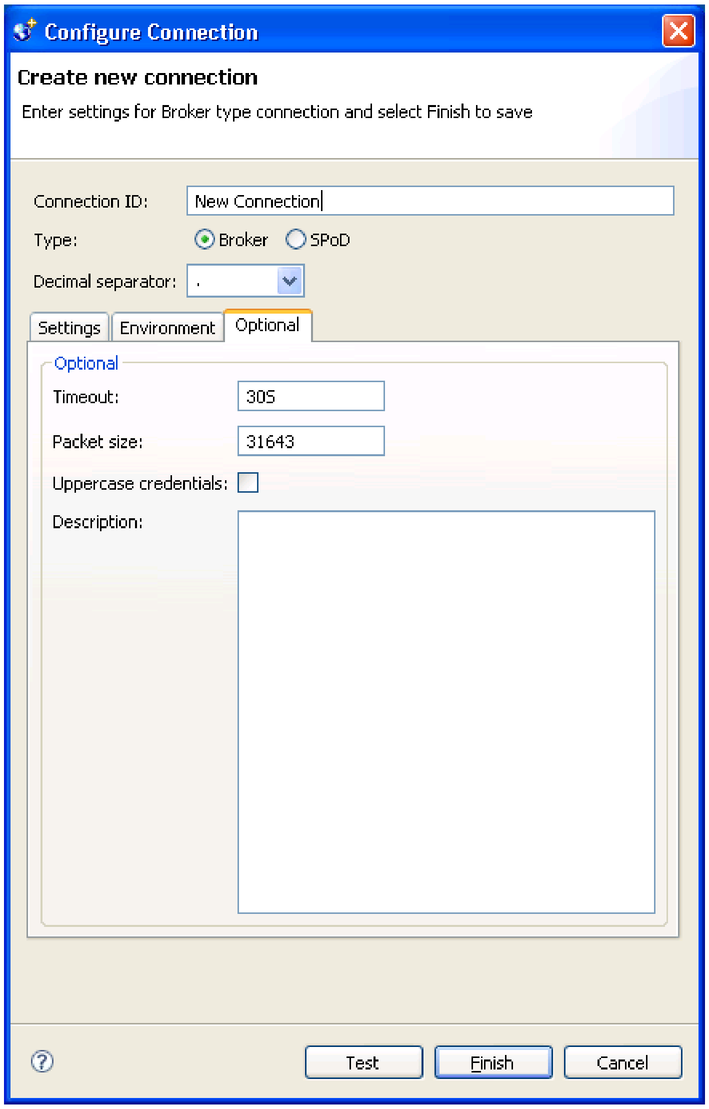
Library	DBID	FNR

To override the current defaults used in the Natural Business Services server environment, supply the connection values in the appropriate table.

Note: Unless you have a compelling reason to override the defaults, we recommend that you do not change these settings.

Optional Tab

This tab contains optional settings for a Broker connection. For example:


Configure Connection
Create new connection
Enter settings for Broker type connection and select Finish to save

Connection ID: New Connection

Type: [✓ Broker] SPoD

Decimal separator: .

Timeout: 30S

Packet size: 31643

Uppercase credentials: ☐

Description:

The optional settings for a Broker connection are:

Option	Description
Timeout	Timeout value used by the Natural Business Services dispatcher when making remote service calls. The default is "30S" (30 seconds).
Packet size	Default packet size for communication messages. If you have many business services using interfaces larger than the default (31643 bytes), increase the size of this field. **Note:** If required, Natural Business Services will increase the size dynamically at runtime, but this will affect performance.
Uppercase credentials	Select this option if the user ID and password must be sent in upper case. **Note:** This option is not recommended for Natural Business Services server environments running on Unix or Linux.
Description	Brief description of the connection.

Add a SPoD Connection

A SPoD connection uses the Natural Single Point of Development (SPoD) protocol to communicate with the Natural Development Server (NDV) running in a Natural environment that contains Natural Business Services. This type of connection is only suitable for developing and testing Natural Business Services applications.

Tip: If you have an existing connection that is similar to the one you want to add, you can copy the connection and then modify it to suit your requirements. For information, see *Copy an Existing Connection*.

To add a SPoD connection:

1 Open the context menu for **Connections** in the **NBS Repositories** view.

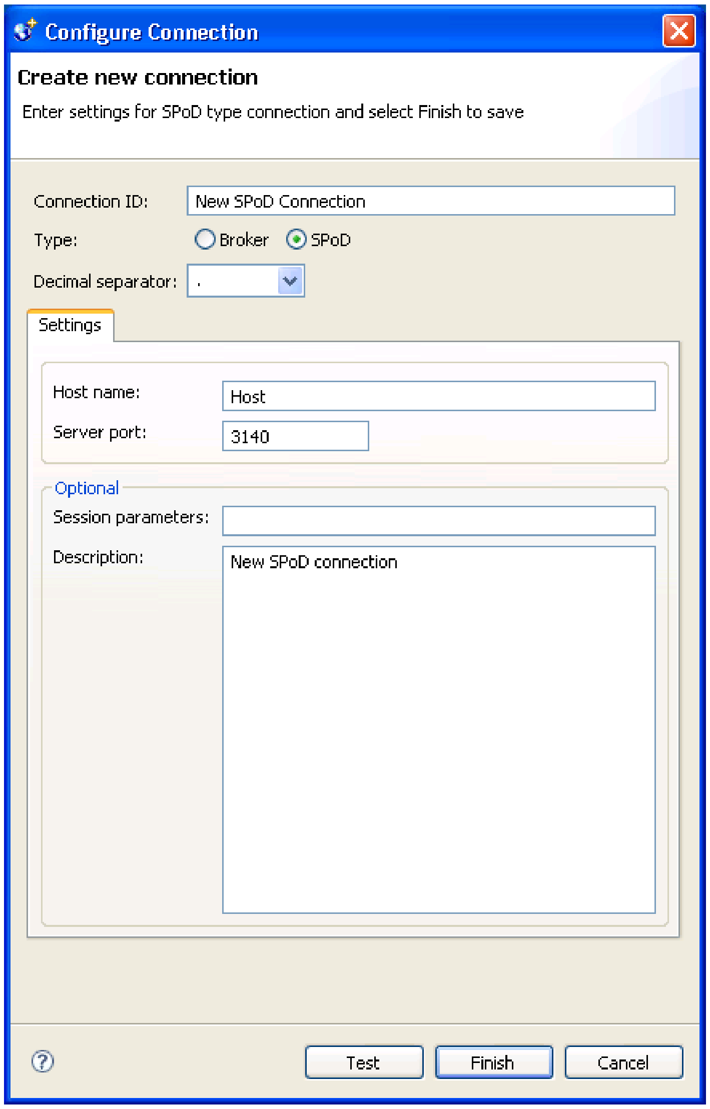
2 Select **New**.

3 Type the name of the connection in **Connection ID**.

The connection ID identifies the server component that provides communications between the client and server. This name is listed in the **NBS Repositories** view and used in code when creating connections to business services. All communications are done by connection ID.

4 Select **SPoD** in **Type**.

The **Settings** tab for a SPoD connection is displayed. For example:



Create new connection

Enter settings for SPoD type connection and select Finish to save

Connection ID: New SPoD Connection

Type: ( ) Broker (●) SPoD

Decimal separator: .

Settings

Host name: Host

Server port: 3140

Optional

Session parameters:

Description: New SPoD connection

5 Specify the following information:

Option	Description
Host name	Name of the server running Natural Development Server (NDV). You may also enter a TCP/IP address.
Server port	TCP/IP port for the NDV server. This value is required. **Note:** You cannot use the Windows services file to look up this value.
Session parameters	Any session parameters required to connect to your environment. This can be a profile setting, such as `PROFILE=NBS82`.
Description	Brief description of the connection.

Note: To test the new connection, select **Test**. For more information, see *Test a Connection*.

6 Select **Finish** to create the new SPoD connection.

The connection is added to the list of available connections.

Copy an Existing Connection

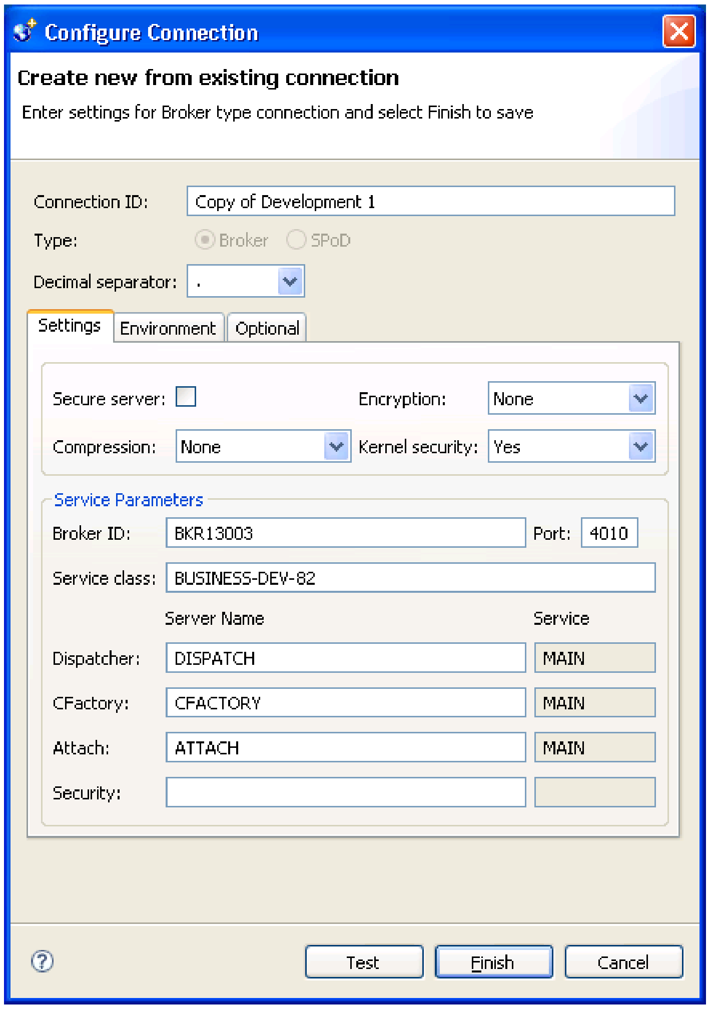
To save time and effort, you can copy an existing connection (Broker or SPoD) and then modify the settings for your requirements.

To copy an existing connection:

1 Open the context menu for an existing connection in the **NBS Repositories** view.

2 Select **New**.

The settings for the existing connection are displayed. For example:



Create new from existing connection

Enter settings for Broker type connection and select Finish to save

Connection ID:	Copy of Development 1
Type:	[✓ Broker] SPoD
Decimal separator:	.

Secure server:	☐	Encryption:	None
Compression:	None	Kernel security:	Yes

Service Parameters

Broker ID:	BKR13003	Port:	4010
Service class:	BUSINESS-DEV-82		

	Server Name	Service
Dispatcher:	DISPATCH	MAIN
CFactory:	CFACTORY	MAIN
Attach:	ATTACH	MAIN
Security:		

3 Change the settings as required.

For example, change the connection name from Copy of `ConnectionName` to another name.

Note: To test the new connection, select **Test**. For more information, see *Test a Connection*.

4 Select **Finish** to save the new connection.

The connection is added to the list of available connections in the **NBS Repositories** view.

Modify a Connection

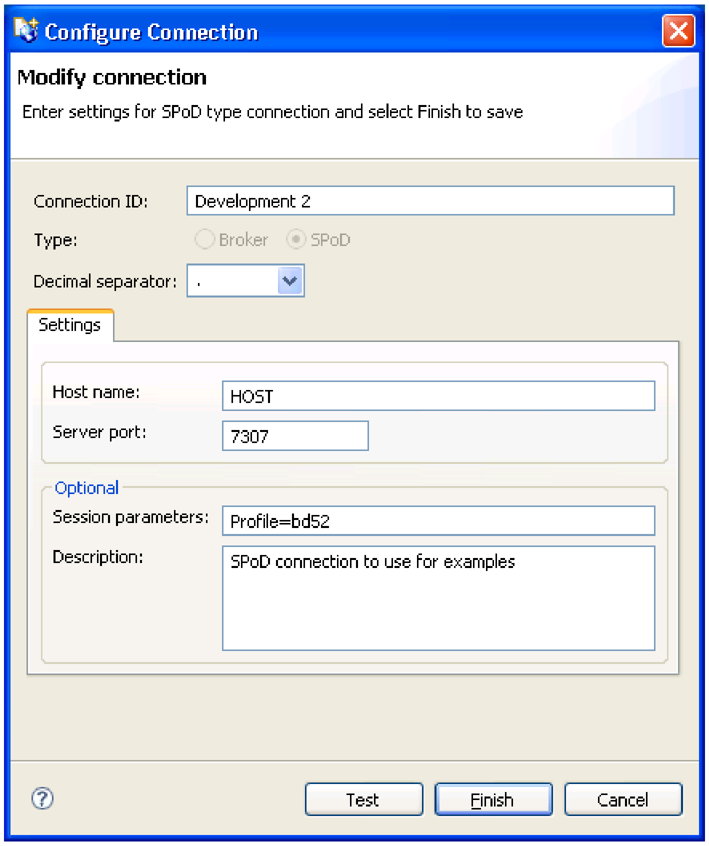
To modify a connection:

1 Open the context menu for the connection in the **NBS Repositories** view.

2 Select **Configure**.

Tip: If **Configure** is not available on the context menu, select **Disconnect**. You cannot modify the settings for a connection that is currently connected.

The **Modify connection** panel is displayed. For example:



Modify connection

Enter settings for SPoD type connection and select Finish to save

Connection ID: Development 2

Type: Broker [✓ SPoD]

Decimal separator: .

Host name: HOST

Server port: 7307

Session parameters: Profile=bd52

Description: SPoD connection to use for examples

This example shows the settings for a SPoD connection.

3 Modify the settings as desired.

For information on the configuration settings, see:

- *Add a Broker Connection*
- *Add a SPoD Connection*

Note: To test the modified connection, select **Test**. For more information, see *Test a Connection*.

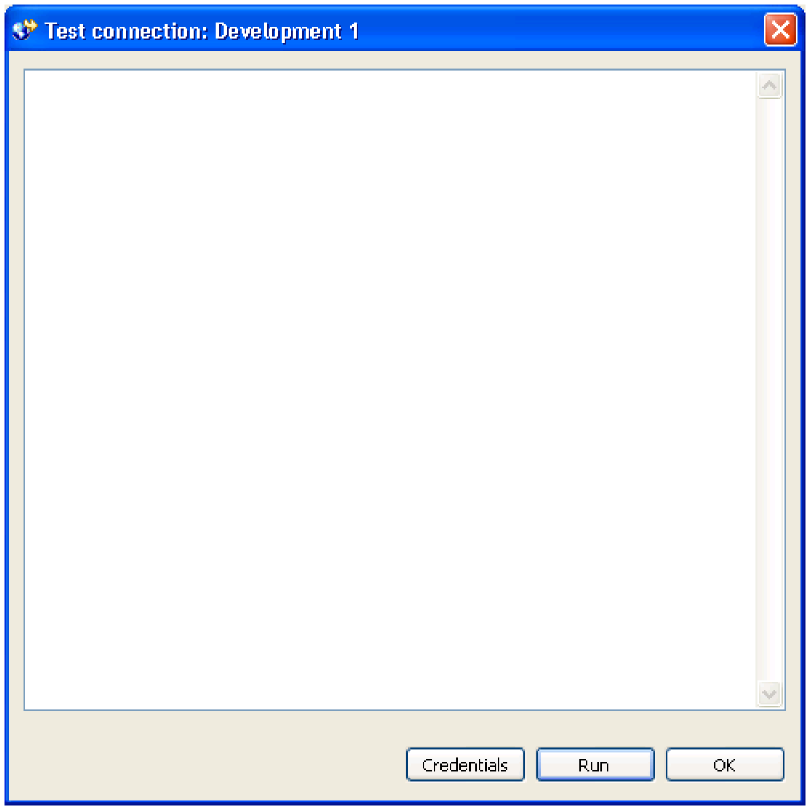
4 Select **Finish** to save the new settings.

Test a Connection

To test a connection:

1 Select **Test** on any Connection wizard panel.

The Test connection window is displayed for the selected connection. For example:



2 Select **Run** to test the connection.

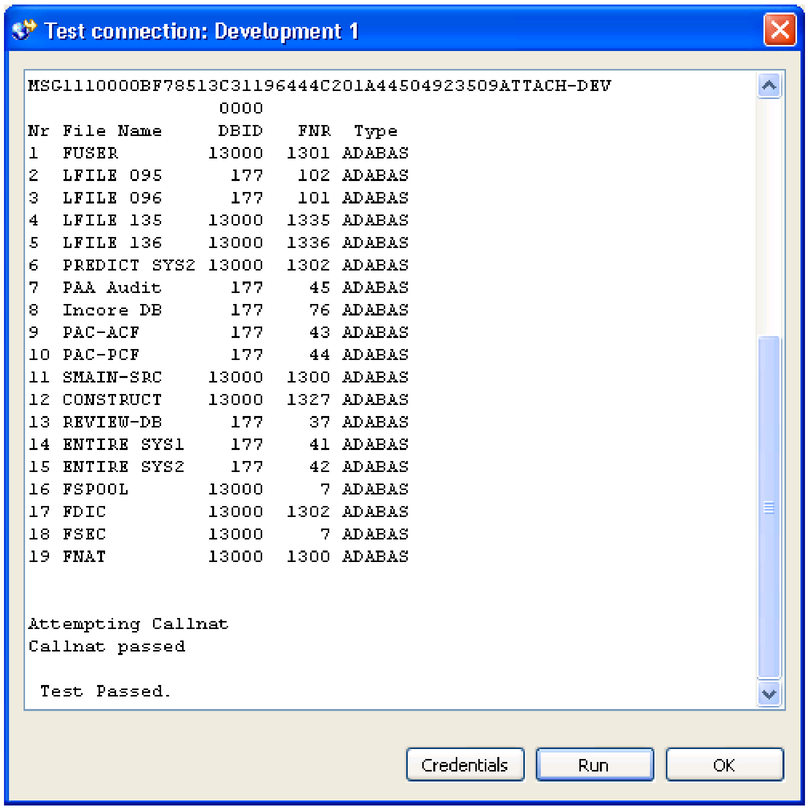
The **Enter User's Credentials** window is displayed.

For a description of this window, see *Access Business Services*.

3 Type the user ID and password for the connection.

4 Select **OK**.

The results of the test are displayed. For example:



Test connection: Development 1

```
MSG1110000BF78513C31196444C201A44504923509ATTACH-DEV
                 0000
Nr File Name      DBID   FNR  Type
1  FUSER         13000  1301 ADABAS
2  LFILE 095       177   102 ADABAS
3  LFILE 096       177   101 ADABAS
4  LFILE 135     13000  1335 ADABAS
5  LFILE 136     13000  1336 ADABAS
6  PREDICT SYS2  13000  1302 ADABAS
7  PAA Audit       177    45 ADABAS
8  Incore DB       177    76 ADABAS
9  PAC-ACF         177    43 ADABAS
10 PAC-PCF         177    44 ADABAS
11 SMAIN-SRC     13000  1300 ADABAS
12 CONSTRUCT     13000  1327 ADABAS
13 REVIEW-DB       177    37 ADABAS
14 ENTIRE SYS1     177    41 ADABAS
15 ENTIRE SYS2     177    42 ADABAS
16 FSPOOL        13000     7 ADABAS
17 FDIC          13000  1302 ADABAS
18 FSEC          13000     7 ADABAS
19 FNAT          13000  1300 ADABAS


Attempting Callnat
Callnat passed

 Test Passed.
```

5 Select **OK** to close the test window.

Delete a Connection

To delete a connection:

1 Open the context menu for the connection in the **NBS Repositories** view.

2 Select **Delete**.

A confirmation window is displayed.

3 Select **OK** to delete the connection.

The connection is removed from the list of available connections.

Use the NBS Repositories View

The **NBS Repositories** view lists all defined business service connections in a tree view. You can expand each connection node to display information about each connection. If the Eclipse plug-in has been activated, the **NBS Repositories** view is displayed when you start Eclipse.

Note: For information on activating the plug-in, see ***Activate the Eclipse Plug-in***.

Use the **NBS Repositories** view to:

- Access Business Services
- Search for Business Services
- Set Trace and Log Options
- Access Connection Options

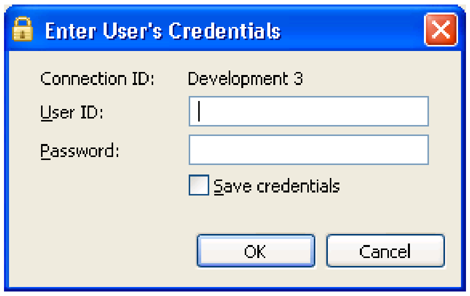
Access Business Services

To access the business services:

1 Open the context menu for the connection in the **NBS Repositories** view.

2 Select **Connect**.

The **Enter User's Credentials** window is displayed. For example:



Enter User's Credentials

Connection ID: Development 3

User ID:

Password:

Save credentials

Optionally, you can select **Save credentials** to save your user ID and password for this connection. To remove saved credentials from the security cache when Eclipse shuts down, select one of the following options:

- To clear logon credentials for this connection from the security cache, open the context menu for the connection in the **NBS Repositories** view and select **Clear logon credentials**.
- To clear logon credentials for all connections, open the context menu for the **Connections** node and select **Clear all logon credentials**.

3 Enter the user ID and password for the connection.

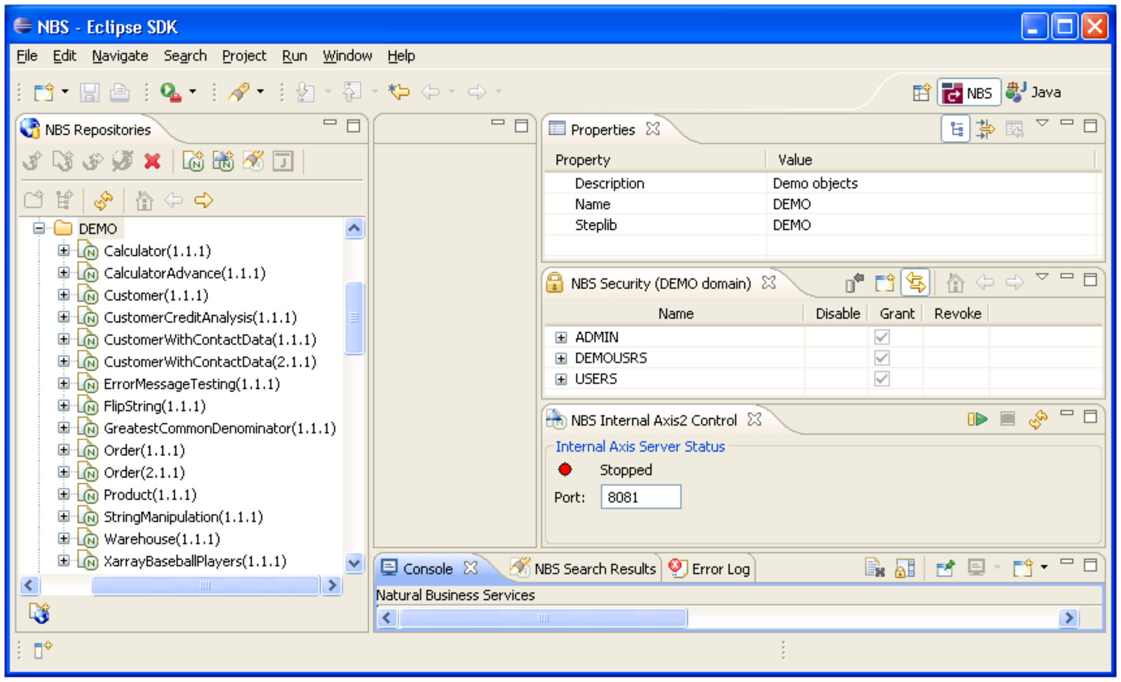
The Natural Business Services server retrieves information for the specified connection.

4 Expand the **Domains** node.

The list of available domains is displayed.

5 Expand the domain containing the business services.

The list of available services is displayed. For example:


NBS - Eclipse SDK
File Edit Navigate Search Project Run Window Help
NBS
Java
NBS Repositories
Properties
NBS Security (DEMO domain)
NBS Internal Axis2 Control
Console
NBS Search Results
Error Log

DEMO
- Calculator(1.1.1)
- CalculatorAdvance(1.1.1)
- Customer(1.1.1)
- CustomerCreditAnalysis(1.1.1)
- CustomerWithContactData(1.1.1)
- CustomerWithContactData(2.1.1)
- ErrorMessageTesting(1.1.1)
- FlipString(1.1.1)
- GreatestCommonDenominator(1.1.1)
- Order(1.1.1)
- Order(2.1.1)
- Product(1.1.1)
- StringManipulation(1.1.1)
- Warehouse(1.1.1)
- XarrayBaseballPlayers(1.1.1)

Property	Value
Description	Demo objects
Name	DEMO
Steplib	DEMO

Name	Disable	Grant	Revoke
ADMIN		✓	
DEMOUSRS		✓	
USERS		✓	

Internal Axis Server Status

● Stopped

Port: 8081

Natural Business Services

You can also use this window to set trace and log options. For information, see *Set Trace and Log Options*.

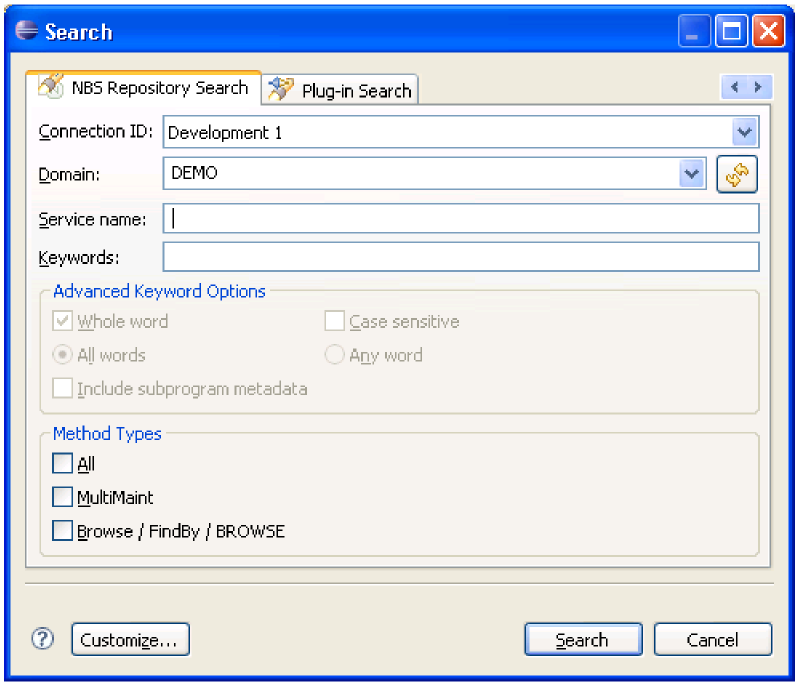
Search for Business Services

To search for all business services within a domain:

1 Open the context menu for the domain in the **NBS Repositories** view.

2 Select **Search for services**.

The **Search** window is displayed for the specified connection and domain. For example:



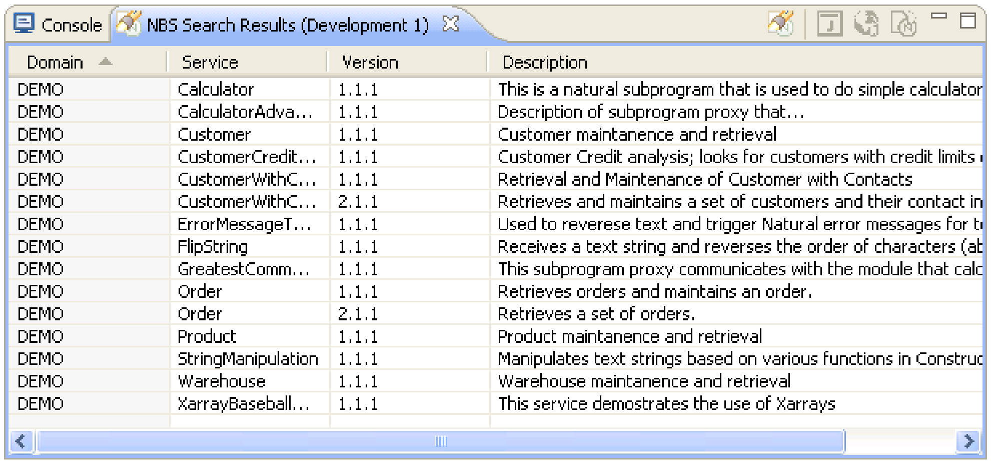
Connection ID: Development 1
Domain: DEMO
Service name:
Keywords:

Advanced Keyword Options
[✓] Whole word
[ ] Case sensitive
(●) All words
( ) Any word
[ ] Include subprogram metadata

Method Types
[ ] All
[ ] MultiMaint
[ ] Browse / FindBy / BROWSE

3 Select **Search**.

Natural Business Services displays the business services in the **NBS Search Results** view. For example:



Domain	Service	Version	Description
DEMO	Calculator	1.1.1	This is a natural subprogram that is used to do simple calculator
DEMO	CalculatorAdva...	1.1.1	Description of subprogram proxy that...
DEMO	Customer	1.1.1	Customer maintanence and retrieval
DEMO	CustomerCredit...	1.1.1	Customer Credit analysis; looks for customers with credit limits
DEMO	CustomerWithC...	1.1.1	Retrieval and Maintenance of Customer with Contacts
DEMO	CustomerWithC...	2.1.1	Retrieves and maintains a set of customers and their contact in
DEMO	ErrorMessageT...	1.1.1	Used to reverese text and trigger Natural error messages for t
DEMO	FlipString	1.1.1	Receives a text string and reverses the order of characters (ab
DEMO	GreatestComm...	1.1.1	This subprogram proxy communicates with the module that calc
DEMO	Order	1.1.1	Retrieves orders and maintains an order.
DEMO	Order	2.1.1	Retrieves a set of orders.
DEMO	Product	1.1.1	Product maintanence and retrieval
DEMO	StringManipulation	1.1.1	Manipulates text strings based on various functions in Construc
DEMO	Warehouse	1.1.1	Warehouse maintanence and retrieval
DEMO	XarrayBaseball...	1.1.1	This service demostrates the use of Xarrays

The **NBS Search Results** view displays the domain and service names, as well as the service version number and description for each service. To change the sort order for **Domain**, **Service**, **Version**, or **Description** in this view, select the corresponding heading. For example, when you select the

Service heading in the example above, the services are redisplayed in ascending order. The arrow in each heading indicates the sort order:

- ▲ indicates ascending order (A to Z)
- ▼ indicates descending order (Z to A)

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

- Search for All Services for a Connection
- Search for Specific Services Within All Domains
- Search for Specific Services within a Domain
- Use Advanced Criteria
- Customize the Search Window

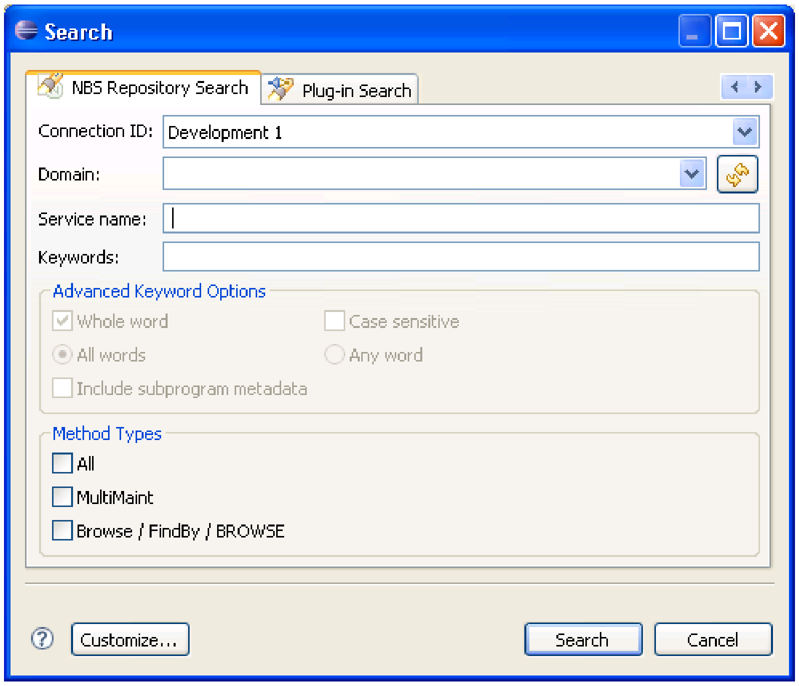
Search for All Services for a Connection

To search for all business services for a connection:

1 Open the context menu for the connection in the **NBS Repositories** view.

2 Select **Search for services**.

The **Search** window is displayed for the specified connection. For example:



Connection ID: Development 1
Domain:
Service name:
Keywords:

3 Select **Search**.

All business services for the specified connection are displayed in the **NBS Search Results** view.

Search for Specific Services Within All Domains

To search for specific business services within all domains:

1 Open the context menu for the connection in the **NBS Repositories** view.

2 Select **Search for services**.

The **Search** window is displayed.

3 Do one of the following:

- Type the name of the business service in **Service name** to return a list of all services with that name in all domains

  Note: Ensure you use the correct case for the service name. For example, use "Calculator", not "calculator", to search for the Calculator service.

- Type a keyword in **Keywords** to return a list of all services with that name in their descriptions

Note: You can also use **Advanced Keyword Options** and **Method Types** to limit the search. For information, see *Use Advanced Criteria*.

4 Select **Search**.

The specified business services are displayed in the **NBS Search Results** view.

Search for Specific Services within a Domain

To search for specific business services within a domain:

1 Open the context menu for the domain in the **NBS Repositories** view.

2 Select the name of the domain from **Domain**.

3 Select **Search for services**.

The **Search** window is displayed.

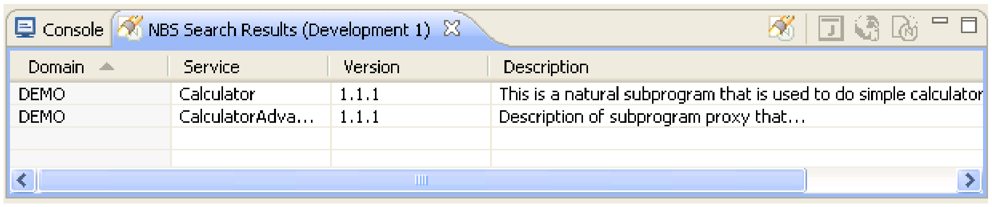
4 Do one of the following:

- Type the name of a business service in **Service name** to return a list of all services with that name in the specified domain
- Type a keyword in **Keywords** to return a list of all services with that name in their descriptions

Note: You can also use **Advanced Keyword Options** and **Method Types** to limit the search. For information, see *Use Advanced Criteria*.

5 Select **Search**.

The specified business services are displayed in the **NBS Search Results** view. For example, if you enter "Calculator" in **Service name**, the following results are displayed:



Domain	Service	Version	Description
DEMO	Calculator	1.1.1	This is a natural subprogram that is used to do simple calculator
DEMO	CalculatorAdva...	1.1.1	Description of subprogram proxy that...

Use Advanced Criteria

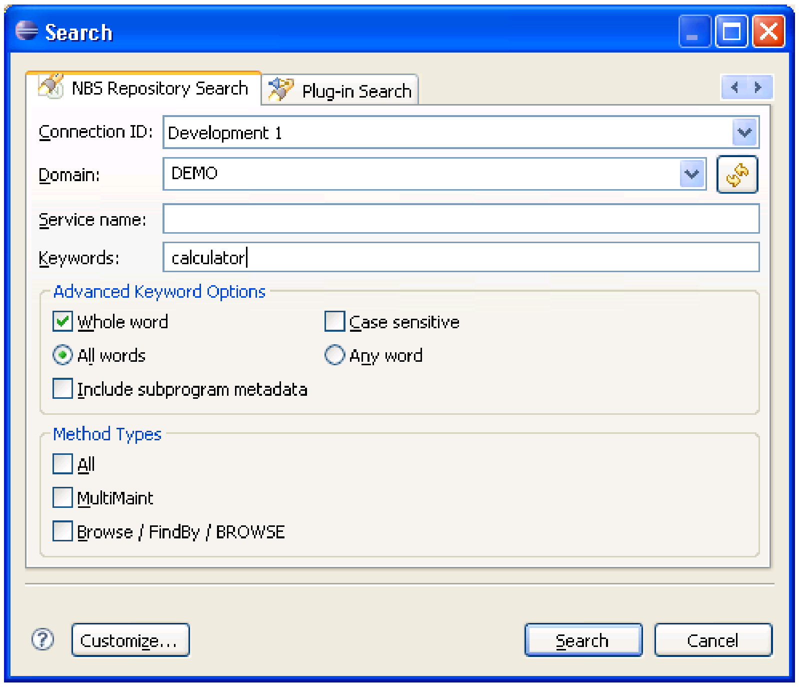
You can further limit the search of business services by using advanced keyword options or specific method types.

Advanced Keyword Options

To limit the search to business services that match the advanced keyword options:

1 Type one or more keywords in **Keywords** in the **Search** window.

For example:



Connection ID: Development 1
Domain: DEMO
Service name:
Keywords: calculator

Advanced Keyword Options
[✓] Whole word
[ ] Case sensitive
(●) All words
( ) Any word
[ ] Include subprogram metadata

Method Types
[ ] All
[ ] MultiMaint
[ ] Browse / FindBy / BROWSE

2 Select from the following advanced keyword options:

Keyword Option	Description
Whole word	Search for whole words only (for example, if you type "Customers" in **Keywords**, business services containing "Customer" will not be returned).
All words	Search for all words specified in **Keywords** (for example, if you type "Customer record" in **Keywords** and select **All words**, only business services containing both "Customer" and "record" will be returned).
Include subprogram metadata	Search business services and their subprogram metadata for services containing the keyword specified in **Keywords**.
Case sensitive	Search for words using a particular case (for example, if you type "Customer" in **Keywords** and select **Case sensitive**, business services containing "customer" will not be returned).
Any word	Search for either word specified in **Keywords** (for example, if you type "Customer record" in **Keywords** and select **All words**, business services containing either "Customer" or "record" will be returned).

3 Select **Search**.

All business services that match the specified options are displayed in the **NBS Search Results** view.

Method Types

You can limit the search to business services containing one of the method types listed. For example, you can search for business services containing:

- All method types
- MultiMaint method types
- Browse or FindBy method types

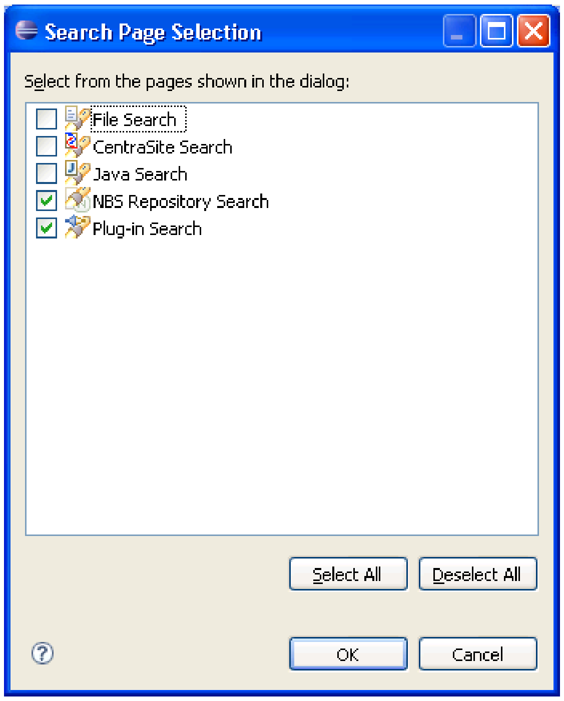
Customize the Search Window

You can customize the **Search** window.

To customize the Search window:

1 Select **Customize**.

The **Search Page Selection** window is displayed. For example:



- [ ] File Search
- [ ] CentraSite Search
- [ ] Java Search
- [✓] NBS Repository Search
- [✓] Plug-in Search

2 Select or deselect the pages you want to include in your customized **Search** window.

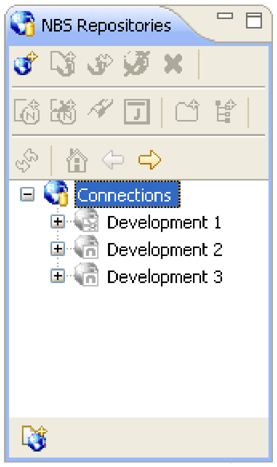
- To display all views, select **Select All**.

- To remove all views and then select one, select **Deselect All**, select the view, and then select **OK**. At least one view must be selected in this window.

3 Select **OK** to confirm the changes.

Set Trace and Log Options

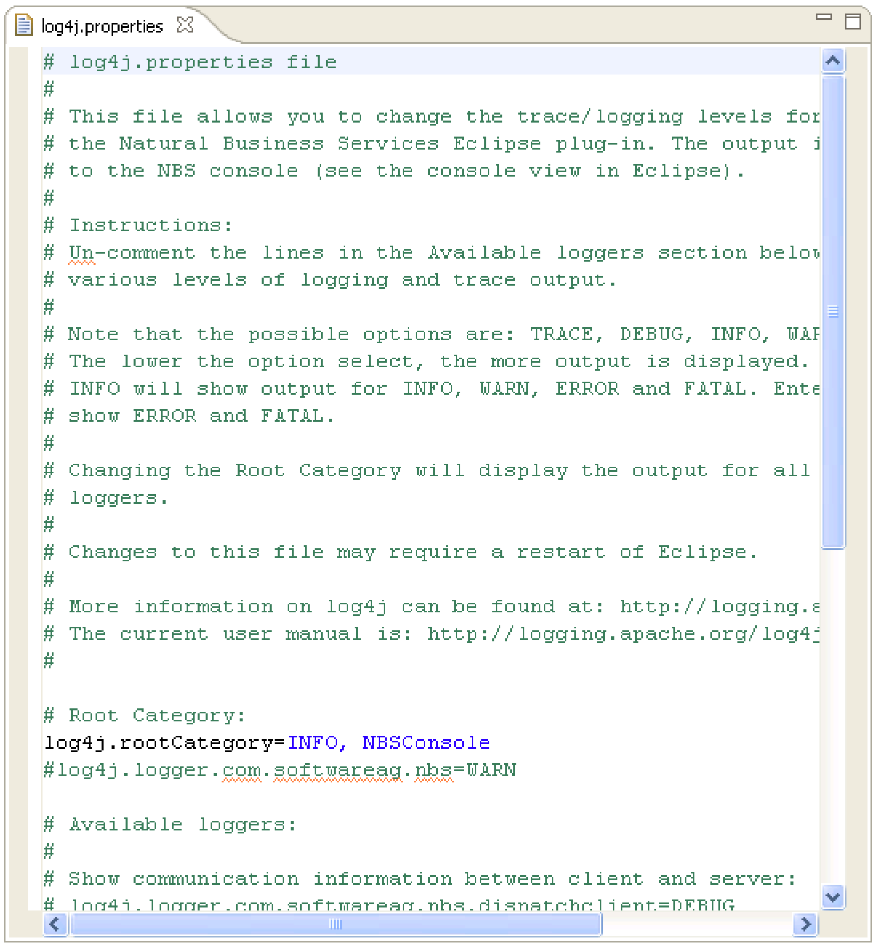
Trace and log functions in the Eclipse plug-in are handled through the *log4j.properties* file using log4j technology. An icon at the bottom of the **NBS Repositories** view allows you to display this file for editing. For example:



To set trace and log options:

1 Select the icon at the bottom of the **NBS Repositories** view.

The *log4j.properties* file is displayed. For example:



log4j.properties

```
# log4j.properties file
#
# This file allows you to change the trace/logging levels for
# the Natural Business Services Eclipse plug-in. The output i
# to the NBS console (see the console view in Eclipse).
#
# Instructions:
# Un-comment the lines in the Available loggers section below
# various levels of logging and trace output.
#
# Note that the possible options are: TRACE, DEBUG, INFO, WAR
# The lower the option select, the more output is displayed.
# INFO will show output for INFO, WARN, ERROR and FATAL. Ente
# show ERROR and FATAL.
#
# Changing the Root Category will display the output for all
# loggers.
#
# Changes to this file may require a restart of Eclipse.
#
# More information on log4j can be found at: http://logging.a
# The current user manual is: http://logging.apache.org/log4j
#

# Root Category:
log4j.rootCategory=INFO, NBSConsole
#log4j.logger.com.softwareag.nbs=WARN

# Available loggers:
#
# Show communication information between client and server:
# log4j.logger.com.softwareag.nbs.dispatchclient=DEBUG
```

2 Change the trace and log output as desired.

The file contains editing instructions.

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

Note: You can use the **Preferences** window to format the trace and log output. For information, see *Preferences for Trace and Log Functions*.

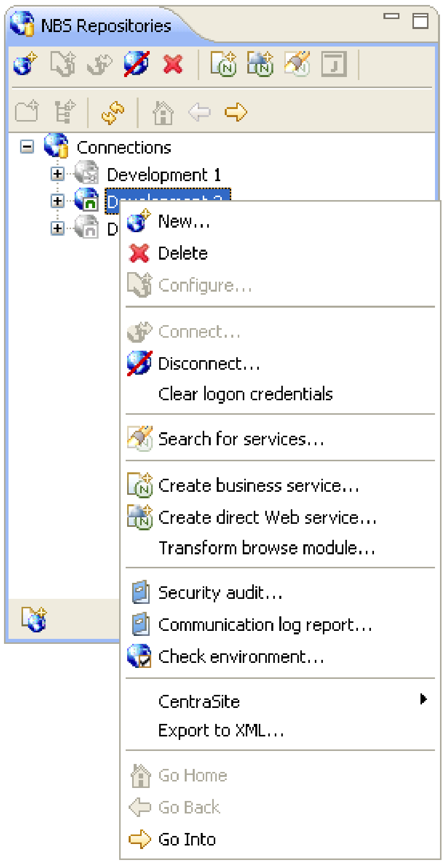
Access Connection Options

Many options are available for connections listed in the **NBS Repositories** view, based on the type of connection (Broker or SPoD).

To access the connection options:

- Open the context menu for a connection in the **NBS Repositories** view.

  The following example shows the options for a SPoD connection:



Create a Directly Enabled Web Service

This option creates a single Web service from a selected Natural subprogram. The wizard generates a directly enabled business service, a Java class, and a Web service class, and then deploys the service to the server of your choice.

Notes:

1. This option is only available for SPoD connections.
2. You can set default values for this wizard in the *serviceDefaults.xml* file, which is installed in the configuration folder.

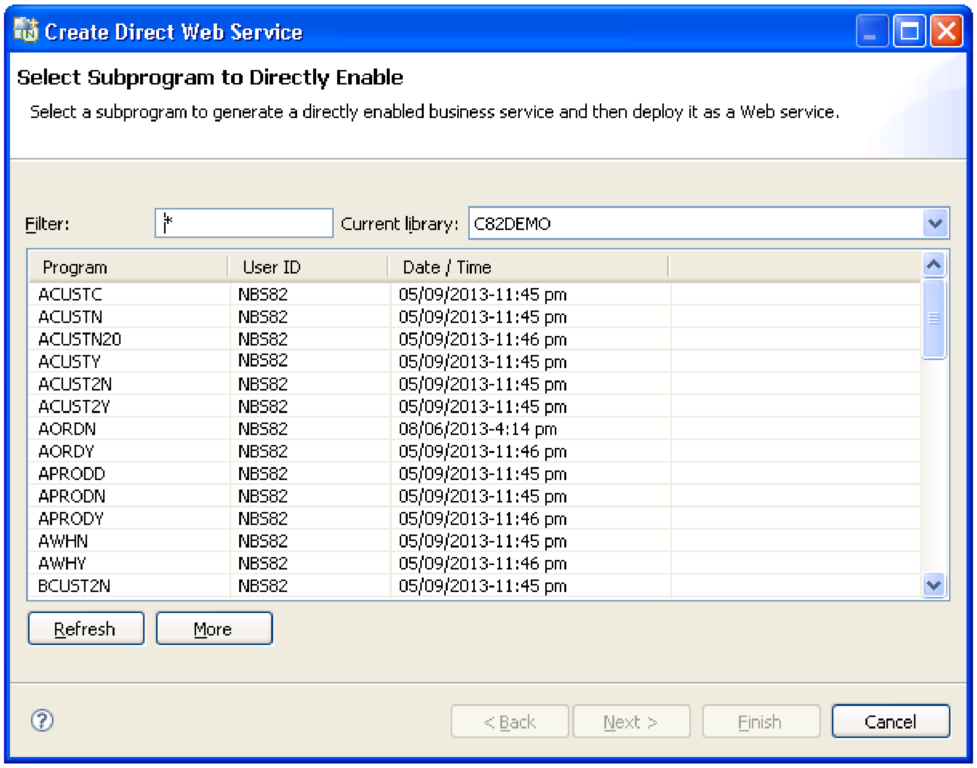
To create a directly enabled Web service:

1 Open the context menu for the SPoD connection in the **NBS Repositories** view.

For an example of this menu, see *Access Connection Options*.

2 Select **Create direct Web service**.

The **Select Subprogram to Directly Enable** panel is displayed. For example:



Select Subprogram to Directly Enable

Select a subprogram to generate a directly enabled business service and then deploy it as a Web service.

Filter: * Current library: C82DEMO

Program	User ID	Date / Time
ACUSTC	NBS82	05/09/2013-11:45 pm
ACUSTN	NBS82	05/09/2013-11:45 pm
ACUSTN20	NBS82	05/09/2013-11:46 pm
ACUSTY	NBS82	05/09/2013-11:45 pm
ACUST2N	NBS82	05/09/2013-11:45 pm
ACUST2Y	NBS82	05/09/2013-11:45 pm
AORDN	NBS82	08/06/2013-4:14 pm
AORDY	NBS82	05/09/2013-11:46 pm
APRODD	NBS82	05/09/2013-11:45 pm
APRODN	NBS82	05/09/2013-11:45 pm
APRODY	NBS82	05/09/2013-11:46 pm
AWHN	NBS82	05/09/2013-11:45 pm
AWHY	NBS82	05/09/2013-11:46 pm
BCUST2N	NBS82	05/09/2013-11:45 pm

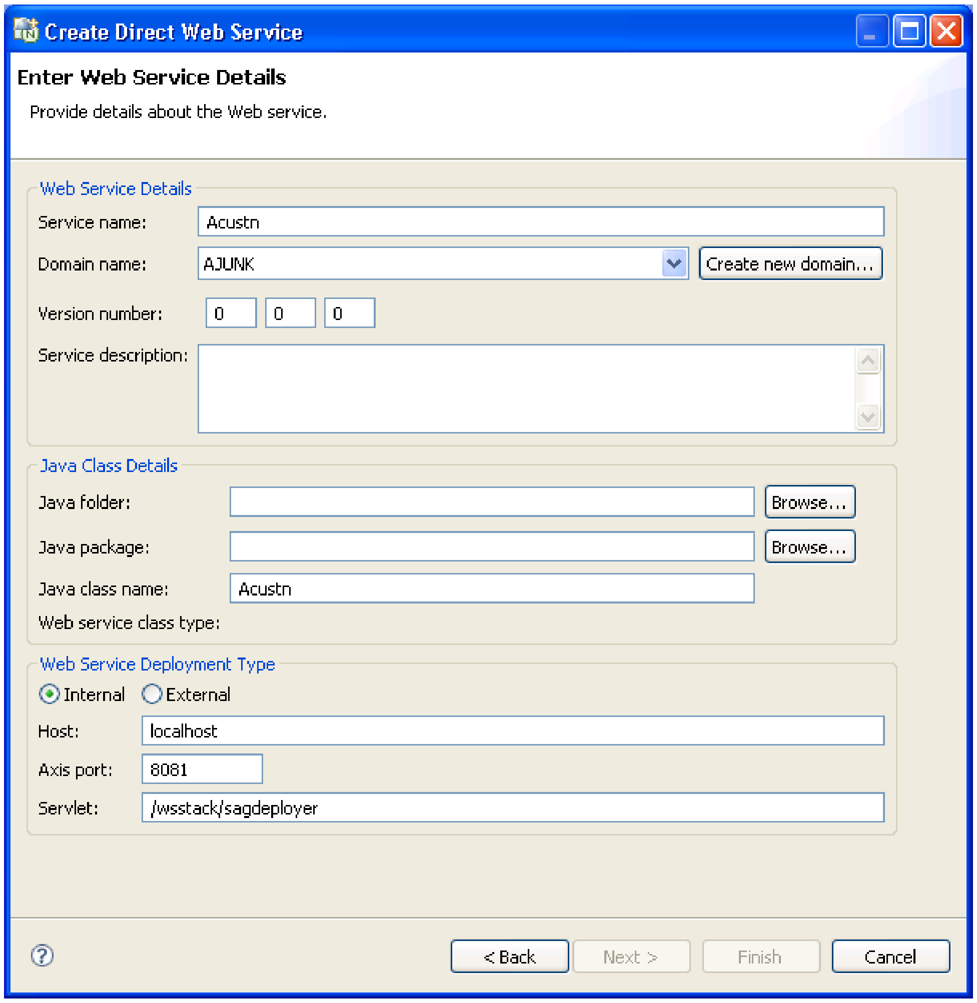
This panel displays the list of subprograms in the current library. Optionally, you can:

Task	Procedure
Display the 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 C82DEMO are displayed.
Change the name of the library.	Select a different library from **Current library**.

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

4 Select **Next**.

The **Enter Web Service Details** panel is displayed. For example:



Enter Web Service Details

Provide details about the Web service.

Web Service Details

Service name: Acustn

Domain name: AJUNK

Version number: 0 0 0

Service description:

Java Class Details

Java folder:

Java package:

Java class name: Acustn

Web service class type:

Web Service Deployment Type

[✓ Internal] External

Host: localhost

Axis port: 8081

Servlet: /wsstack/sagdeployer

To determine default values for several of the input fields on this panel, the wizard uses the *serviceDefaults.xml* file. This file is installed in the configuration folder.

Optionally, you can use this panel to:

Task	Procedure
Change the name of the service.	Type a new name in **Service name**.
Change the name of the domain.	Type a new name in **Domain name**.
Create a new domain.	Select **Create new domain** (if one does not currently exist). For information, see *Create a New Domain*.
Provide the location of the Java folder.	Select a Java folder from **Java folder**.
Provide the name of the Java package.	Select a Java project from **Java project**.
Change the name of the Java class for the service.	Type a new name in **Java class name**.
Use an external Web service deployment type.	Select **External** and specify the Axis port number and Servlet address. **Note:** For information about the WS-Stack plug-in, refer to the webMethods shared components documentation.

5 Enter the following information for the Web service:

Option	Description
Version number	Number for this version of the Web service. For example, enter "1 1 1" for the first version.
Service description	Brief description of the service.
Java folder	Name of the Java folder for the service.
Internal Axis port	ID for the internal Axis port.

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

Create a REQUEST DOCUMENT Client

This option generates a subprogram that allows Natural to call an external Web service. The generated Natural subprogram uses REQUEST DOCUMENT and PARSE XML statements to call the service and interpret the response.

Notes:

1. You must use a SPoD connection to create a REQUEST DOCUMENT client, although you can use the generated code with any connection.

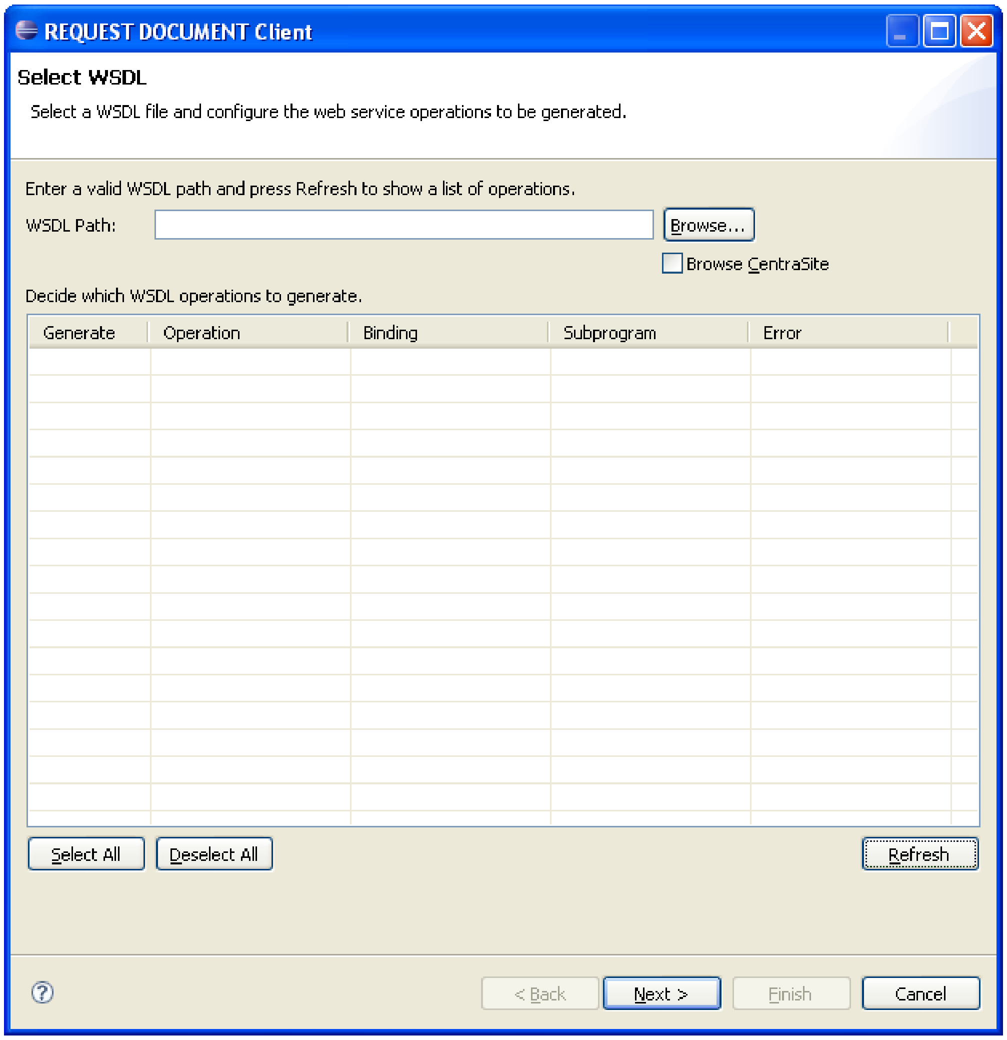
2. To use this feature, the Natural nucleus/profile must be set up to correctly handle XML. For information, see *Activate PARSE XML Statement* and *Activate REQUEST DOCUMENT Statement* in the Natural documentation.

The generated subprogram includes parameter data areas (PDAs) that map to the request and response portions of a Web service method (operation). The subprogram then maps the input parameters to an XML document, which is sent to the Web service via the REQUEST DOCUMENT statement. The response is verified and parsed in the response PDA and the parameters are generated as #INPUT and #OUTPUT level 1 structures. In addition, the generated subprogram includes an error PDA that informs users of any errors.

To create a REQUEST DOCUMENT client:

1 Select **New > REQUEST DOCUMENT Client** on the **File** menu.

The **Select WSDL** panel is displayed. For example:



Select WSDL

Select a WSDL file and configure the web service operations to be generated.

Enter a valid WSDL path and press Refresh to show a list of operations.

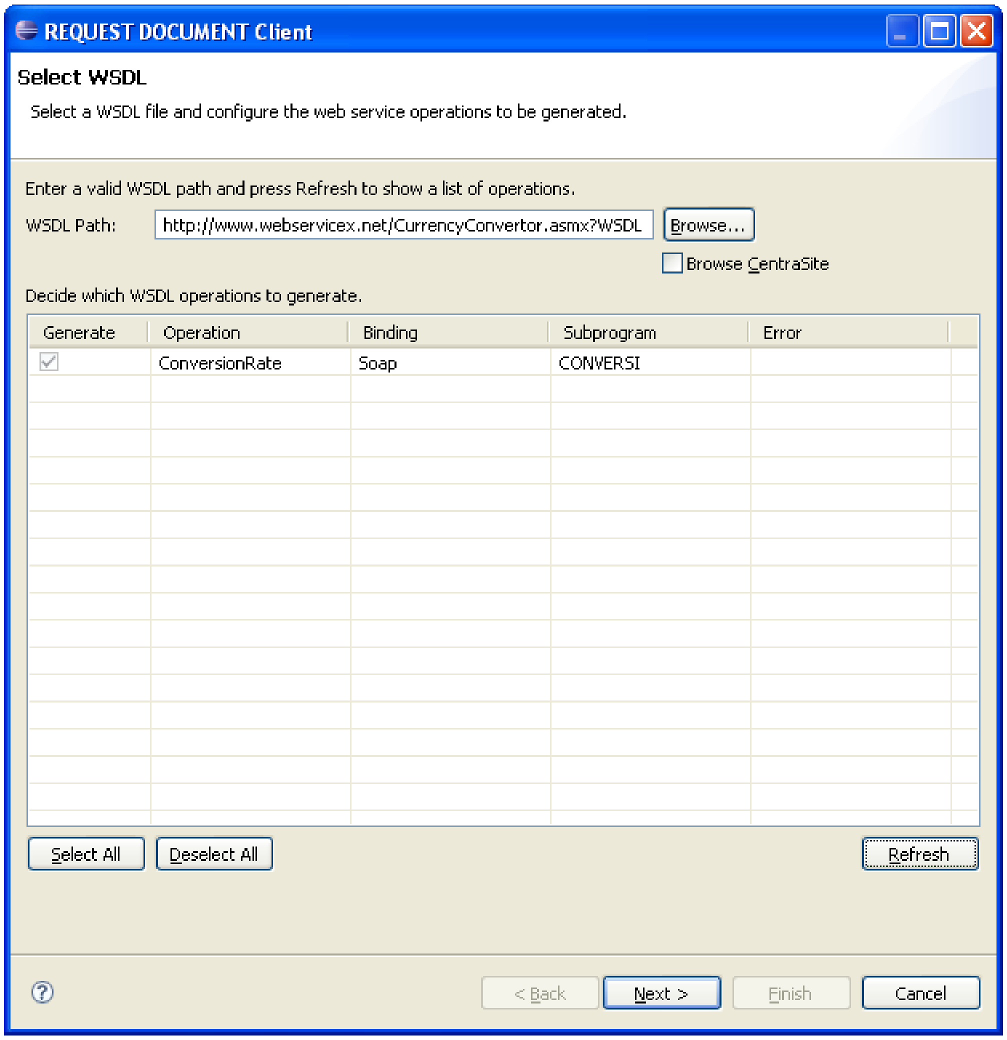
WSDL Path:

Decide which WSDL operations to generate.

Generate	Operation	Binding	Subprogram	Error

2 Select a WSDL path in **WSDL Path**.

You can browse for a WSDL path in the file system (select **Browse**) or from CentraSite (select **Browse CentraSite**). You can also type a valid WSDL path using an HTTP location. After specifying the path name, the wizard scans the selected WSDL file for each Web service method (operation), generates a separate subprogram for each operation, and displays them in the table. For example:



Select WSDL

Select a WSDL file and configure the web service operations to be generated.

Enter a valid WSDL path and press Refresh to show a list of operations.

WSDL Path: http://www.webservicex.net/CurrencyConvertor.asmx?WSDL

Decide which WSDL operations to generate.

Generate	Operation	Binding	Subprogram	Error
✓	ConversionRate	Soap	CONVERSI	

This window indicates the following for each operation:

- Whether a subprogram is generated
- Which operation is generated
- Which binding is used
- What the generated subprogram is named

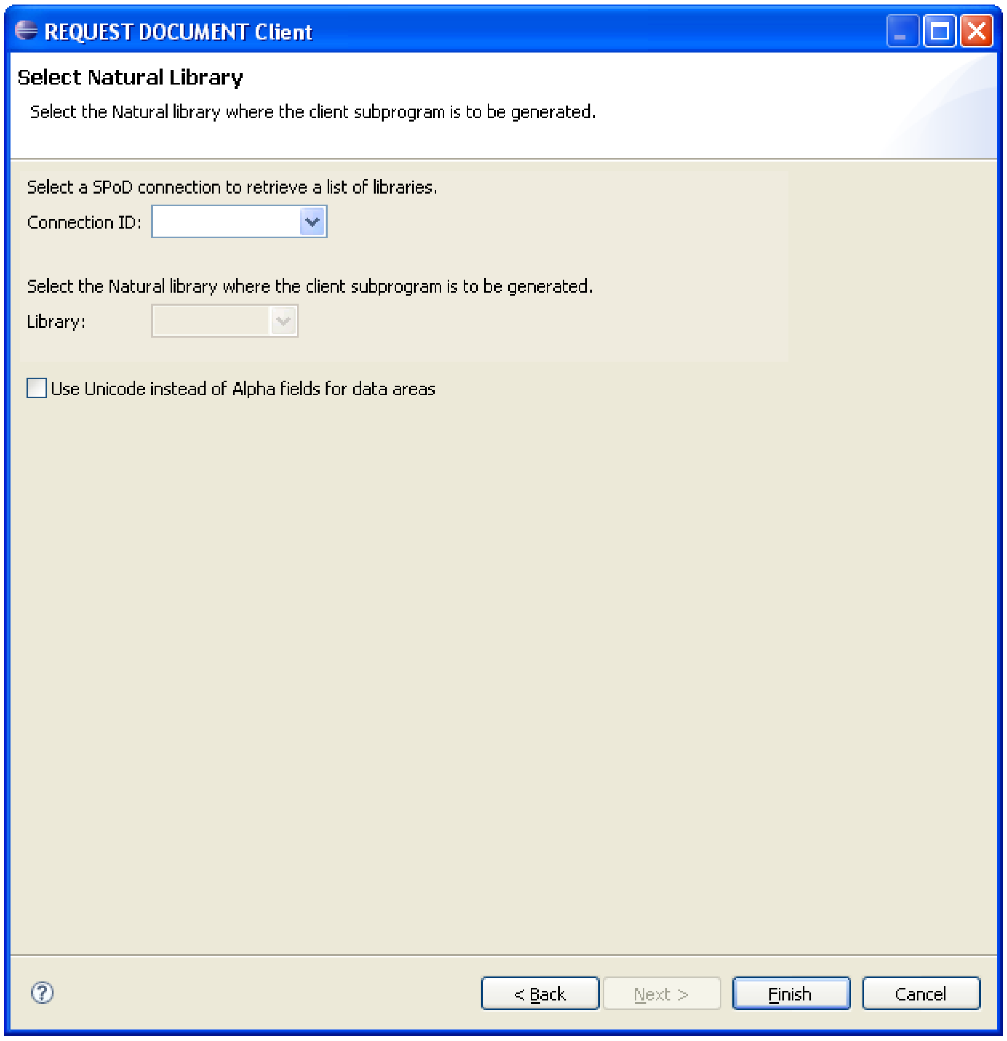
If desired, you can change the generated subprogram name in this window.

3 Select or deselect the operations to generate.

Tip: Use **Select All** or **Deselect All** to quickly select all or one operation.

4 Select **Next**.

The **Select Natural Library** panel is displayed. For example:


REQUEST DOCUMENT Client

Select Natural Library

Select the Natural library where the client subprogram is to be generated.

Select a SPoD connection to retrieve a list of libraries.

Connection ID:

Select the Natural library where the client subprogram is to be generated.

Library:

☐ Use Unicode instead of Alpha fields for data areas

Note: If the Web service passes Unicode data, you can select **Use Unicode instead of Alpha fields for data areas** and Unicode format will be used for variables in the data areas (instead of Alpha format).

5 Select a SPoD connection in **Connection ID**.

The wizard retrieves a list of libraries for that connection.

6 Select the Natural library in which to generate the subprogram in **Library**.

7 Select **Finish** to generate the subprogram.

User Exits for the REQUEST-DOCUMENT Subprogram

The user exits available for the REQUEST-DOCUMENT subprogram are:

User Exit	Description
SEARCH-AND-REPLACE-INITIALIZATION	Allows you to add your own substitutions to the standard search and replace array. For more information, see *Modify the Search and Replace Array*.
BEFORE-REVERSE-SUBSTITUTION	Allows modifications to the standard search and replace array before the search and replace reversal occurs. Use this exit to customize your own reversal substitutions if they are different from the original substitutions. This exit is used in conjunction with the AFTER-REVERSE-SUBSTITUTION user exit.
AFTER-REVERSE-SUBSTITUTION	Allows modifications to the standard search and replace array after the search and replace reversal occurs. This exit is used in conjunction with the BEFORE-REVERSE-SUBSTITUTION user exit.

Modify the Search and Replace Array

To ensure that the correct data is received from and presented to the end user, the data is massaged coming from and going to the Web service. For example, to use the ' character in the xml data, the character is sent as &apos. Before this character goes into the data base, the &apos is replaced by '. The REQUEST-DOCUMENT subprogram handles this functionality for the standard key characters. For example:

```
*************************************************************************************
DEFINE SUBROUTINE SETUP-SUBSTITUTION-STRINGS
*************************************************************************************
/* Determine double quotes in based on platform
#DOUBLE-QUOTE := 'A'
IF #DOUBLE-QUOTE < '0' THEN /* Letters LT numbers
  #DOUBLE-QUOTE := H'7F' /* EBCDIC
ELSE
  #DOUBLE-QUOTE := H'22' /* ASCII
END-IF
/* setup search and replace strings;
/* note & must be first because of & substitution
#SEARCH-STRING(1) := '&'
#REPLACE-STRING(1) := '&amp;'
#SEARCH-STRING(2) := "'"
#REPLACE-STRING(2) := '&apos;'
#SEARCH-STRING(3) := #DOUBLE-QUOTE
#REPLACE-STRING(3) := '&quot;'
#SEARCH-STRING(4) := '<'
#REPLACE-STRING(4) := '&lt;'
#SEARCH-STRING(5) := '>'
```

```
#REPLACE-STRING(5) := '&gt;'
**SAG DEFINE EXIT SEARCH-AND-REPLACE-INITIALIZATION
**SAG END-EXIT
END-SUBROUTINE /* SETUP-SUBSTITUTION-STRINGS
*******************************************************************************************
DEFINE SUBROUTINE SUBSTITUTION
*******************************************************************************************
FOR #REP-IND = 1 TO #MAX-REPLACE
  EXAMINE #REP-MANIPULATION-STRING FOR #SEARCH-STRING(#REP-IND )
  REPLACE WITH #REPLACE-STRING(#REP-IND )
END-FOR
END-SUBROUTINE /* PERFORM-SUBSTITUTION
*******************************************************************************************
DEFINE SUBROUTINE REVERSE-SUBSTITUTION
*******************************************************************************************
**SAG DEFINE EXIT BEFORE-REVERSE-SUBSTITUTION
**SAG END-EXIT
FOR #REP-IND = 1 TO #MAX-REPLACE
  #REP-MANIPULATION-STRING := #SEARCH-STRING(#REP-IND)
  #SEARCH-STRING(#REP-IND) := #REPLACE-STRING(#REP-IND)
  #REPLACE-STRING(#REP-IND) := #REP-MANIPULATION-STRING
END-FOR
**SAG DEFINE EXIT AFTER-REVERSE-SUBSTITUTION
**SAG END-EXIT
END-SUBROUTINE /* REVERSE-SUBSTITUTION
```

Test the Generated Subprogram

To test the generated subprogram:

1. Create a program that will pass the parameters to the generated subprogram.
2. Supply the requested input.
3. Verify the output after the CALLNAT has been issued.

Transform a Browse Module

This option transforms a Natural Construct-generated browse module into object browse modules for use in a client/server environment.

Note: This option is only available for SPoD connections.

To access the Transform Browse wizard:

1. Open the context menu for the SPoD connection in the **NBS Repositories** view.

For information, see *Access Connection Options*.

2 Select **Transform browse module**.

The **Select Browse Module** panel is displayed. For more information, see *Eclipse Plug-in* in Verify Transformation Specifications Panel.

Check Environment Information for a Connection

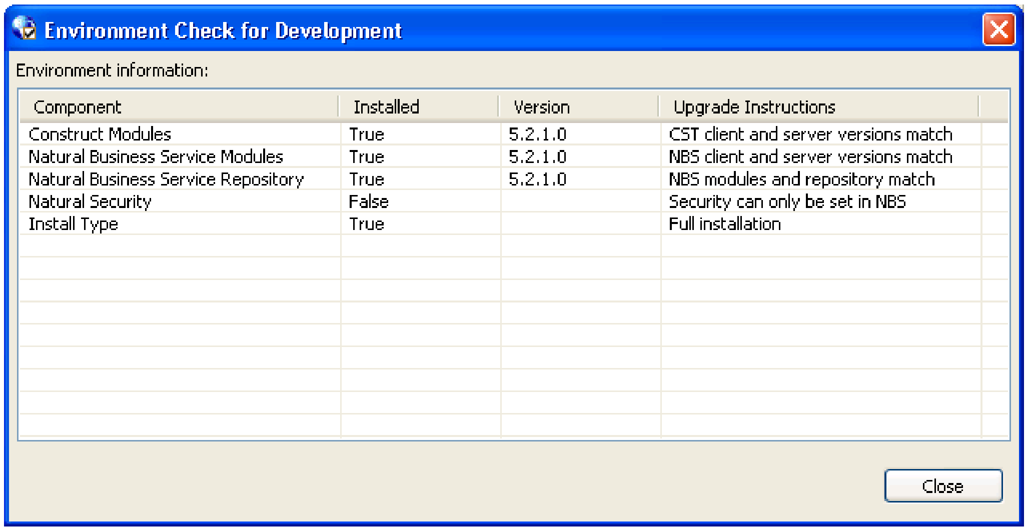
To display the environment information:

1 Open the context menu for the connection in the **NBS Repositories** view.

For an example of this menu, see *Access Connection Options*.

2 Select **Check environment**.

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



Environment Check for Development

Environment information:

Component	Installed	Version	Upgrade Instructions
Construct Modules	True	5.2.1.0	CST client and server versions match
Natural Business Service Modules	True	5.2.1.0	NBS client and server versions match
Natural Business Service Repository	True	5.2.1.0	NBS modules and repository match
Natural Security	False		Security can only be set in NBS
Install Type	True		Full installation

This window displays information about the environment for the current connection.

3 Select **Close**.

Access CentraSite

This option adds Natural Business Services repository metadata to CentraSite. The information includes business services, subprograms, proxies, and Natural Construct model, file, and field names, as well as information on relationships (for example, a business service belongs to a domain, a subprogram uses a certain file, etc.). You can also remove the repository metadata from CentraSite. For information, see *Remove Concepts*.

Import Custom Concepts into the CentraSite Server

Before you can add concepts to CentraSite, custom concept archive files must be imported into the CentraSite server. These files are located in the CentrasiteImports folder within the NBS install folder (for example, *C:\Program Files\Software AG\Natural Business Services\V8.2\CentrasiteImports\Concept-*.zip*).

To import custom concepts to the CentraSite server:

1 Open CentraSite Control and log on.

2 Select **Import**.

3 Select **Archive** in **Import type**.

4 Select the custom concept archive file.

5 Select **Import**.

6 Repeat steps 2 to 5 for each custom archive file.

Add Concepts

To add repository metadata to CentraSite:

1 Open the context menu for the connection in the **NBS Repositories** view.

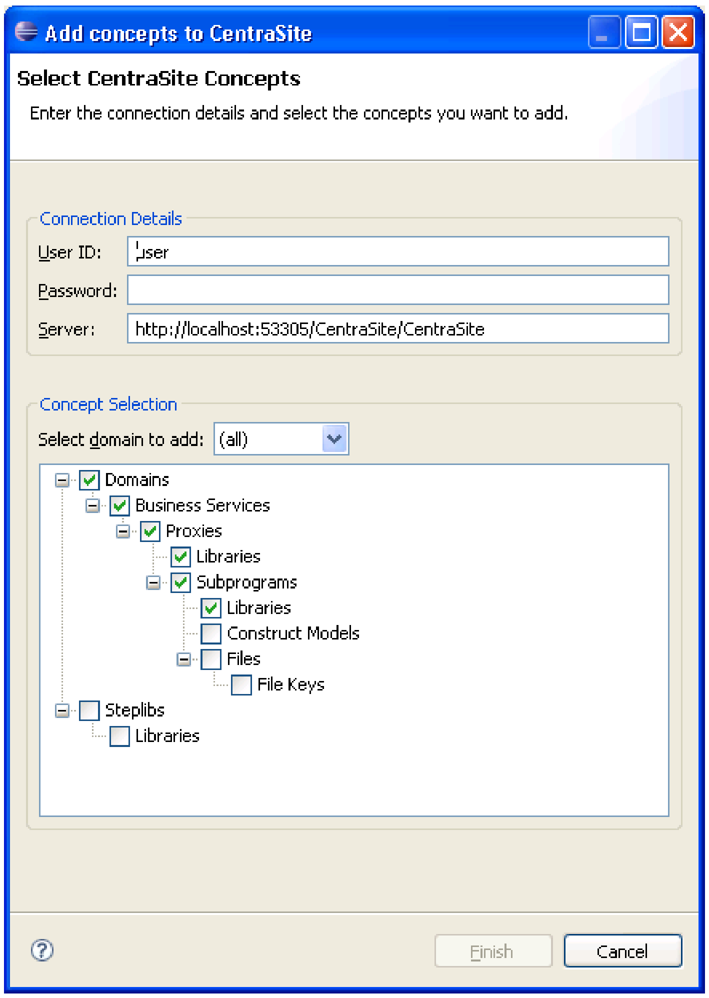
For an example of this menu, see *Access Connection Options*.

2 Select **CentraSite**.

The CentraSite options are displayed.

3 Select **Add concepts**.

The **Select CentraSite Concepts** window is displayed. For example:


Add concepts to CentraSite

Select CentraSite Concepts

Enter the connection details and select the concepts you want to add.

Connection Details

User ID: user

Password:

Server: http://localhost:53305/CentraSite/CentraSite

Concept Selection

Select domain to add: (all)

- [✓] Domains
  - [✓] Business Services
    - [✓] Proxies
      - [✓] Libraries
      - [✓] Subprograms
        - [✓] Libraries
        - [ ] Construct Models
        - [ ] Files
          - [ ] File Keys
- [ ] Steplibs
  - [ ] Libraries

Tip: You can also access this window from the context menu for a domain. If you do, the domain will be selected automatically in **Select domain to add**.

4 Type the user ID for CentraSite in **User ID**.

5 Type the password for CentraSite in **Password**.

6 Verify the location of the server port for CentraSite in **Server**.

7 Verify which nodes in the repository tree you want to add to CentraSite.

By default, the information for all domains, business services, proxies, libraries, and subprograms is added to CentraSite. You can select other nodes in the repository tree (listed in **Concept Selection**) or deselect any nodes you do not want to add.

- To add one domain only, select the domain in **Select domain to add**.
- To include details about the step library chains, select **Steplibs** in the repository tree.

8 Select **Finish** to add the selected nodes to CentraSite.

Remove Concepts

This action will remove ALL repository metadata entries (i.e., any registry object that has one of the Natural Business Services custom concepts as its object type).

To remove repository metadata from CentraSite:

1 Open the context menu for the connection in the **NBS Repositories** view.

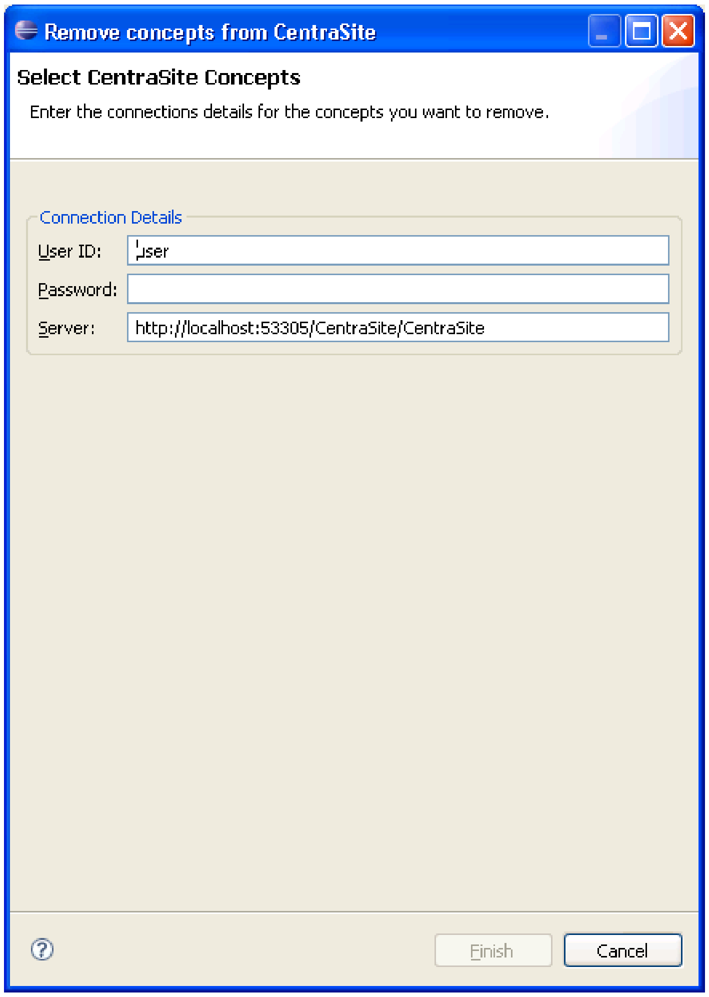
For an example of this menu, see ***Access Connection Options***.

2 Select **CentraSite**.

The CentraSite options are displayed.

3 Select **Remove concepts**.

The **Select CentraSite Concepts** window is displayed. For example:



Select CentraSite Concepts

Enter the connections details for the concepts you want to remove.

Connection Details

User ID: user

Password:

Server: http://localhost:53305/CentraSite/CentraSite

4 Type the user ID for the CentraSite concepts you want to remove in **User ID**.

5 Type the password for the CentraSite concepts in **Password**.

6 Verify the location of the server port for CentraSite in **Server**.

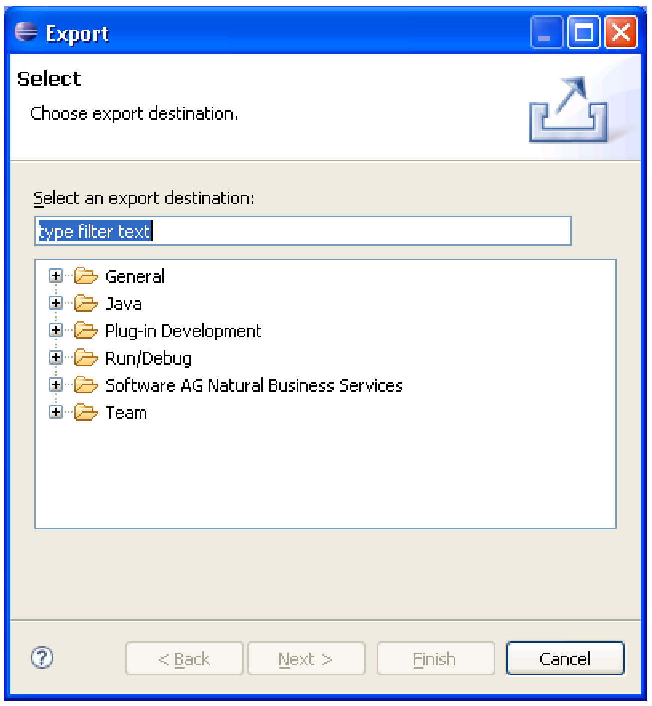
7 Select **Finish**.

Export Repository Metadata to XML

To export repository metadata to XML:

1 Select **Export** on the **File** menu.

The **Export** window is displayed. For example:

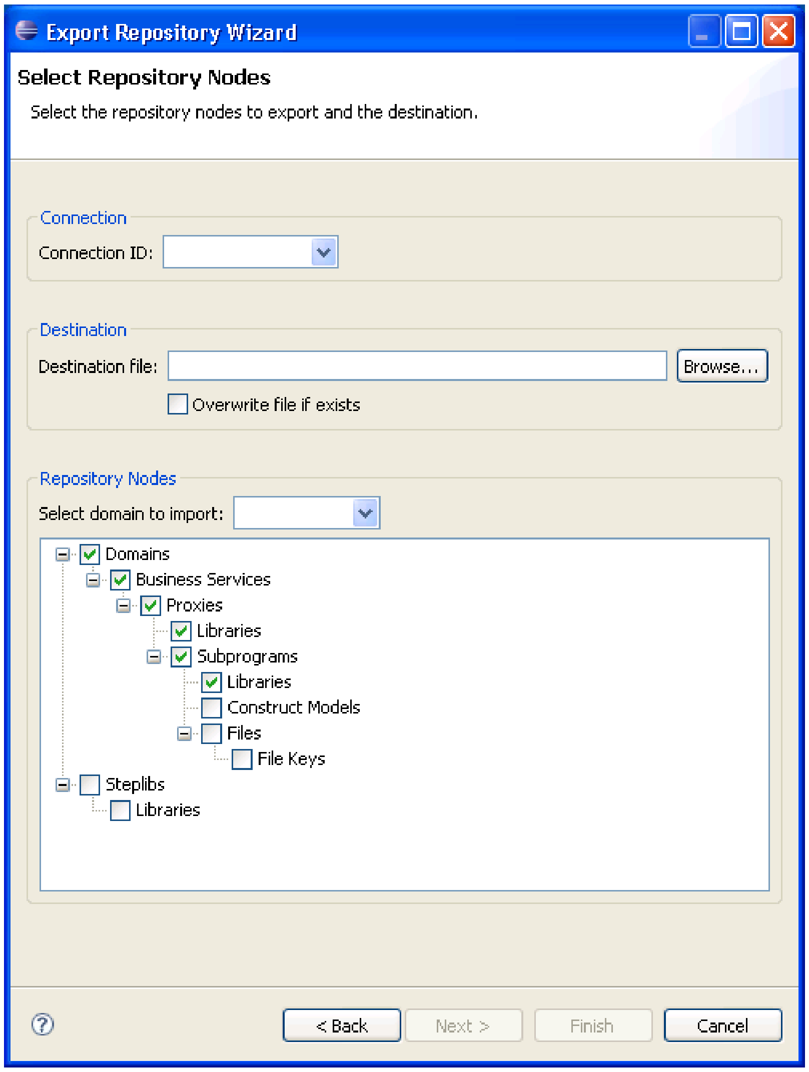


2 Expand the **Software AG Natural Business Services** node.

3 Select **NBS Repository**.

4 Select **Next**.

The **Select Repository Nodes** window is displayed. For example:



Tip: You can also access this window using the context menu for an open connection or for a domain listed in the **NBS Repositories** view.

5 Select the Connection ID for the repository metadata you want to export.

6 Select an output file name in **Destination file**.

To overwrite the existing file, select **Overwrite file if exists**.

Tip: If you type the file name, ensure the name includes the .xml extension.

7 Select the domain containing the repository metadata you want to export.

To export the metadata for all domains, select **(all)**.

8 Select the repository metadata you want to include in the exported file.

9 Select **Finish** to export the selected repository metadata.

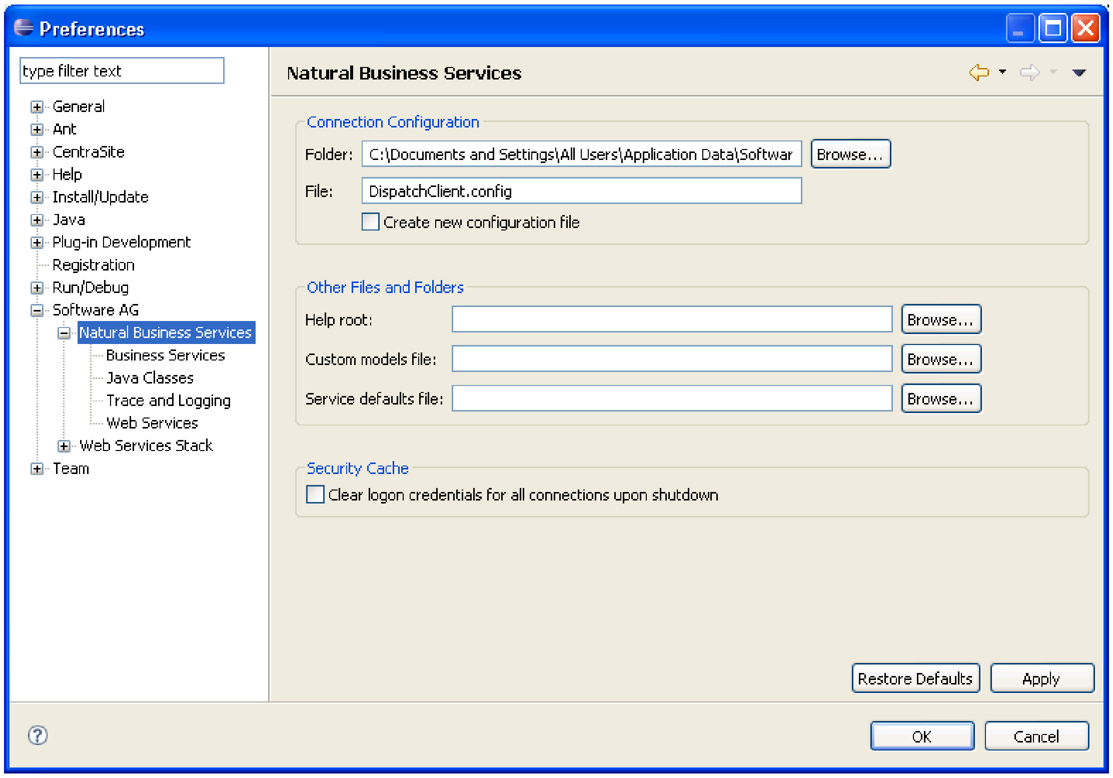
Set Natural Business Service Preferences

This section describes how to set preferences for Natural Business Services (NBS).

To set preferences for NBS:

1 Select **Preferences** > **Software AG** > **Natural Business Services** on the **Window** menu.

The **Preferences** window for Natural Business Services is displayed. For example:



Folder: C:\Documents and Settings\All Users\Application Data\Softwar

File: DispatchClient.config

Using this window, you can:

Task	Procedure
Change the configuration folder used.	Select another folder from **Folder**.
Create a new configuration file.	Select **Create new configuration** and enter the name of the new file in **File**.
Specify the folder containing the help file.	Select the folder from **Help root**.
Specify the file containing custom models.	Select the file from **Custom models file**.
Specify the file containing default values for business services.	Select the file from **Service defaults file**.
Remove saved credentials from the security cache.	Select **Clear logon credentials for all connections upon shutdown** to remove the saved credentials from the security cache for all connections when Eclipse shuts down. **Tip:** You can also clear all logon credentials by opening the context menu for the **Connections** node in the **NBS Repositories** view and selecting **Clear all logon credentials**.
Restore all default values for the configuration file.	Select **Restore Defaults**.

2 Select **OK** to apply the changes and close the **Preferences** window.

Note: To apply the changes without closing the window, select **Apply**.

In addition to these preferences, you can also set the following:

- Preferences for Business Services
- Preferences for Java Classes
- Preferences for Trace and Log Functions
- Preferences for Web Services

Preferences for Business Services

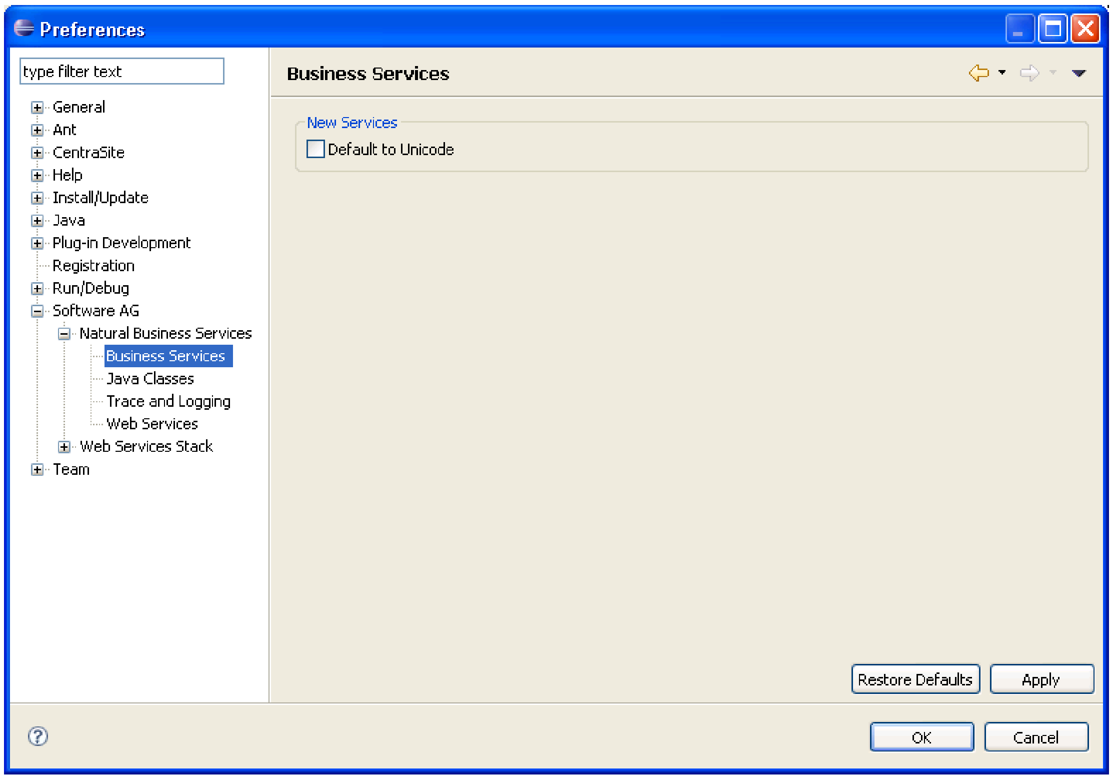
Use this setting to have NBS create the interface for all new services using Unicode field formats (U data type).

Note: Services that use a Unicode interface require a special EntireX runtime server definition. For information, see *Use Unicode Parameters for Your Business Service*.

To set the business service preferences:

1 Select the **Business Services** node.

The Business Services information is displayed. For example:



☐ Default to Unicode

The only option available in this window is **Default to Unicode**, which allows all new services to be Unicode enabled.

2 Select **Default to Unicode**.

3 Select **OK** to apply the changes and close the **Preferences** window.

Note: To apply the changes without closing the window, select **Apply**.

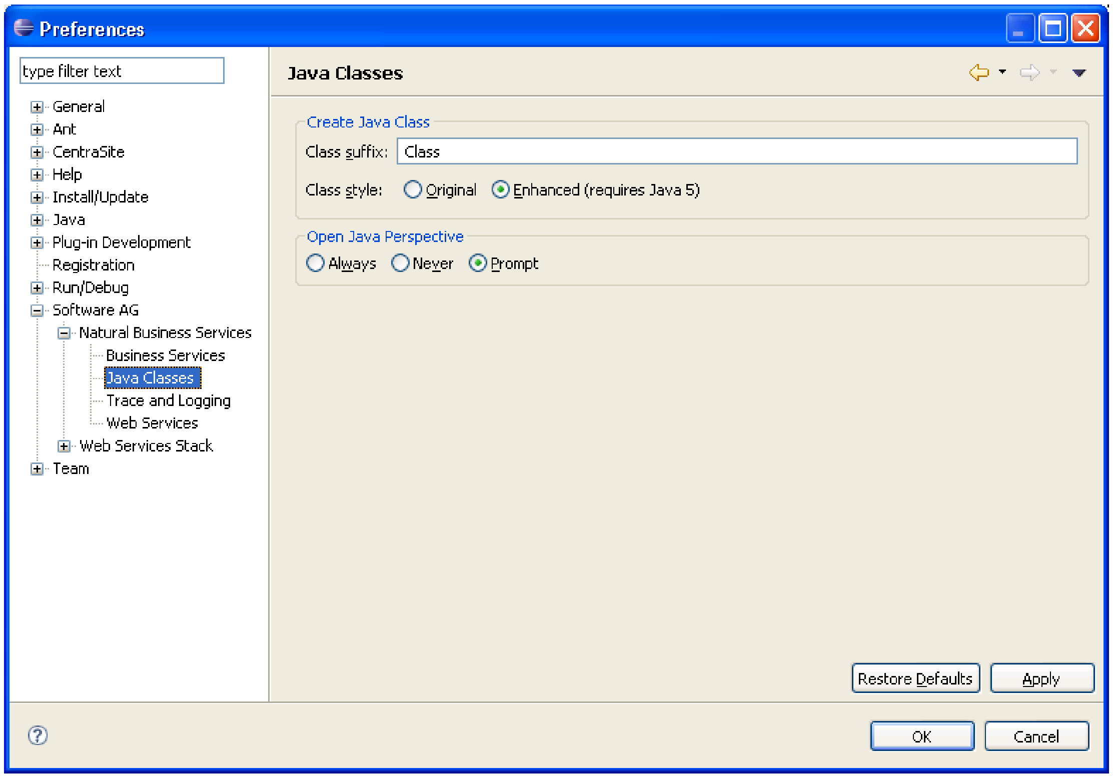
Preferences for Java Classes

Use this setting to define preferences for creating new Java classes.

To set the Java class preferences:

1 Select the **Java Classes** node.

The Java Classes information is displayed. For example:



Java Classes

Create Java Class

Class suffix: Class

Class style: Original [✓ Enhanced (requires Java 5)]

Open Java Perspective

Always Never [✓ Prompt]

2 Type a class suffix in **Class suffix**.

3 Select the target Java Virtual Machine (VM) version in **Class style**.

The class styles are:

- Original (Java VM 1.4)
- Enhanced (Java VM 5)

4 Select one of the Java perspective options.

These options are:

- Always (open the Java perspective when creating Java classes)
- Never (do not open the Java perspective)
- Prompt (allow the user to select the option)

5 Select **OK** to apply the changes and close the **Preferences** window.

Note: To apply the changes without closing the window, select **Apply**.

Preferences for Trace and Log Functions

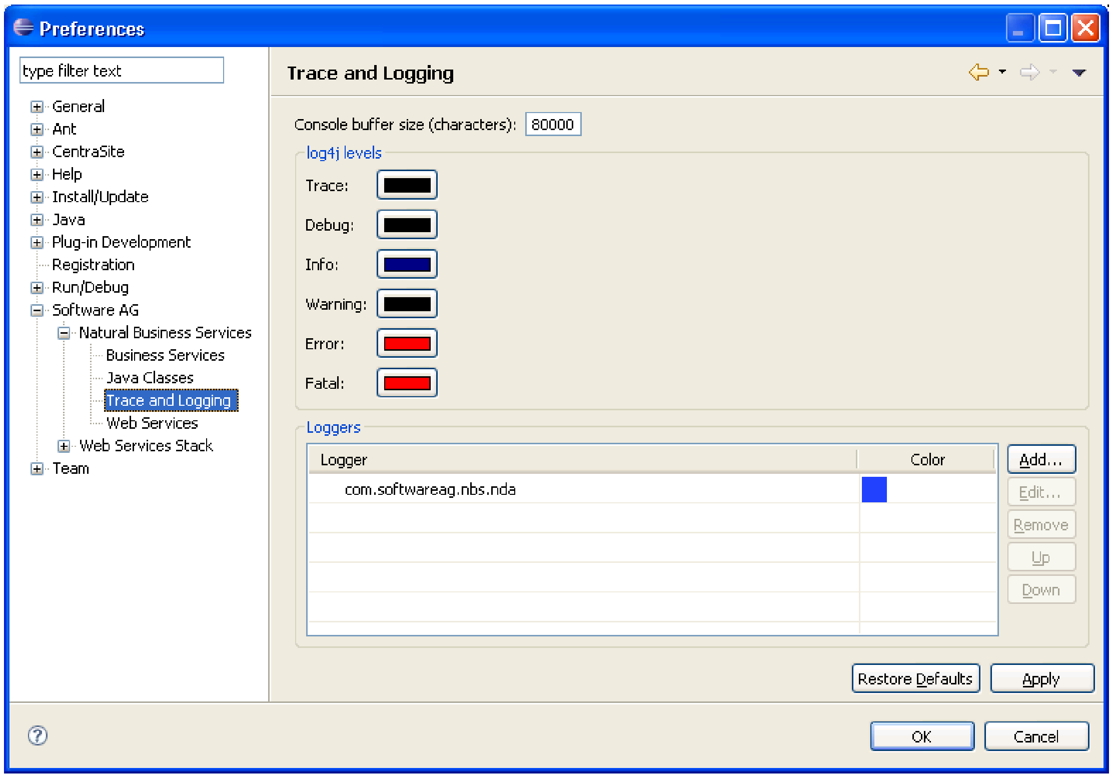
Use this setting to format the trace and log output.

Note: For information on setting trace and log options, see *Set Trace and Log Options*.

To format the trace and log output:

1 Select the **Trace and Logging** node.

The Trace and Logging information is displayed. For example:



Console buffer size (characters): 80000

log4j levels

Trace:	⬛
Debug:	⬛
Info:	🟦
Warning:	⬛
Error:	🟥
Fatal:	🟥

Loggers

Logger	Color
com.softwareag.nbs.nda	🟦

This window displays the default buffer size reserved for the NBS Console in the Eclipse plug-in, the current colors for different log4j levels, and a list of the overridden logger files and corresponding colors.

Note: Color settings specified in the Loggers section take precedence over color settings specified in the log4j levels section.

2 Accept or change the default buffer size reserved for the NBS Console.

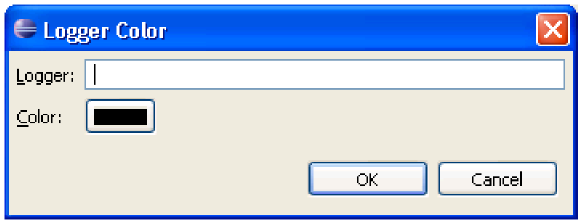
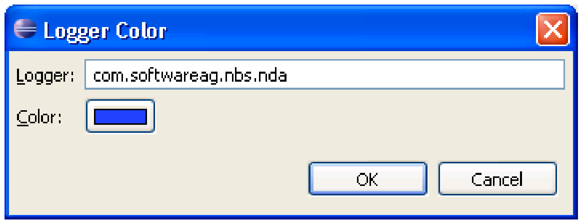
3 Accept or change the current color for each log4j level.

When you select a color, the Color window is displayed. Use this window to select the new color.

4 Accept or change the current logger files and colors.

Note: The logger names listed in this window are derived from the *log4j.properties* file.

You can use the buttons in the Loggers section of this window to:

Task	Procedure
Add a new logger file.	Select **Add**. The **Logger Color** window is displayed. For example:  Logger Color Logger: Color: OK Cancel Type the name of the logger file in **Logger** and select **Color**. The Color window is displayed to select a color to identify the file in the NBS Console view.
Change an existing logger file.	Select the file you want to change and select **Edit**. The **Logger Color** window is displayed, showing information for the file you selected. For example:  Logger Color Logger: com.softwareag.nbs.nda Color: OK Cancel Change the name of the logger file in **Logger** and/or select **Color**. The Color window is displayed to select a new color.
Remove a logger file.	Select the file you want to delete and select **Remove**. The selected logger file is removed from the Loggers section.
Move a logger file up or down in the list.	Select the file you want to move and select **Up** or **Down**. This changes the order of precedence for the files. If one logger file overlaps another, the highest one in the list will be used.

5 Select **OK** to apply the changes and close the **Preferences** window.

Note: To apply the changes without closing the window, select **Apply**.

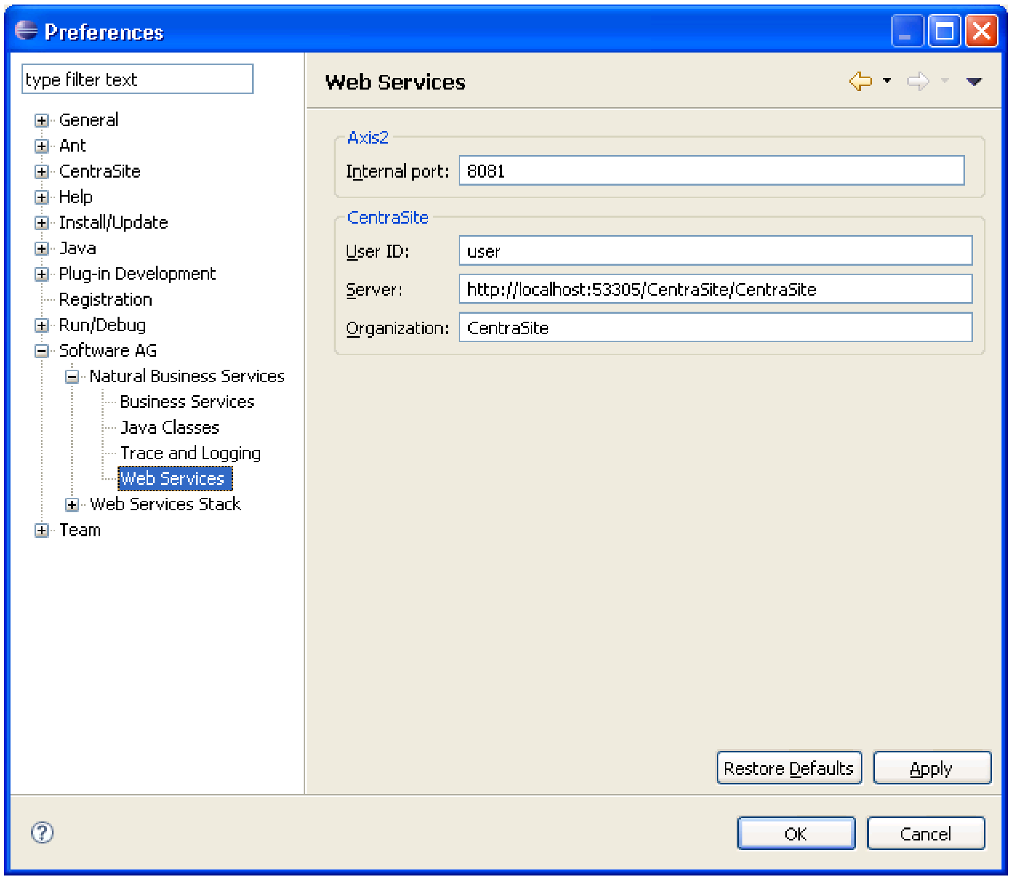
Preferences for Web Services

Use this setting to define the internal port for Axis2 and CentraSite details.

To set the Web service preferences:

1 Select the **Web Services** node.

The Web Services information is displayed. For example:



Axis2

Internal port: 8081

CentraSite

User ID: user

Server: http://localhost:53305/CentraSite/CentraSite

Organization: CentraSite

The current Axis2 internal port number is displayed, as well as information on CentraSite.

2 Accept or change the Axis2 internal port number.

3 Accept or change the CentraSite user ID, server location, and organization name.

4 Select **OK** to apply the changes and close the **Preferences** window.

Note: To apply the changes without closing the window, select **Apply**.

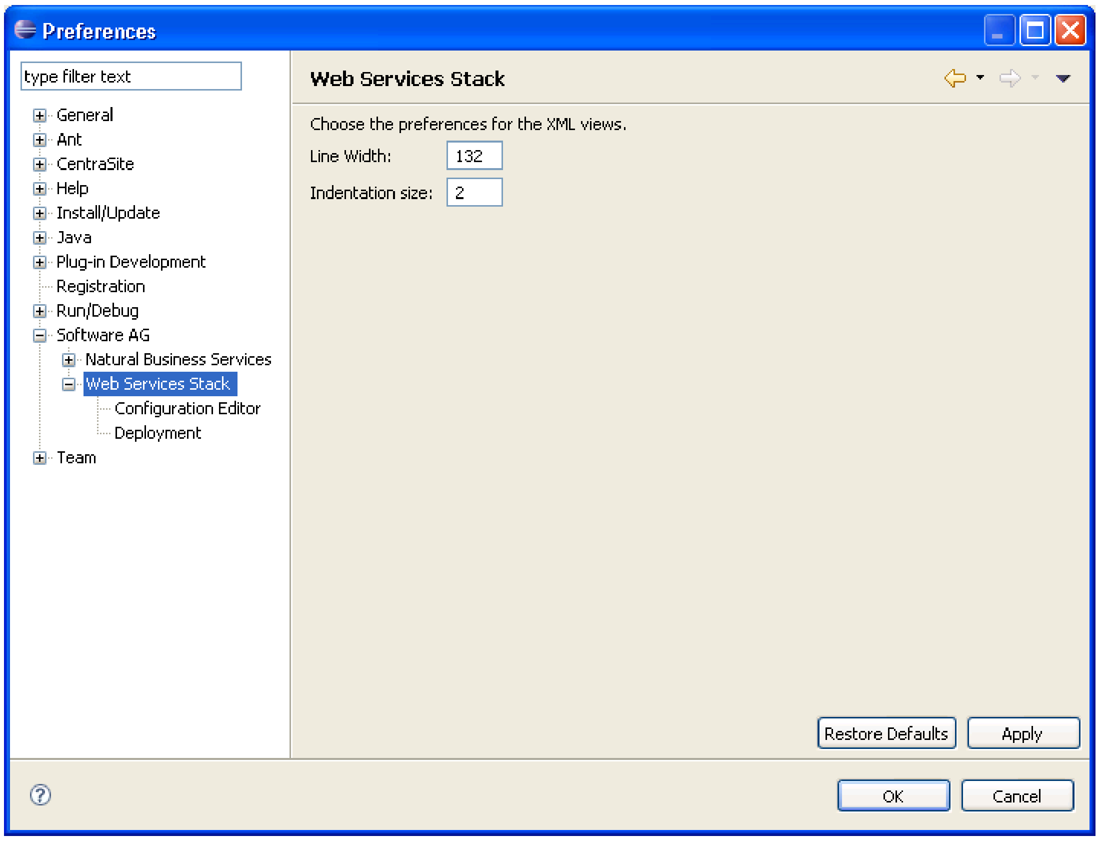
Set Web Services Stack Preferences

This section describes how to set Web service stack preferences for the WS-Stack plug-in.

To set Web service preferences for the WS-Stack plug-in:

1 Select **Preferences** > **Software AG** > **Web Services Stack** on the **Window** menu.

The **Preferences** window for Web Services Stack is displayed. For example:



Web Services Stack

Choose the preferences for the XML views.

Line Width: 132

Indentation size: 2

Use this window to:

Task	Procedure
Change the line width used for XML views.	Enter another value in **Line width**.
Change the size of the indentation used for XML views.	Enter another value in **Indentation size**.

2 Select **OK** to apply the changes and close the **Preferences** window.

Note: To apply the changes without closing the window, select **Apply**.

In addition to these preferences, you can also set the following:

- Preferences for the Configuration Editor
- Preferences for Deployment

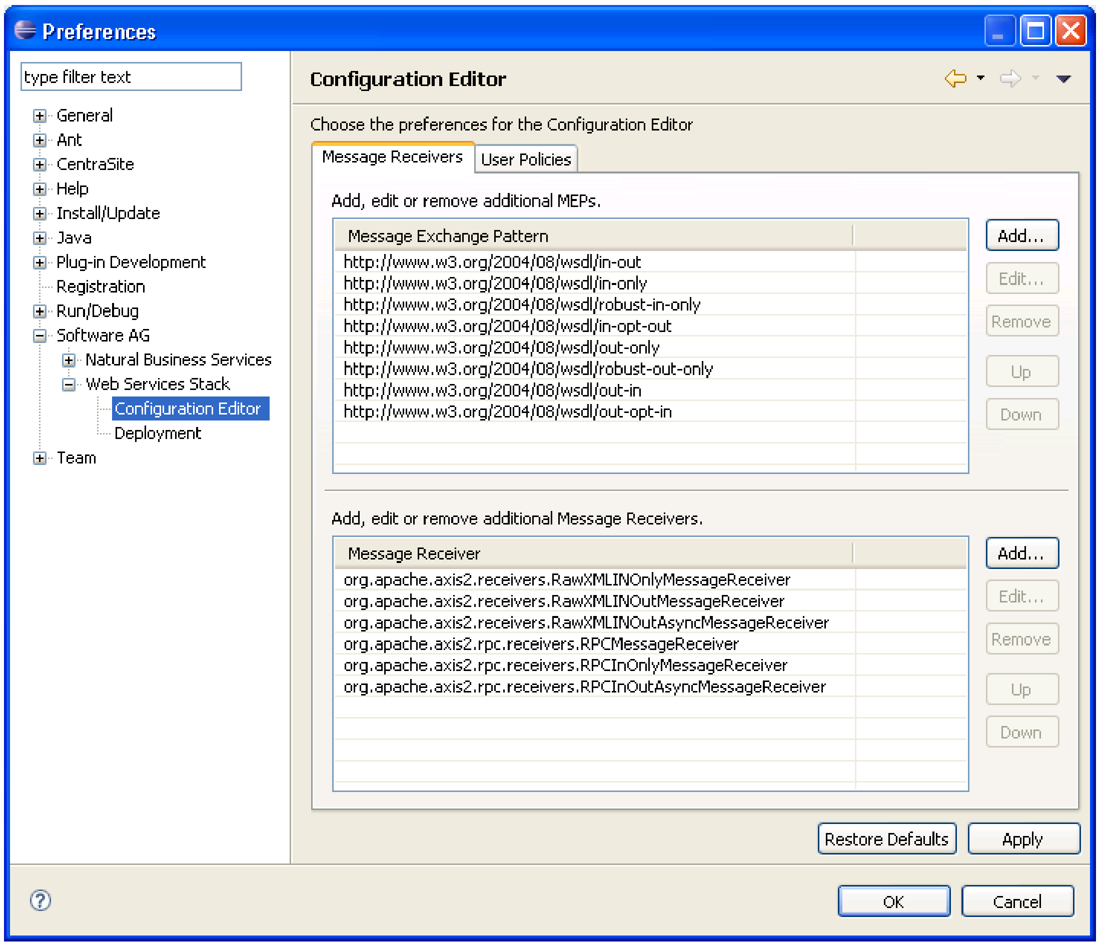
Preferences for the Configuration Editor

Use this setting to define preferences for the Web services Configuration editor.

To set preferences for the Configuration editor:

1 Select the **Configuration Editor** node.

The Configuration Editor information is displayed for message exchange patterns and message receivers. For example:



Configuration Editor

Choose the preferences for the Configuration Editor

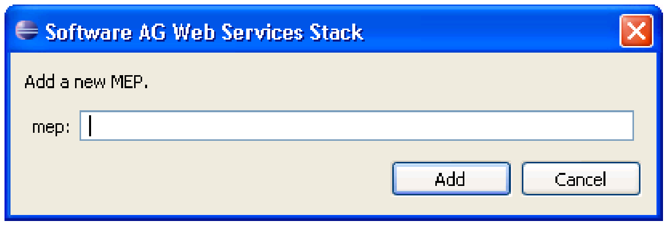
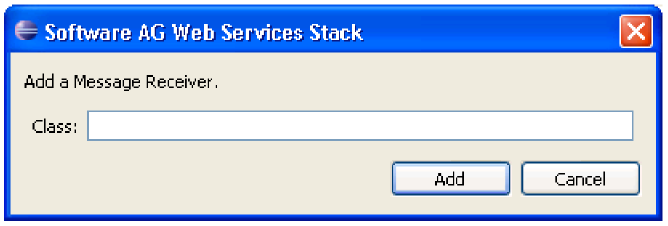
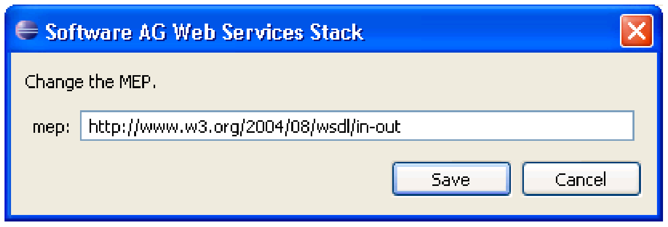
Add, edit or remove additional MEPs.

Message Exchange Pattern
http://www.w3.org/2004/08/wsdl/in-out
http://www.w3.org/2004/08/wsdl/in-only
http://www.w3.org/2004/08/wsdl/robust-in-only
http://www.w3.org/2004/08/wsdl/in-opt-out
http://www.w3.org/2004/08/wsdl/out-only
http://www.w3.org/2004/08/wsdl/robust-out-only
http://www.w3.org/2004/08/wsdl/out-in
http://www.w3.org/2004/08/wsdl/out-opt-in

Add, edit or remove additional Message Receivers.

Message Receiver
org.apache.axis2.receivers.RawXMLINOnlyMessageReceiver
org.apache.axis2.receivers.RawXMLINOutMessageReceiver
org.apache.axis2.receivers.RawXMLINOutAsyncMessageReceiver
org.apache.axis2.rpc.receivers.RPCMessageReceiver
org.apache.axis2.rpc.receivers.RPCInOnlyMessageReceiver
org.apache.axis2.rpc.receivers.RPCInOutAsyncMessageReceiver

The current message exchange patterns and message receivers are displayed. Use this window to:

Task	Procedure
Add a new message exchange pattern.	Select **Add** in Message Exchange Pattern. The Add a new MEP window is displayed. For example:  Software AG Web Services Stack Add a new MEP. mep: Enter the name of the message exchange pattern and select **Add**. The pattern is added to the list.
Add a new message receiver.	Select **Add** in Message Receiver. The Add a Message Receiver window is displayed. For example:  Software AG Web Services Stack Add a Message Receiver. Class: Enter the name of the message receiver and select **Add**. The receiver is added to the list.
Edit a message exchange pattern or message receiver.	Select the pattern or receiver in the list and select **Edit**. The pattern or receiver is displayed for editing. For example:  Software AG Web Services Stack Change the MEP. mep: http://www.w3.org/2004/08/wsdl/in-out Change the pattern (or receiver) information and select **Save** to save the changes.
Remove a message exchange pattern or message receiver.	Select the pattern or receiver in the list and select **Remove**.
Change the sequence in which the message	Select the pattern or receiver in the list and select **Up** or **Down**.

Task	Procedure
exchange patterns or message receivers are listed.	
Set user policies.	Select the User Policies tab. For information, see ***User Policies Tab***.

2 Accept or change the message exchange patterns, message receivers, and/or user policies for Web services.

3 Select **OK** to apply the changes and close the **Preferences** window.

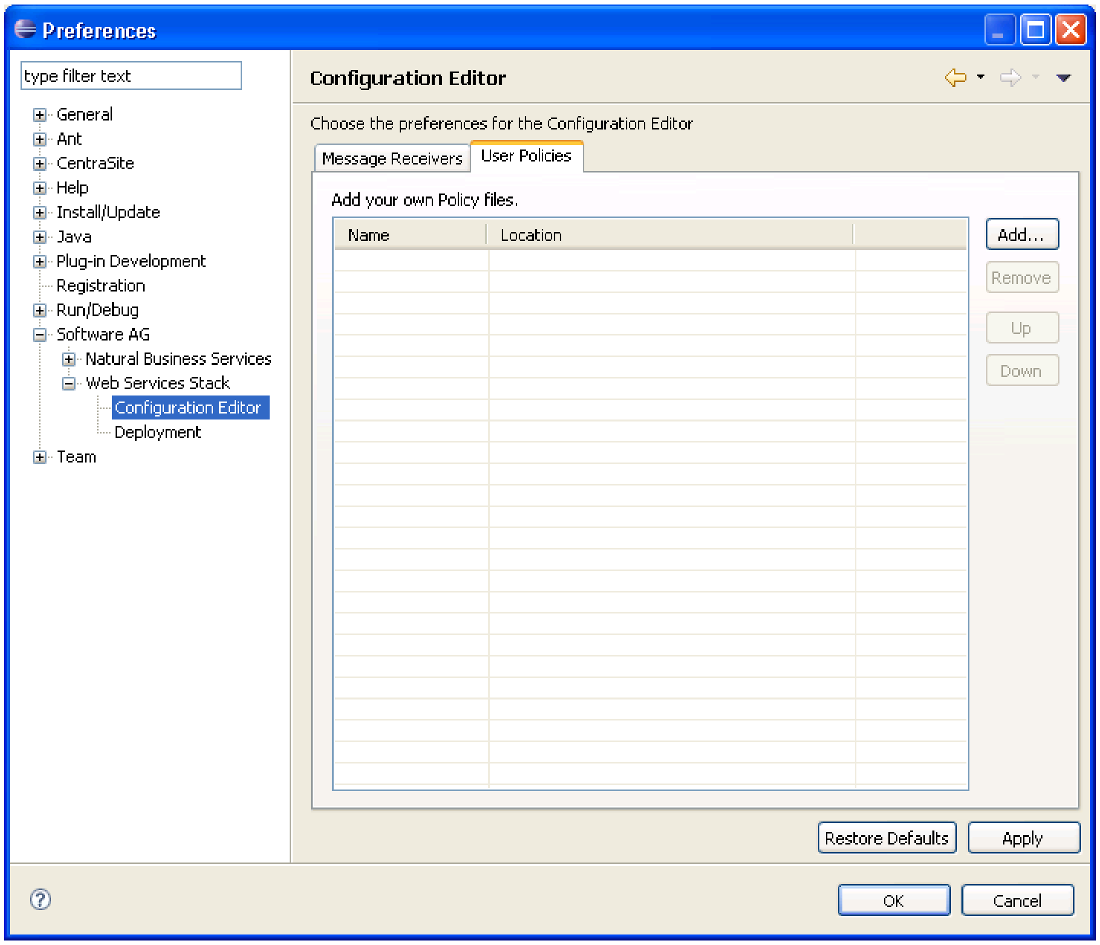
Note: To apply the changes without closing the window, select **Apply**.

User Policies Tab

To set user policies for Web services:

- Select the User Policies tab in the **Configuration Editor** node.

  The Configuration Editor information is displayed for user policies. For example:



Use this window to:

Task	Procedure
Add a new user policy.	Select **Add** and browse for the XML file containing the user policy you want to add.
Remove an existing user policy.	Select the user policy in the list and select **Remove**.
Change the sequence in which the user policies are listed.	Select the user policy and select **Up** or **Down**.

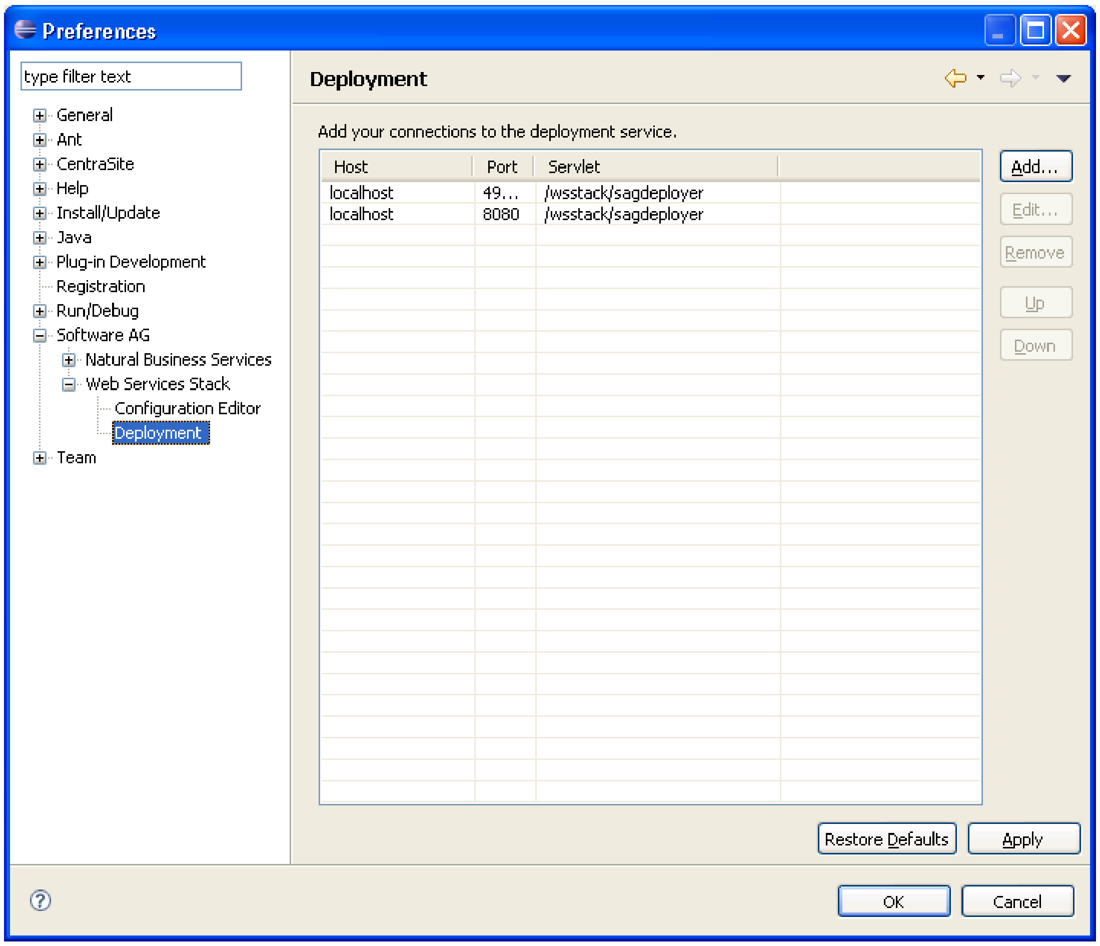
Preferences for Deployment

Use this setting to define preferences for the deployment of Web services.

To set preferences for Web service deployment:

1 Select the **Deployment** node.

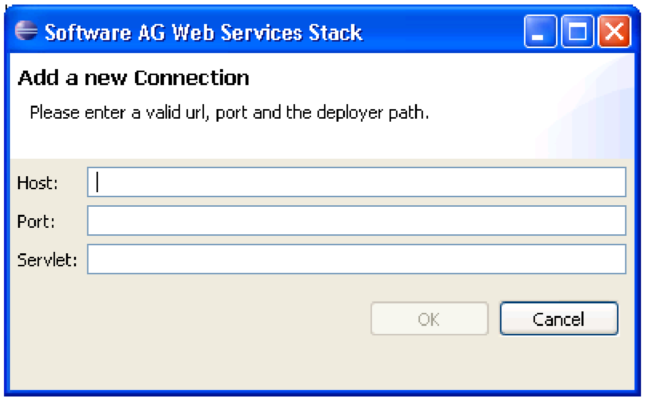
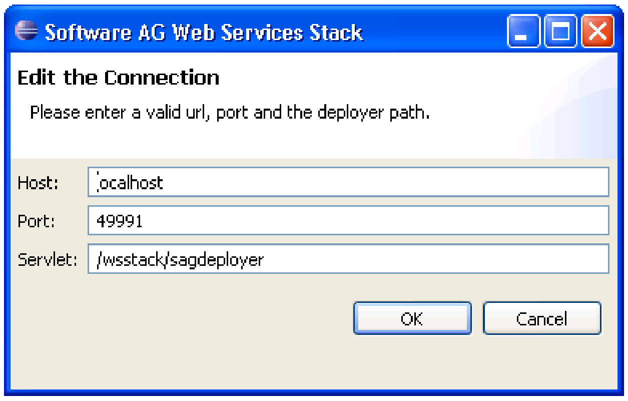
The Deployment information for Web services is displayed. For example:



Add your connections to the deployment service.

Host	Port	Servlet
localhost	49...	/wsstack/sagdeployer
localhost	8080	/wsstack/sagdeployer

The current connections for deployment services are displayed. Use this window to:

Task	Procedure
Add a new deployment connection.	Select **Add**. The **Add a New Connection** window is displayed. For example:  Software AG Web Services Stack **Add a new Connection** Please enter a valid url, port and the deployer path. Host: Port: Servlet: Enter the host name, port number, and servlet address for the new connection and select **OK**. The connection is added to the list.
Edit a deployment connection.	Select the connection in the list and select **Edit**. The connection is displayed for  Software AG Web Services Stack **Edit the Connection** Please enter a valid url, port and the deployer path. Host: localhost Port: 49991 Servlet: /wsstack/sagdeployer editing. For example: Change the connection information and select **OK** to save the changes.
Remove a deployment connection.	Select the connection in the list and select **Remove**.
Change the sequence in which the connections are listed.	Select the connection in the list and select **Up** or **Down**.

2 Accept or change the deployment connections for Web services.

3 Select **OK** to apply the changes and close the **Preferences** window.

Note: To apply the changes without closing the window, select **Apply**.

3 Developing Business Services

Note: You must use a SPoD connection to create, test, or deploy business services.

Create a Business Service

A business service consists of a collection of methods related to a common business entity. Use the Business Service wizard in the Eclipse plug-in to create and maintain a business service. This wizard generates a subprogram proxy to translate data (including Unicode formats) and then adds an entry in the **NBS Repositories** view. It also generates additional subprograms as required.

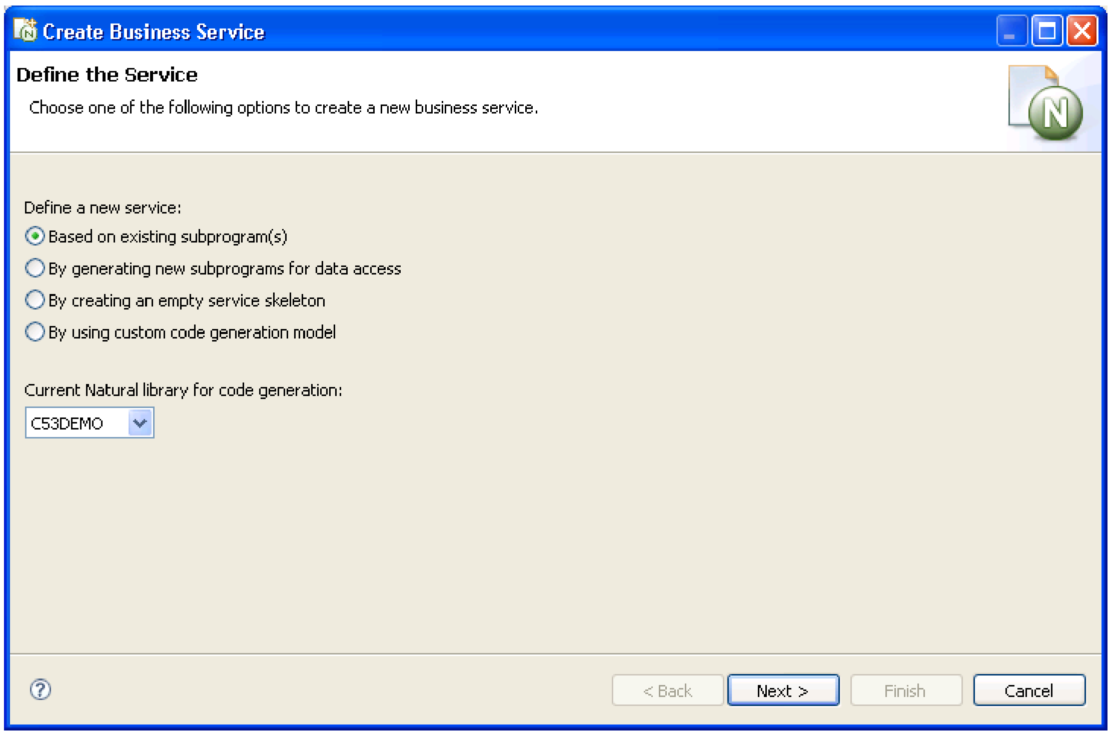
To create a new business service:

1 Open the context menu for the SPoD connection in the **NBS Repositories** view.

For an example of this menu, see *Access Connection Options*.

2 Select **Create business service**.

The **Define the Service** panel is displayed. For example:



Define the Service

Choose one of the following options to create a new business service.

Define a new service:

[✓ Based on existing subprogram(s)]

By generating new subprograms for data access

By creating an empty service skeleton

By using custom code generation model

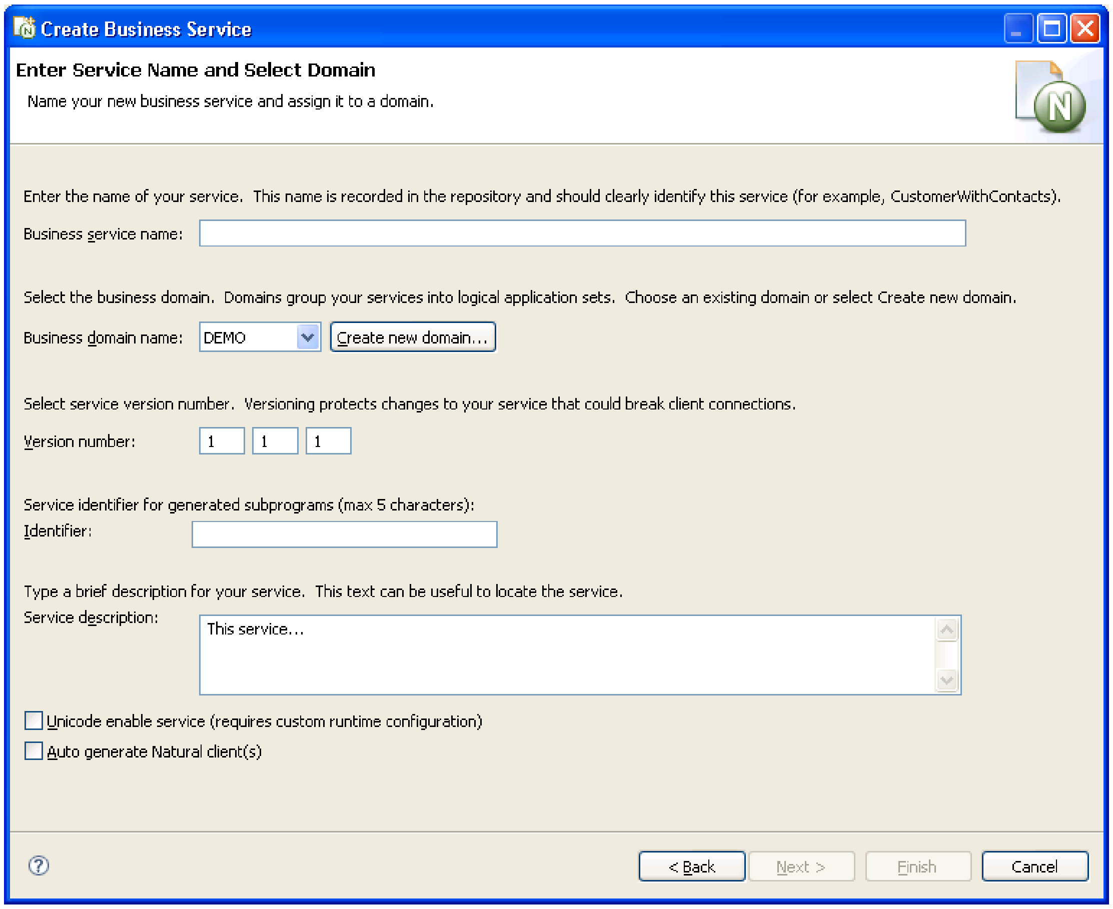
Current Natural library for code generation: C53DEMO

3 Select one of the following options:

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

Optionally, you can change the library in which to generate the service.

Depending on which option you select, one or more additional panels are displayed. After specifying the appropriate information, select **Next** on the last specification panel. The **Enter Service Name and Select Domain** panel is displayed. For example:



Enter Service Name and Select Domain

Name your new business service and assign it to a domain.

Enter the name of your service. This name is recorded in the repository and should clearly identify this service (for example, CustomerWithContacts).

Business service name:

Select the business domain. Domains group your services into logical application sets. Choose an existing domain or select Create new domain.

Business domain name: DEMO

Select service version number. Versioning protects changes to your service that could break client connections.

Version number: 1 1 1

Service identifier for generated subprograms (max 5 characters):

Identifier:

Type a brief description for your service. This text can be useful to locate the service.

Service description: This service...

☐ Unicode enable service (requires custom runtime configuration)

☐ Auto generate Natural client(s)

Use this panel to name your business service and assign it 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** if your service has Unicode fields (U data type). Services that use a Unicode interface require a special EntireX runtime server definition. For information, see *Use Unicode Parameters for Your Business Service*.
Generate Natural clients.	Select **Auto generate Natural clients**. This option automatically generates Natural clients, which can be used to invoke the business service from a Natural environment. The Business Service wizard generates a Natural client for each server proxy your business service creates. For information, see *Natural Business Services Subprogram-Proxy-Client Model*.

4 Enter the name of your service in **Business service name**.

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

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

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 for your service in **Service description**.

This description can be useful when searching for a service.

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.

Note: For information on testing your 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.

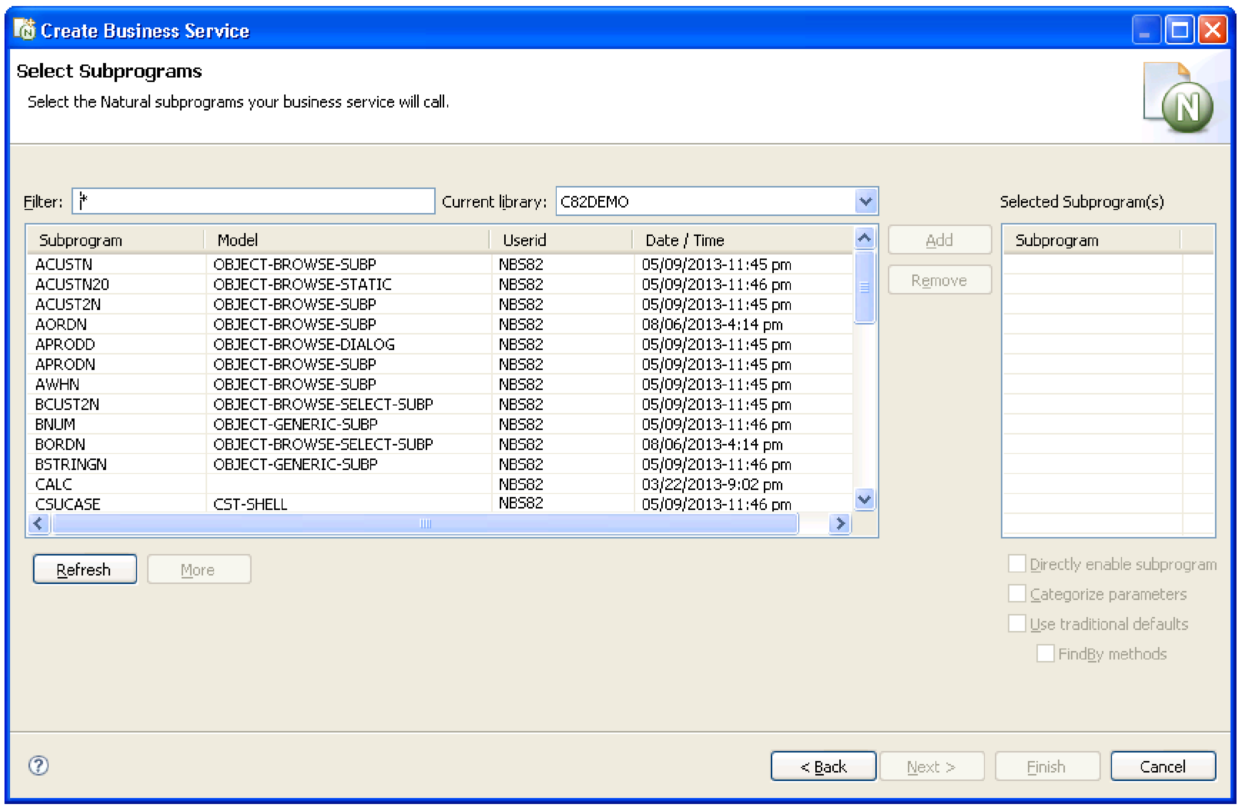
- Categorize parameters

  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 the **Based on existing subprogram(s)** option on the **Define the Service** panel, the **Select Subprograms** panel is displayed. For example:


Create Business Service
Select Subprograms
Select the Natural subprograms your business service will call.

Filter: *   Current library: C82DEMO

Subprogram	Model	Userid	Date / Time
ACUSTN	OBJECT-BROWSE-SUBP	NBS82	05/09/2013-11:45 pm
ACUSTN20	OBJECT-BROWSE-STATIC	NBS82	05/09/2013-11:46 pm
ACUST2N	OBJECT-BROWSE-SUBP	NBS82	05/09/2013-11:45 pm
AORDN	OBJECT-BROWSE-SUBP	NBS82	08/06/2013-4:14 pm
APRODD	OBJECT-BROWSE-DIALOG	NBS82	05/09/2013-11:45 pm
APRODN	OBJECT-BROWSE-SUBP	NBS82	05/09/2013-11:45 pm
AWHN	OBJECT-BROWSE-SUBP	NBS82	05/09/2013-11:45 pm
BCUST2N	OBJECT-BROWSE-SELECT-SUBP	NBS82	05/09/2013-11:45 pm
BNUM	OBJECT-GENERIC-SUBP	NBS82	05/09/2013-11:46 pm
BORDN	OBJECT-BROWSE-SELECT-SUBP	NBS82	08/06/2013-4:14 pm
BSTRINGN	OBJECT-GENERIC-SUBP	NBS82	05/09/2013-11:46 pm
CALC		NBS82	03/22/2013-9:02 pm
CSUCASE	CST-SHELL	NBS82	05/09/2013-11:46 pm

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 for the current library is displayed.

   - If all subprograms cannot be displayed in the space provided, use the scroll bar 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 C82DEMO will be displayed.
   - If desired, select a different library from **Current library**.
2. Select the subprogram you want to use.
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:

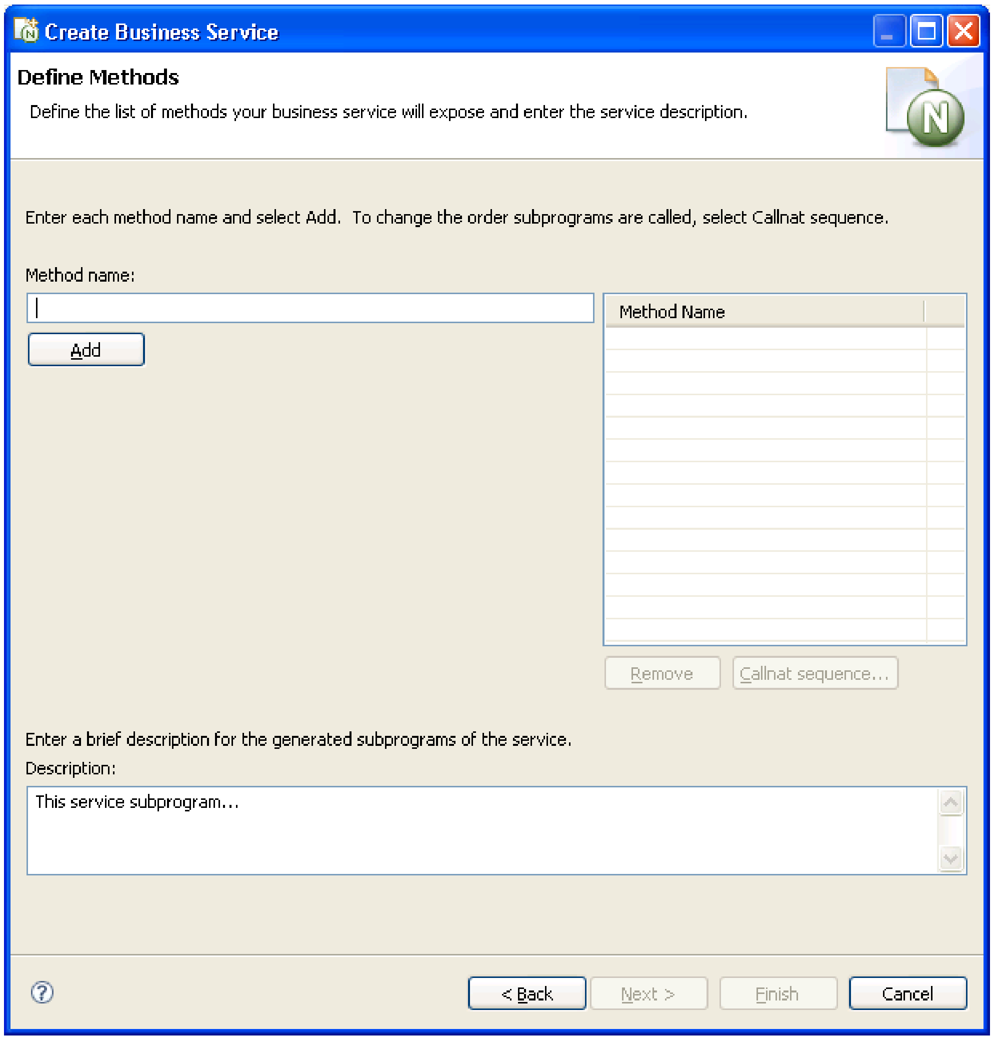
- Generate a subprogram proxy
- Populate the service repository on the server
- Generate a subprogram to call one or more existing subprograms

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:



Define Methods

Define the list of methods your business service will expose and enter the service description.

Enter each method name and select Add. To change the order subprograms are called, select Callnat sequence.

Method name:

Add

Method Name

Remove

Callnat sequence...

Enter a brief description for the generated subprograms of the service.

Description:

This service subprogram...

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.

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 subprogram**.

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

Note: 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.

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

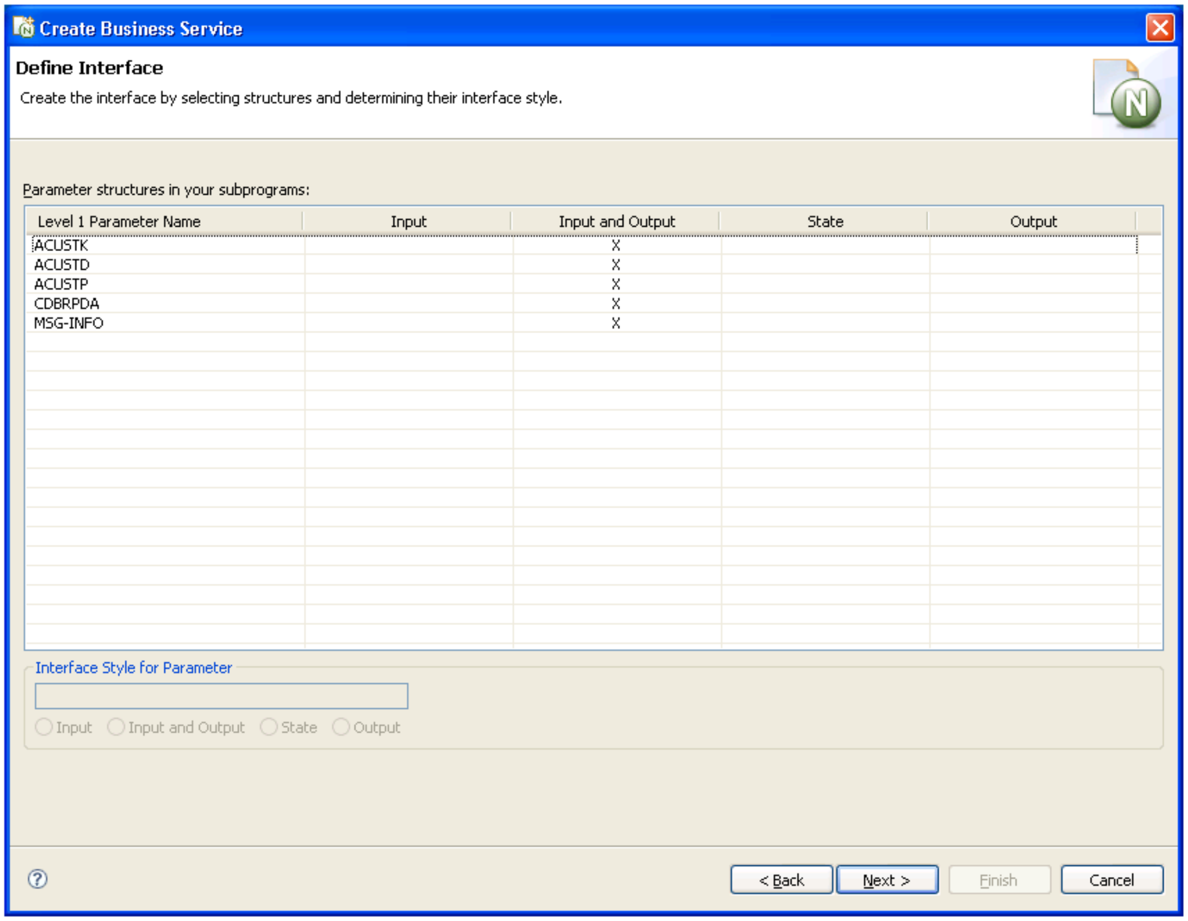
When categorizing parameters, you must ensure that parameters containing the same name (but different values) are not placed in the same category. If this happens, the subprogram will generate but not compile and you will have to decide how to handle the duplicate names.

When two subprograms include 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.

When you select this option on the **Select Subprograms** panel, the **Define Interface** panel is displayed when you select **Next**. For example:


Create Business Service
Define Interface
Create the interface by selecting structures and determining their interface style.
Parameter structures in your subprograms:

Level 1 Parameter Name	Input	Input and Output	State	Output
ACUSTK		X		
ACUSTD		X		
ACUSTP		X		
CDBRPDA		X		
MSG-INFO		X		

Interface Style for Parameter
Input
Input and Output
State
Output
< Back
Next >
Finish
Cancel

This panel lists each level 1 structure in the selected subprograms. By default, the parameters use the input/output interface style.

To change the interface style for a parameter, select the parameter and 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/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.

Tip: If parameters are not unique within each parameter structure, a compile time error may occur. You can fix this error in the Natural editor.

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

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 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 new code to keep.

For example, if you regenerated the CalculatorAdvance service in the DEMO domain, 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. For information, see ***Define Methods***.

Change the Method Callnat Sequence

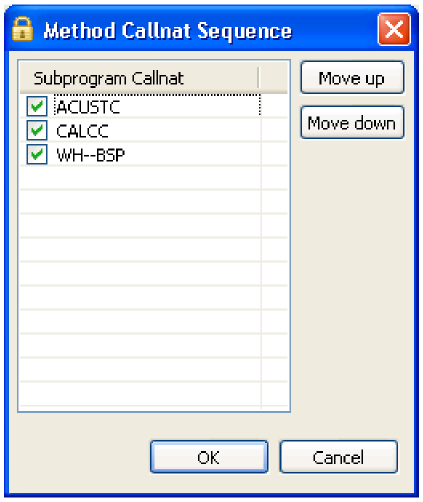
By default, the subprograms are executed in the order they were selected on the **Select subprogram(s)** panel.

To change the method Callnat sequence:

1 Select a method from **Method Name** 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 (select a subprogram and select **Move up** or **Move down**) or to deselect a subprogram that should not be executed with this method.

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 current library when the wizard is invoked.

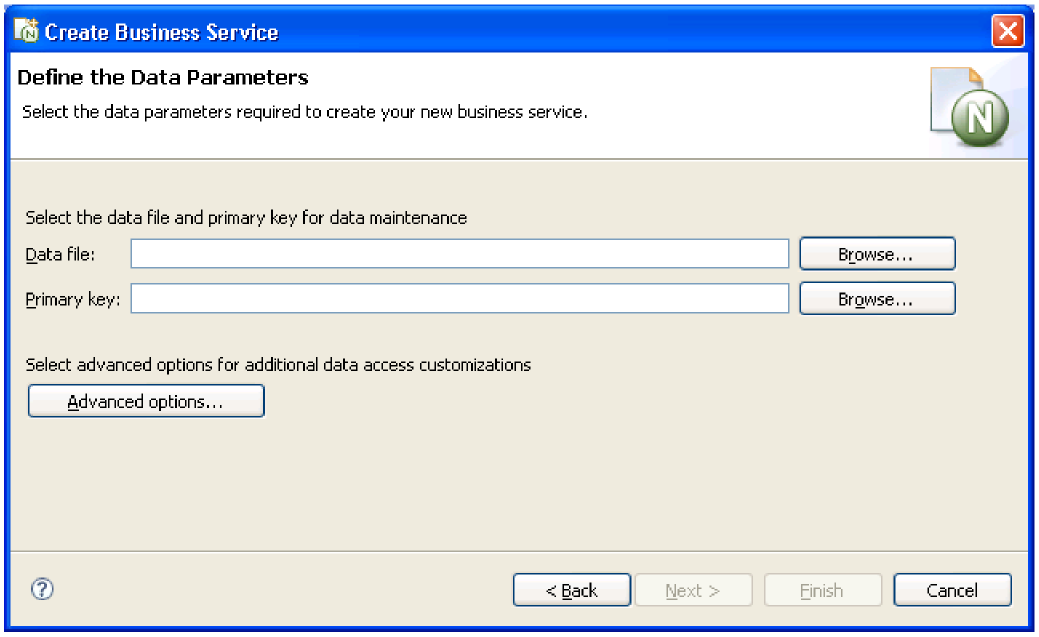
Important: If you make any changes to the exposed interface in this subprogram (i.e., changes to the PDAs), you must regenerate the service proxy. For information, see *Regenerate a Service Proxy*.

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 **NBS Repositories** view), as well as at least one Natural Construct object subprogram. Depending on which data access type is selected on the **Select 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

When you select the **By generating new subprograms for data access** option on the **Define the Service** panel, the **Define the Data Parameters** panel is displayed. For example:



Create Business Service

Define the Data Parameters

Select the data parameters required to create your new business service.

Select the data file and primary key for data maintenance

Data file:

Primary key:

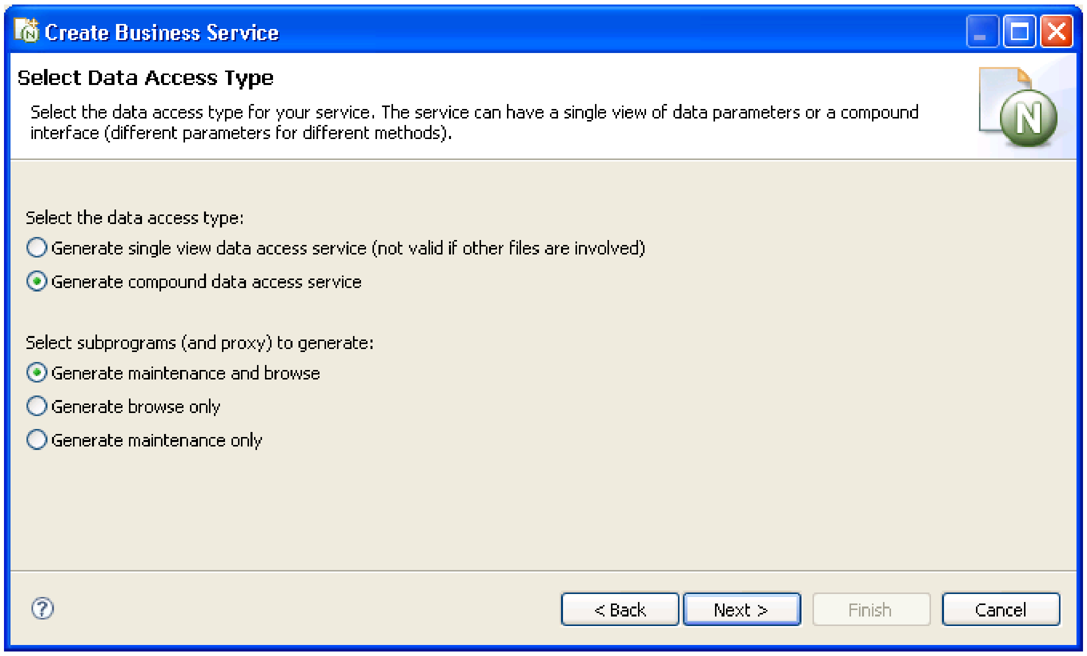
Select advanced options for additional data access customizations

Use this panel to indicate the name of the file and the primary key used for data maintenance. To select advanced options for additional data access customizations, select **Advanced options**. For information about this window, see *Specify Advanced Options for Data Access*.

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

   This file must currently exist in Predict.
2. Select the primary key for the specified file from **Primary key**.
3. Select **Next**.

   The **Select Data Access Type** panel is displayed. For example:



Select Data Access Type

Select the data access type for your service. The service can have a single view of data parameters or a compound interface (different parameters for different methods).

Select the data access type:

( ) Generate single view data access service (not valid if other files are involved)

(✓) Generate compound data access service

Select subprograms (and proxy) to generate:

(✓) Generate maintenance and browse

( ) Generate browse only

( ) Generate maintenance only

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.
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.

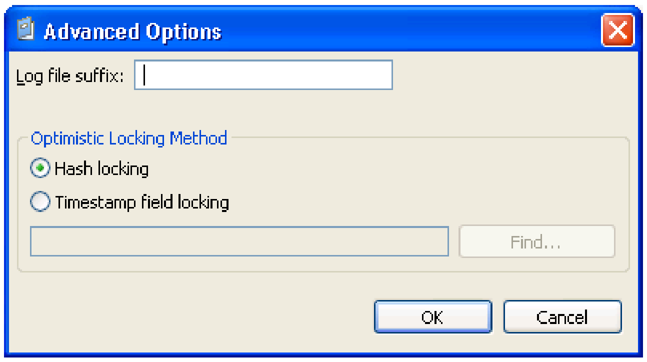
Specify Advanced Options for Data Access

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 **Define the Data Parameters** panel.

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



The first option in this window allows the wizard to generate code to maintain a log file whenever data is modified through this business service. The second option specifies the record-locking method. 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.

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.

3 Select one of the record-locking options.

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.

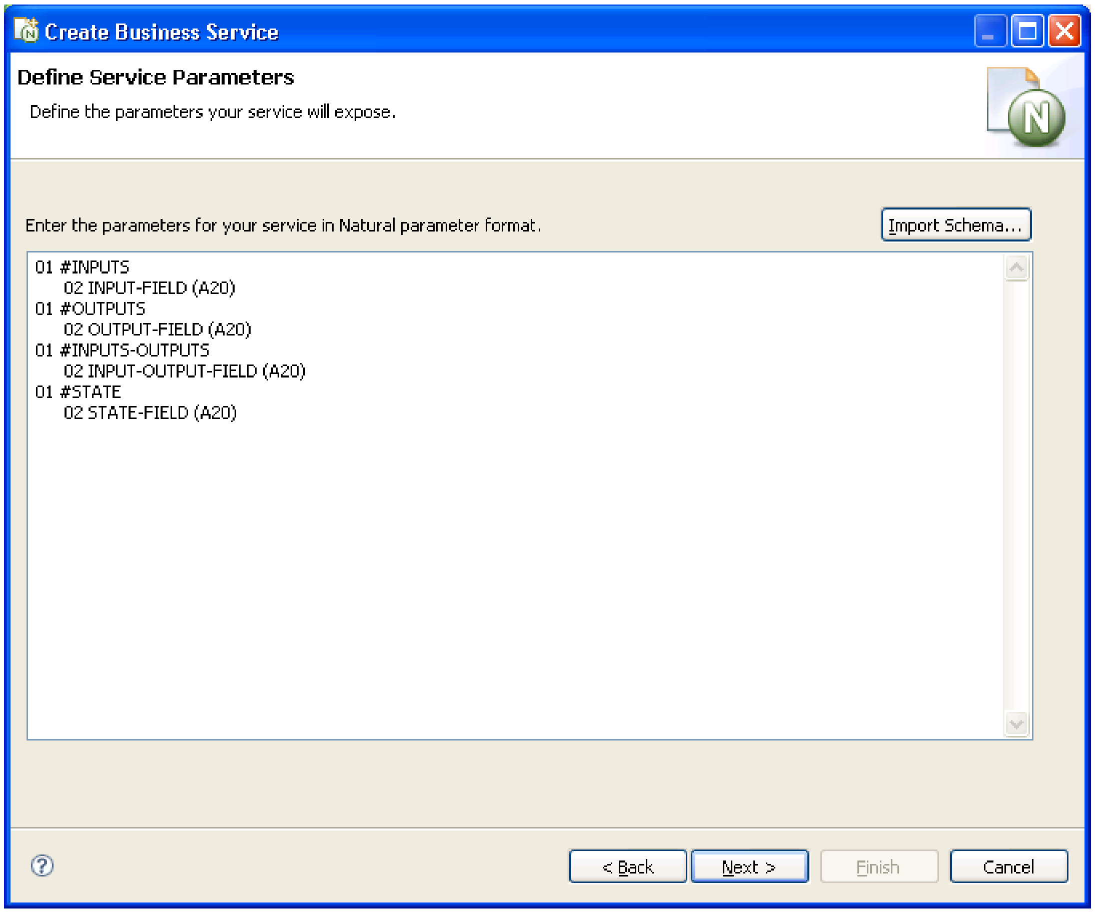
The Business Service wizard automatically generates the standard methods, but you can also add custom methods to the object browse-select subprogram. For information about this model, see *Object-Browse-Select-Subp Model*.

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 service repository on the server. You can edit the new subprogram in the program editor view.

Note: For information on determining the module names that belong to a business service and loading the modules into the editor, see *Edit Service Modules*.

When you select **By creating an empty service skeleton** on the **Define the Service** panel, the **Define Service Parameters** panel is displayed. For example:


Create Business Service
Define Service Parameters
Define the parameters your service will expose.
Enter the parameters for your service in Natural parameter format.
Import Schema...
< Back
Next >
Finish
Cancel

```
01 #INPUTS
    02 INPUT-FIELD (A20)
01 #OUTPUTS
    02 OUTPUT-FIELD (A20)
01 #INPUTS-OUTPUTS
    02 INPUT-OUTPUT-FIELD (A20)
01 #STATE
    02 STATE-FIELD (A20)
```

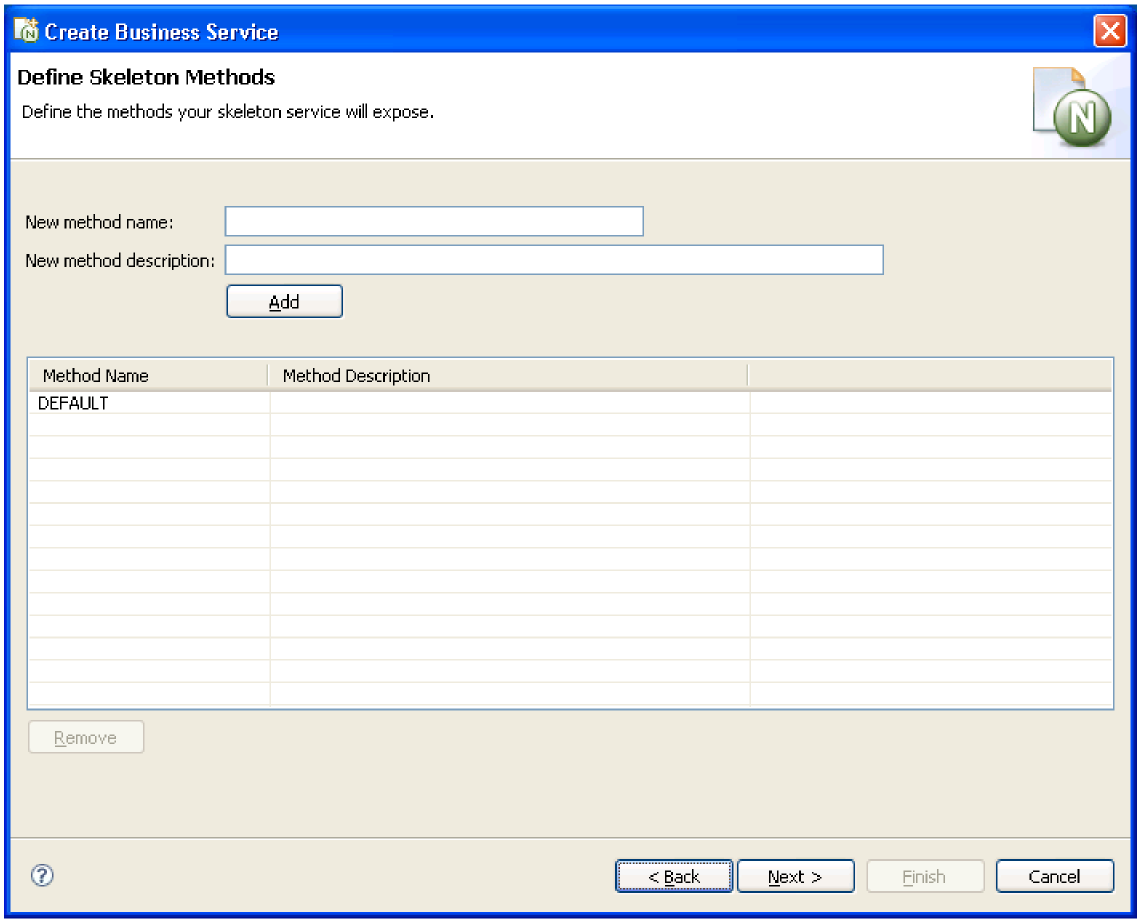
Use this panel to define parameters for the business service. Optionally, you can select **Import schema** to import an external schema to use as the data parameters for the service skeleton. For information, see *Import Schema*.

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:



Create Business Service

Define Skeleton Methods

Define the methods your skeleton service will expose.

New method name:

New method description:

Method Name	Method Description
DEFAULT	

   This panel lists the methods your skeleton service will expose to the user (DEFAULT in this example).

   - To add a new method, type the method name in **New method name**, a brief description of the method in **New method description**, and select **Add**.
   - To remove a method, select the method name and select **Remove**.

3. Select **Next**.

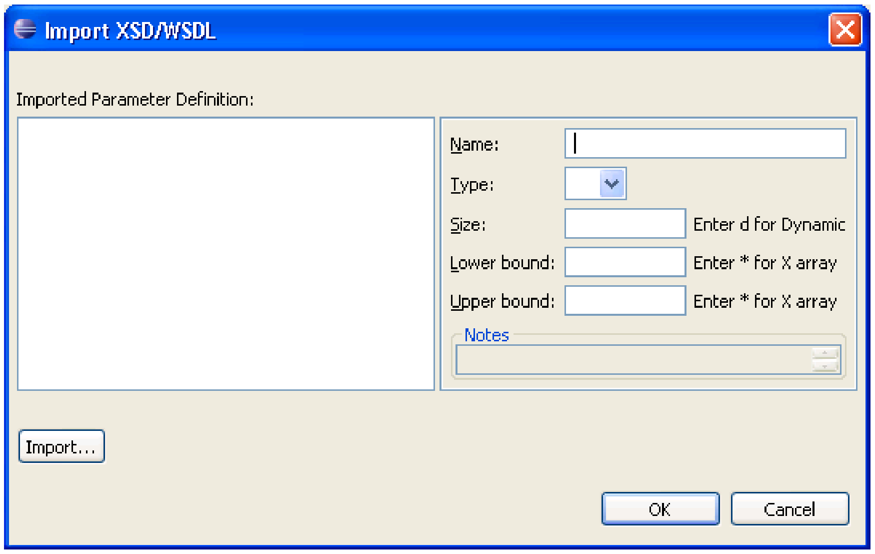
   The last specification panel for the Business Service wizard is displayed.

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 dynamic or X-array variables). The schema can 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.

2 Select **Import**.

A selection window is displayed to select the schema to import. Once the fields are displayed in **Imported parameter definition**, you can change the following options for a selected field:

Option	Description
Name	Name of the field.
Type	Data type.
Size	Use of dynamic variables (either type or remove "D").
Lower bound	Use of X-array variables for the lower bound (either type or remove "*").
Upper bound	Use of X-array variables for the upper bound (either type or remove "*").

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.

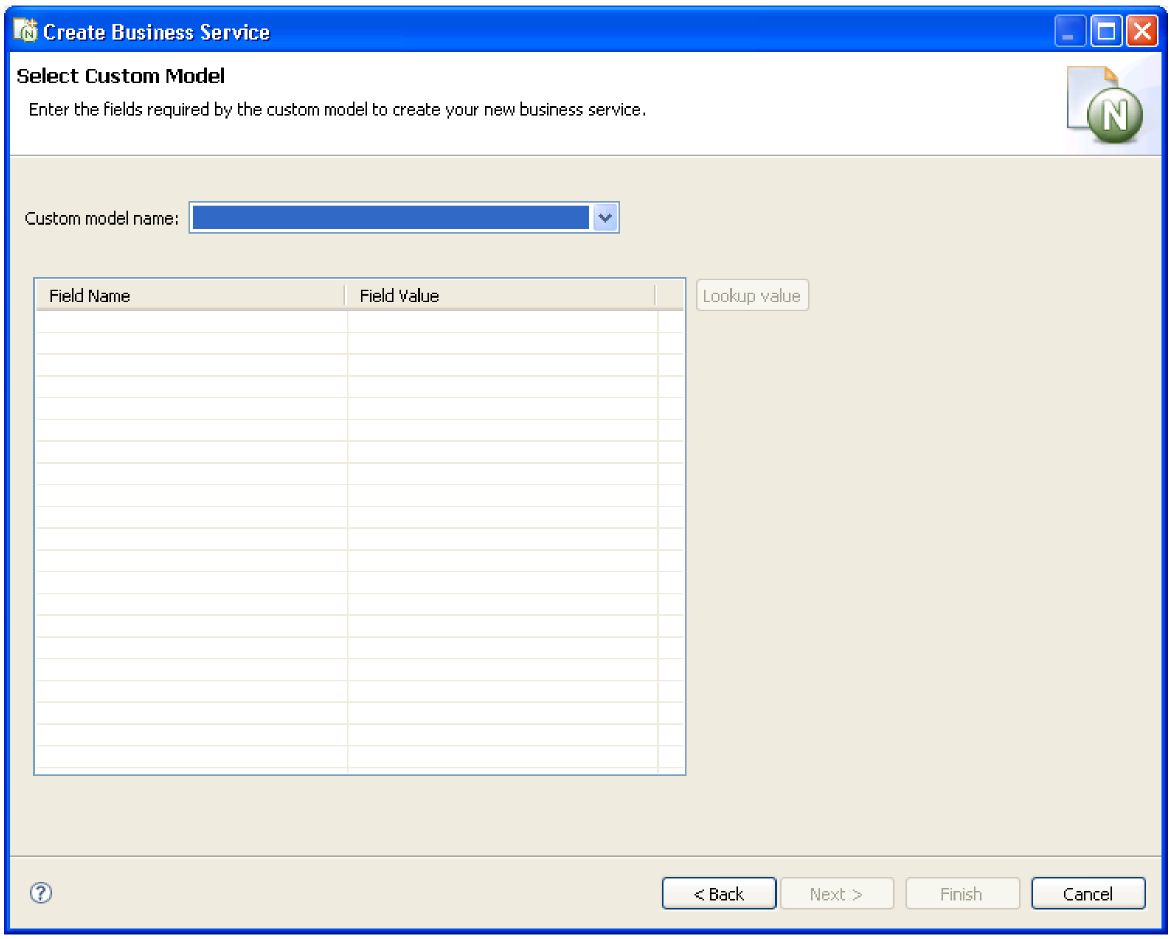
Note: To implement this option, the **Custom models file** field in the **Preferences** window for Natural Business Services must be set to a valid *models.xml* file. For information, see *Set Natural Business Service Preferences*.

During installation, the default *models.xml* file is copied to the following folder:

C:\Documents and Settings\All Users\Application Data\Software AG\Natural Business Services\Configuration

Note: If the *models.xml* file currently exists in this folder, the new file will not be copied. This protects any changes you made to your *models.xml* file from being overwritten. In this case the new *models.xml* file is also copied to the NBS install folder (*C:\Program Files\Software AG\Natural Business Services\Vn.n*).

When you select the **By using custom code generation model** option on the **Define the Service** panel, the **Select Custom Model** panel is displayed. For example:

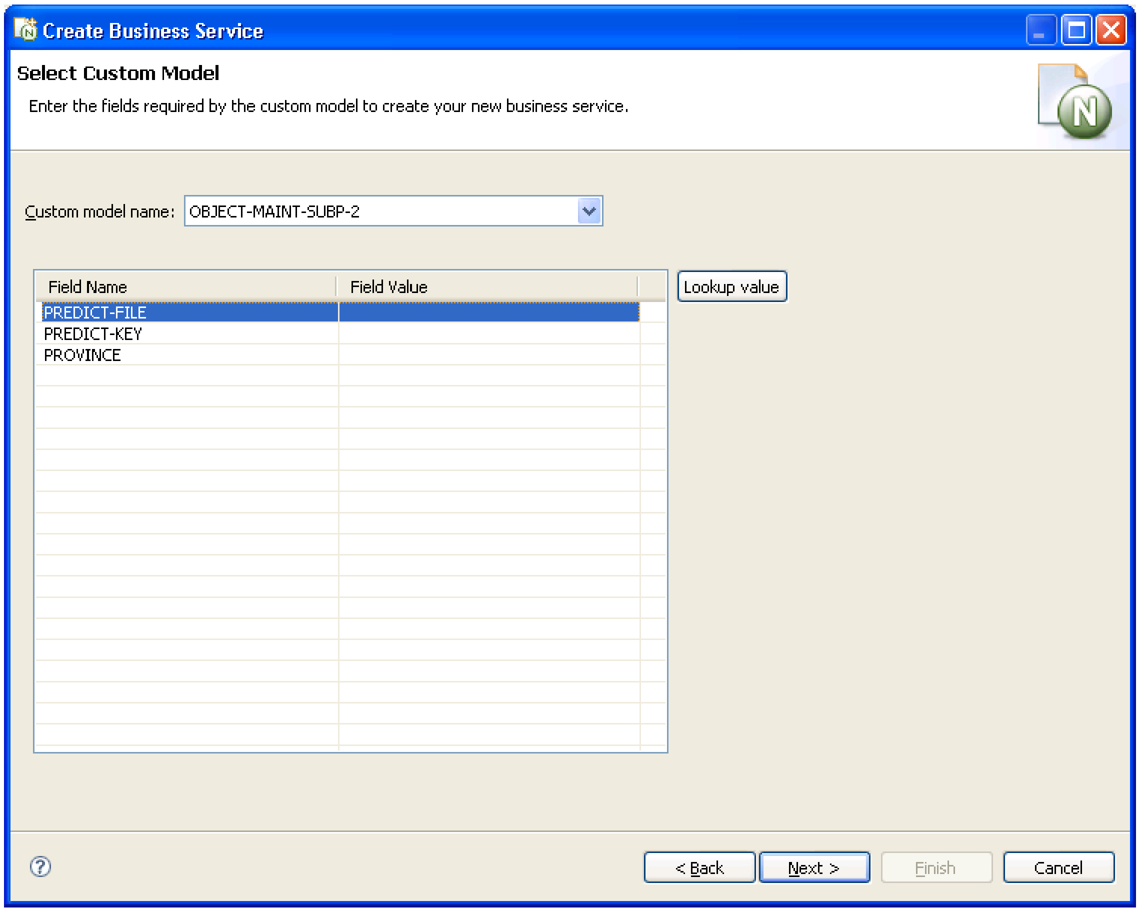


1. Select the custom model from **Custom model name**.

   The fields available for the custom model are displayed.

2. Specify the field names and values.

   If a field does not have lookup values available, the value must be manually entered. If lookup logic is available, the **Lookup value** button becomes active when you select the field. For example:


Create Business Service
Select Custom Model
Enter the fields required by the custom model to create your new business service.
Lookup value
< Back
Next >
Finish
Cancel

Custom model name: OBJECT-MAINT-SUBP-2

Field Name	Field Value
PREDICT-FILE	
PREDICT-KEY	
PROVINCE	

Note: Although lookup values for Predict files and keys are supplied, they must be enabled. You can also define your own lookup logic for a custom model. For information, see *Define Lookup Values*.

3. Select **Next**.

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

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

Define Lookup Values

This section describes how to enable the supplied lookup logic for Predict files and keys, as well as how to define your own lookup logic for a custom model.

Two CustomModelLookups extensions are supplied with the Business Service wizard:

- PredictFileCustomLookup (displays a list of Predict files)
- PredictKeyCustomLookup (displays a list of key fields for the selected Predict file)

To enable the supplied Predict extensions, the *models.xml* file for the custom model must include a binding between a parameter field name and a CustomModelLookups extension contribution.

To enable the supplied lookup logic for Predict files and keys:

- Modify the *models.xml* file (defined in the Natural Business Service Preferences window) and bind the parameter field names for your custom model to the Predict extensions listed above.

  Note: For more information, see the comments in the supplied default *models.xml* file (located in *C:\Documents and Settings\All Users\Application Data\Software AG\Natural Business Services\Configuration*).

If desired, you can also define your own lookup options for a custom model.

To define custom lookup logic:

1. Create a plug-in project to store your custom lookup code.
2. Add the com.softwareag.nbs.ui plugin as a dependency in your plug-in project.
3. Create a java class that extends com.softwareag.nbs.generation.server.CustomLookupBase.
4. Implement logic in the lookup() method for your java class to return a String value (this value will be automatically set in the service wizard custom model field table).
5. Add an extension contribution to the com.softwareag.nbs.ui.CustomModelsLookups extension point.
6. Name the extension with a unique ID and set the class to the java class created in Step 3.
7. Modify the *models.xml* file (defined in the Natural Business Service Preferences window) and bind the parameter field names for your custom model to the new custom models lookup contribution.

Note: For more information, see the comments in the supplied default *models.xml* file.

Regenerate a Business Service

Note: This option is only available for SPoD connections.

To regenerate a business service:

1. Open the context menu for the business service in the **NBS Repositories** view.

   For an example of this menu, see *Access Connection Options*.
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 Service Proxy

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.

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 Java class) must be regenerated as well.

To regenerate a business service proxy:

1. Open the context menu for the business service in the **NBS Repositories** view.
2. Select **Regenerate service proxy(s)**.

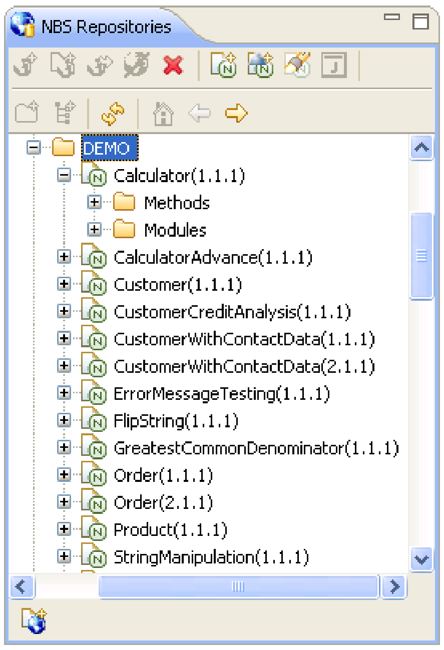
   The Eclipse plug-in regenerates the service proxy (or proxies) without displaying the wizard panels. New metadata is downloaded from the server before regeneration.

Test a Business Service Method

To test a method used by a business service:

1 Expand the business service node in the **NBS Repositories** view.

Folders containing the methods and modules used for the business service are displayed. For example:


NBS Repositories
DEMO
Calculator(1.1.1)
Methods
Modules
CalculatorAdvance(1.1.1)
Customer(1.1.1)
CustomerCreditAnalysis(1.1.1)
CustomerWithContactData(1.1.1)
CustomerWithContactData(2.1.1)
ErrorMessageTesting(1.1.1)
FlipString(1.1.1)
GreatestCommonDenominator(1.1.1)
Order(1.1.1)
Order(2.1.1)
Product(1.1.1)
StringManipulation(1.1.1)

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*.

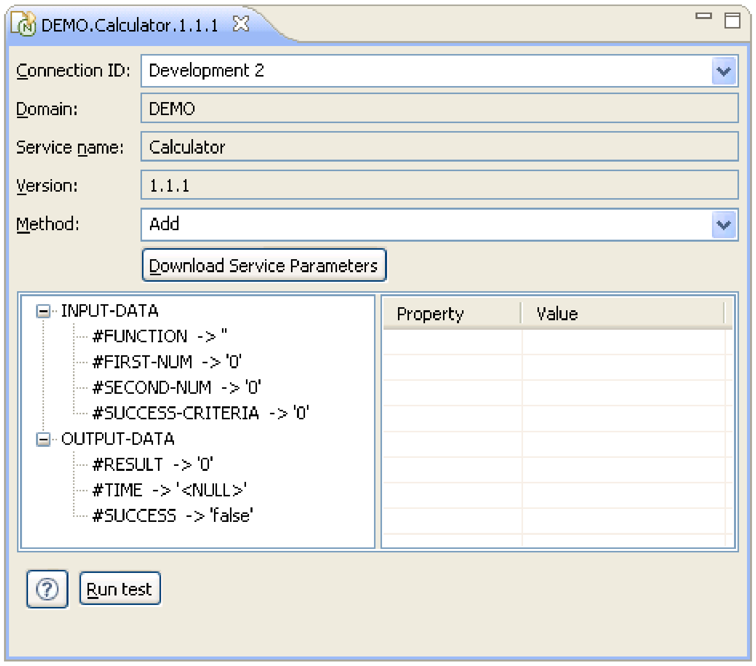
Test a Business Service

To test your business service:

1 Open the context menu for the business service in the **NBS Repositories** view.

2 Select **Test**.

The test window is displayed. For example:



Connection ID:	Development 2
Domain:	DEMO
Service name:	Calculator
Version:	1.1.1
Method:	Add

```
INPUT-DATA
    #FUNCTION -> ''
    #FIRST-NUM -> '0'
    #SECOND-NUM -> '0'
    #SUCCESS-CRITERIA -> '0'
OUTPUT-DATA
    #RESULT -> '0'
    #TIME -> '<NULL>'
    #SUCCESS -> 'false'
```

3 Select a method from **Method**.

For this example, select **Add**.

4 Select #**FIRST-NUM**.

The properties and values for #FIRST-NUM are displayed.

5 Type "1" in the **Value** property.

6 Select #**SECOND-NUM**.

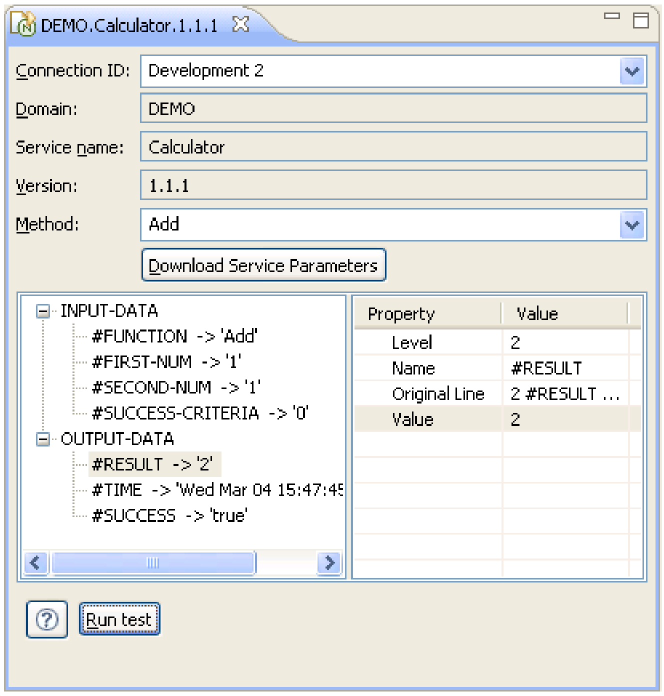
7 Type "1" in the **Value** property.

8 Select #**RESULT**.

9 Type "2" in the **Value** property.

10 Select **Run test**.

The results of the test are displayed in #SUCCESS. For example:



DEMO.Calculator.1.1.1

Connection ID: Development 2

Domain: DEMO

Service name: Calculator

Version: 1.1.1

Method: Add

Download Service Parameters

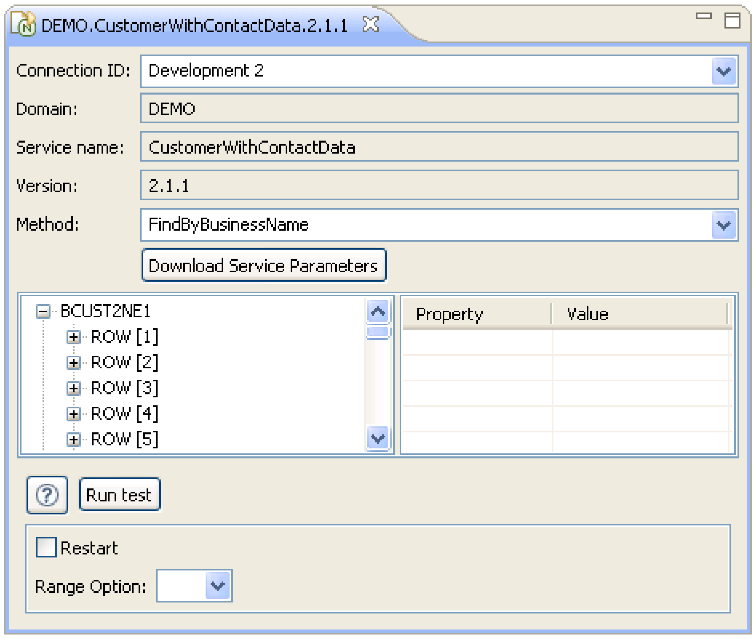
```
INPUT-DATA
    #FUNCTION -> 'Add'
    #FIRST-NUM -> '1'
    #SECOND-NUM -> '1'
    #SUCCESS-CRITERIA -> '0'
OUTPUT-DATA
    #RESULT -> '2'
    #TIME -> 'Wed Mar 04 15:47:45
    #SUCCESS -> 'true'
```

Property	Value
Level	2
Name	#RESULT
Original Line	2 #RESULT ...
Value	2

You can use this window to test all methods available for this business service.

Test a Business Service with Multiple Rows

If multiple rows occur because an object-browse or object-browse-select subprogram was used to create the service, two additional fields are available at the bottom of the test window. For example:


DEMO.CustomerWithContactData.2.1.1
Connection ID: Development 2
Domain: DEMO
Service name: CustomerWithContactData
Version: 2.1.1
Method: FindByBusinessName
Download Service Parameters
BCUST2NE1
ROW [1]
ROW [2]
ROW [3]
ROW [4]
ROW [5]
Property
Value
Run test
Restart
Range Option:

These fields provide an alternate, easier way to populate the CDBUPDA.RANGE-OPTION and CDBUPDA.RESTART parameters. The additional fields are:

Field	Description
Restart	Select this option to restart the test for multiple rows.
Range Option	Select a range option to limit the test results. The range options are: ■ * ■ = ■ > ■ >= ■ < ■ <=

Delete a Business Service

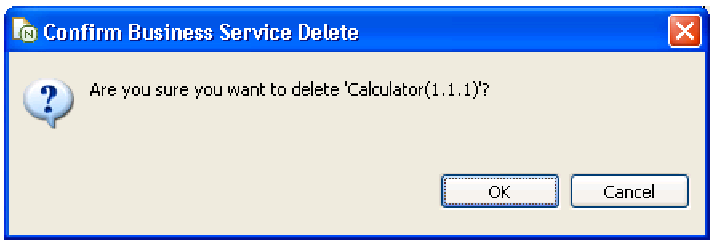
Note: Removing a business service will only delete the service from the repository, it will not delete the Natural modules.

To delete (remove) a business service:

1 Open the context menu for the business service in the **NBS Repositories** view.

2 Select **Delete**.

A confirmation window is displayed. For example:



Are you sure you want to delete 'Calculator(1.1.1)'?

3 Select **OK** to delete the business service.

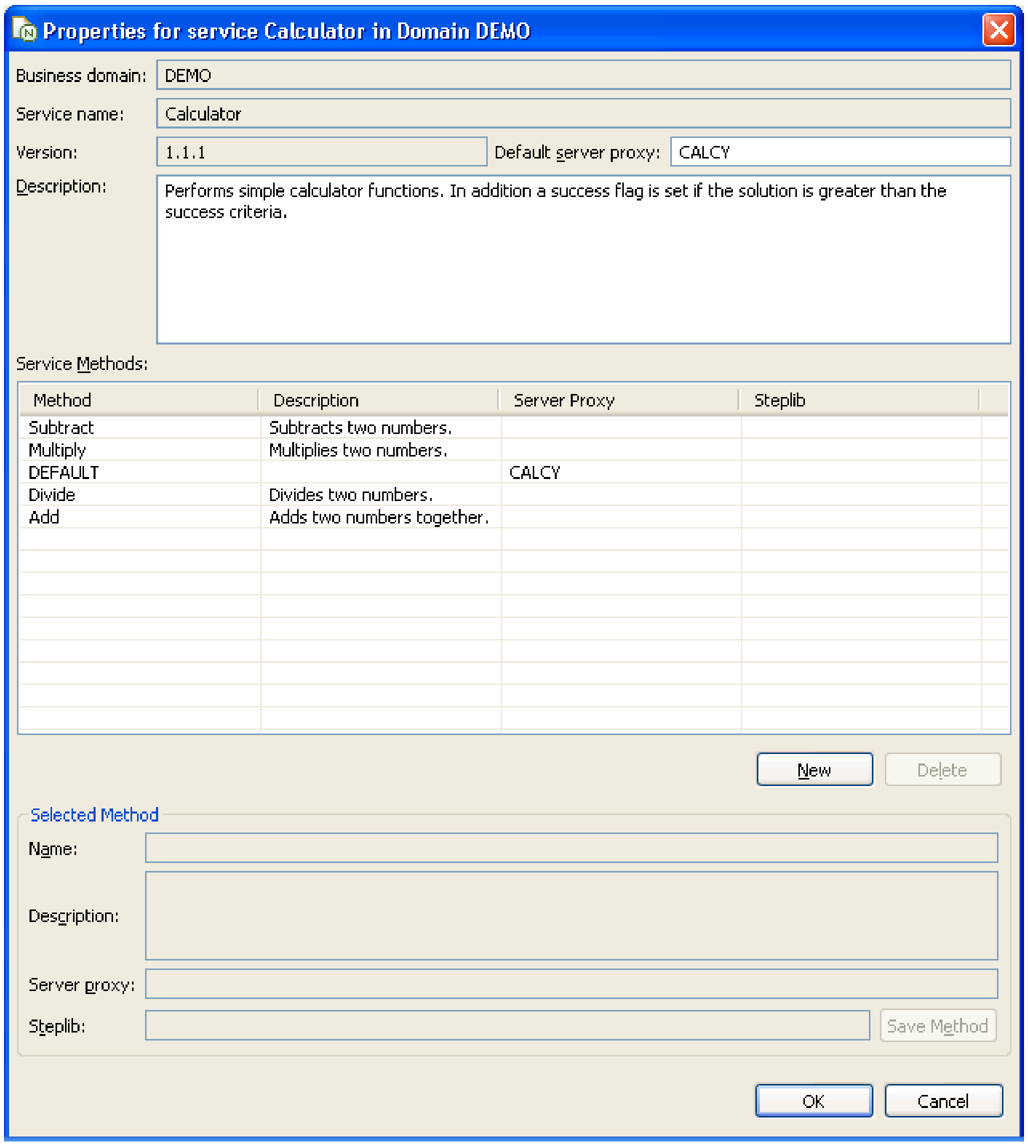
Edit a Service Definition

To edit a business service definition:

1 Open the context menu for the business service in the **NBS Repositories** view.

2 Select **Edit**.

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



Business domain: DEMO

Service name: Calculator

Version: 1.1.1 Default server proxy: CALCY

Description: Performs simple calculator functions. In addition a success flag is set if the solution is greater than the success criteria.

Service Methods:

Method	Description	Server Proxy	Steplib
Subtract	Subtracts two numbers.		
Multiply	Multiplies two numbers.		
DEFAULT		CALCY	
Divide	Divides two numbers.		
Add	Adds two numbers together.		

Use this window to:

Task	Procedure
Change the business service description.	Type a new description in **Description**.
Edit an existing method.	Select the method from **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.	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**.

Task	Procedure
Delete a method.	Select the method from **Service Methods** and select **Delete**. The method is removed from **Service Methods**.

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

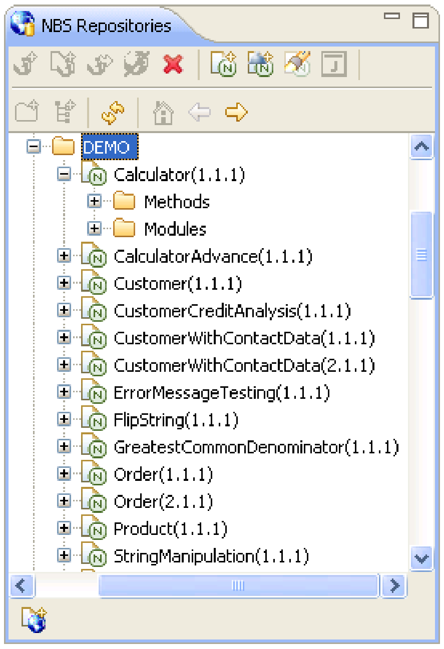
Edit Service Modules

Note: You must use a SPoD connection to edit the service modules.

To edit the modules used for a business service:

1 Expand the business service node in the **NBS Repositories** view.

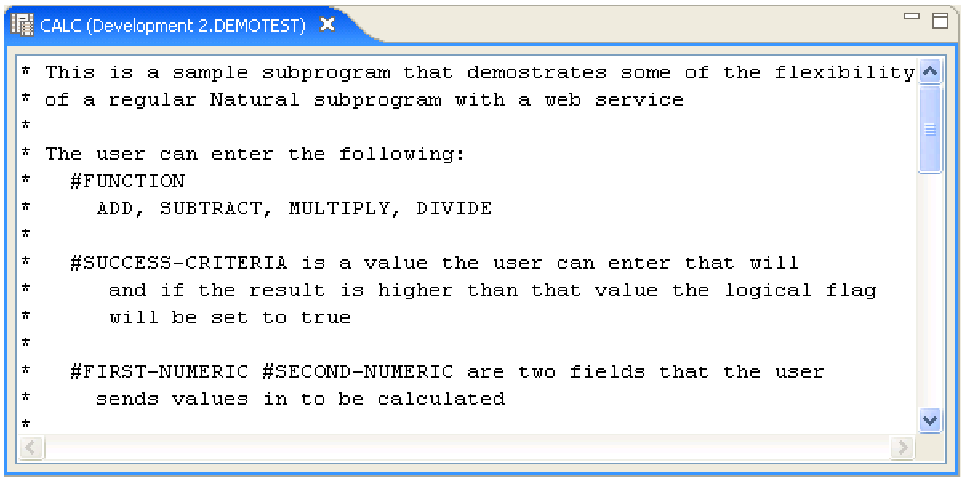
The methods and modules used for the business service are displayed as nodes in the repository view. For example:


NBS Repositories
DEMO
Calculator(1.1.1)
Methods
Modules
CalculatorAdvance(1.1.1)
Customer(1.1.1)
CustomerCreditAnalysis(1.1.1)
CustomerWithContactData(1.1.1)
CustomerWithContactData(2.1.1)
ErrorMessageTesting(1.1.1)
FlipString(1.1.1)
GreatestCommonDenominator(1.1.1)
Order(1.1.1)
Order(2.1.1)
Product(1.1.1)
StringManipulation(1.1.1)

2 Do one of the following:

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

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



```
* This is a sample subprogram that demostrates some of the flexibility
* of a regular Natural subprogram with a web service
*
* The user can enter the following:
*   #FUNCTION
*     ADD, SUBTRACT, MULTIPLY, DIVIDE
*
*   #SUCCESS-CRITERIA is a value the user can enter that will
*      and if the result is higher than that value the logical flag
*      will be set to true
*
*   #FIRST-NUMERIC #SECOND-NUMERIC are two fields that the user
*     sends values in to be calculated
*
```

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

Important: 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 Service Proxy*.

4 Using the Custom Ant Tasks

- Get latest version from source control
- Compile source code
- Create jar (Java archive) files

For example:

```
<Project name=”sample project”>
  <target name=”build”>
       <cvs … >   <!—get latest version
       <javac …  > <!—compile the source code
   </target>
</Project>
```

Natural Business Services includes several custom Ant tasks. These tasks help generate and deploy Java classes based on business services created on the server.

Prerequisites

The custom Ant tasks provided with Natural Business Services require the following prerequisites:

- Ant 1.6
- The following .jar files on the class path:
  - junit-4.3.1.jar
  - entirex.jar
  - nbsrt4j.jar
  - ndvserveraccess.jar
  - nbsAnt.jar
  - velocity-dep-1.4.jar
  - nbs.ui.jar
- CentraSite .jar file(s)

  This file(s) is only required if deploying to CentraSite.

Global Properties

The custom Ant tasks use the following global properties. These properties must be set using the **Property** node in Ant:

Property	Value
nbs.userID	User ID to connect to the Natural Business Services (NBS) server.
nbs.connectionID	Connection ID (located in the *DispatchClient.Config* file).
nbs.password	Password for the specified user ID.
nbs.config.location	Location of the *DispatchClient.Config* file.
nbs.target.vm	Java Virtual Machine (VM) version to target (original or enhanced).
nbs.templates	Location of the template (*.vm) files used by Natural Business Services (for example: <`install dir`>/*templates*).

Supplied Ant Tasks

This section describes the Ant tasks supplied with Natural Business Services. All tasks are located in the com.softwareag.nbs.ant.tasks package.

The supplied Ant tasks are:

- Axis2Deploy
- CentraSiteRegister
- CreateAAR
- CreateClass
- PrintConfig
- Regenerate

Axis2Deploy

This task deploys an Axis2 Web service archive file (.aar) to an Axis2 web application server. The attributes for this task are:

Attribute	Optional	Description
aarfilename	No	Archive file to deploy.
targetfolder	No	Name of the folder in which to copy the .aar file.

CentraSiteRegister

This task registers a Web service with CentraSite. The attributes for this task are:

Attribute	Optional	Description
description	No	Brief description of the Web service to register.
organization	No	Name of the CentraSite organization.
server	No	URL for the CentraSite server (for example: *http://<hostname>:53305/CentraSite/CentraSite*).
userid	No	User ID for CentraSite.
password	No	Password for CentraSite.
wsdl	No	URL for the WSDL (Web Service Description Language) file for the Web service.

CreateAAR

This task creates a Web service archive file (.aar) for deployment to Axis2. The attributes for this task are:

Attribute	Optional	Description
destfile	No	Path for the .aar file to create.
servicestemplate	No	Path for the .xml template to use (*services.xml.vm*).

Child Nodes

Child Node	Description and Attributes
fileset	Standard Ant fileset node. Each fileset node is used to include files in the archive. The `dir` attribute must be specified. Any files found by the fileset will be added to the .aar file with a path relative to the `dir` attribute. For example, if the `dir` attribute is *c:\temp* and a file is found at *c:\temp\a\b\c.class*, the archive will contain `a\b\c.class`, with `a` and `b` as folders.
webservice	Web service entry to add to the *services.xml* file.

CreateClass

This task creates a Java class based on a business service on the server.

Note: If the `webservice` attribute is set to true, the CreateClass task will not generate the .aar file because the Java code has not yet been compiled. Use the **CreateAAR** task instead.

The attributes for this task are:

Attribute	Optional	Description
axisversion	Yes	Version of Axis files to generate. The default is "Axis" (i.e., not Axis2).
destdir	Yes	Path for the directory in which to store the generated files.
metadir	Yes	Path for the directory in which to store the specifications (*.nbsMetadata* file).
domain/servicename/version	Yes	Specifications for the business service.
junit	Yes	Indicates whether to generate Junit test(s). The default is "false".
package	Yes	Java package to use for the generated class.
webservice	Yes	Indicates whether to generate files to expose the generated class as a Web service. The default is "false".
failonerror	Yes	Indicates whether the CreateClass task will pass or fail when an error occurs while creating the client proxy class. The default is "true". ■ If this attribute is set to "false" and an error occurs, the task will continue. ■ If this attribute is set to "true" and an error occurs, the task will end.
multithread	Yes	Indicates whether to use multiple threads to create the classes. The default is "true"; for each service found on the server, a separate thread will be created to download metadata for the business service and create the class. The advantage to this option is that if the user has a slow connection to the server, several calls can be made at once to download metadata for different services. To disallow this option, set this attribute to "false".

Child Nodes

Child Node	Description and Attributes
serviceset	List containing business services to generate.
serverchange	List of business services based on server activity.

PrintConfig

This task prints a copy of the configuration settings used by Natural Business Services. The following example shows output from this task:

```
[nbs.printconfig] Configuration
[nbs.printconfig] Configuration File: S:/NBS82/Runtime/DispatchClient.config
[nbs.printconfig] ConnectionID: 82-NEW-DEV
[nbs.printconfig] User ID: DEV
[nbs.printconfig] Password: ****
[nbs.printconfig] Templates: ↩
S:\NBS82\Java\Workspace3.3\com.softwareag.nbs.ui\templates
```

Regenerate

This task regenerates files created by the CreateClass task. The attributes for this task are:

Attribute	Optional	Description
destdir	Yes	Path to the directory containing the files to regenerate.
metadir	Yes	Path to the directory containing the specifications (folder in which the *.nbsMetadata* files are located).

Child Nodes

Child Node	Description and Attributes
fileset	List of .java or .nbsMetadata files to regenerate. If the file is .java, the Regenerate task searches for the corresponding .nbsMetadata file in the directory specified in the metadir attribute.

Child Node Types

This section describes the Child node types for the Ant tasks supplied with Natural Business Services. All node types are located in the com.softwareag.nbs.ant package.

The Child node types are:

- ServerChange
- ServiceSet

- WebService

ServerChange

This task uses filtering criteria (such as domain, user ID, and date) to search the Natural Business Services log file on the server and find any changes to business services.

Note: All purge changes are ignored because you cannot generate a class for a business service that has been purged.

The attributes for this task are:

Attribute	Optional	Description
domain	Yes	Limits the list of services to a particular domain.
duration	Yes	Finds changes made during the specified interval. The duration format is: `[count][interval]` Valid intervals are: ■ H (hour) ■ M (minute) ■ d (day) ■ w (week) ■ m (month) ■ y (year) For example, "2d" indicates changes made in the past two days. The default is "1m" (1 month).
userid	Yes	Lists the changes made by the specified user.

ServiceSet

This task lists the business services available for the specified connection ID and domain. The attributes for this task are:

Attribute	Optional	Description
connectionid	Yes	Connection ID for which to list the services. If this attribute is not specified, the nbs.connectionID global property will be used.
domain	No	Natural Business Services domain containing the specified services.
includepattern	Yes	Regular expression pattern used to identify a business service name. If a name matches this pattern, it will be included.
excludepattern	Yes	Regular expression pattern used to identify a business service name. If a name matches this pattern, it will be excluded.

Note: If the `includepattern` and `excludepattern` attributes are both specified, `includepattern` will be applied before `excludepattern`.

WebService

This task specifies an Axis Web service. The attributes for this task are:

Attribute	Optional	Description
name	No	Name of the Web service.
classfullname	No	Full class name (including package) of the class for the Web service.

Example of Using Custom Ant Tasks

The following XML example demonstrates the use of custom Ant tasks supplied with Natural Business Services:

```
<?xml version="1.0" ?>
- <project name="Test" default="build" basedir=".">
- <!--
 TODO: change the below path to match your nbs installation folder
  -->
  <property name="nbs.install" value="C:\Program Files\Software AG\Natural Business ↩
Services\V8.2\" />
- <!--
 TODO: Modify the below settings to match your configuration
  -->
- <target name="init">
  <property name="nbs.userID" value="GUEST" />
  <property name="nbs.connectionID" value="Default Broker" />
  <property name="nbs.password" value="" />
  <property name="nbs.config.location" value="${nbs.install}\dispatchclient.config" ↩
/>
  <property name="nbs.target.jvm" value="1.4" />
  <property name="nbs.templates" value="${nbs.install}\templates\java" />
  </target>
- <!--
 jars to be used while ant is running
  -->
- <path id="runtimeJars">
- <fileset dir="${nbs.install}\javaruntime">
  <include name="*.jar" />
  </fileset>
- <fileset dir="${nbs.install}\EclipsePlugin\plugins">
  <include name="com.softwareag.nbs.ui_*.jar" />
  </fileset>
  </path>
- <!--
```

```
 jars to be used by the compiler
  -->
- <path id="compileJars">
- <fileset dir="${nbs.install}\javaruntime">
  <include name="*.jar" />
  </fileset>
  </path>
- <!--
 NBS Task definitions
  -->
  <taskdef classname="com.softwareag.nbs.ant.tasks.Test" name="nbstest" ↩
classpathref="runtimeJars" loaderref="NBSRef" />
  <taskdef classname="com.softwareag.nbs.ant.tasks.CreateClass" ↩
classpathref="runtimeJars" name="nbs.createclass" loaderref="NBSRef" />
  <taskdef classname="com.softwareag.nbs.ant.tasks.Regenerate" name="nbs.regenerate" ↩
classpathref="runtimeJars" loaderref="NBSRef" />
  <taskdef classname="com.softwareag.nbs.ant.tasks.CreateAAR" name="nbs.createaar" ↩
classpathref="runtimeJars" loaderref="NBSRef" />
  <taskdef classname="com.softwareag.nbs.ant.tasks.Axis2Deploy" ↩
name="nbs.axis2deploy" classpathref="runtimeJars" loaderref="NBSRef" />
  <taskdef classname="com.softwareag.nbs.ant.tasks.CentraSiteRegister" ↩
name="nbs.centrasite" classpathref="runtimeJars" loaderref="NBSRef" />
- <!--
 NBS Type definitions
  -->
  <typedef name="serviceset" classname="com.softwareag.nbs.ant.ServiceSet" ↩
classpathref="runtimeJars" loaderref="NBSRef" />
  <typedef name="serverchange" classname="com.softwareag.nbs.ant.ServerChange" ↩
classpathref="runtimeJars" loaderref="NBSRef" />
  <typedef name="webservice" classname="com.softwareag.nbs.ant.WebService" ↩
classpathref="runtimeJars" loaderref="NBSRef" />
- <target name="build" depends="init">
- <!--
 Sample create class
  -->
- <nbs.createclass destdir="c:\temp\root\src" metadir="c:\temp\root\.metadata" ↩
domain="DEMO" servicename="product" version="1.1.1" package="ant1.ant2" junit="true" ↩
webservice="true" axisversion="2">
  <serviceset domain="DEMO" includePattern="Calc.*" />
  <serverchange duration="1y" userid="PWRUSR" domain="DEMO" />
  </nbs.createclass>
- <!--
 Sample regenerate
  -->
- <nbs.regenerate destdir="c:\temp\root\src" metadir="c:\temp\root\.metadata">
- <fileset dir="c:\temp\root\src" casesensitive="false">
  <include name="**/*.java" />
  </fileset>
  </nbs.regenerate>
- <!--
 Compile the above generated classes
  -->
  <mkdir dir="c:\temp\bin" />
  <javac executable="C:\Program Files\Java\jdk1.5.0_11\bin\javac.exe" fork="true" ↩
classpathref="compileJars" compiler="javac1.5" destdir="c:\temp\bin" ↩
srcdir="c:\temp\root\src" />
  </target>
  </project>
```

Note: Java 1.4 runtime is supported.

5 Administering Business Services

Configure the Eclipse Plug-in

Before using the Natural Business Services Eclipse plug-in for 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

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

To create a new steplib configuration:

1 Select the connection for the steplib configuration in the **NBS Repositories** view.

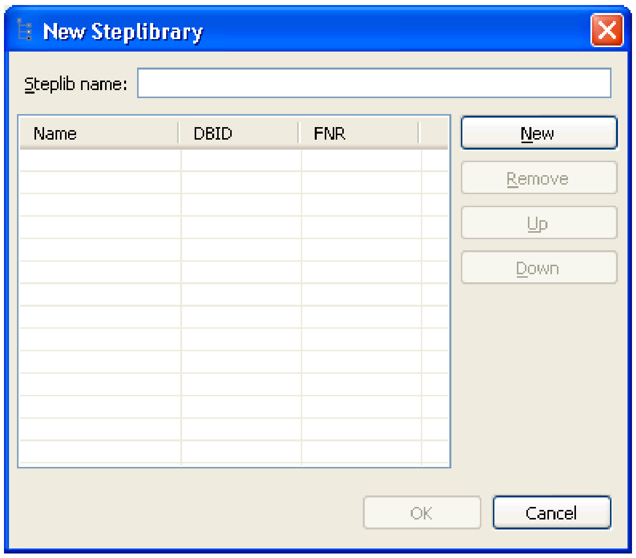
  If required, enter the user ID and password for the connection.

2 Expand the **Configuration** node.

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

4 Select **Add new steplib configuration**.

  The **New Steplibrary** window is displayed. For example:



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

6 Select **New**.

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 **OK** to add the new steplib configuration.

Modify an Existing Steplib Configuration

To modify an existing steplib configuration:

1 Select the connection for the steplib configuration in the **NBS Repositories** view.

If required, enter the user ID and password for the connection.

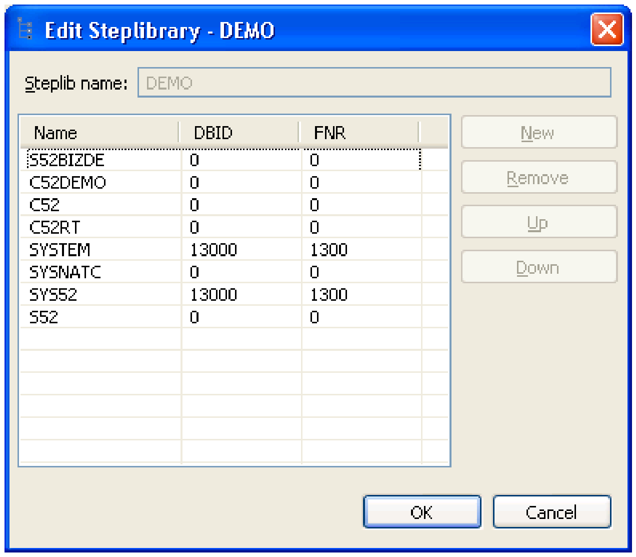
2 Expand the **Configuration** node.

3 Expand the **Steplibs** node.

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

5 Select **Edit**.

The **Edit Steplibrary** window is displayed. For example:



Steplib name: DEMO

Name	DBID	FNR
S52BIZDE	0	0
C52DEMO	0	0
C52	0	0
C52RT	0	0
SYSTEM	13000	1300
SYSNATC	0	0
SYS52	13000	1300
S52	0	0

Use this window to:

Task	Procedure
Remove a library from the steplib configuration.	Select the library and select **Remove**.
Add a library to the steplib configuration.	Select **New** and 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.
Change the order of libraries in the steplib configuration.	Select the library and either **Up** or **Down**.

6 Select **OK** to save changes to the steplib configuration.

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:

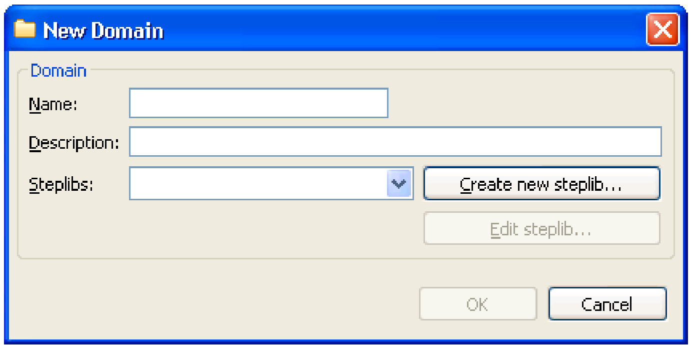
1 Select the connection for the domain in the **NBS Repositories** view.

If required, enter the user ID and password for the connection.

2 Open the context menu for **Domains**.

3 Select **Add New Domain**.

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



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

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

6 Select at least one step library chain from **Steplibs**.

Optionally, you can select **Create new steplib** to create a new step library definition or select **Edit steplib** to edit an existing steplib configuration.

Note: For more information, see *Create a New Steplib Configuration* and *Modify an Existing Steplib Configuration*.

7 Select **OK** to add the new domain.

Modify an Existing Domain

To modify an existing domain definition:

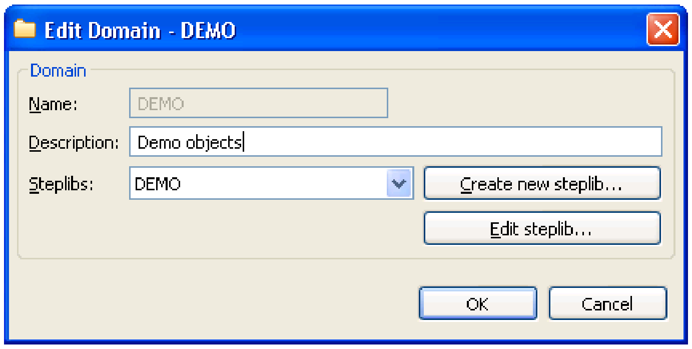
1 Select the connection for the domain in the **NBS Repositories** view.

If required, enter the user ID and password for the connection.

2 Open the context menu for the domain you want to edit.

3 Select **Edit**.

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



Name: DEMO
Description: Demo objects
Steplibs: DEMO

Use this window to:

Task	Procedure
Change the description of the domain.	Type a new description in **Description**.
Select another steplib configuration to use for this domain.	Select the configuration from **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*.

4 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 Select the connection for the domain in the **NBS Repositories** view.

If required, enter the user ID and password for the connection.

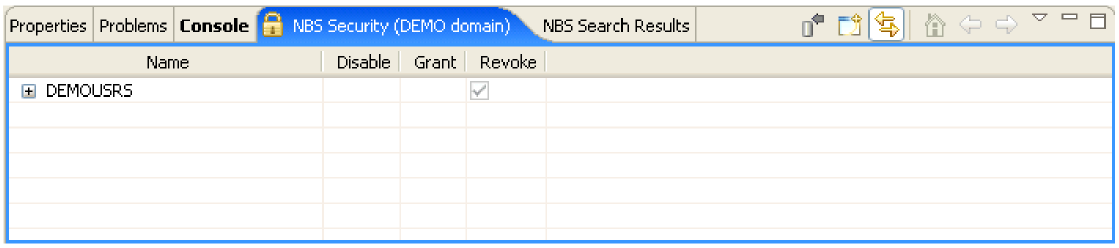
2 Expand the **Domains** node.

The list of available domains is displayed.

3 Open the context menu for the domain you want to secure.

4 Select **View security**.

The **NBS Security** view is displayed. For example:



Name	Disable	Grant	Revoke
⊞ DEMOUSRS			✓

This view displays the security groups for the selected domain in **Name**. You can expand the group node to display the business services in the group and then expand the business service node to display the methods for the service.

5 Select **Disable**, **Grant**, or **Revoke** 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.

6 Open the context menu for the group, service, and/or method.

7 Select **Update** to save the changes.

Add a Security Group

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

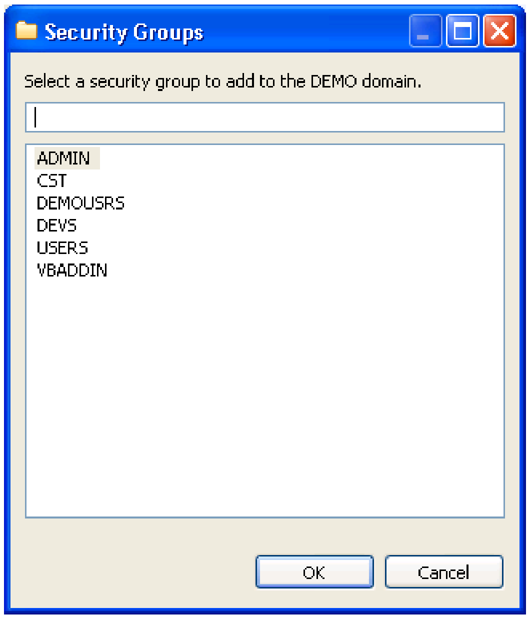
To add a security group to the domain:

1 Open the context menu for one of the groups listed in the **NBS Security** view.

Tip: You can also select the toolbar option in the **NBS Security** view.

2 Select **Add group**.

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



Select a security group to add to the DEMO domain.

ADMIN
CST
DEMOUSRS
DEVS
USERS
VBADDIN

3 Select the security group you want to add.

4 Select **OK**.

The security group is added to the domain.

Delete a Security Group

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

To delete (remove) a security group from a domain:

1 Open the context menu for the group in the **NBS Security** view.

2 Select **Delete group**.

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 Open the context menu for the SPoD connection in the **NBS Repositories** view.

For an example of this menu, see *Access Connection Options*.

2 Select **Connect**.

The **Enter User's Credentials** window is displayed.

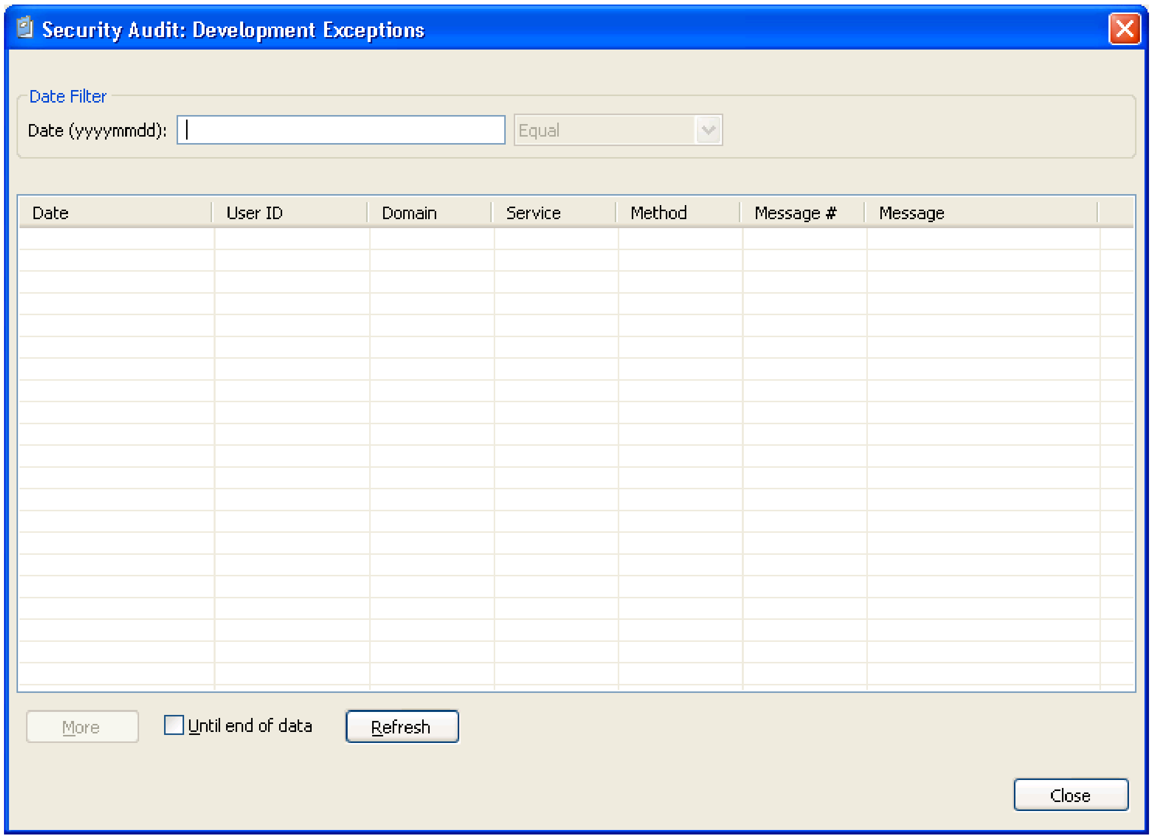
3 Type the user ID and password for the connection.

4 Select **OK**.

5 Open the context menu for the connection.

6 Select **Security audit**.

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



This window displays the date of each audit activity, the user ID of the user who performed the action, and the domain, service, and method names. It also displays the message number and text for each exception. Optionally, you can filter the information based on a date. For example, to display all activity on a particular date, enter that date in Date and select **Refresh**.

After entering the date, you can further refine the information by selecting one of the following options from the drop-down list before selecting **Refresh**:

- Less than
- Less than or equal
- Equal
- Greater than
- Greater than or equal

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

7 Select **Close** to close the **Security Audit** window.

Audit Domains

Use this option to display activities within a domain.

To audit a domain:

1 Select the connection for the domain in the **NBS Repositories** view.

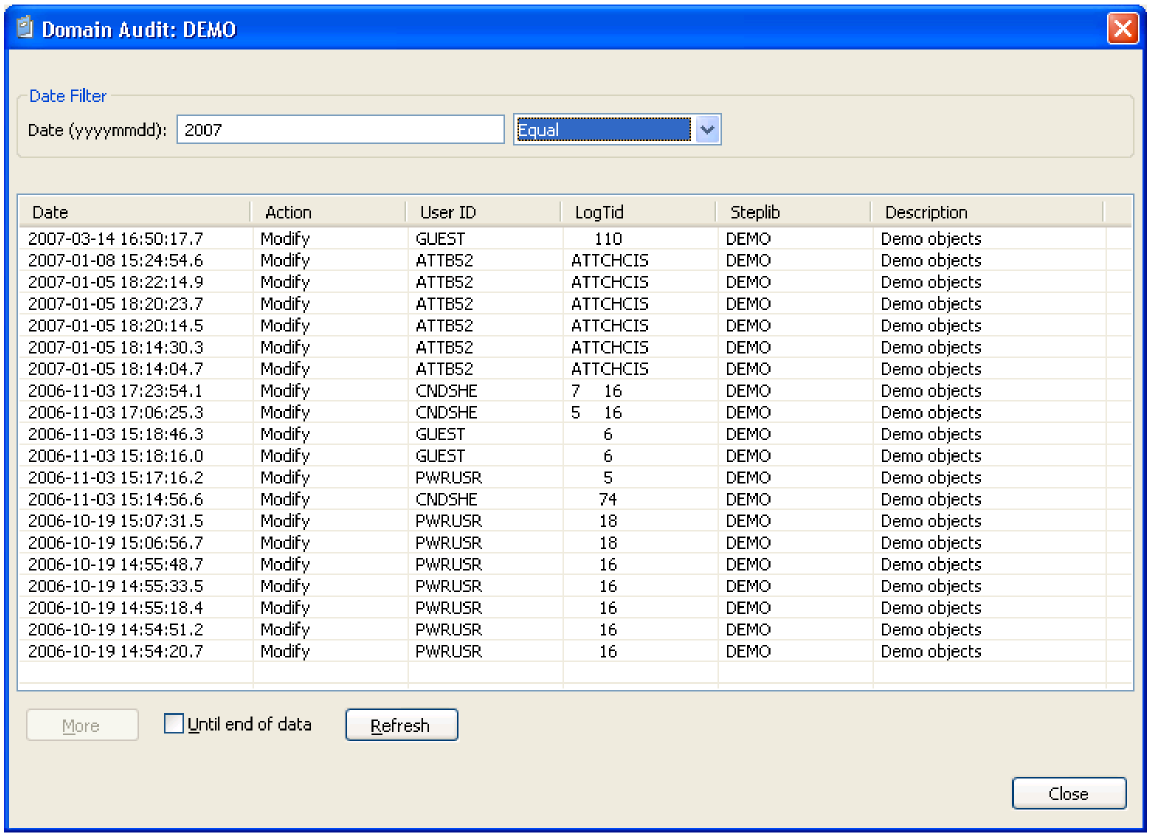
If required, enter the user ID and password for the connection.

2 Expand the **Domains** node.

3 Open the context menu for the domain you want to audit.

4 Select **Audit**.

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



Date (yyyymmdd): 2007 Equal

Date	Action	User ID	LogTid	Steplib	Description
2007-03-14 16:50:17.7	Modify	GUEST	110	DEMO	Demo objects
2007-01-08 15:24:54.6	Modify	ATTB52	ATTCHCIS	DEMO	Demo objects
2007-01-05 18:22:14.9	Modify	ATTB52	ATTCHCIS	DEMO	Demo objects
2007-01-05 18:20:23.7	Modify	ATTB52	ATTCHCIS	DEMO	Demo objects
2007-01-05 18:20:14.5	Modify	ATTB52	ATTCHCIS	DEMO	Demo objects
2007-01-05 18:14:30.3	Modify	ATTB52	ATTCHCIS	DEMO	Demo objects
2007-01-05 18:14:04.7	Modify	ATTB52	ATTCHCIS	DEMO	Demo objects
2006-11-03 17:23:54.1	Modify	CNDSHE	7 16	DEMO	Demo objects
2006-11-03 17:06:25.3	Modify	CNDSHE	5 16	DEMO	Demo objects
2006-11-03 15:18:46.3	Modify	GUEST	6	DEMO	Demo objects
2006-11-03 15:18:16.0	Modify	GUEST	6	DEMO	Demo objects
2006-11-03 15:17:16.2	Modify	PWRUSR	5	DEMO	Demo objects
2006-11-03 15:14:56.6	Modify	CNDSHE	74	DEMO	Demo objects
2006-10-19 15:07:31.5	Modify	PWRUSR	18	DEMO	Demo objects
2006-10-19 15:06:56.7	Modify	PWRUSR	18	DEMO	Demo objects
2006-10-19 14:55:48.7	Modify	PWRUSR	16	DEMO	Demo objects
2006-10-19 14:55:33.5	Modify	PWRUSR	16	DEMO	Demo objects
2006-10-19 14:55:18.4	Modify	PWRUSR	16	DEMO	Demo objects
2006-10-19 14:54:51.2	Modify	PWRUSR	16	DEMO	Demo objects
2006-10-19 14:54:20.7	Modify	PWRUSR	16	DEMO	Demo objects

☐ Until end of data

This window displays the date of each audit activity, the action performed, and the user ID of the user who performed the action. It also displays the terminal ID for the log file, the names of the steplibs, and a brief description of the domain. Optionally, you can filter the information based on a date. For example, to display all activity on a particular date, enter that date in Date and select **Refresh**.

After entering the date, you can further refine the information by selecting one of the following options from the drop-down list before selecting **Refresh**:

- Less than
- Less than or equal
- Equal
- Greater than
- Greater than or equal

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

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

Audit Business Services

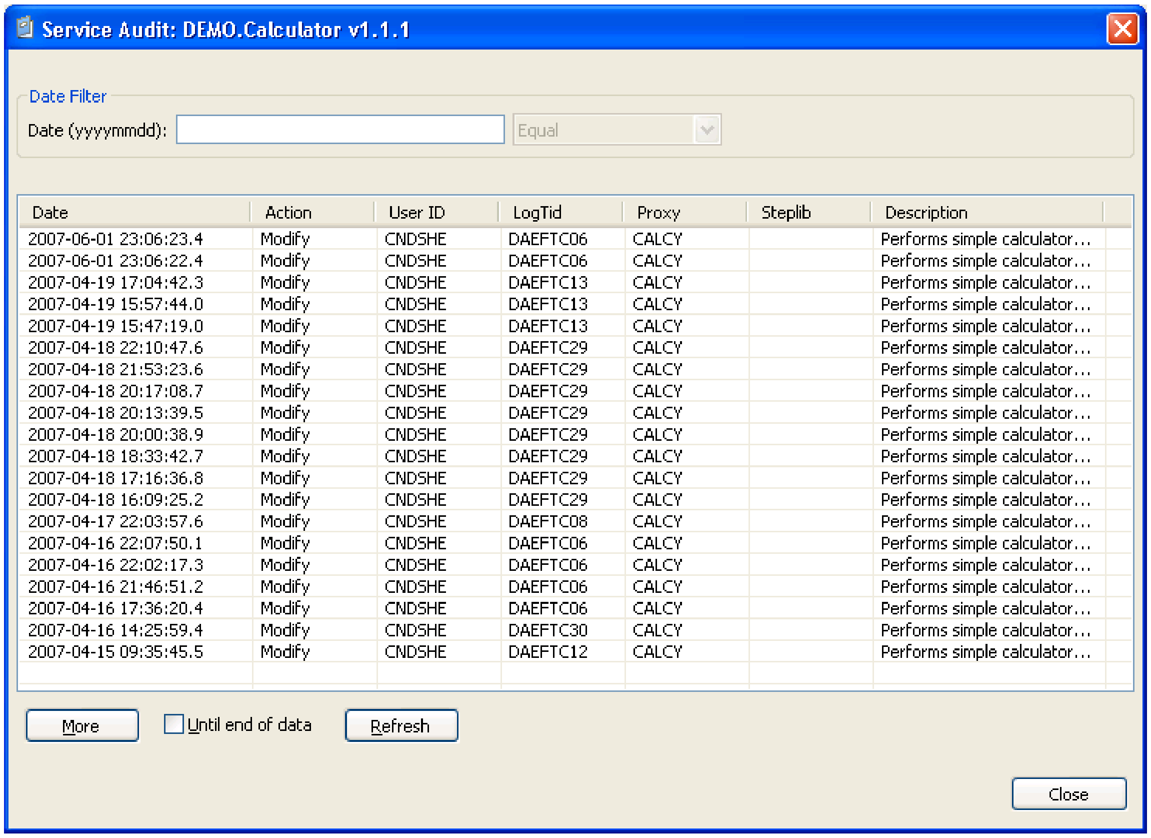
Use this option to determine which users have been adding new business services or modifying existing business services.

To audit a business service:

1 Open the context menu for the business service in the **NBS Repositories** view.

2 Select **Audit**.

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



Date	Action	User ID	LogTid	Proxy	Steplib	Description
2007-06-01 23:06:23.4	Modify	CNDSHE	DAEFTC06	CALCY		Performs simple calculator...
2007-06-01 23:06:22.4	Modify	CNDSHE	DAEFTC06	CALCY		Performs simple calculator...
2007-04-19 17:04:42.3	Modify	CNDSHE	DAEFTC13	CALCY		Performs simple calculator...
2007-04-19 15:57:44.0	Modify	CNDSHE	DAEFTC13	CALCY		Performs simple calculator...
2007-04-19 15:47:19.0	Modify	CNDSHE	DAEFTC13	CALCY		Performs simple calculator...
2007-04-18 22:10:47.6	Modify	CNDSHE	DAEFTC29	CALCY		Performs simple calculator...
2007-04-18 21:53:23.6	Modify	CNDSHE	DAEFTC29	CALCY		Performs simple calculator...
2007-04-18 20:17:08.7	Modify	CNDSHE	DAEFTC29	CALCY		Performs simple calculator...
2007-04-18 20:13:39.5	Modify	CNDSHE	DAEFTC29	CALCY		Performs simple calculator...
2007-04-18 20:00:38.9	Modify	CNDSHE	DAEFTC29	CALCY		Performs simple calculator...
2007-04-18 18:33:42.7	Modify	CNDSHE	DAEFTC29	CALCY		Performs simple calculator...
2007-04-18 17:16:36.8	Modify	CNDSHE	DAEFTC29	CALCY		Performs simple calculator...
2007-04-18 16:09:25.2	Modify	CNDSHE	DAEFTC29	CALCY		Performs simple calculator...
2007-04-17 22:03:57.6	Modify	CNDSHE	DAEFTC08	CALCY		Performs simple calculator...
2007-04-16 22:07:50.1	Modify	CNDSHE	DAEFTC06	CALCY		Performs simple calculator...
2007-04-16 22:02:17.3	Modify	CNDSHE	DAEFTC06	CALCY		Performs simple calculator...
2007-04-16 21:46:51.2	Modify	CNDSHE	DAEFTC06	CALCY		Performs simple calculator...
2007-04-16 17:36:20.4	Modify	CNDSHE	DAEFTC06	CALCY		Performs simple calculator...
2007-04-16 14:25:59.4	Modify	CNDSHE	DAEFTC30	CALCY		Performs simple calculator...
2007-04-15 09:35:45.5	Modify	CNDSHE	DAEFTC12	CALCY		Performs simple calculator...

This window displays the date of each audit activity, the action performed, and the user ID of the user who performed the action. It also displays the terminal ID for the log file, the names of the subprogram proxies and steplibs, and a brief description of the domain. Optionally, you can filter the information based on a date. For example, to display all activity on a particular date, enter that date in **Date** and select **Refresh**.

After entering the date, you can further refine the information by selecting one of the following options from the drop-down list before selecting **Refresh**:

- Less than
- Less than or equal
- Equal
- Greater than
- Greater than or equal

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

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

Deploy a Domain

You can deploy a business service domain to another repository or environment.

Note: You can only deploy the subprograms from the **Business Services** node, not from the **Domains** node (i.e., you can deploy all the services in a domain, but not the subprograms; you can only deploy subprograms for a specific service).

To deploy a domain:

1 Select the connection for the domain in the **NBS Repositories** view.

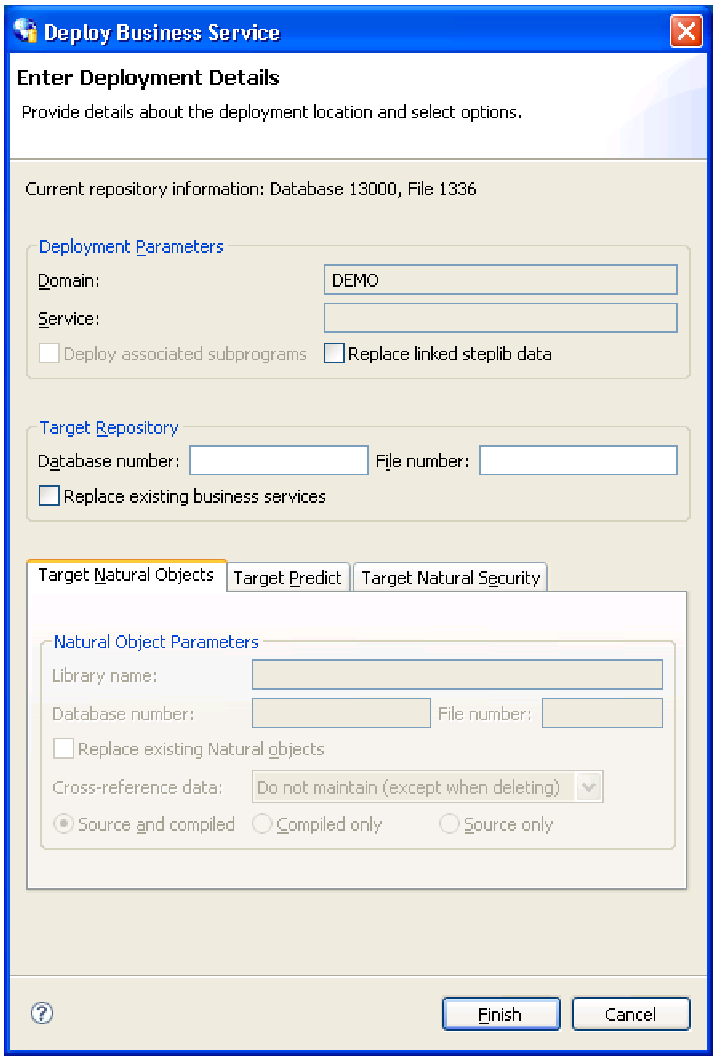
If required, enter the user ID and password for the connection.

2 Expand the **Domains** node.

3 Open the context menu for the domain you want to deploy.

4 Select **Deploy**.

The **Enter Deployment Details** window is displayed. For example:


Deploy Business Service

Enter Deployment Details

Provide details about the deployment location and select options.

Current repository information: Database 13000, File 1336

Deployment Parameters

Domain: DEMO

Service:

☐ Deploy associated subprograms ☐ Replace linked steplib data

Target Repository

Database number: File number:

☐ Replace existing business services

Natural Object Parameters

Library name:

Database number: File number:

☐ Replace existing Natural objects

Cross-reference data: Do not maintain (except when deleting)

[✓ Source and compiled] Compiled only Source only

This window displays information about the current repository, as well as the name of the domain you selected.

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 the Library name is blank, the Natural modules in the source library will be deployed to the same library in the target FUSER file.

Optionally, you can use this window to:

Task	Procedure
Update steplib data for the business service domain.	Select **Replace linked steplib data** and type the database and file numbers for the target repository in **Target Repository**
Replace the current business services in the domain with updated services.	Select **Replace existing business services** and type the database and file numbers for the target repository in the appropriate fields. **Note:** If the **Replace** option is not selected and the domain exists in the target library, it will not be replaced.

5 Type the database and file numbers for LFILE 136 in **Target Repository**.

6 Select **Finish** to deploy the domain.

Deploy a Business Service

You can deploy the information in the business service repository for a single business service or, optionally, deploy the subprograms associated with the service. The subprograms are Natural objects that were created by Natural Construct models.

Notes:

1. Any subprograms that these subprograms call, or any subprograms called by user exit code, will not be deployed.
2. You can only deploy the associated subprograms from the business service node, not from the **Domains** node.
3. You must use a SPoD connection to deploy a business service.

If a business service uses two subprograms, for example, the business service wizard generates the following subprograms:

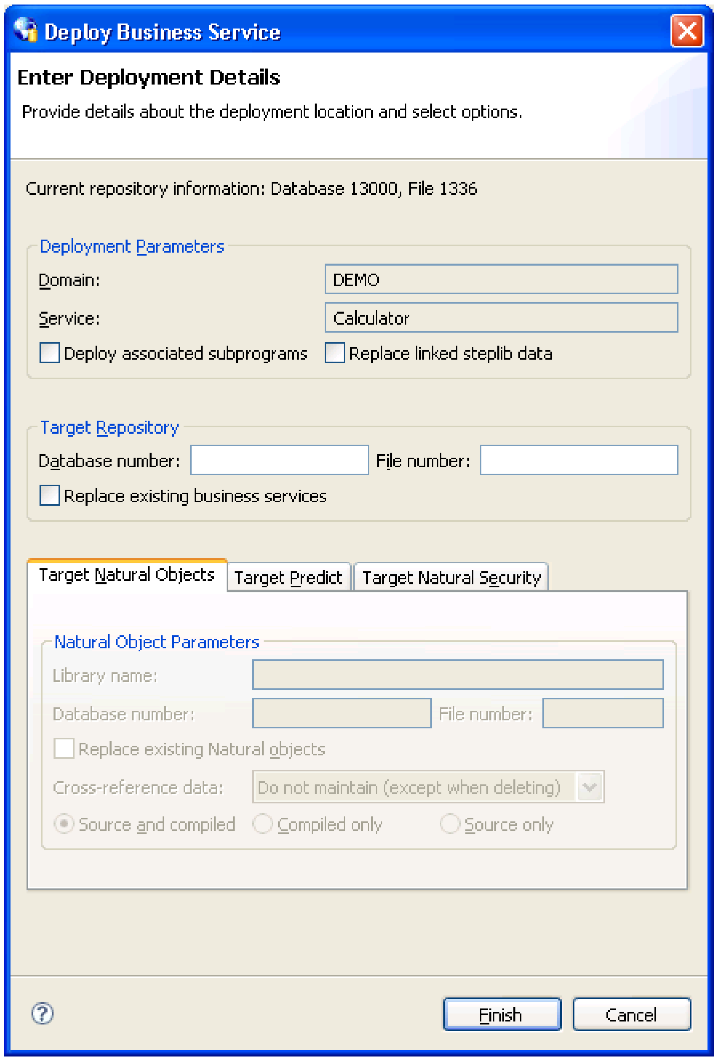
- The two subprograms (selected on the wizard panel)
- The wrapper subprogram
- The subprogram proxy for the business service

To deploy a business service:

1 Open the context menu for the business service in the **NBS Repositories** view.

2 Select **Deploy**.

The **Enter Deployment Details** window is displayed. For example:


Deploy Business Service

Enter Deployment Details

Provide details about the deployment location and select options.

Current repository information: Database 13000, File 1336

Deployment Parameters

Domain: DEMO

Service: Calculator

☐ Deploy associated subprograms ☐ Replace linked steplib data

Target Repository

Database number: File number:

☐ Replace existing business services

Natural Object Parameters

Library name:

Database number: File number:

☐ Replace existing Natural objects

Cross-reference data: Do not maintain (except when deleting)

[✓ Source and compiled] Compiled only Source only

This window displays information about the current repository, as well as the name of the domain and business service you selected.

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 the Library name is blank, the Natural modules in the source library will be deployed to the same library in the target FUSER file.

Optionally, you can use this window to do the following tasks:

Task	Procedure
Deploy the subprograms associated with the business service.	Select **Deploy associated subprograms** and specify the target settings. For information, see *Deploy Associated Subprograms*.
Update steplib data.	Select **Replace linked steplib data** and type the database and file numbers for LFILE 136 in **Target Repository**.
Replace the business service definition in the target repository (LFILE 136).	Select **Replace existing business services** and type the database and file numbers for LFILE 136 in the appropriate fields. **Note:** If this option is not selected and the business service definition exists in the target repository, it will not be replaced.

3 Type the database and file numbers for the LFILE 136 system file in **Target Repository**.

The target system file will store the business service definition.

4 Select **Finish** to deploy the business service.

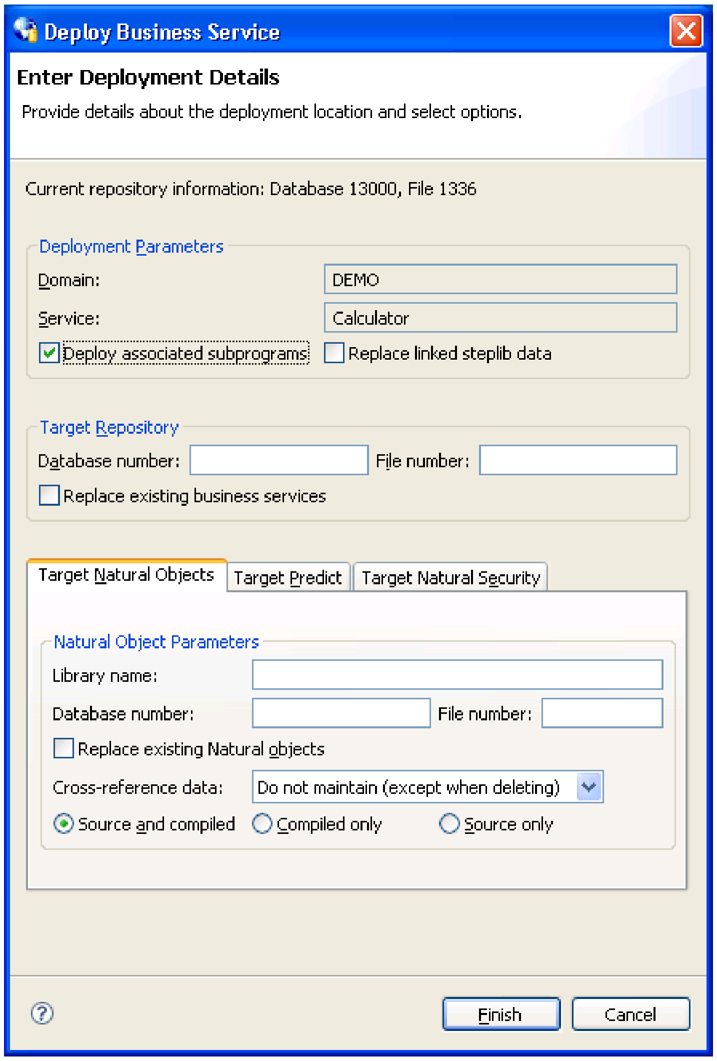
Deploy Associated Subprograms

You can deploy all subprograms associated with a 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 **Enter Deployment Details** window, the target setting tabs become active. For example:



Enter Deployment Details

Provide details about the deployment location and select options.

Current repository information: Database 13000, File 1336

Deployment Parameters

Domain:	DEMO
Service:	Calculator

[✓] Deploy associated subprograms
[ ] Replace linked steplib data

Target Repository

Database number: File number:

[ ] Replace existing business services

Natural Object Parameters

Library name:

Database number: File number:

[ ] Replace existing Natural objects

Cross-reference data: Do not maintain (except when deleting)

[● Source and compiled] ○ Compiled only ○ Source only

The tabs for the target settings are:

- Target Natural Objects
- Target Predict

- Target Natural Security

Target Natural Objects

Specify the following target settings for the Natural objects:

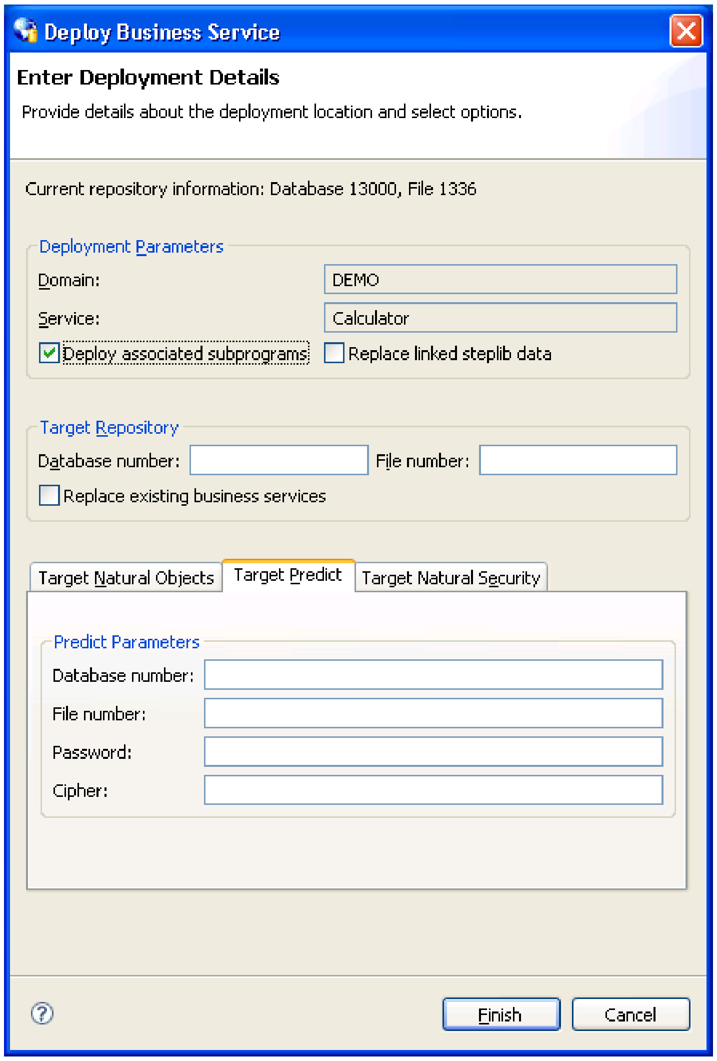
Target Setting	Description
Library name	Name of the Natural library in which to copy all Natural objects that implement the business service being deployed.
Database number	Database ID for the FUSER (or FNAT) file that will store the Natural objects (subprograms, data areas, etc).
File number	File number 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 the existing Natural code (subprograms, data areas, etc.) in the target FUSER (or FNAT).	Select **Replace existing Natural objects**.
Maintain cross-reference data using an alternate method.	Select another setting in **Cross-reference data**. Cross-reference settings 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:


Deploy Business Service

Enter Deployment Details

Provide details about the deployment location and select options.

Current repository information: Database 13000, File 1336

Deployment Parameters

Domain: DEMO

Service: Calculator

[✓] Deploy associated subprograms [ ] Replace linked steplib data

Target Repository

Database number: File number:

[ ] Replace existing business services

Target Natural Objects | Target Predict | Target Natural Security

Predict Parameters

Database number:

File number:

Password:

Cipher:

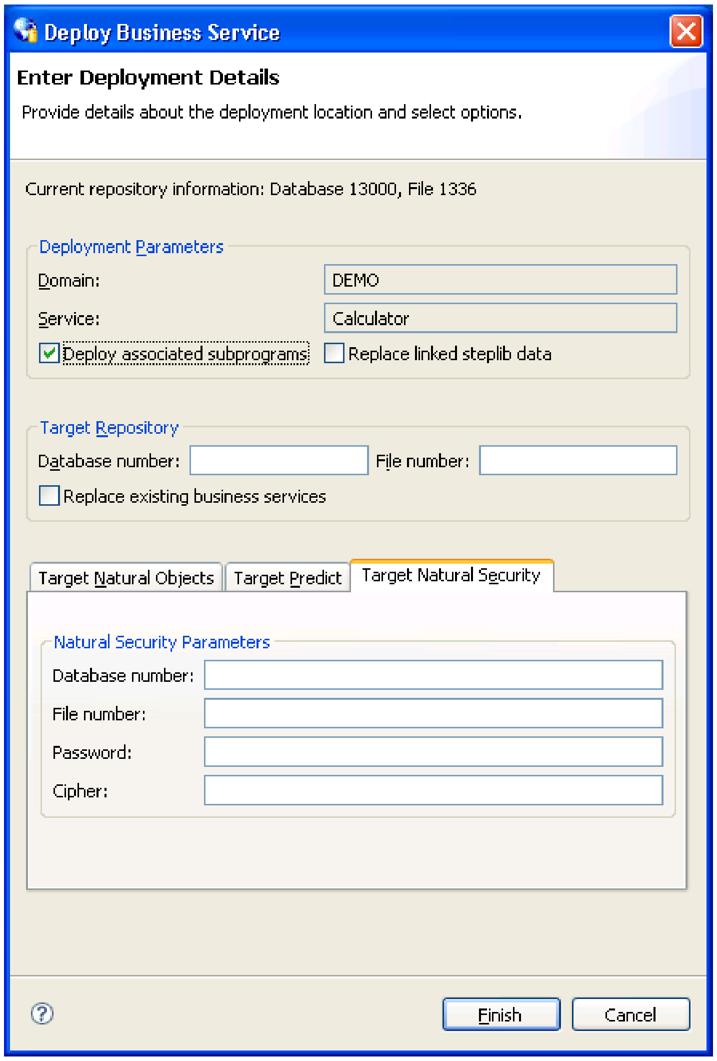
Finish Cancel

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:



Enter Deployment Details

Provide details about the deployment location and select options.

Current repository information: Database 13000, File 1336

Deployment Parameters

Domain: DEMO

Service: Calculator

[✓] Deploy associated subprograms [ ] Replace linked steplib data

Target Repository

Database number: File number:

[ ] Replace existing business services

Target Natural Objects | Target Predict | Target Natural Security

Natural Security Parameters

Database number:

File number:

Password:

Cipher:

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.

Access the Communication Log Report

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

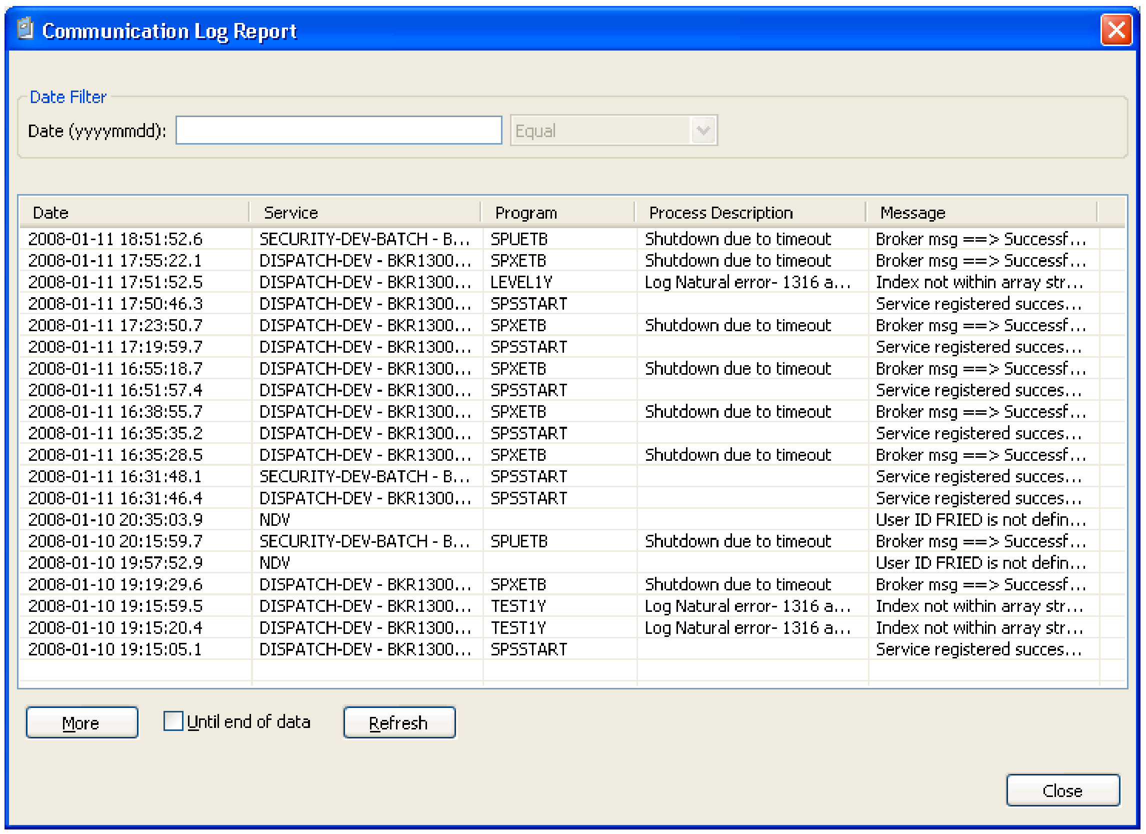
To access the Communication Log report:

1 Open the context menu for the connection in the **NBS Repositories** view.

For an example of this menu, see *Access Connection Options*.

2 Select **Communication log report**.

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



Date (yyyymmdd):

Date	Service	Program	Process Description	Message
2008-01-11 18:51:52.6	SECURITY-DEV-BATCH - B...	SPUETB	Shutdown due to timeout	Broker msg ==> Successf...
2008-01-11 17:55:22.1	DISPATCH-DEV - BKR1300...	SPXETB	Shutdown due to timeout	Broker msg ==> Successf...
2008-01-11 17:51:52.5	DISPATCH-DEV - BKR1300...	LEVEL1Y	Log Natural error- 1316 a...	Index not within array str...
2008-01-11 17:50:46.3	DISPATCH-DEV - BKR1300...	SPSSTART		Service registered succes...
2008-01-11 17:23:50.7	DISPATCH-DEV - BKR1300...	SPXETB	Shutdown due to timeout	Broker msg ==> Successf...
2008-01-11 17:19:59.7	DISPATCH-DEV - BKR1300...	SPSSTART		Service registered succes...
2008-01-11 16:55:18.7	DISPATCH-DEV - BKR1300...	SPXETB	Shutdown due to timeout	Broker msg ==> Successf...
2008-01-11 16:51:57.4	DISPATCH-DEV - BKR1300...	SPSSTART		Service registered succes...
2008-01-11 16:38:55.7	DISPATCH-DEV - BKR1300...	SPXETB	Shutdown due to timeout	Broker msg ==> Successf...
2008-01-11 16:35:35.2	DISPATCH-DEV - BKR1300...	SPSSTART		Service registered succes...
2008-01-11 16:35:28.5	DISPATCH-DEV - BKR1300...	SPXETB	Shutdown due to timeout	Broker msg ==> Successf...
2008-01-11 16:31:48.1	SECURITY-DEV-BATCH - B...	SPSSTART		Service registered succes...
2008-01-11 16:31:46.4	DISPATCH-DEV - BKR1300...	SPSSTART		Service registered succes...
2008-01-10 20:35:03.9	NDV			User ID FRIED is not defin...
2008-01-10 20:15:59.7	SECURITY-DEV-BATCH - B...	SPUETB	Shutdown due to timeout	Broker msg ==> Successf...
2008-01-10 19:57:52.9	NDV			User ID FRIED is not defin...
2008-01-10 19:19:29.6	DISPATCH-DEV - BKR1300...	SPXETB	Shutdown due to timeout	Broker msg ==> Successf...
2008-01-10 19:15:59.5	DISPATCH-DEV - BKR1300...	TEST1Y	Log Natural error- 1316 a...	Index not within array str...
2008-01-10 19:15:20.4	DISPATCH-DEV - BKR1300...	TEST1Y	Log Natural error- 1316 a...	Index not within array str...
2008-01-10 19:15:05.1	DISPATCH-DEV - BKR1300...	SPSSTART		Service registered succes...

This window shows the first 20 communication status and error messages in descending order by date (the default). To sort by another column, select that column heading.

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

Optionally, you can use this window to:

- Restrict the list of messages by entering a date and specifying selection 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 by selecting **More**. A scroll bar is displayed to scroll through the additional messages.
- Display all messages by selecting **Until end of data** and then selecting **More**. A scroll bar is displayed to scroll through the messages.

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

Note: This log is 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 **NBS Repositories** view.

2 Select **Generate Natural client**.

The Eclipse **Console** view shows the name of the subprogram proxy generated for the Natural client.

6 Developing Client Proxy Classes and Web Services

Generate a Client Proxy Class

You can use the Natural Business Services Eclipse plug-in to create proxy classes in Java. Client proxy classes provide access to business services running in Natural. The generated classes are plain Java classes. Typically, proxy classes have properties and methods that map to their Natural counterparts.

Note: This procedure differs, based on whether the target Java Virtual Machine (VM) is original or enhanced.

During generation, the wizard creates a folder called .nbsMetadata containing the specifications for the Java class. This allows the specifications to be associated with a project rather than a workspace. The metadata can be stored in CVS, as well as exported with the project.

Note: A Java project must currently exist. For information, refer to the Eclipse documentation.

This section covers the following topics:

- Java Original
- Java Enhanced
- Deploy the Client Proxy Class as a Web Service Class

Java Original

This section describes how to generate a client proxy class for Java Virtual Machine (VM) version 1.4.

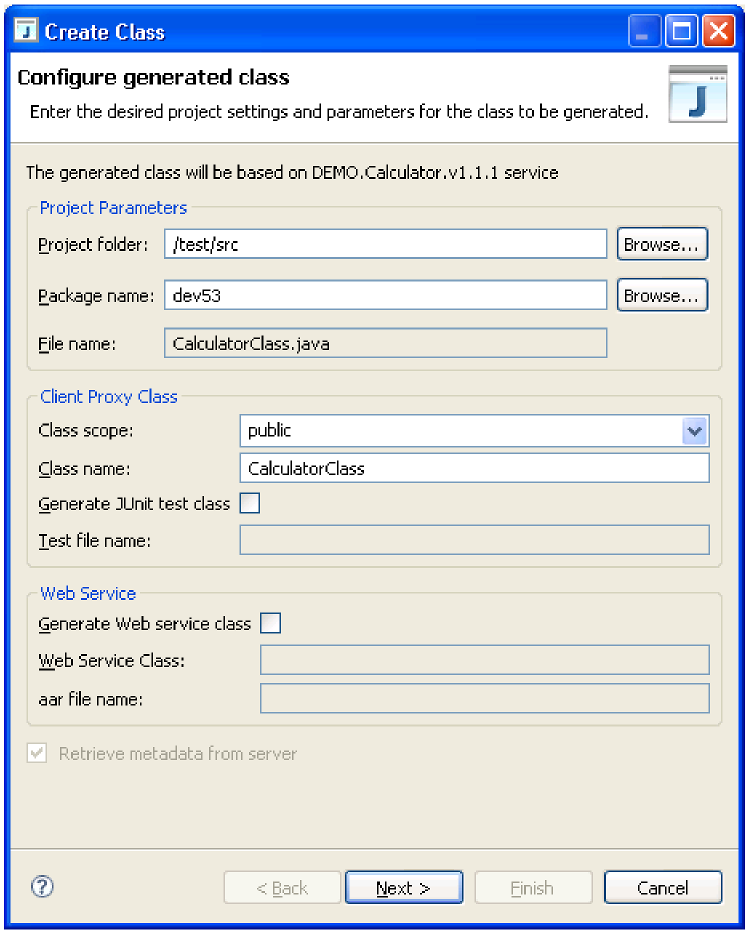
Note: For information on setting the VM version, see *Preferences for Java Classes*.

To generate a client proxy class for Java original:

1 Open the context menu for a business service in the **NBS Repositories** view.

2 Select **Create class**.

The **Configure generated class** panel is displayed. For example:


Create Class
Configure generated class
Enter the desired project settings and parameters for the class to be generated.

The generated class will be based on DEMO.Calculator.v1.1.1 service

Project Parameters
Project folder: /test/src
Package name: dev53
File name: CalculatorClass.java

Client Proxy Class
Class scope: public
Class name: CalculatorClass
Generate JUnit test class ☐
Test file name:

Web Service
Generate Web service class ☐
Web Service Class:
aar file name:

☑ Retrieve metadata from server

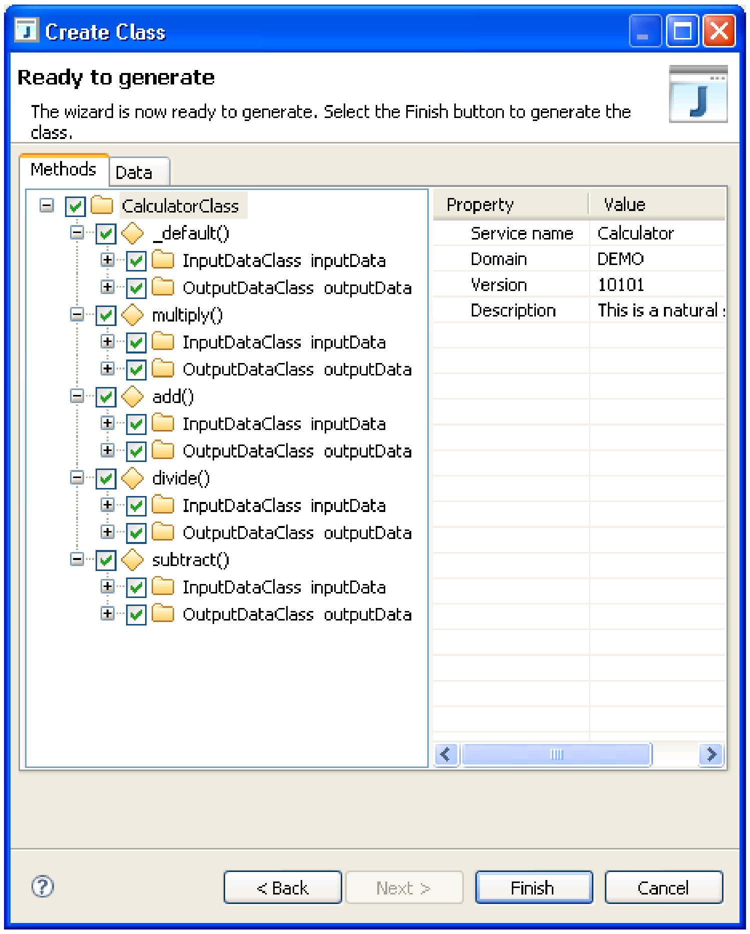
This panel displays the default configuration settings for the class to be generated. Optionally, you can:

Task	Procedure
Change the project folder.	Select a new root source folder in **Project folder**.
Change the package name.	Select a new package in **Package name**.
Change the scope of the class.	Select the class scope from **Class scope**. By default, public is the class scope.
Rename the class.	Enter a new name in **Class name**.
Create a class that can be run in the JUnit testing framework.	Select **Generate JUnit test class**. This test class will contain default tests for each method used by your business service. The name of the file used to store the test is displayed in **Test file name**.
Create additional Web service deployment files and classes.	Select **Generate Web service class** and enter the Web service class name and .aar file name. For more information, see *Deploy the Client Proxy Class as a Web Service Class*.

Task	Procedure
Rename the Web service.	Enter a new name in **Web service name**.
Refresh metadata from the server.	Select **Retrieve metadata from server**.

3 Select **Next**.

The wizard downloads metadata from the service repository on the server and displays the **Ready to generate** panel. For example:

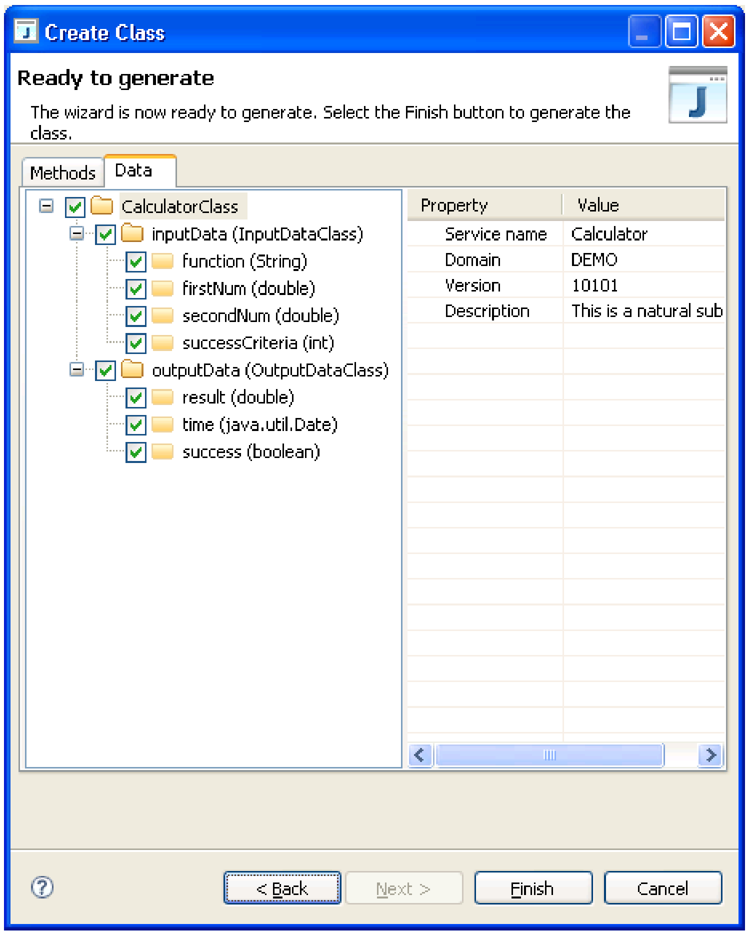


Property	Value
Service name	Calculator
Domain	DEMO
Version	10101
Description	This is a natural :

Use this panel to customize the methods and data parameters your class will support. The **Methods** tab allows you to customize how class methods will be generated based on your existing business service methods.

4 Select the **Data** tab.

For example:


Create Class

Ready to generate

The wizard is now ready to generate. Select the Finish button to generate the class.

```
⊟ ✓ CalculatorClass
   ⊟ ✓ inputData (InputDataClass)
         ✓ function (String)
         ✓ firstNum (double)
         ✓ secondNum (double)
         ✓ successCriteria (int)
   ⊟ ✓ outputData (OutputDataClass)
         ✓ result (double)
         ✓ time (java.util.Date)
         ✓ success (boolean)
```

Property	Value
Service name	Calculator
Domain	DEMO
Version	10101
Description	This is a natural sub

The **Data** tab allows you to customize how Get and Set methods for your business service properties will be generated.

5 Select **Finish** to generate the client proxy class.

Java Enhanced

This section describes how to generate a client proxy class that takes advantage the enhanced features of Java Virtual Machine (VM) version 5. These features include:

- Try/Catch statements that throw exceptions for server errors when calling business service methods for a class
- Method definitions that include the correct input/output parameters based on the style of business service

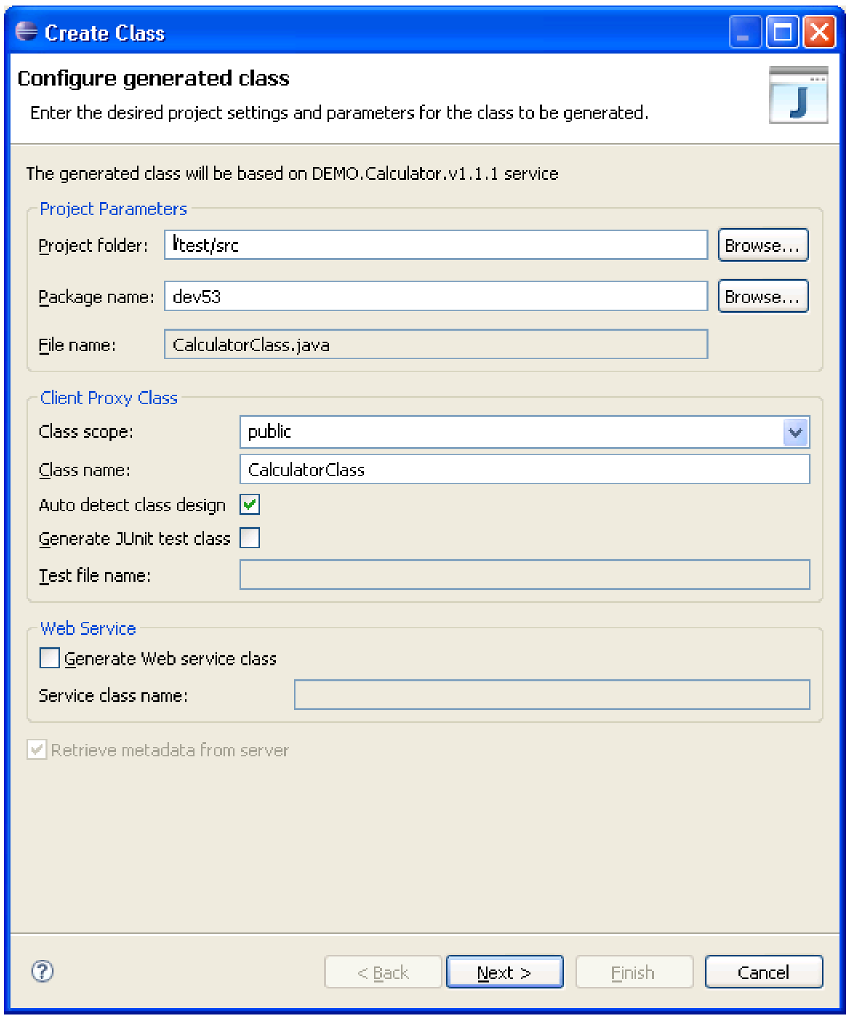
Note: For information on setting the VM version, see *Preferences for Java Classes*.

To generate a client proxy class for Java enhanced:

1 Open the context menu for a business service in the **NBS Repositories** view.

2 Select **Create class**.

The **Configure generated class** panel is displayed. For example:



Configure generated class

Enter the desired project settings and parameters for the class to be generated.

The generated class will be based on DEMO.Calculator.v1.1.1 service

Project Parameters

Project folder:	/test/src
Package name:	dev53
File name:	CalculatorClass.java

Client Proxy Class

Class scope:	public
Class name:	CalculatorClass
Auto detect class design	✓
Generate JUnit test class	☐
Test file name:	

Web Service

☐ Generate Web service class

Service class name:

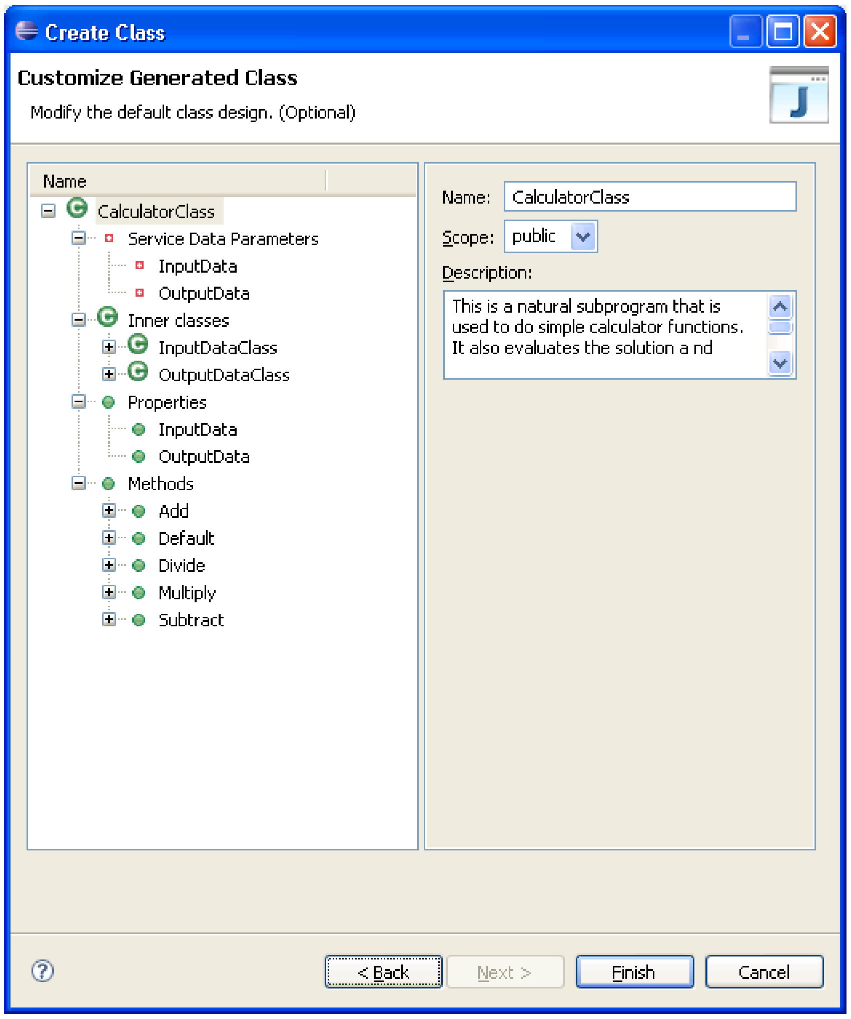
✓ Retrieve metadata from server

This panel displays the default configuration settings for the class to be generated. Optionally, you can:

Task	Procedure
Change the project folder.	Select a new root source folder in **Project folder**.
Change the package name.	Select a new package in **Package name**.
Change the scope of the class.	Select the class scope from **Class scope**. By default, public is the class scope.
Rename the class.	Enter a new name in **Class name**.
Copy the field structure as defined in the PDAs generated for the business service subprograms.	Deselect **Auto detect class design**. If this option is selected, the wizard will determine which model to use when generating the subprograms on the server and design an object class hierarchy that is appropriate for the model. For example, the generated Get method will accept the primary key as input and return an object that includes all fields. The wizard recognizes subprograms generated by the Object-Maint-Subp, Object-Browse-Subp, and Object-Browse-Select-Subp models.
Create a class that can be run in the JUnit testing framework.	Select **Generate JUnit test class**. This test class will contain default tests for each method used by your business service. The name of the file used to store the test is displayed in **Test file name**.
Create additional Web service deployment files and classes.	Select **Generate Web service class** and enter the Web service class name in **Service class name**. For more information, see *Deploy the Client Proxy Class as a Web Service Class*.
Rename the Web service.	Enter a new name in **Web service name**.
Refresh metadata from server.	Select **Retrieve metadata from server**.

3 Select **Next**.

The wizard downloads metadata from the service repository on the server and displays the **Customize Generated Class** panel. For example:



Name: CalculatorClass

Scope: public

Description:

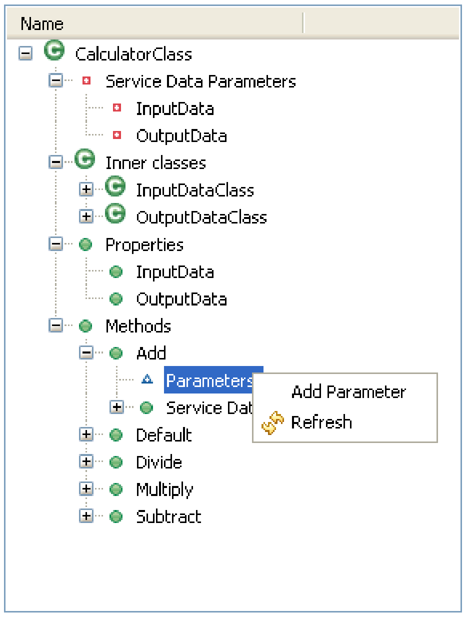
This is a natural subprogram that is used to do simple calculator functions. It also evaluates the solution a nd

Use this panel to design how each method will work when generating Java classes for services that were not generated by Natural Construct models.

4 Expand a method listed in **Methods**.

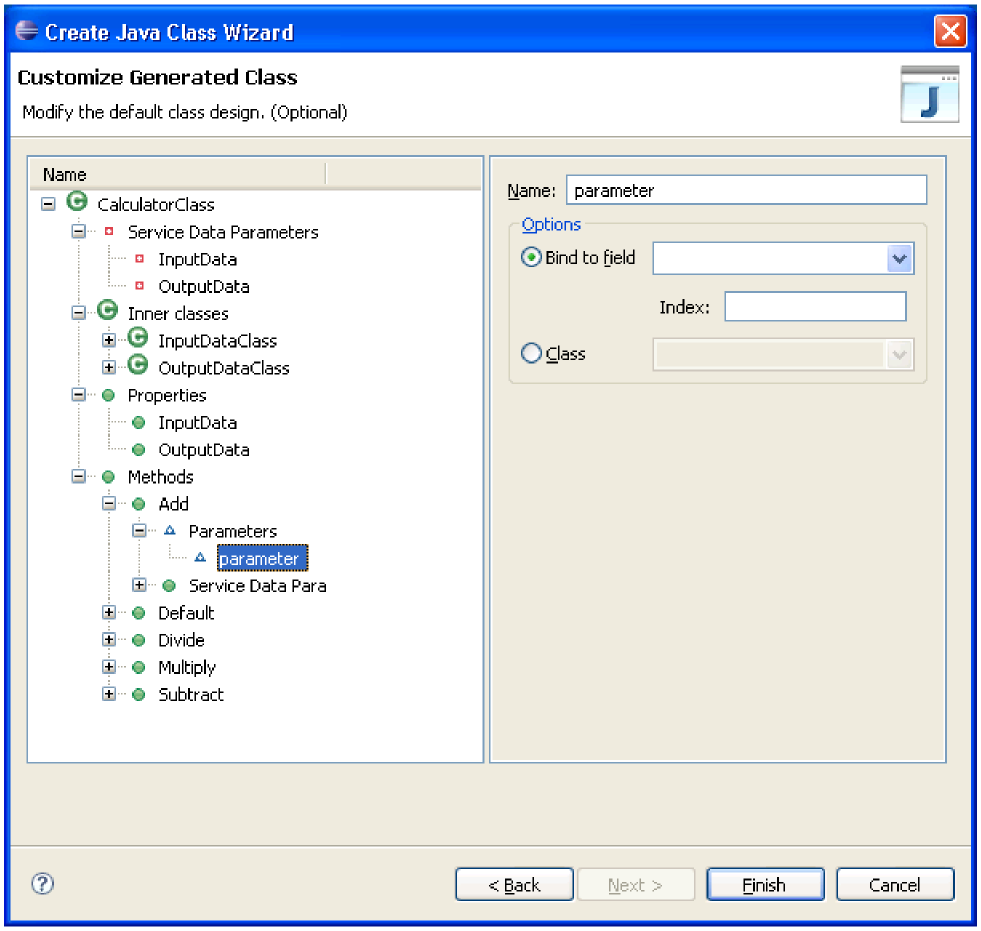
5 Open the context menu for Parameters.

For example:



6 Select **Add Parameter**.

The parameter settings are displayed. For example:



Use this panel to add a parameter to the method and bind it to an existing field in the service. Optionally, you can:

Task	Procedure
Provide an index value for the Natural field.	Type the value in **Index**.
Bind the parameter to an inner class.	Select **Class** and specify the class.
Select the return value for a method.	Select the method (for example, Add) and specify the return settings. For information, see *Select Return Value for a Method*.

7 Type the name of the parameter in **Name**.

8 Select the Natural field in **Bind to field**.

9 Select **Finish** to generate your classes and files.

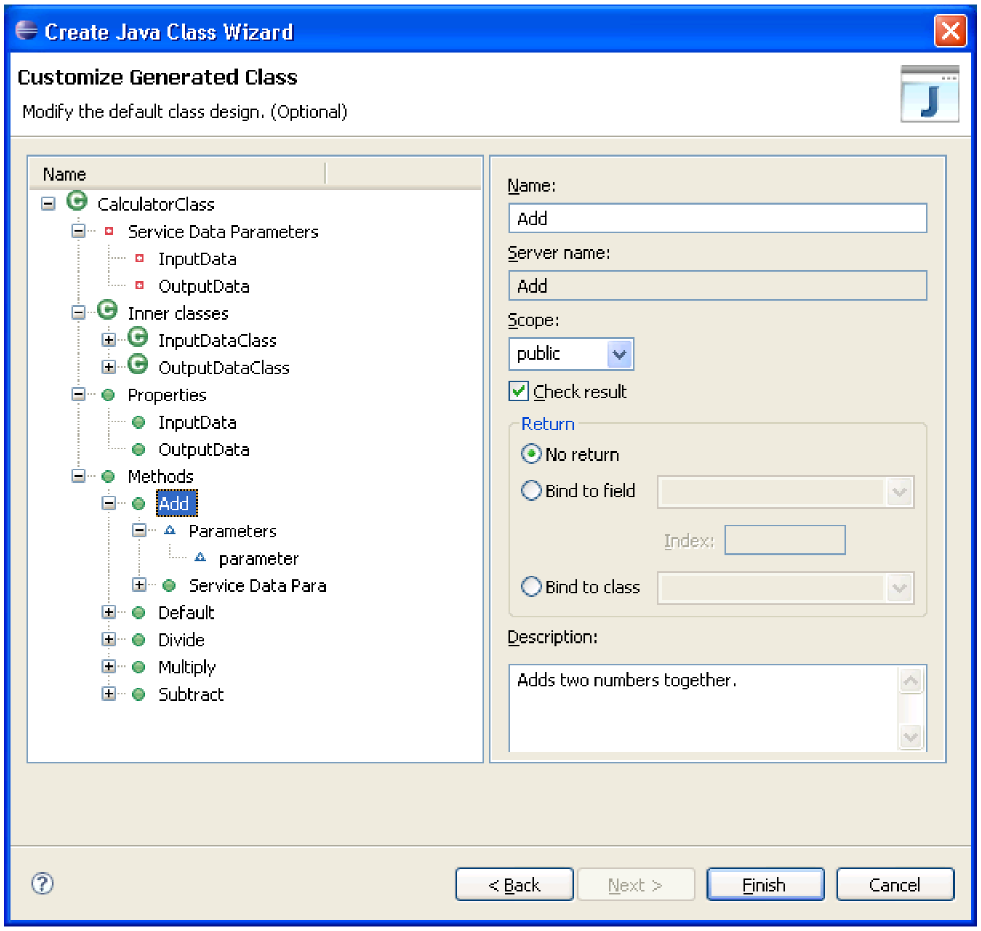
The new class is displayed in the program editor view.

Select Return Value for a Method

To select the return value for a method:

1 Select a method listed in **Methods**.

Information about the method is displayed. For example:



Create Java Class Wizard

Customize Generated Class

Modify the default class design. (Optional)

Name
- CalculatorClass
  - Service Data Parameters
    - InputData
    - OutputData
  - Inner classes
    - InputDataClass
    - OutputDataClass
  - Properties
    - InputData
    - OutputData
  - Methods
    - Add
      - Parameters
        - parameter
      - Service Data Para
    - Default
    - Divide
    - Multiply
    - Subtract

Name: Add

Server name: Add

Scope: public

[✓] Check result

Return

[✓ No return] Bind to field Index: Bind to class

Description: Adds two numbers together.

This panel displays the name of the method on the client and on the server, as well as the scope (public or private) and description of the method. Optionally, you can:

Task	Procedure
Change the name of the method.	Type a new name in **Name**.
Change the scope of the method.	Select the new scope in **Scope**.
Remove the option to verify the result.	Deselect **Check result**.
Change the description of the method.	Type a new description in **Description**.

2 Select one of the following binding options for the return value:

- Bind to field

  To bind the return value to a field, select the field and, optionally, provide an index value.

- Bind to class

  To bind the return value to an inner class, select the class.

For example, if you select **Bind to field** and OUTPUT-DATA.#RESULT, the Add method changes from:

```
public void Add() throws BusinessServiceException
```

to:

```
public double Add(double firstNum,
                  double secondNum)
       throws BusinessServiceException
```

Deploy the Client Proxy Class as a Web Service Class

When generating a client proxy class, one of the wizard options generates a class and descriptor files to deploy the client proxy class as a Web service class. Web service deployment is handled by the WS-Stack plug-in. For information about this plug-in, refer to the webMethods shared components documentation.

Note: To deploy the client proxy class as a Web service class, **Generate Web service class** must be selected on the **Configure Generated Class** panel.

This section covers the following topics:

- Add CentraSite Associations
- Deploy to an Internal Axis Server
- Deploy to an External Application Server

- Override the Generated Connection Settings

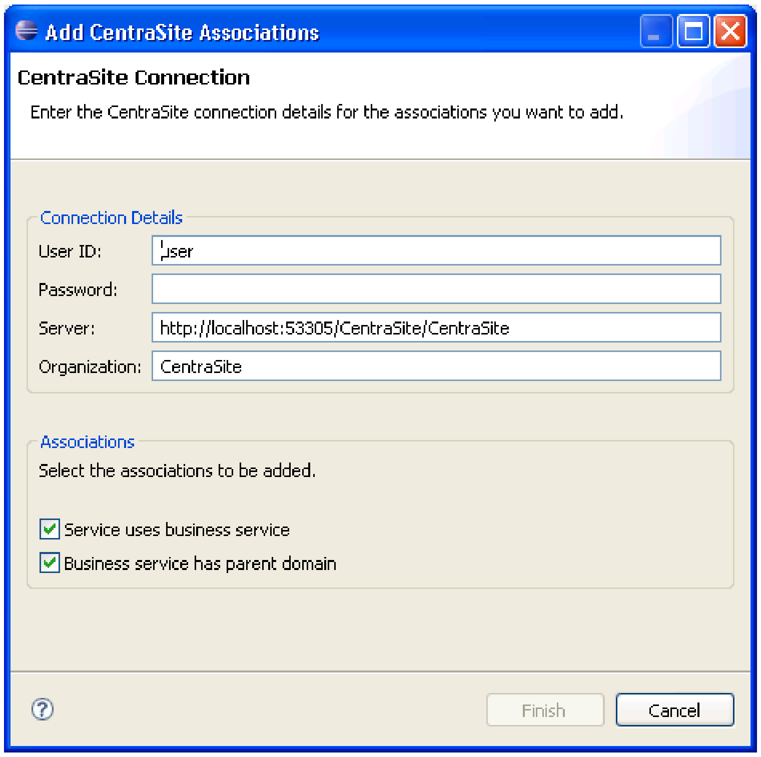
Add CentraSite Associations

The WS-Stack plug-in will not add an association between a Web service being added to CentraSite and the NBS repository metadata in CentraSite. You must add each association manually.

To add associations to CentraSite:

1. Open the context menu for the .aar file in the **Package Explorer** view.
2. Select **Business Services > Add CentraSite association**.

   The **CentraSite Connection** window is displayed. For example:


Add CentraSite Associations
CentraSite Connection
Enter the CentraSite connection details for the associations you want to add.
Connection Details

User ID: user
Password:
Server: http://localhost:53305/CentraSite/CentraSite
Organization: CentraSite

Associations
Select the associations to be added.

[✓] Service uses business service
[✓] Business service has parent domain

Use this window to add up to two associations between a Web service and the NBS repository metadata to CentraSite. These associations are:

Association	Description
Service uses business service	Association between the Web service and its corresponding NBS business service metadata (for example, CalculatorClassWS Uses DEMO.Calculator.v1.1.1).
Business service has parent domain	Association between the NBS business service metadata and its corresponding NBS domain metadata (for example, DEMO.Calculator.v1.1.1 HasParent DEMO).

3 Confirm the connection details.

4 Type the CentraSite password in **Password**.

5 Select the associations you want to add.

6 Select **Finish**.

The selected associations are added to CentraSite.

Deploy to an Internal Axis Server

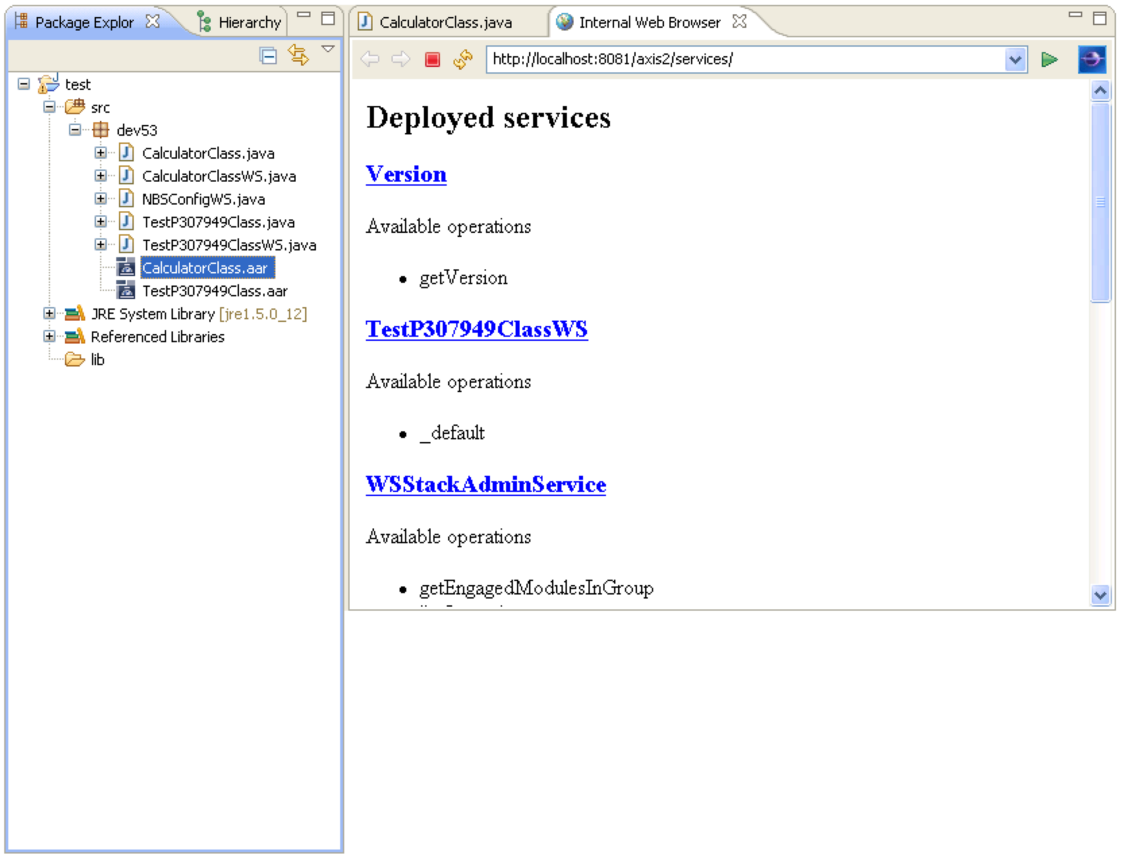
The Web service can be deployed to an internal Axis server. The internal server will use your PC name as the web server name.

To deploy to an internal Axis server:

1 Open the context menu for the .aar file in the **Package Explorer** view.

2 Select **Business Services > Internal Axis Deploy**.

The deployed service is displayed in the Internal Web Browser view. For example:



Deployed services

Version

Available operations

- getVersion

TestP307949ClassWS

Available operations

- _default

WSStackAdminService

Available operations

- getEngagedModulesInGroup

Deploy to an External Application Server

The Web service can be deployed to an external application server. This functionality is handled by the WS-Stack plug-in. For information, refer to the webMethods shared components documentation.

Before you can deploy the Web service, you must prepare the application server.

To prepare the application server for Web service deployment:

- Copy the *NBSAxis2.war* file to the *..\tomcat\webapps\Axis2 Install* folder.

Note: While deploying a Web service, the wizard can generate debug log statements. For information, see *Generate log4J Log Statements*.

Tip: After making changes (such as changing the Java runtime version used by your classes and then redeploying the class), you may have to restart the application server.

Tip: If an error occurs while deploying a service to an external application server, ensure that the Java version running on the server is correct. To determine which version is running, open the context menu for your Java project and select **Properties**. View the Java Build Path and verify the JRE system library setting.

Override the Generated Connection Settings

When deploying a client proxy class as a Web service class, the Java Class wizard creates two files: the .aar file and the NBSConfigWS class. This class is responsible for creating the dispatcher and setting the credentials used to communicate with the server. If desired, you can override the connection settings generated into the NBSConfigWS class by overwriting the settings in the *NBSAxis.properties* file (located in *C:\Documents and Settings\All Users\Application Data\Software AG\Natural Business Services\Configuration*).

Customize a Generated Client Proxy Class

After generating a client proxy class, you can customize the code that was generated for a method or field used by the class. However, regeneration will overwrite your changes unless you place the following annotation immediately preceding the customized code:

```
@NBSPreserve("")
```

The annotation preserves the code for that method or field only.

Note: Although providing a comment about the customization is optional, you must include the comment indicators (i.e., the brackets and quotation marks).

Use the Generated Client Proxy Classes

This section describes the contents of the generated client proxy classes, how to execute the methods of a client proxy, and how to use the Natural Business Services runtime client dispatcher. The following topics are covered:

- Contents of a Generated Client Proxy Class
- Execute Methods of a Client Proxy Class

- Use the Natural Business Services Runtime Client Dispatcher

Contents of a Generated Client Proxy Class

Generated client proxy classes have the following contents:

- Init Method
- Methods
- Data

Init Method

This method contains a parameter called IRemoteCaller, which communicates with the server (it is also called a client dispatcher). You must create an instance of a class that supports the IRemoteCaller interface. Use the NBS runtime and ServiceFactory contained in the dispatchclient package to do this. For example:

```
import com.softwareag.nbs.bshelper.*;
import com.softwareag.nbs.dispatchclient.*;

Config.getDispatchClientConfig("c:/eclipse/plugins/com.softwareag.nbs.ui_5.3.1/DispatchClient.config");
IRemoteCaller dispatcher = ServiceFactory.createDispatcher("Default Broker");
dispatcher.setUserID("GUEST");
dispatcher.setPassword("");
dispatcher.logon();

CalculatorClass bs = new CalculatorClass();
bs.init(dispatcher);
```

Note: Although calling the `dispatcher.logoff()` method is not mandatory, it is recommended. Calling the logoff method is important, especially with SPoD connections, as it releases server connections sooner rather than waiting for them to be released during garbage collection. Calling the logon method is mandatory.

When using the ServiceFactory to create an instance of IRemoteCaller, you must pass in the name of a connection ID. For example:

```
IRemoteCaller dispatcher = ServiceFactory.createDispatcher("My Connection");
```

This is the name you specified when creating new connections in the **NBS Repositories** view. You must also load the configuration file using the getDispatchClientConfig method of the Config object. For example:

```
Config.getDispatchClientConfig("c:/My NBS Files/DispatchClient.config");
```

Methods

Each method in the class corresponds to a method in the business service.

Data

Each level 1 field from a PDA used in a business service becomes a Get/Set function of the class. Each group within a PDA becomes a subclass. For example, if the following definition is specified:

```
01 Group1
  02 Group2
    03 Field1 (A10)
```

The following is generated in the client proxy class:

```
public class Group1 {

  public class Group2 {

     public String getField1() {
       return pda_inputData.getString("Field1",_ax);
     }
  }
}
```

Execute Methods of a Client Proxy Class

When executing a method of a client proxy class, use the BusinessServiceResult return value to check for communication and/or runtime errors. For example:

```
BusinessServiceResult bsr = businessSerivce.add(); // Invoke the add method.
if (bsr.isSuccess())
{
   // Success
}
else
{
   String s = bsr.getDispatchResult().toString();
}
```

If your business service was generated using code generation patterns, or it uses a standard error message parameter group, verify whether business service errors or warnings are present. The msg field will also contain status messages. For example:

```
if (bs.getMessage().getReturnCode() == "E")
{
  String s = bs.getMessage().getMsg(); // Error, warning or status message
  Int I = bs.getMessage().getMsgNr(); // Error number
}
```

Use the Natural Business Services Runtime Client Dispatcher

The runtime client dispatcher that uses the IRemoteCaller interface has properties and methods you can use to provide additional functionality.

Set Security

When creating dispatcher objects, you must set the correct user ID and password to be used for remote calls. For example:

```
dispatcher.setUserID("GUEST");
dispatcher.setPassword("secret");
```

Use Transactions

You can invoke business services in a transactional context by using the `startTransaction`, `commit`, and `abort` methods. For example:

```
Service1 sv1 = new Service1Class();
Service2 sv2 = new Service2Class();

IRemoteCaller dispatcher = ServiceFactory.createDispatcher("Default Broker");
sv1.init(dispatcher);
sv2.init(dispatcher);

dispatcher.startTransaction();
// Call methods of sv1 and sv2.
...
// If successful calls
dispatcher.commit();
...
// If an error in a method
dispatcher.abort();
```

Note: To control transactions, the same dispatcher object must be used for each business service. For more information, see the Java transaction example in the Samples folder. You can copy the Samples folder from the installation CD.

Generate log4j Log Statements

While creating a Java class and deploying a Web service to an external application server (for example, Apache Tomcat), the wizard can generate debug log statements into each Web service method that corresponds to a business service method.

Each Web service request is recorded in a log file with the following information:

- Name of the Web service (unique identifier)

- User ID (found in DispatchClient.config)
- Timestamp indicating when the application server retrieves the Web service (Start of Request)
- Timestamp indication when the Web service leaves the application server (End of Request)
- Calculated time difference between the two timestamps (Request Elapsed Time)

To enable logging:

1 Edit the *log4j.properties* file.

This file is located in a subfolder in the *\NBSAxis2\WEB-INF\classes* folder where your application server is running.

2 Add the following line:

```
log4j.logger.[package]=DEBUG, CONSOLE
```

where `[package]` is the Java package for your project containing the generated Java and Web service classes. This will log all Web services in this package to the log file.

Tip: You can also include the class name after the package name, for example `[package].[classnameWS]`, which will enable logging for that class only.

3 Restart your application server.

Entries will be written to log files for standard output.

The log files use the following naming convention:

```
stdout_[date].log ↩
```

These files are stored in different locations, depending on the application server you are using. (For Apache Tomcat, they are stored in the Logs subfolder in which Tomcat is installed.)

For more information, see:

- Log4J manual: **http://logging.apache.org/log4j/1.2/manual.html**
- Using Log4J from Tomcat: **http://tomcat.apache.org/tomcat-5.5-doc/logging.html**