

# Construct Spectrum SDK for Microsoft .NET Framework

## **Manual Order Number: SPV451-023IBW**

This document applies to Construct Spectrum SDK Version 4.5 and to all subsequent releases.

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

Readers' comments are welcomed. Comments may be addressed to the Documentation Department at the address on the back cover or to the following e-mail address:

[Documentation@softwareag.com](mailto:Documentation@softwareag.com)

© Copyright Software AG 2003

All rights reserved

Printed in the Federal Republic of Germany

The name Software AG and/or all Software AG product names are either trademarks or registered trademarks of Software AG. Other company and product names mentioned herein may be trademarks of their respective owners.

# TABLE OF CONTENTS

## 1. WELCOME

Overview . . . . .	2
Supplied Wizards. . . . .	4
Prerequisite Knowledge. . . . .	5
Purpose and Structure of this Document. . . . .	6
Document Conventions. . . . .	7
Naming Conventions for Examples. . . . .	8
Other Resources. . . . .	9
Related Documentation . . . . .	9
Construct Spectrum SDK . . . . .	9
Construct Spectrum . . . . .	10
Natural Construct. . . . .	10
Other Documentation . . . . .	11
Related Courses. . . . .	11

## 2. CONFIGURING THE SDK

Introduction . . . . .	14
Configuring the Runtime Component . . . . .	16
Step 1: Specify the Configuration Editor Settings . . . . .	16
Step 2: Set the Runtime Dispatcher. . . . .	18

## 3. USING THE SDK

Preparing a Natural Subprogram for Client/Server Access. . . . .	20
Step 1: Generate the Subprogram Proxy . . . . .	20
Step 2: Verify Spectrum Data (Optional) . . . . .	21
Generating a New Web Service . . . . .	23
Step 1: Invoke the Web Service Wizard (WSW) . . . . .	24
Step 2: Select the Subprogram Proxy . . . . .	25
Step 3: Configure the Methods (Optional) . . . . .	27
Step 4: Configure the Parameter Data Areas (Optional) . . . . .	29
Step 5: Configure the Web Service (Optional) . . . . .	31
Step 6: Generate the Service . . . . .	32
Regenerating an Existing Web Service. . . . .	35
Making Changes Accessible to Users. . . . .	36

Using Business Data Types (BDTs) .....	37
BDT Modifiers .....	37
Supplied BDTs .....	38
Validating Input .....	42
Testing Your Web Service .....	43
Building a Web Application .....	47
Step 1: Create the Web Application .....	48
Step 2: Generate a Web Page .....	51
Configure HTML Option .....	56
Advanced Options Window .....	58
Configure Methods Option .....	61
Configure Tertiary Display Option .....	62
Configure Browse Keys Option .....	63
Regenerating a Web Page .....	63
Step 3: Generate a Menu .....	64
Regenerating a Menu .....	65

#### 4. CUSTOMIZING

Modifying the Web.config File .....	68
Event Hierarchy and Inheritance .....	71
.SWS Mapping .....	72
Creating a Request for a Spectrum Security Token .....	73
Adding Custom Code .....	74
Web Service Engine (WSE) Events .....	74
ValidateUser Event .....	74
BeforeCallnat Event .....	74
AfterCallnat Event .....	75
ExceptionThrown Event .....	75
Event Arguments .....	75
SPEEventArgs .....	75
ValidateUserEventArgs .....	76
BeforeCallnatEventArgs .....	76
AfterCallnatEventArgs .....	76
ExceptionThrownEventArgs .....	77
Create a Custom Event Handler Class .....	77
Modify the Web.config File for Custom Code .....	78
Creating a Custom BDT .....	79
Use the BDT During Generation .....	79
BDTs.xml File .....	80
BDT Node .....	80
DataType Node .....	81
Adding a Web Services Root Directory .....	82

**5. TIPS AND TECHNIQUES**

Tips for Data Updates . . . . . 84

    Verifying Namespace Names . . . . . 84

    Excluding Attributes During an Update . . . . . 84

    Using the CDAOBJ2 Parameter Data Area . . . . . 85

Miscellaneous Tips . . . . . 86

    Overriding Web Service Wizard BDT Defaults . . . . . 86

    Setting Up Steplibs . . . . . 86

        Development Environment . . . . . 86

        Runtime Environment . . . . . 86

    Changing the Dispatcher . . . . . 87

        Development Environment . . . . . 87

        Runtime Environment . . . . . 87

    Correcting an Invalid Subprogram Method List . . . . . 87

Supplied Samples . . . . . 88

    Sample SOAP Messages . . . . . 88

        Input for the DELETE Action . . . . . 88

        Input for the EXISTS, GET, and NEXT Actions . . . . . 88

        Input for the INITIALIZE Action . . . . . 89

        Input for the STORE Action . . . . . 89

        Input for the UPDATE Action . . . . . 89

Troubleshooting Common Errors . . . . . 90

    Setting Zero Suppression. . . . . 90

    Correcting Errors During an Update . . . . . 90

**APPENDIX A: GLOSSARY . . . . .91**

**INDEX . . . . .103**



---

## WELCOME

Welcome to *Construct Spectrum SDK for Microsoft .NET Framework*, designed to help developers create and customize Web services using the Construct Spectrum software development kit (SDK). This chapter provides an overview of the SDK and outlines the prerequisite knowledge assumed for this documentation. It describes the purpose and structure of the document, the document conventions, and the naming conventions for examples. It also lists other sources of information about Construct Spectrum and Natural Construct.

The following topics are covered:

- **Overview**, page 2
- **Prerequisite Knowledge**, page 5
- **Purpose and Structure of this Document**, page 6
- **Document Conventions**, page 7
- **Naming Conventions for Examples**, page 8
- **Other Resources**, page 9

## Overview

The Construct Spectrum SDK for Microsoft .NET framework creates Web services that invoke Natural objects from a Windows platform via standard XML conventions (SOAP, WSDL, XSD, etc.). These XML-based Web services provide a standard way to call and integrate existing or new Natural objects into other platforms and systems.

The Construct Spectrum SDK for Microsoft .NET framework simplifies the creation of Spectrum components; you do not have to write, develop, code, or compile in any language other than Natural. This SDK supports and utilizes the newest features of Visual Basic.NET, part of Microsoft's Visual Studio.NET. Those who want to continue using Visual Basic can do so. Others can use the new framework that does not require Visual Basic coding, but takes advantage of the .NET platform. For advanced customizations, you can code in your choice of .NET languages, including Visual Basic, C#, J#, and JavaScript.NET.

Using the .NET technology, you can test existing Construct Spectrum runtime and generation components that were written in Visual Basic and compiled as ActiveX/COM components. Or you can migrate these components to .NET and its common language runtime.

---

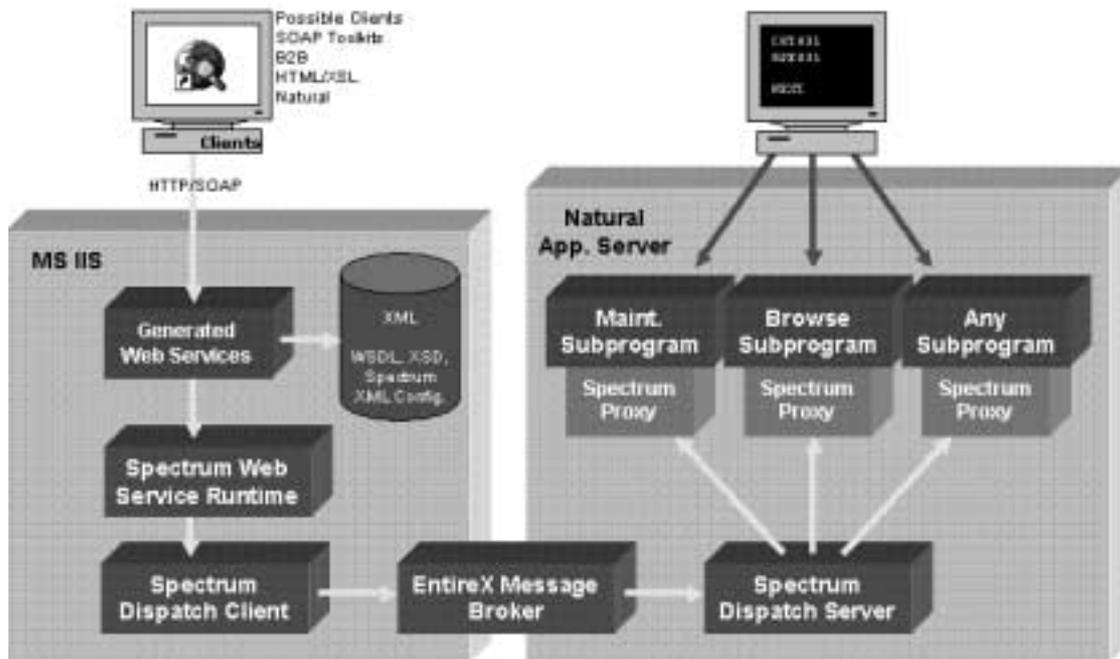
**Note:** If you currently only use the Spectrum runtime components to code custom applications, you can continue to do so after migrating to Visual Studio.NET.

---

The Construct Spectrum SDK for Microsoft .NET framework performs two functions:

- Generates Web services for Natural objects.
- Integrates the Web services into application frameworks that support web (HTML) interfaces, based on .NET technology.

The diagram on the following page illustrates the architecture of the Construct Spectrum SDK for Microsoft .NET framework:



### Architecture of the Construct Spectrum SDK for Microsoft .NET Framework

To create a Web service, you can use an existing Natural Construct-generated object browse or object maintenance subprogram on the mainframe or any Natural subprogram that does not have contain a user interface. Natural Construct-generated objects include various methods, standards, and security integration that make generating a complete application time efficient and standardized. However, the ability to hook into any Natural subprogram provides the practical flexibility required in a production shop.

Natural Construct-generated objects interface with the middleware and client environments through a subprogram proxy and the Construct Spectrum-supplied middleware. Using the Subprogram-Proxy model, you can create the subprogram proxy and security metadata in seconds. With Construct Spectrum installed, mainframe application code is then ready to be accessed by any of the following:

- Web service
- HTML page
- Visual Basic application
- ActiveX component

## Supplied Wizards

When you select Web Services SDK for Microsoft .NET from the Start menu, the following submenu is displayed:



Web Services SDK for Microsoft .NET Submenu

In addition to the Configuration Editor and SOAP Client Testing Tool, the SDK supplies the following wizards:

Wizard	Description
Web Application Builder	
Web Application	Defines the location of the public and private directories containing files for a web application.
Page	Takes information from a Web service and generates a web page to create a B2C environment. For Natural Construct-generated objects, the Web Page wizard for SWS (Spectrum Web Services) determines how Natural Construct handles messages and whether the page should be set up as a maintenance- or browse-style screen.
Menu	Generates a menu (navigation bar) to tie web pages together.
Web Service	Generates a Web service that can take data from the PDAs used by a Natural subprogram, massage it if required (for example, to add commas to dollar values so 7000 becomes 7,000 or to add a compulsory value, such as CITY must equal Toronto), and then place it in an XML message — or reverse this process by taking the data from an XML message and placing it in the PDAs.  For an example of generating a Web service using this wizard, see <b>Generating a New Web Service</b> , page 23.

For information about the Configuration editor, see **Using the Configuration Editor**, page 50, *Construct Spectrum SDK Reference*. For information about the SOAP Client testing tool, see **Testing Your Web Service**, page 43.

## Prerequisite Knowledge

This documentation does not provide information about the following topics. We assume that you are either familiar with the topics or have access to other sources of information about them.

- Natural Construct
- Microsoft .NET
- Predict
- Natural programming language and environment
- EntireX
- XML
- Web service technologies

See **Other Resources**, page 9, for sources of information about Natural Construct and Construct Spectrum.

## Purpose and Structure of this Document

*Construct Spectrum SDK for Microsoft .NET Framework* describes how to use the Construct Spectrum SDK for Microsoft .NET framework to generate a Web service that invokes a Natural subprogram (business object) over the Inter/Intranet via the W3C SOAP standard.

Other chapters in *Construct Spectrum SDK for Microsoft .NET Framework* are:

Chapter	Title	Topics
2	<b>Configuring the SDK,</b> page 13	Describes how to install and configure the Construct Spectrum SDK for Microsoft .NET framework.
3	<b>Using the SDK,</b> page 19	Describes how to generate and regenerate a Web service, prepare a Natural subprogram, use BDTs to format data, test your Web service, and build a web application for the service. This chapter also describes how input is validated when using the wizards.
4	<b>Customizing,</b> page 67	Describes the administrative functions, such as how to modify the Web.config file or create a request for a security token. It also describes how to add custom code, create a custom BDT, and add a Web services root directory.
5	<b>Tips and Techniques,</b> page 83	Contains tips and techniques you can consult when using the Construct Spectrum SDK to create Web services.
Appendix A	<b>Appendix A: Glossary,</b> page 165	Contains definitions of terms used throughout the Construct Spectrum documentation set.

## Document Conventions

This documentation uses the following typographical conventions:

<b>Example</b>	<b>Description</b>
Introduction	Bolded text in cross references indicates chapter and section titles.
“A”	Items within quotation marks indicate values you must enter.
Browse model, GotFocus, Enter	Mixed case text indicates names of: <ul style="list-style-type: none"><li>• Natural Construct and Construct Spectrum editors, fields, files, functions, models, panels, parameters, subsystems, variables, and dialogs</li><li>• Visual Basic classes, constants, controls, dialogs, events, files, menus, methods, properties, and variables</li><li>• Keys</li></ul>
Alt+F1	A plus sign (+) between two key names indicates that you must press the keys together to invoke a function. For example, Alt+F1 means hold down the Alt key while pressing the F1 key.
CHANGE-HISTORY	Uppercase text indicates the names of Natural command keywords, command operands, data areas, help routines, libraries, members, parameters, programs, statements, subprograms, subroutines, user exits, and utilities.
<i>Construct Spectrum SDK for Microsoft .NET Framework, variable name</i>	Italicized text indicates: <ul style="list-style-type: none"><li>• Book titles</li><li>• Placeholders for information you must supply</li></ul>
[ <i>variable</i> ]	In syntax and code examples, values within square brackets indicate optional items.
{WHILE UNTIL}	In syntax examples, values within brace brackets indicate a choice between two or more items; each item is separated by a vertical bar ( ).

## Naming Conventions for Examples

The examples used throughout this documentation follow an established naming conventions. The following table describes these conventions:

<b>Term</b>	<b>Description</b>
Dispatcher	Indicates the name of the server component that provides broker communications between the client and server.
SpectrumWebServices	Indicates the name of the Web services root directory. For example, from the web: <code>http://[machinename]/SpectrumWebServices</code> From the physical location: <code>C:\inetpub\wwwroot\SpectrumWebServices</code>

---

## Other Resources

This section provides information about other resources you can use to learn more about Construct Spectrum and Natural Construct. For more information about these documents and courses, contact the nearest Software AG office or visit the website at [www.softwareag.com](http://www.softwareag.com) to order documents or view course schedules and locations. You can also use the website to email questions to Customer Support.

## Related Documentation

This section lists other documentation in the Construct Spectrum and Natural Construct documentation set.

### Construct Spectrum SDK

- *Construct Spectrum SDK Reference*  
This documentation is for developers creating Natural modules and ActiveX Business Objects to support applications that will run in the Natural mainframe environment and a Windows environment and/or an internet server.
- *Construct Spectrum SDK for Web Applications*  
This documentation is for developers creating the web components of applications. It describes how to use the Construct Spectrum wizards in Visual Basic to generate HTML templates, page handlers, and object factory entries. It also contains detailed information about customizing, debugging, deploying, and securing web applications.
- *Construct Spectrum SDK for Client/Server Applications*  
This documentation is for developers creating client components for applications that will run in the Natural mainframe environment (server) and a Windows environment (client).
- *Construct Spectrum Messages*  
This manual is for application developers, application administrators, and system administrators who want to investigate messages returned by Construct Spectrum runtime and SDK components.

## Construct Spectrum

- *Construct Spectrum Administration*  
This documentation is for administrators who want to use the Construct Spectrum Administration subsystem to set up and manage Construct Spectrum applications.
- *Construct Spectrum and SDK Vn Release Notes*  
These notes contain information on new features, enhancements, and other changes in this version of Construct Spectrum.
- *Construct Spectrum Reference*  
This documentation is for application developers and administrators who need quick access to information about Construct Spectrum application programming interfaces (APIs) and utilities.
- *Construct Spectrum and SDK Installation Guide for Windows*  
This documentation describes how to install and set up the Construct Spectrum runtime and SDK components on the client.
- *Construct Spectrum and SDK Installation Guide for Mainframes*  
This documentation describes how to install and set up the Construct Spectrum runtime and SDK components on the mainframe.

## Natural Construct

- *Natural Construct Installation Guide for Mainframes*  
This manual provides essential information for setting up the latest version of Natural Construct, which is needed to operate the Construct Spectrum programming environment.
- *Natural Construct Generation*  
This manual describes how to use the Natural Construct models to generate applications that will run in a mainframe environment.
- *Natural Construct Administration and Modeling*  
This manual describes how to use the Administration subsystem of Natural Construct and how to create new models.
- *Natural Construct Help Text*  
This manual describes how to create online help for applications that run on server platforms.
- *Natural Construct Getting Started Guide*  
This documentation introduces new users to Natural Construct and provides step-by-step instructions to create several common processes.

---

## Other Documentation

This section lists documents published by WH&O International:

- *Natural Construct Tips & Techniques*  
This book provides a reference of tips and techniques for developing and supporting Natural Construct applications.
- *Natural Construct Application Development User's Guide*  
This guide describes the basics of generating Natural Construct modules using the supplied models.
- *Natural Construct Study Guide*  
This guide is intended for programmers who have never used Natural Construct.

## Related Courses

In addition to documentation, the following courses are available from Software AG:

- A self-study course on Natural Construct fundamentals
- An instructor-led course on building applications with Natural Construct
- An instructor-led course on modifying the existing Natural Construct models or creating your own models



---

## CONFIGURING THE SDK

This chapter describes how to configure the runtime component for the Construct Spectrum SDK for Microsoft .NET framework.

The following topics are covered:

- **Introduction**, page 14
- **Configuring the Runtime Component**, page 16

## Introduction

The Construct Spectrum SDK for Microsoft .NET framework runs in the following environments:

Environment	Description
Development	<p>During development, the Natural code (subprograms) may be located in a different Natural environment than the Spectrum runtime code. The location of the Natural code used for development purposes is determined by the settings in the Spectrum Configuration editor.</p> <p>The dispatcher used for development purposes is the dispatcher set as “Active” in the Configuration editor.</p>
Runtime	<p>Construct Spectrum requires a Natural runtime environment on the backend. This environment handles functions such as EntireX connections, security, and Spectrum services (attach managers and dispatchers).</p> <p>The runtime environment is setup on the mainframe by specifying the FNAT and LFILES for the Natural Construct file and two Construct Spectrum files. On the client, the runtime environment for Construct Spectrum is configured using the Spectrum Service Manager. For more information, see <b>Configuring the Runtime Component</b>, page 15, <i>Construct Spectrum and SDK Installation Guide for Windows</i>.</p> <p>The dispatcher used for the runtime environment for Web services is set in the Web.config file in the SpectrumWebServices root directory.</p>
<b>Note:</b>	<p>The steplib chain for code accessed by a runtime environment is specified on the Maintain Application Service Definitions panel in the Spectrum Administration subsystem. For information on accessing this panel, see <b>Step 2: Verify Spectrum Data (Optional)</b>, page 21.</p>

---

After installing the SDK, three key areas require configuration settings:

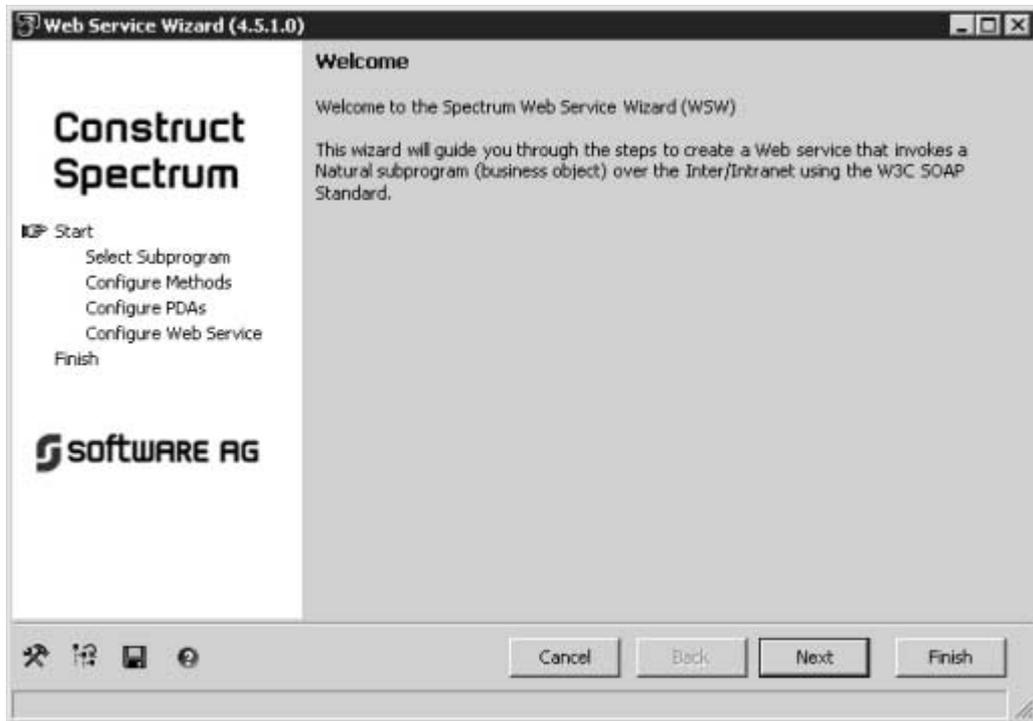
- IIS (Internet Information Server) on the Web services box.  
This configuration is done automatically during installation of the runtime component.
- Spectrum Service Manager on the client.  
The Spectrum Service Manager provides the dispatcher connectivity to the server. If Construct Spectrum is currently installed, this configuration is done. For more information, see **Configuring the Runtime Component**, page 15, *Construct Spectrum and SDK Installation Guide for Windows*.
- Configuration editor and Web.config file on the client.  
This configuration sets up the runtime component for your generated Web services. For more information, see the following section.

## Configuring the Runtime Component

- To configure the Spectrum Web services runtime component:
  - ❑ **Step 1: Specify the Configuration Editor Settings**, page 16
  - ❑ **Step 2: Set the Runtime Dispatcher**, page 18

### Step 1: Specify the Configuration Editor Settings

- To specify the Configuration editor settings:
  - 1 Select the Spectrum Web Service wizard (WSW) from the Start menu.  
The Start window is displayed:



Start Window for the Web Service Wizard

- 2 Click the Configuration editor icon in the lower, left corner of the window.  
The configuration profiles are displayed. For more information, see **Using the Configuration Editor**, page 50, *Construct Spectrum SDK Reference*.

---

**Note:** You can also select Spectrum Configuration from the Start menu.

---

- 3 Select the Spectrum 451 profile.  
Ensure that Spectrum 451 is the active profile or click Set as Active.
- 4 Select the Settings for Profile 'Spectrum 451' tab.  
The settings for the default Spectrum 451 configuration profile are displayed.
- 5 Verify that the configuration settings are correct for your environment.  
If the settings are not correct, you can modify them as desired. The Global, Natural Environment, and Spectrum settings are described in **Modify the Profile Settings**, page 52, *Construct Spectrum SDK Reference*. The Web service settings are:

Setting	Description
HTTP Web Services Root	Displays the default name of the Web service root directory on the web.
Local Web Services Root	Displays the default name of the Web service root directory on the physical machine.
BDT Configuration File	Displays the default name for the BDT configuration file.
HTTP Application Root	Displays the default name of the web application root directory on the web.
Local Application Root	Displays the default name of the web application root directory on the physical machine.
Code Frame	Displays the default code frame settings.
Project Templates	Displays the default location of the project templates.
Default Web Service Server	Displays the default code frame settings.

- 6 Click OK to close the Configuration editor.

## Step 2: Set the Runtime Dispatcher

The dispatcher for your runtime environment determines the location from which data is retrieved.

➤ To set the dispatcher for your runtime environment:

- 1 Open the global Web.config file.  
This file is located in the Web services root directory (SpectrumWebServices).

---

**Warning:**

Some settings in the Web.config file are required by the Web Service wizard and should not be modified. For example, configSections contains the IIS (Internet Information Server) settings that associate the .sws files to the Web Services Engine (WSE).

---

- 2 Scroll to the <Spectrum> node and verify the Dispatcher setting.
- 3 Save the Web.config file.  
For more information on modifying the settings in the Web.config file, see **Modifying the Web.config File**, page 68.
- 4 Click Cancel to close the file.  
The Construct Spectrum SDK for Microsoft .NET framework is now configured and ready to use.

---

**Tip:** The development dispatcher, which determines the location of the Natural PDAs, is based on the default dispatcher specified in the Configuration editor. For information, see **Using the Configuration Editor**, page 50, *Construct Spectrum SDK Reference*.

---

---

## USING THE SDK

This chapter describes how to generate and regenerate a Web service using the Construct Spectrum SDK for Microsoft .NET framework. It includes sections on how to prepare a Natural subprogram for client/server access and how to use BDTs to format data, as well as how input is validated and how to test your Web service. This chapter also describes how to build a web application for your Web service.

The following topics are covered:

- **Preparing a Natural Subprogram for Client/Server Access**, page 20
- **Generating a New Web Service**, page 23
- **Regenerating an Existing Web Service**, page 35
- **Using Business Data Types (BDTs)**, page 37
- **Validating Input**, page 42
- **Testing Your Web Service**, page 43
- **Building a Web Application**, page 47

## Preparing a Natural Subprogram for Client/Server Access

Before generating a Web service, you must prepare the Natural subprogram for client/server access. To do this, generate a subprogram proxy on the mainframe for a Natural subprogram. Generation of the proxy automatically populates the Spectrum data with the appropriate methods, domain, object name, and version. It also “flattens” the subprogram parameters for transfer across the wire.

➤ To prepare a Natural subprogram for client/server access:

- ❑ **Step 1: Generate the Subprogram Proxy**, page 20
- ❑ **Step 2: Verify Spectrum Data (Optional)**, page 21

The following sections describe these steps in detail.

### Step 1: Generate the Subprogram Proxy

To access a Natural subprogram, you must first create a subprogram proxy to transmit messages between the mainframe and the client. The following example generates a subprogram proxy for the ORD-MSO maintenance subprogram in SPECDEMO.

---

**Note:** For more information about generating a subprogram proxy, see **Using the Subprogram-Proxy Model**, page 103, *Construct Spectrum SDK Reference*.

---

➤ To generate the subprogram proxy:

- 1 Log onto the SPECDEMO library.  
If this library is not available, ask your Natural administrator to enable access.
- 2 Enter “ncstg” at the Next prompt.  
The Generation main menu is displayed.
- 3 Type the following values in the fields indicated:
  - “M” in Function
  - “MYPROXY” in Module
  - “SUBPROGRAM-PROXY” in Model
- 4 Press Enter.  
The Standard Parameters are displayed. The values entered here determine how the message is placed on the wire. The module name and system are defaulted from the Generation main menu.
- 5 Modify the title and description to indicate the creation of a proxy for the Order object.
- 6 Select “ORD-MSO” from Subprogram.

- 7 Select “DEMO” from Domain.  
Domains help to categorize and add security in the Spectrum environment. They must be setup in the Spectrum environment before a subprogram proxy can be created.
- 8 Type “MyOrder” in Object name.
- 9 Press Enter.  
You are returned to the Generation main menu.
- 10 Enter “G” in Function.  
Natural Construct generates the subprogram proxy and populates the Spectrum file with your object information.
- 11 Enter “W” in Function to stow the subprogram proxy.  
Your subprogram proxy has been generated and saved. If everything is configured and you have connectivity, all the work on the mainframe is now completed.

## Step 2: Verify Spectrum Data (Optional)

Next, you can optionally verify that your data was entered into the Construct Spectrum Administration subsystem on the mainframe. The following example verifies the Spectrum data for the MYPROXY subprogram generated in the previous step.

➤ To verify that your data was entered into Construct Spectrum:

- 1 Enter “.” in Function to exit from Natural Construct.
- 2 Log onto the SYSSPEC library.  
If this library is not available, ask your Natural administrator to enable access.
- 3 Enter “menu” at the Next prompt.  
The Construct Spectrum Administration Subsystem main menu is displayed:

```

BS__MAIN          Construct Spectrum Administration Subsystem          CDLAYMN1
Mar 11              Main Menu                                         08:55 AM

                Functions
                -----
                SA   System Administration
                AA   Application Administration

                ?   Help
                .   Terminate
                -----
Function ..... _

Command .....
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11---PF12---
      help retn quit          flip          main

```

Construct Spectrum Administration Subsystem Main Menu

- 4 Enter "AA" in Function.  
The Application Administration main menu is displayed.
- 5 Enter "MM" in Function.  
The Application Administration Maintenance menu is displayed.
- 6 Enter "AS" in Function.  
The Maintain Application Service Definitions panel is displayed:

```

BSIF__MP          Construct Spectrum Administration Subsystem          BSIF__11
Jan 30           Maintain Application Service Definitions              3:15 PM

Action (A,B,C,D,M,N,P)  _

Domain.....: _____ *
Object.....: _____
Version.....: 00 / 00 / 00
Description.....: _____
Default subprogram proxy: _____
Steplibs.....: _____ *

01           Method Name          Subprogram          Steplibs *
-----
1  _____          _____          _____
2  _____          _____          _____
3  _____          _____          _____
4  _____          _____          _____
5  _____          _____          _____

Command: _____
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
confm help  retrn quit          flip pref  bkwrdr frwr          main

```

### Maintain Application Service Definitions Panel

- 7 Type "B" in Action.
- 8 Type DEMO in Domain.
- 9 Press Enter.
- 10 Select MYPROXY.  
Details of your subprogram proxy should be displayed. Because ORD-MSO is a Natural Construct-generated object subprogram, the appropriate methods are included.
- 11 Enter "." in Action.  
The Application Administration Maintenance menu is displayed.
- 12 Enter "DO" in Function.  
The Maintain Domain Table panel is displayed.
- 13 Enter "B" in Action to display a browse window listing all available domains.
- 14 Select "DEMO".  
Verify that DEMO has been defined to Steplibs (name of a step library grouping).
- 15 Enter "." in Action.  
The Application Administration Maintenance menu is redisplayed.
- 16 Enter "ST" in Function.  
The Maintain Steplib Table panel is displayed.

- 17 Enter “B” in Action.
- 18 Select the DEMO step library grouping.  
Verify that SPECDEMO is listed as a steplib.  
You are now ready to generate a Web service for your Natural subprogram.

## Generating a New Web Service

After preparing the Natural subprogram on the mainframe, you are ready to generate a Web service on the client. The following example generates a Web service for the MYPROXY subprogram proxy created in **Preparing a Natural Subprogram for Client/Server Access**, page 20.

- To generate a new Web service using the Web Service wizard:
  - ❑ **Step 1: Invoke the Web Service Wizard (WSW)**, page 24
  - ❑ **Step 2: Select the Subprogram Proxy**, page 25
  - ❑ **Step 3: Configure the Methods (Optional)**, page 27
  - ❑ **Step 4: Configure the Parameter Data Areas (Optional)**, page 29
  - ❑ **Step 5: Configure the Web Service (Optional)**, page 31
  - ❑ **Step 6: Generate the Service**, page 32

---

**Note:** If you do not override any settings for the steps marked as optional, the default settings are used.

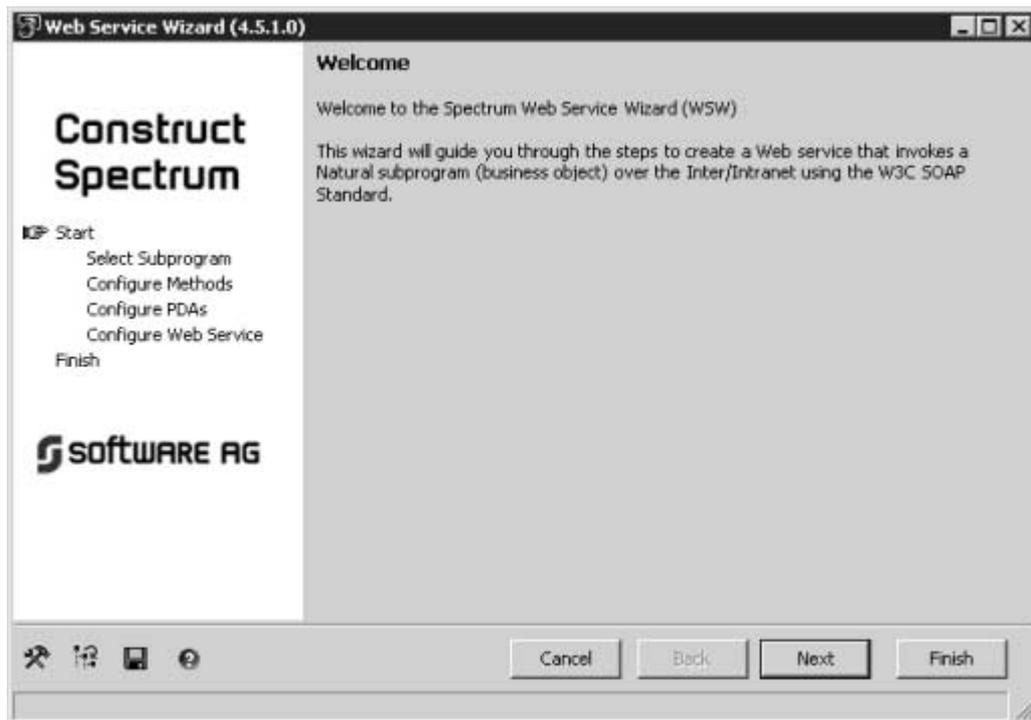
---

These steps are described in detail in the following sections.

For information on validation edits, see **Validating Input**, page 42.

## Step 1: Invoke the Web Service Wizard (WSW)

- To invoke the WSW:
  - 1 Select the Spectrum Web Service wizard from the Start menu.  
The Start window is displayed:



Start Window

- 2 Click Next to proceed.  
The Select Subprogram window is displayed.

---

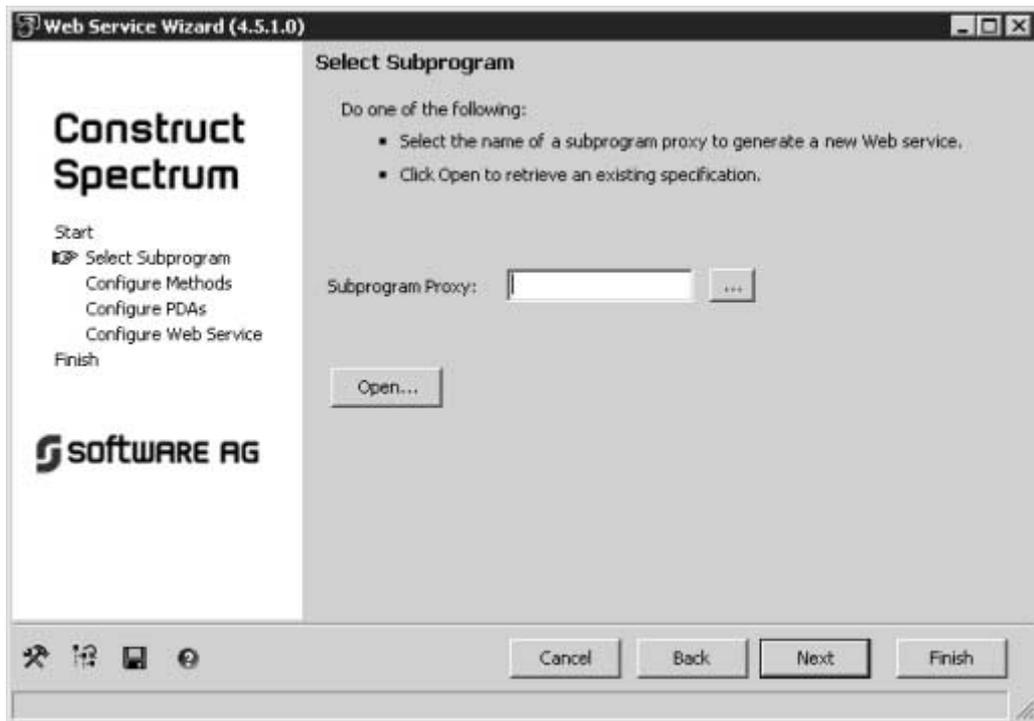
**Note:** You can save the specifications for your Web service at any time by clicking the Save icon in the lower, left corner of the window. The Web Service wizard creates a .swsd file.

---

## Step 2: Select the Subprogram Proxy

The subprogram proxy is a Natural subprogram generated using the Natural Construct Subprogram-Proxy model. The subprogram proxy prepares data for transmission across a wire. Each Natural subprogram accessed by a web server must have a corresponding subprogram proxy. For information, see **Preparing a Natural Subprogram for Client/Server Access**, page 20.

The following example shows the Select Subprogram window:



Select Subprogram Window

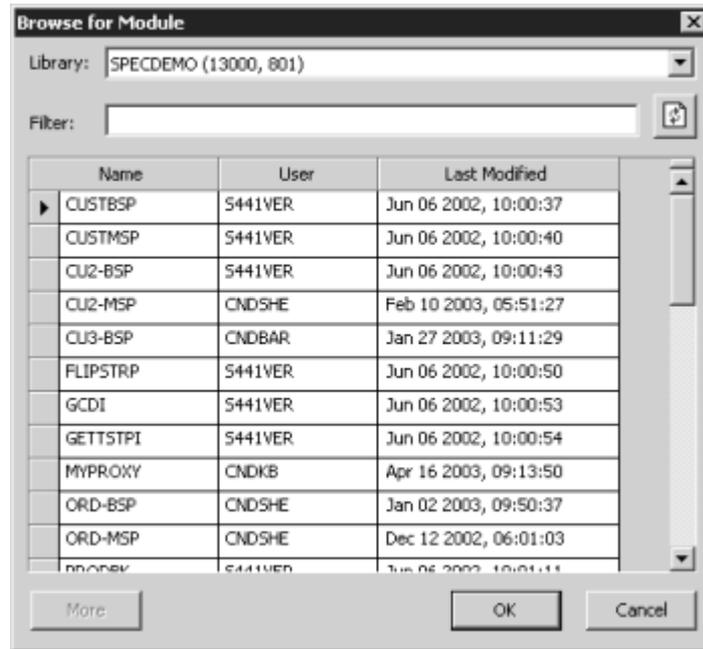
Use this window to indicate the name of the subprogram proxy for which you are generating the Web service.

---

**Note:** To retrieve the specifications for a Web service that was previously saved, click Open.

---

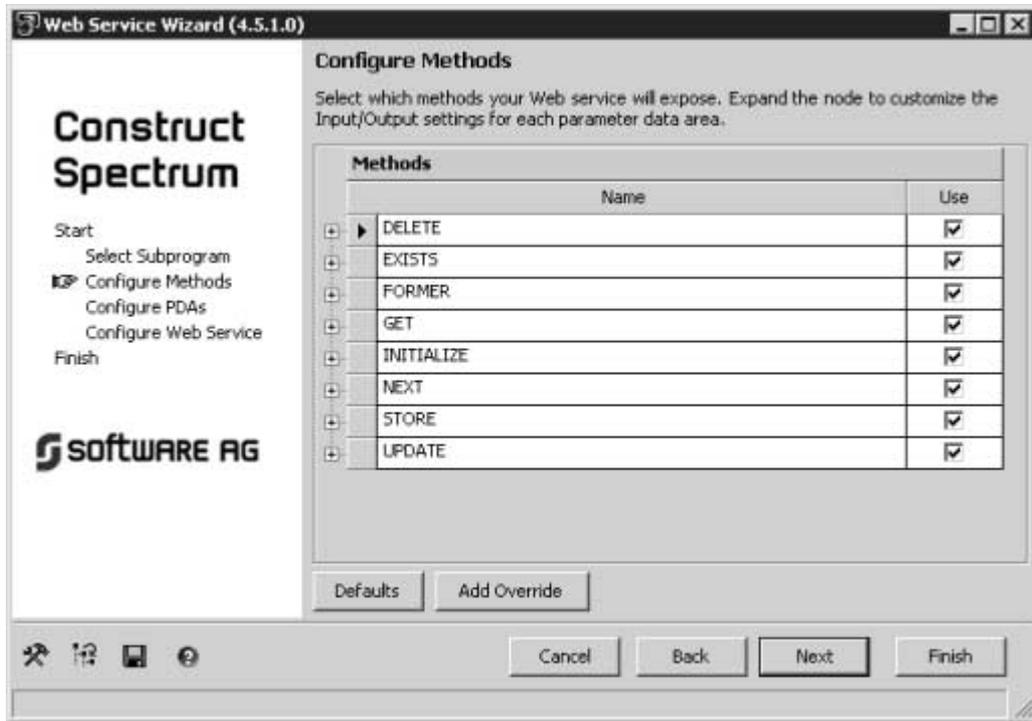
- To select a subprogram proxy:
- 1 Click the browse button (...) for Subprogram Proxy.  
The Browse for Module window is displayed:



Browse for Module Window

- 2 Select "MYPROXY".
- 3 Click OK.  
The Select Subprogram window is displayed, showing MYPROXY in Subprogram Proxy.
- 4 Click Next to proceed.  
The Configure Methods window is displayed.

### Step 3: Configure the Methods (Optional)



Configure Methods Window

When you generate a Web service for a Natural subprogram, you must decide which methods (actions) will be permitted on the client. For example, most subprograms support the DEFAULT method, browse subprograms also support the BROWSE method, and maintenance subprograms support the methods shown above.

Typically, the WSW-generated defaults are sufficient. However, you can use this window to customize the methods and Input/Output settings your Web service will expose to the user, as well as any field overrides for mainframe and server data.

- To configure the methods:
  - 1 Expand the node for each method.
  - 2 Define any field overrides for the method as follows:
    - To add a new field override, click Add Override, select a field name from the drop-down list, and specify the override value and XML format or tag name.
    - To remove an existing field override, select the field name and press Delete.

A field override initializes custom data in the PDA before sending the data to the mainframe. The data does not have to be public to the client. An example of this functionality is the #FUNCTION field in the CDAOBJ2 PDA for a maintenance subprogram proxy. This field should be hidden from the client, but populated when a request is made to the mainframe. Another example of this functionality is overriding an input value. For example, CITY='TORONTO' forces the CITY value to be Toronto, even if another value was passed to the website.

- 3 Verify the INPUT/OUTPUT settings for each PDA used by the method. To re-display all default values, click Default.

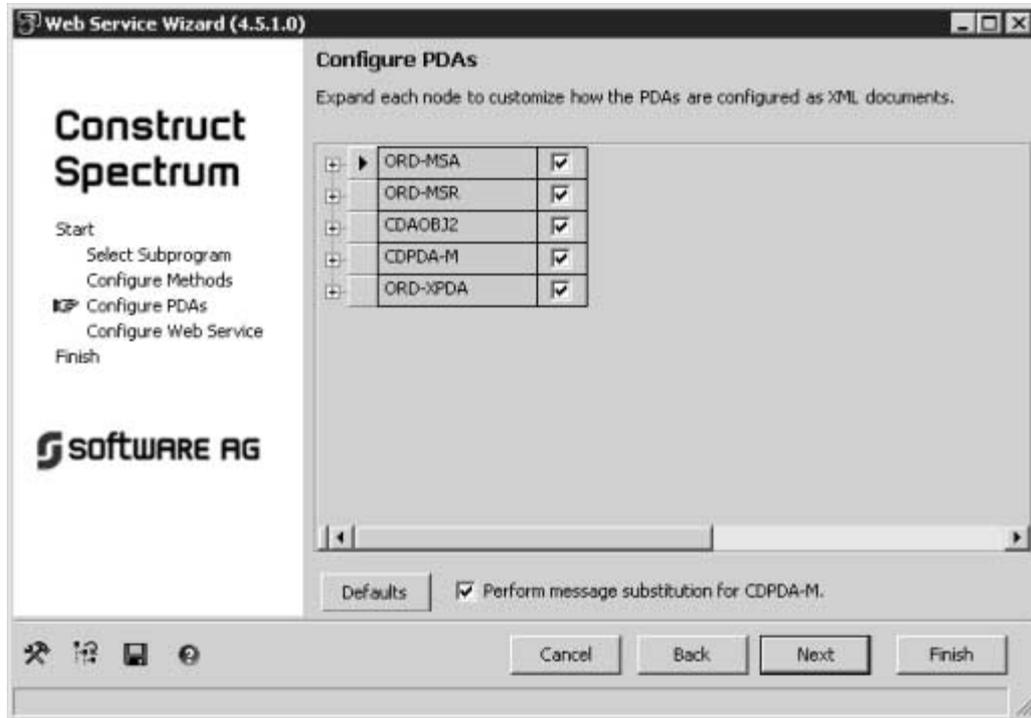
---

**Note:** If you intend to use the Web Page wizard for SWS (Spectrum Web Services), we recommend that you select the appropriate BDTs and levels of redefine within the data PDAs (maintenance PDA for the maintenance object and row PDA for the browse object). As state must be maintained, do not eliminate any fields from a Natural Construct-generated maintenance object (although the fields can be hidden by the Web Page wizard for SWS). Accept the defaults for any other Natural Construct-supplied PDA.

---

- 4 Click Next to proceed.  
The Configure PDAs window is displayed.

## Step 4: Configure the Parameter Data Areas (Optional)



Configure PDAs Window

Use this window to indicate which mainframe fields the web server will make public to the client. You can also use this window to do the following:

- **Override global settings**  
To override the global Web.config file settings, use BDT modifiers. For more information, see **BDT Modifiers**, page 37.
- **Create multilingual Web services**  
If you are using message numbers in CDPDA-M for your Web service (because the subprogram was generated with the message number option), select Perform message substitution for CDPDA-M. At runtime, Construct Spectrum scans for the :1::2::3: place holders and retrieves the appropriate message. This option is selected by default if CDPDA-M is available for this Web service.

---

**Tip:** If you are not using message numbers, de-select this option to eliminate unnecessary CPU usage.

---

---

**Note:** To re-display all default values, click Defaults.

---

- To configure the PDAs:
- 1 Expand each node.  
The read-only settings for the specified PDA show the level, name, and Natural format for each field in a PDA.
  - 2 Scroll to the right to display the modifiable settings.  
The BDT setting refers to the business data type for the corresponding field. BDTs provide a way to present data to the user in a format that is consistent and based on business conventions, rather than on programming language conventions. For example, a BDT can format a phone number with dashes (-) so it is easily recognized as a phone number. For more information, see **Using Business Data Types (BDTs)**, page 37.
  - 3 Select or deselect fields in each PDA used by your Web service.
- 

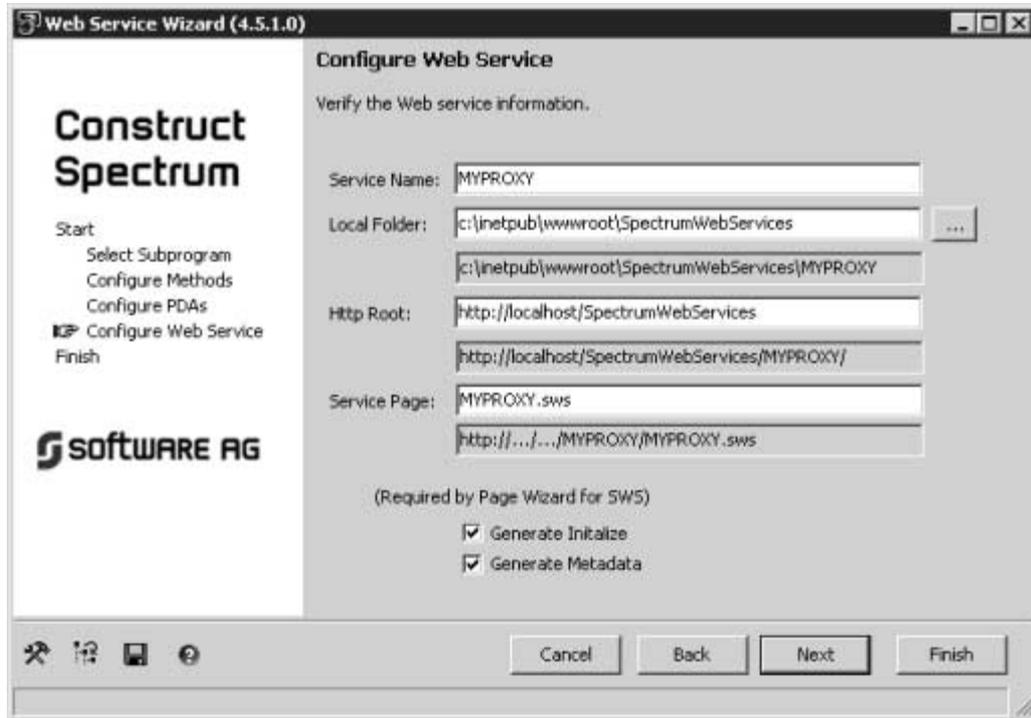
**Warning:**

If a field is not selected and an update is performed, the data in the non-selected field is blanked out at runtime. De-selection of a field implies that the field value is blank when it reaches the mainframe, unless a value is assigned by a field override.

---

- 4 Define the XML tag name or format for each selected field, if required.
- 5 Click Next to proceed.  
The Configure Web Service window is displayed.

## Step 5: Configure the Web Service (Optional)



Configure Web Service Window

Use this window to verify the service information, such as the name of the Web service, the location of the local folder, the URL to invoke the service, and the name of the Web service page.

You can also use this window to generate local initialization data (for example, create WSE\_Initialize) or an xml file containing metadata for the Web Page wizard for SWS (metadata.xml), such as the name of the model used to generate the subprogram proxy, the name of the primary key and the hold field, the types of PDAs used by the service, a description of the object, and the Predict data.

---

**Note:** If you are using the Web Page wizard for SWS to create the web page, do not deselect Generate Initialize or Generate Metadata.

---

➤ To configure the Web service:

1 Verify the following settings:

- Name of the Web service
- Location of the local folder containing the Web service; use the browse button (...) to select another location
- URL used to invoke the Web service, excluding the name of the Web service page
- Name of the Web service page

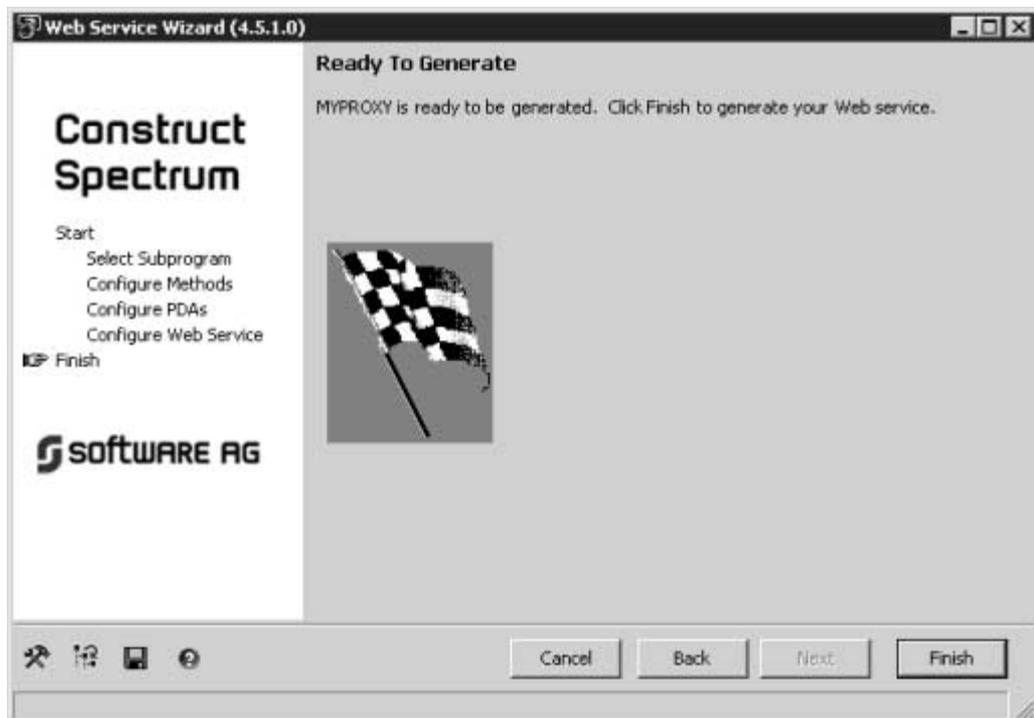
The location of the Web service page is determined by combining the URL and the Service Page file name. For example:

```
http://localhost/SpectrumWebServices/MYPROXY/MYPROXY.sws
```

2 Click Next to proceed.

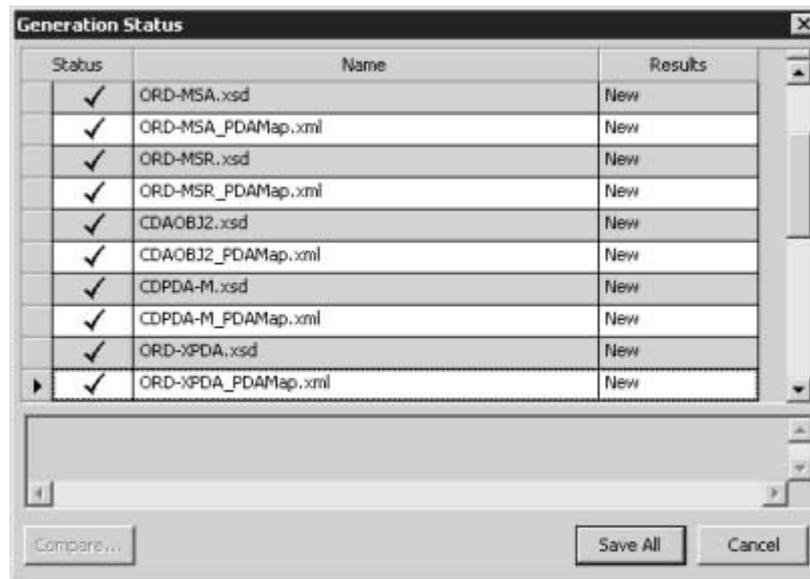
The Ready to Generate window is displayed.

## Step 6: Generate the Service



Ready to Generate Window

- To generate the Web service:
- 1 Click Finish.  
The Generation Status window is displayed:



Generation Status Window After Generating a Web Service

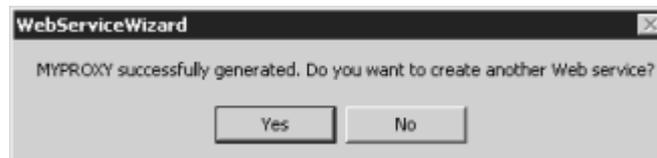
This window shows the files that were generated for your Web service. To enable the Web service, save these files to disk.

---

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

---

- 2 Click Save All to save the files.  
The Create New Service window is displayed:



Create New Service Window

- 3 Do one of the following:
  - To create another service, click Yes.
  - To close the wizard, click No.

You have successfully generated and saved a new Web service for a Natural subprogram.

- For information on testing your service, see **Testing Your Web Service**, page 43.
- For information on creating a web application for your web service, including web pages and menus, see **Building a Web Application**, page 47.

## Regenerating an Existing Web Service

In addition to creating new Web services, you can use the Web Service wizard to retrieve and regenerate the specifications for an existing Web service.

---

**Warning:**

Regeneration may affect the code generated by the Web Page wizard for SWS.

---

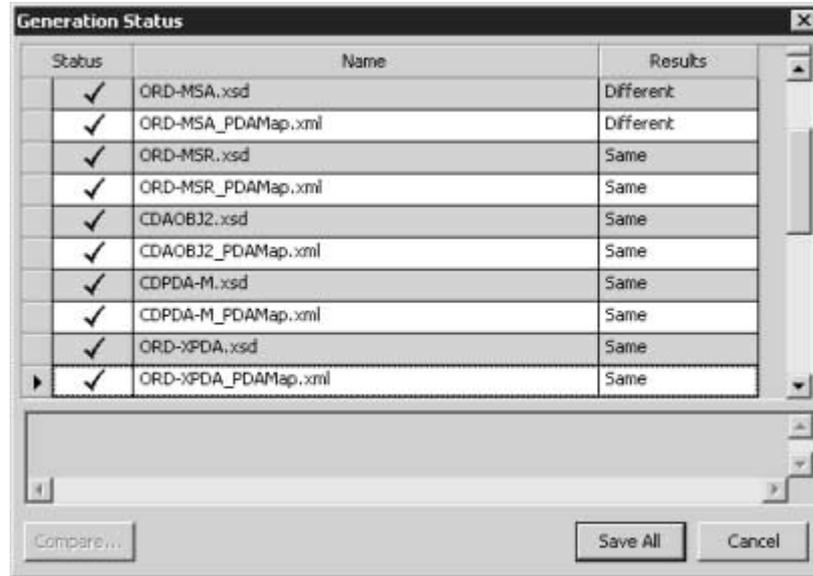
- To regenerate an existing Web service:
- 1 Select the Spectrum Web Service wizard from the Start menu.  
The Start window is displayed.
  - 2 Click Next.  
The Select Subprogram window is displayed.
  - 3 Click Open.
  - 4 Locate the .sws file for the existing Web service in the private directory in the local folder in the Web services root directory (SpectrumWebServices).
  - 5 Click OK.  
The selected subprogram is displayed in Subprogram Proxy. You can change the name of the subprogram proxy used by the Web service, as well as other settings defined in subsequent wizard windows. For information on options available in these windows, see **Generating a New Web Service**, page 23.

---

**Tip:** If the parameters on the mainframe change, the changes may not be incorporated if the cache is not cleared. To clear the cache, click the Spectrum Cache icon in the lower, left corner of the window, select the cache to be cleared, and click Delete. For more information, see **Using The Spectrum Cache**, page 68, *Construct Spectrum SDK Reference*.

---

- 6 Click Finish in the Ready to Generate window to regenerate the Web service. The Generation Status window is displayed:



Generation Status Window after Regenerating a Web Service

The Results column indicates whether a file is the same as the previous generation or different from the previous generation. If a file is different, select the file and click Compare to invoke your code comparison tool. For more information, see **Use Reports with a Code Comparison Tool**, page 67, *Construct Spectrum SDK Reference*.

## Making Changes Accessible to Users

If you make changes to a generated Web service and do not change the Web.config file, the updates will not be accessible to users invoking your service. To make the changes accessible, do one of the following:

- Recycle IIS (Internet Information Server)
- Unload the Root application
- Re-save the Web.config file

## Using Business Data Types (BDTs)

Business data types (BDTs) help ensure that information is displayed in a way that is consistent and easy to understand. BDTs convert Natural data types into values displayed to the user in a browse or maintenance window. For example, a BDT can reformat a telephone number that was entered without dashes.

Using BDTs offers three primary benefits:

- **Consistency**  
BDTs ensure that each data type is displayed to the user in a consistent format.
- **Flexibility**  
BDTs recognize a variety of input formats, which makes using the Web service easier.
- **Accuracy**  
BDTs centralize the validation code for a data type and provide a consistent mechanism for returning validation error messages.

Construct Spectrum SDK for Microsoft .NET framework supplies a number of pre-defined BDTs you can use — or you can create your own. If there is a piece of information whose format you are constantly validating, consider creating a BDT to handle it. Once a BDT has been created, you can use it in other Web services. For more information, see **Creating a Custom BDT**, page 79.

### BDT Modifiers

Some of the BDTs include modifiers, additional parameters you can use to further refine the display of data. For example, you can use BDTAlpha with a modifier of CASE=U to convert the contents of a field into uppercase.

You can also specify more than one BDT for a field. For example, you can use BDT-Numeric with modifiers of ROUND=1 | GS=ON to round the field value to one decimal and display the group separator (separator for numeric values in the thousands).

---

**Note:** Note that the | character is used to separate multiple BDT modifiers.

---

## Supplied BDTs

Construct Spectrum SDK for Microsoft .NET framework supplies several BDTs you can use to format input data. The following table lists the supplied BDTs, as well as the Web service and Predict keywords:

<b>BDT Name</b>	<b>Type</b>	<b>Applied to</b>	<b>Predict Keyword</b>
BDTAlpha	Alpha	Alphanumeric data	BDT_Alpha
BDTBoolean	Boolean	Data that can have a value of either true or false	BDT_Boolean
BDTCurrency	Currency	Currency data	BDT_Currency
BDTDateTime	Date Time	Date and time data	BDT_Date BDT_Time
BDTHexByte	Hex byte	Hex data in a string format	BDT_HexByte
BDTNumeric	Numeric	Numeric data	BDT_Numeric
BDTPhone	Phone	Data that represents a phone number	BDT_Phone
BDTPostalCode	Postal code	Data that represents a postal code	BDT_PostalCode

The following sections describe the supplied BDTs and the modifiers each supports.

**Note:** BDTPhone and BDTPostalCode do not support modifiers.

### BDTAlpha

BDTAlpha is applied to alphanumeric data.

<b>Modifier</b>	<b>Description</b>
TRIM=L T LT	Trims leading spaces (L), trailing spaces (T), or leading and trailing spaces (LT). Default is no trimming. This affects ConvertToDisplay and ConvertFromDisplay behavior.
CASE=U L	Forces the text into uppercase (U) or lowercase (L). Default is to not change the case. This affects ConvertToDisplay and ConvertFromDisplay behavior.

## BDT Boolean

BDTBoolean is applied to data that can have a value of either False or True.

Modifier	Description
O/I=<False> <True>	Displays Output (False) or Input (True) settings. The default is Output.
EM=<False> <True>	Displays the <False> string for False and the <True> string for True. Default is EM=False True. ConvertFromDisplay compares the formatted data to the <False> and <True> strings and recognizes a match if the value matches unambiguously to the beginning of either string. This is not case-sensitive.

**Note:** The left side of the edit mask always represents the negative (false) and the right side always represents the positive (true).

The following examples show various types of edit mask values, user input, and each result.

EM Value	Formatted Value	Raw Value
EM=False True	T	True
	t	True
	tr	True
	TRU	True
	F	False
	false	False
	yes <blank>	Error: Invalid Error: Invalid
EM=True False	true	False
	F	True
EM=Off On	off	False
	on	True
	o	Error: Ambiguous
EM= X	x	True
	<blank>	False
	xx	Error: Invalid

**Note:** If you are using the Web Page wizard for SWS (Spectrum Web Services) and have set non-standard BDTBoolean values, you must modify the evaluate-Boolean function in the SPEShared.js file.

## BDTCurrency

BDTCurrency is applied to any currency values.

Modifier	Description
ZERO=OFF ON	Suppresses (OFF) or displays (ON) zero values. The default is ZERO=ON.

## BDTDateTime

BDTDateTime is applied to any date or time value.

Modifier	Description
ISDate=<True> <False>	Converts Date (True) or Time (False).
XML=<True> <False>	Converts to or from an XML compliant string.
DisplayDateFormat DisplayTimeFormat	ConvertToDisplay formats the Date (DisplayDateFormat) or Time (DisplayTimeFormat). For the syntax for the format string, see MSDN.

## BDTHexByte

BDTHexByte is applied to hex data in a string format.

Modifier	Description
HexPrefix HexSuffix	Indicates the string used at the beginning (HexPrefix) or end (HexSuffix) of hex data.
<b>Note:</b>	If HexSuffix is not specified, spaces are used by default.

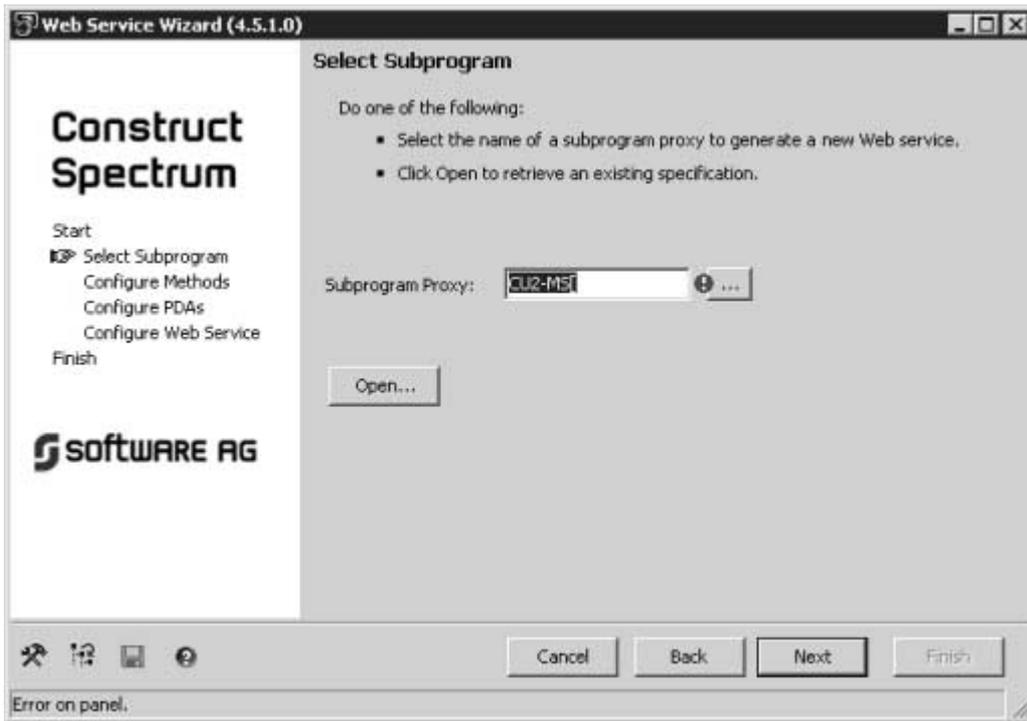
## BDTNumeric

BDTNumeric is applied to any numeric data.

<b>Modifier</b>	<b>Description</b>
DEC= <i>n</i>	Forces the display of <i>n</i> decimal places. Default is to display as many decimal places as there are significant decimal digits when the Natural format is not provided, or to use a fixed number of decimal places if the Natural format is provided. In this latter case, use DEC=-1 to ignore the Natural format and display significant decimal digits only.
ROUND= <i>n</i>	Rounds the value to <i>n</i> decimal places. If <i>n</i> is negative, it rounds to the left of the decimal place. Default is no rounding.
GS=OFF ON	Used to suppress (OFF) or display (ON) group separators (thousands separators). Default is GS=OFF.
ZERO=OFF ON	Suppresses (OFF) or displays (ON) zero values (for output only). Default is ZERO=OFF.
SIGN=OFF ON	Suppresses (OFF) or displays (ON) the sign for positive numbers. Default is SIGN=OFF.
SCIENTIFIC=OFF ON	Displays the value in normal (OFF) or scientific notation (ON). Default is SCIENTIFIC=OFF.
EM= <i>xxx</i>	Any format string understood by the Visual Basic Format function. ConvertToDisplay uses the Format function to format the value according to that format string.

## Validating Input

If an error occurs in any wizard window, Construct Spectrum displays a circle containing an exclamation mark beside the field in error. The following example shows an error in the Subprogram Proxy field:



Spectrum SDK for Microsoft .NET Window Showing Validation Error

The Status bar message describes the error, or you can move the cursor over the icon to display a description of the error. In this example, the specified subprogram proxy (CU2-MSI) is not found anywhere in the steplib chain. To continue creating the Web service, select a valid subprogram proxy.

## Testing Your Web Service

Use the SOAP Client to test your generated Web service. SOAP (Simple Object Access Protocol) is part of the Web service definition and contains the XML message sent via HTTP. A Web service sends PDA data in the form of an XML document inside a SOAP document. The SOAP document contains the format of the message used on the wire to receive and respond to Web service requests, including information that indicates which method is being invoked. The response is returned via a SOAP document. Standard SOAP technologies, such as the SOAP:Fault section, are used to transmit error and status information.

---

**Tip:** You can use the SOAP Client to test any Web service, not just a Spectrum-generated Web service.

---

In the SOAP Client, the SOAP actions correspond to the methods defined for the sub-program. The URL is the URL set up in the configuration file with the addition of “.sws”. For example:

```
http://<machine name>/SpectrumWebServices/CU2-MSP/CU2-MSP.sws
```

---

**Note:** To view the incoming and outgoing data while your Web service is running, use the Trace utility in the Microsoft SOAP toolkit.

---

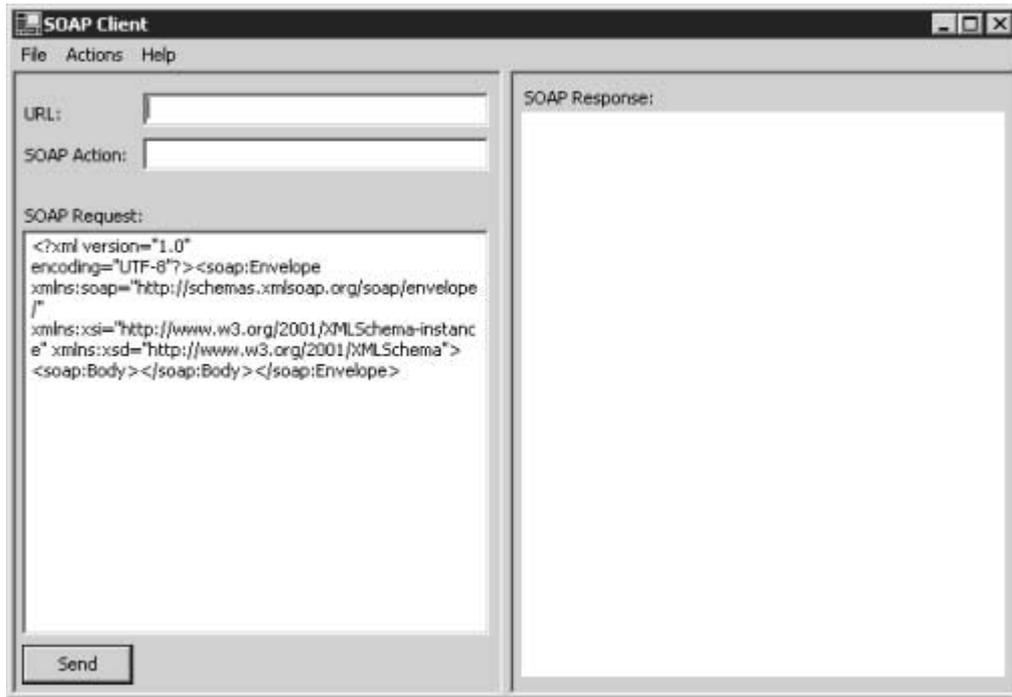
---

**Note:** The SOAP Client is case-sensitive.

---

For examples of input to the SOAP Client, see **Sample SOAP Messages**, page 88.

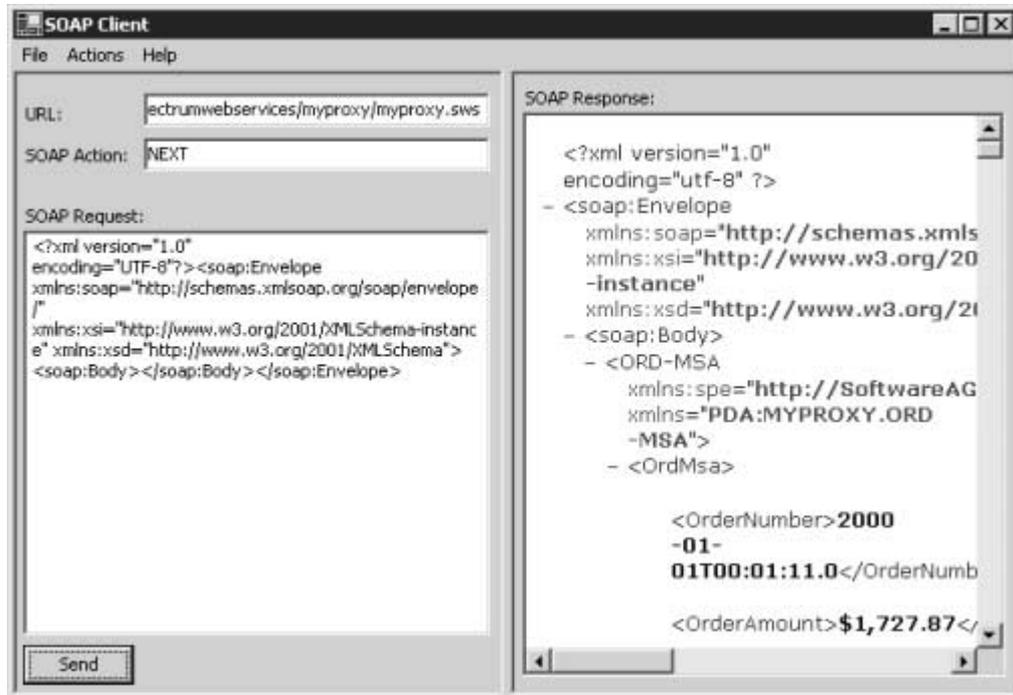
- To test your generated Web service:
- 1 Select SOAP Client from the Start menu.  
The SOAP Client window is displayed:



SOAP Client Window

- 2 Type the URL noted in **Step 5: Configure the Web Service (Optional)**, page 31.  
For example:  
`http://localhost/SpectrumWebServices/MYPROXY/MYPROXY.sws`
- 3 Add `/ordmsp.sws` to the end of this URL.  
For example:  
`http://localhost/SpectrumWebServices/MYPROXY/ordmsp.sws`
- 4 Type NEXT in SOAP Action for a Natural Construct-generated maintenance subprogram.  
The method name entered in this field corresponds to the SOAP action sent for the request.

- Click Send and wait for data to return.  
The resulting XML document is displayed to the right. For example:



SOAP Client Showing XML Document

In this example, the key is blank; the next value will be the first value in the file. Assuming that this value is 111, the SOAP request must add additional code to get the next value.

- Collapse OrdMsa by clicking the - (minus sign) and copy the following:

```
<OrdMsaId>111</OrdMsaId>
</ORD-MSA>
```

- Paste this text into the <soap:Body> tag.
- Add the beginning of ORD-MSA to the top of this. For example:

```
<ORD-MSA xmlns:spe="http://SoftwareAG.com/Spectrum/SPE451"
xmlns="PDA:MYPROXY.ORD-MSA">
```

This creates the following SOAP request:

```
<?xml version="1.0" encoding="UTF-8"?><soap:Envelope xmlns:soap="http://
/schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/
2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body><ORD-MSA xmlns:spe="http://SoftwareAG.com/Spectrum/SPE451"
xmlns="PDA:MYPROXY.ORD-MSA"><OrdMsaId>111</OrdMsaId> </ORD-MSA>
</soap:Body></soap:Envelope>
```

## 9 Click Send.

The next value is displayed (111, in this example).

You can also test other maintenance methods, such as DELETE, EXISTS, GET, INITIALIZE, STORE, and UPDATE.

---

**Tip:** Save time by using the File menu to save and load previously issued requests.

---

---

**Tip:** Use the NEXT method to retrieve sample values.

---

## Building a Web Application

After generating and testing your Web service, use the supplied Web Application Builder to build a web application for the service. This application consists of a menu and a collection of related web pages.

Applications built using the Web Application Builder have a unique architecture that runs completely in the browser and submits Web service requests to previously generated services. The builder creates an HTML page containing JavaScript that uses Microsoft XML ActiveX components to directly parse and submit XML SOAP documents to the IIS (Internet Information Server) hosting your generated Web services. The JavaScript then receives the XML SOAP response and applies a generated XSL template to transform the data into an HTML user interface. This architecture has numerous advantages, including a better separation of business and presentation logic and increased network performance. Network performance is increased because the browser only sends and receives the data (in XML format) — not the entire HTML presentation.

---

**Note:** Applications built using the Web Application Builder only run in Internet Explorer 5.0 or higher and require the ActiveX scripting security settings in Internet Explorer to be enabled.

---

- To build a web application using the Web Application Builder:
  - ❑ **Step 1: Create the Web Application**, page 48
  - ❑ **Step 2: Generate a Web Page**, page 51
  - ❑ **Step 3: Generate a Menu**, page 64

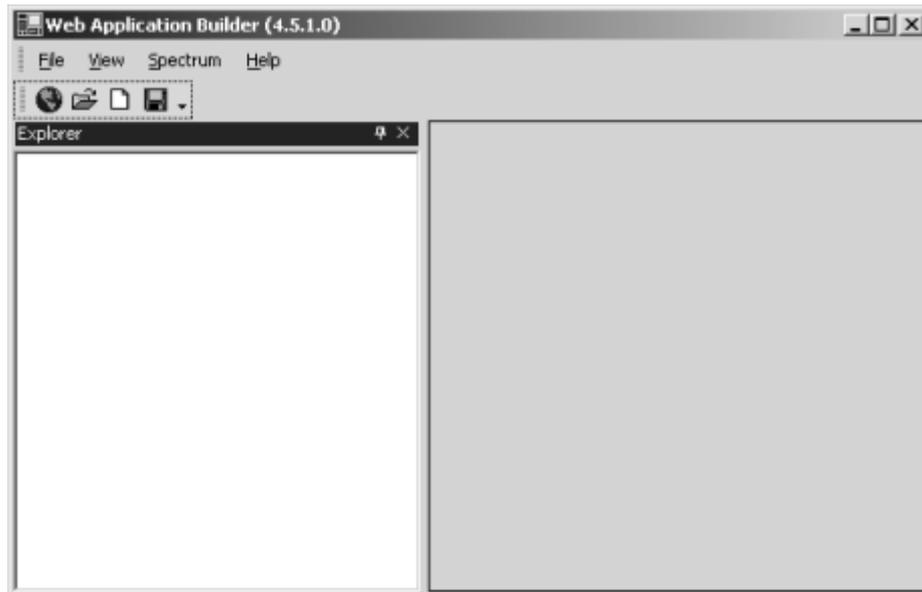
## Step 1: Create the Web Application

---

**Note:** You cannot regenerate a web application.

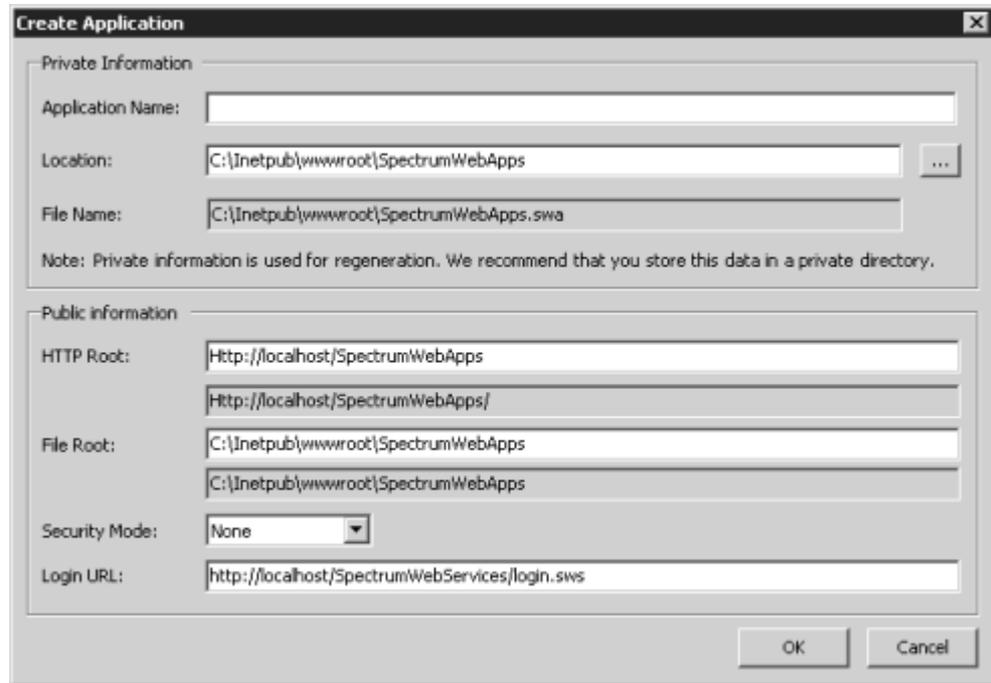
---

- To create a web application:
- 1 Select Web Application Builder from the Web Services SDK for Microsoft .NET submenu.  
The Web Application Builder window is displayed:



Web Application Builder Window

- 2 Select New > Web Application from the File menu.  
The Create Application window is displayed:



Create Application Window

Use this window to verify the default location of the private and public files for your web application. You can also specify the security mode for public files. Valid security mode options are None, Token, or Password.

---

**Note:** If you select Token from Security Mode, type the name of the web server that creates and stores the token in Login URL.

---

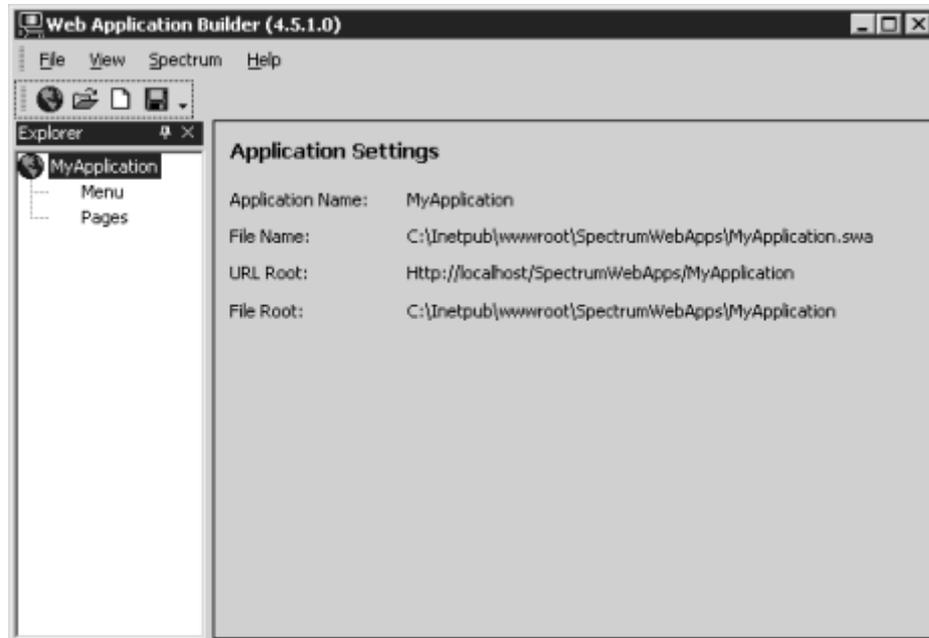
---

**Note:** The security mode specified in this window must be the same as the security mode defined in the Web.config file for the Web service. For more information, see **Modifying the Web.config File**, page 68.

---

- 3 Type “MyApplication” in Application Name.  
As you type the name, it is reflected in File Name for private files and HTTP Root and File Root for public files.

- Click OK.  
MyApplication is displayed in the Web Application Builder window:



Application Settings for New Web Application

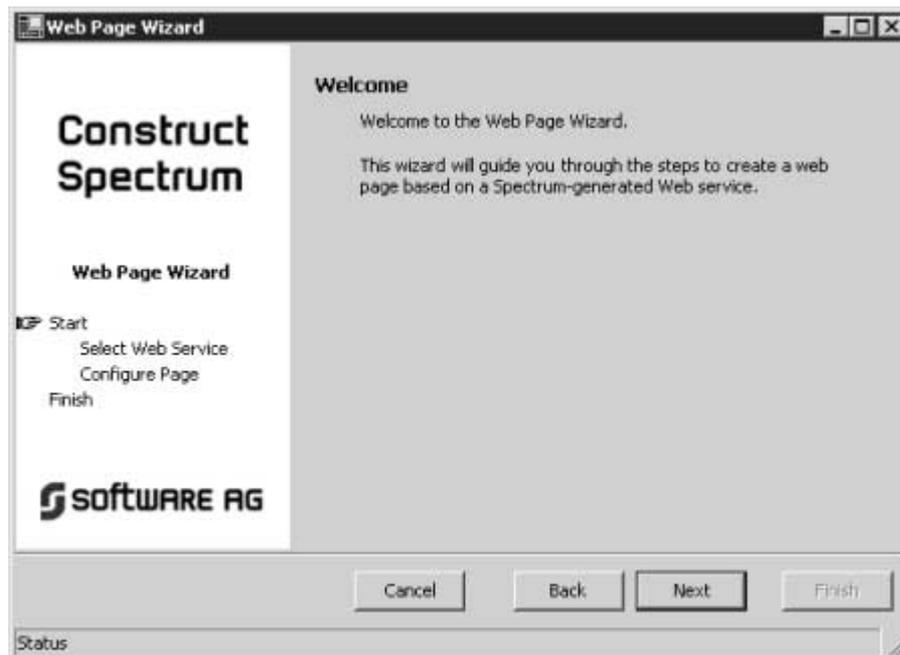
This window shows the application settings for your new web application. The Web Application Builder creates support files for the application based on the directory structure shown in this window.

You can use this window to invoke the Web Page wizard and Menu wizard to generate a web page and menu.

## Step 2: Generate a Web Page

Next, use the Web Page wizard to generate a maintenance or browse page for your web application. The following example generates a maintenance web page for the Web service generated in **Generating a New Web Service**, page 23.

- To generate a web page for your web application:
  - 1 Right-click Pages in the Explorer view in the Web Application Builder window for your web application.  
The Web Page wizard Start window is displayed:



Web Page Wizard Start Window

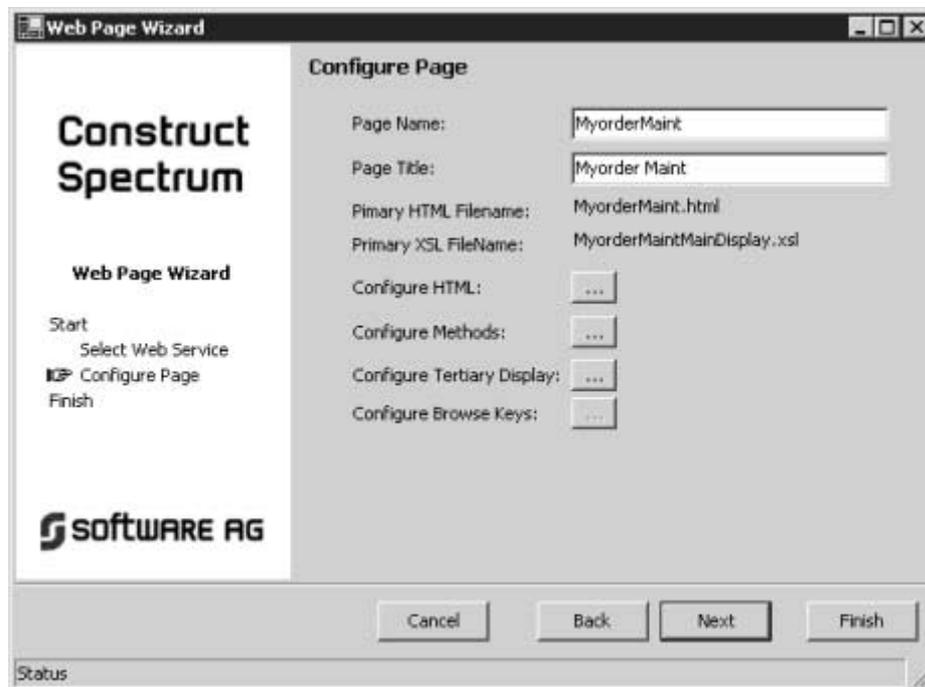
- 2 Click Next.  
The Select Web Service window is displayed:



Select Web Service Window

- 3 Click the browse button (...) for Web Service URL and select your Web service.  
For this example, select "MyProxy".

- Click Next.  
The Configure Page window is displayed:



Configure Page Window

This window shows the default name of the web page, page title, and the primary HTML and XSL file names. Optionally, you can use this window to access the following options:

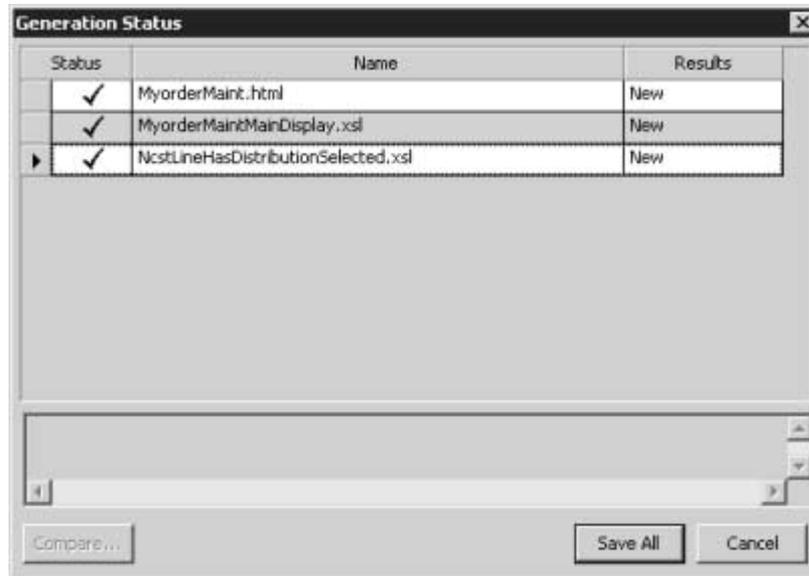
- **Configure HTML**  
This option allows you to tailor the HTML for your web page. For information, see **Configure HTML Option**, page 56. It includes advanced options, such as linking web pages or defining the value list.
- **Configure Methods**  
This option allows you to indicate which methods are available on your web page. For information, see **Configure Methods Option**, page 61.
- **Configure Tertiary Display**  
This option allows you to define the headings for tertiary data displayed on your web page. For information, see **Configure Tertiary Display Option**, page 62.
- **Configure Browse Keys**  
This option allows you to define which fields are available to browse on your web page. For information, see **Configure Browse Keys Option**, page 63.

- 5 Click Next.  
The Ready to Generate window is displayed:



Ready to Generate Window

- Click Finish.  
The Generation Status window is displayed:

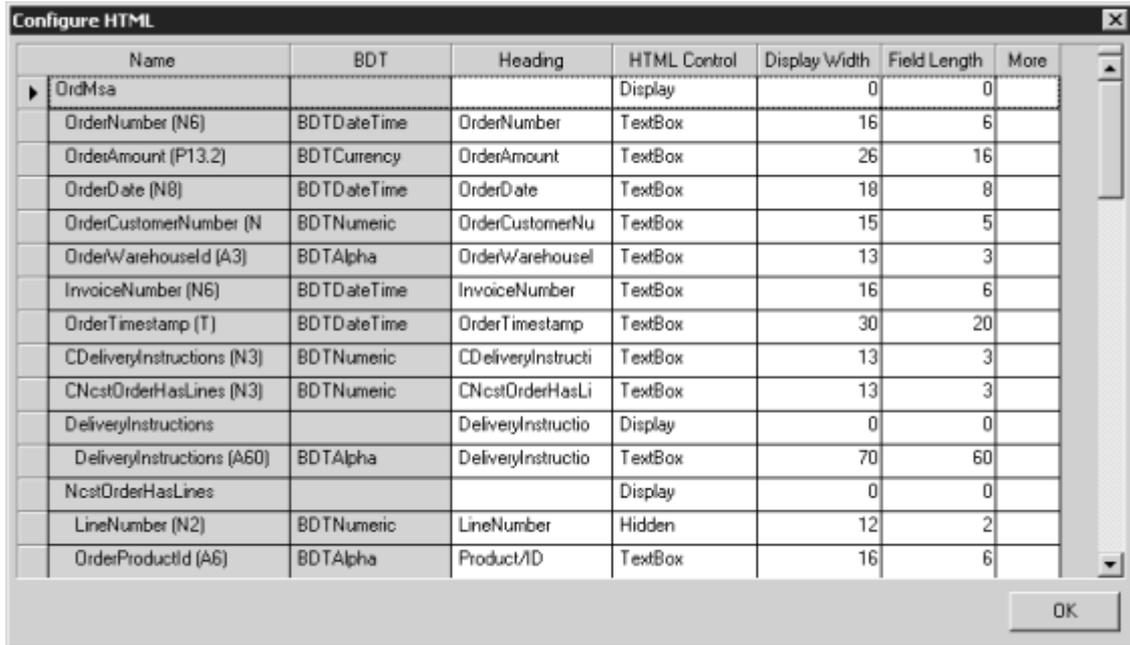


Generation Status Window

- Click Save All to save the web page files.  
Your web page is added to your web application. Next, you can generate a menu for your web application. For information, see **Step 3: Generate a Menu**, page 64.

## Configure HTML Option

- To configure the HTML for your web page:
  - 1 Select Configure HTML in the Configure Page window. The Configure HTML window is displayed:



Configure HTML Window

Use this window to indicate how information is displayed on your web page. For example, you can select:

Field	Description
Heading	Changes the heading displayed for the field.
HTML Control	Changes the HTML control (for example, changes a text box to a combo box). For a list of HTML controls, see the table on the following page.
Display Width	Changes the display size (width) of the field.
Field Length	Changes the length of data that can be entered in the field.
More	Sets advanced HTML options, such as linking your web page to other web pages or specifying the value list (see the following sections). Double-click More to display the options.

---

**Tip:** If you select a display option of Hidden at a group level for an HTML control, no field is displayed in the group — regardless of the display options selected at the field level.

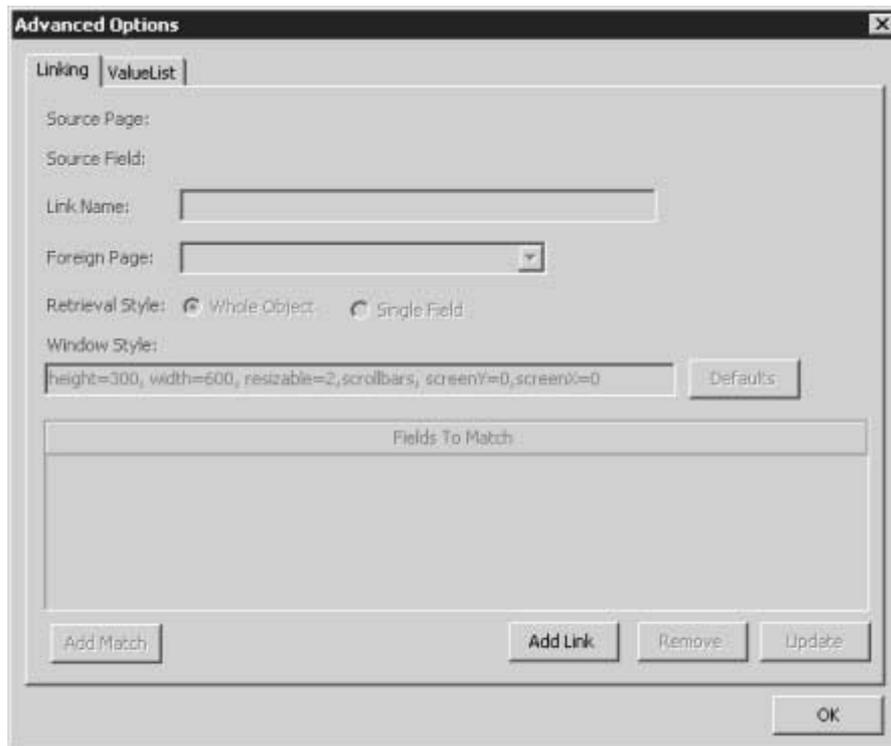
---

The following table describes each HTML control, as well as any requirements for the control and the values to which the control applies:

<b>HTML Control</b>	<b>Description</b>	<b>Requirements</b>	<b>Applies To</b>
CheckBox	Check box	Boolean	Boolean values
Combo Box	Drop-down selection box	Values provided by the Value list	Elementary values
Display	Read only	None	All variables
Hidden	Invisible	None	All variables
RadioButton	Selection buttons	Values provided by the Value list	Elementary values
TextArea	Multi-line text box	One-dimensional array	One-dimensional arrays
TextBox	Standard input	None	All variables

### Advanced Options Window

- To set advanced HTML options:
- 1 Select More for a field in the Configure HTML window. The browse button (...) is displayed.
  - 2 Click the browse button. The Advanced Options window is displayed, showing the options available on the Linking tab:



Linking Tab in the Advanced Options Window

The Linking tab allows you to specify links between web pages, update links between pages, or remove links between pages. For example, you can link a customer browse page to an order browse page. When users select a customer number on the customer page, they can drill-down to details about all orders for that customer. Or you can link an order browse page to an order maintenance page. When users select an order from the order browse page, they can drill-down to the order maintenance page.

- To define a link:
- 1 Click Add Link.  
The names of the Source Page and Source Field are displayed and other fields in the window become active.
  - 2 Type a name for this link in Link Name.
  - 3 Select the page you want to link your web page to from Foreign Page.  
When linking a maintenance page to a browse page, the code is generated as follows:
    - a look up for a whole object  
For example, you can browse the Customer file and select a customer for maintenance activities. All the data for the Customer object is then displayed on the maintenance page for that customer.
    - a lookup for a single field  
For example, if you do not know the warehouse ID, but you know the name of the warehouse, you can browse the Warehouse file until you find the one you want, select the warehouse, and the warehouse ID field is populated.
  - 4 Select the matching field from Fields to Match.  
This option differs, depending on the retrieval style:
    - a look up for a whole object  
For this option, provide the matching fields for this object. If the object was Order line, for example, users can provide an order and line number and then information for the whole object is returned, such as the quantity, unit price, line total, etc.
    - a look up for a single field  
For this option, the matching fields are similar to a look up. For example, you can attach a browse page to the warehouse ID field so users can browse through the Warehouse file by name. After they select a warehouse, however, only the warehouse ID field is populated on the maintenance page.
  - 5 Click OK.

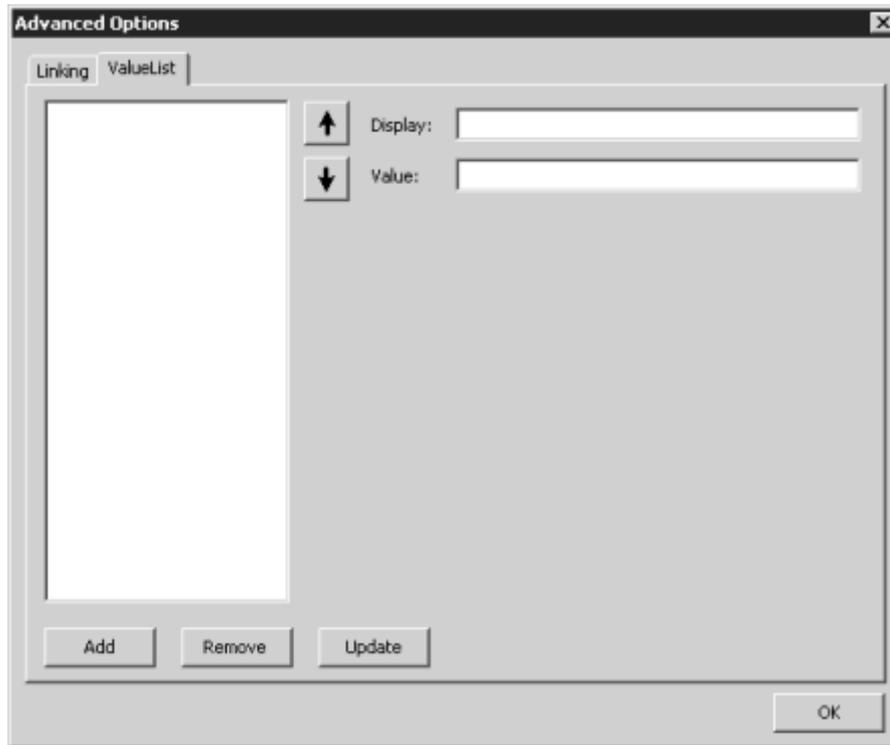
---

**Note:** Typically, linked web pages show up as windows. To change the size or display characteristics of the windows, click Defaults and specify the style using the parameters provided in Window Style.

---

If the Linking option is defined, an L is displayed in the corresponding More column in the Configure HTML window.

The following example shows the ValueList tab:



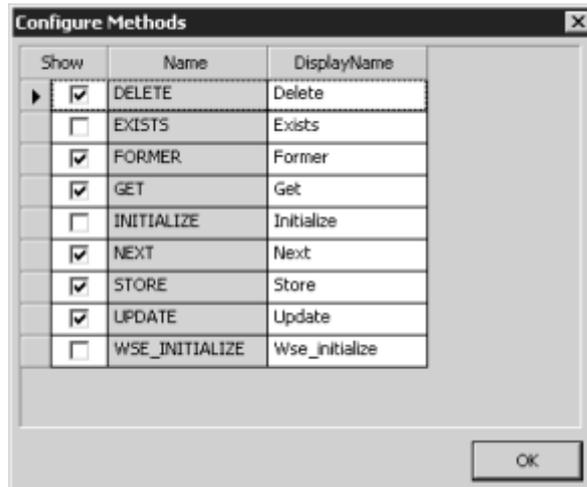
ValueList Tab in the Advanced Options Window

The value list is automatically populated from table verification rules attached to fields in Predict. Use the ValueList tab to add, remove, or update the list of valid values for a combo box or radio button.

If the ValueList option is defined, a V is displayed in the corresponding More column in the Configure HTML window.

## Configure Methods Option

- To configure the methods available on your web page:
  - 1 Select Configure Methods in the Configure Page window. The Configure Methods window is displayed:



Configure Methods Window

This example shows the methods available for a maintenance web page. Use this window to indicate which methods are available on your web page or to change the name displayed on the web page for that method (for example, change “Store” to “Add”).

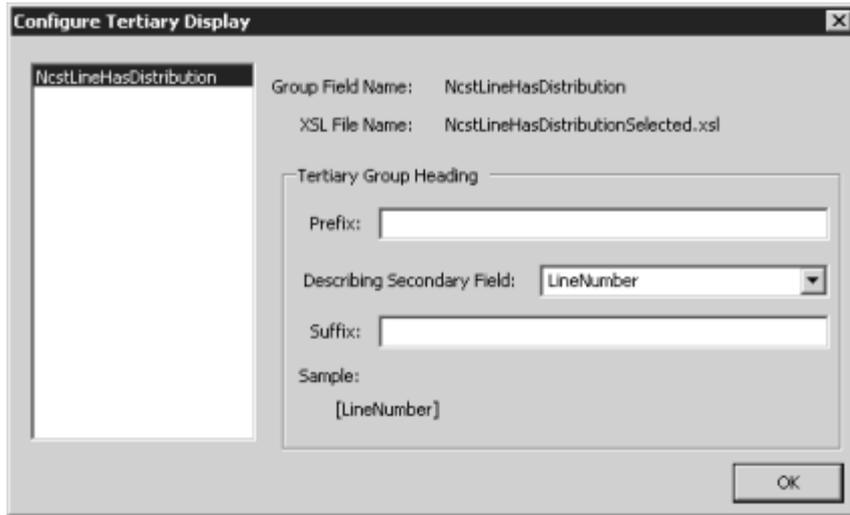
---

**Tip:** To initialize the page locally, select WSE\_INITIALIZE.

---

## Configure Tertiary Display Option

- To configure the tertiary display for your web page:
- 1 Select Configure Tertiary Display in the Configure Page window. The Configure Tertiary Display window is displayed:



Configure Tertiary Display Window

When displaying a tertiary group, a two-dimensional array must be presented in a one-dimensional window. To handle this complexity, only one occurrence of one of the dimensions is displayed at one time. For example, each order line is associated with one or more distributions. Only the distributions associated with the current order line are displayed at one time. To distinguish between lines, a field from the secondary file (for example, line number) can be displayed as part of the header, along with a prefix and/or suffix.

Use this window to define the headings displayed in the tertiary grid. In this example, you can use Prefix to define the beginning of the heading for LineNumber and Suffix to define the ending. For example, if:

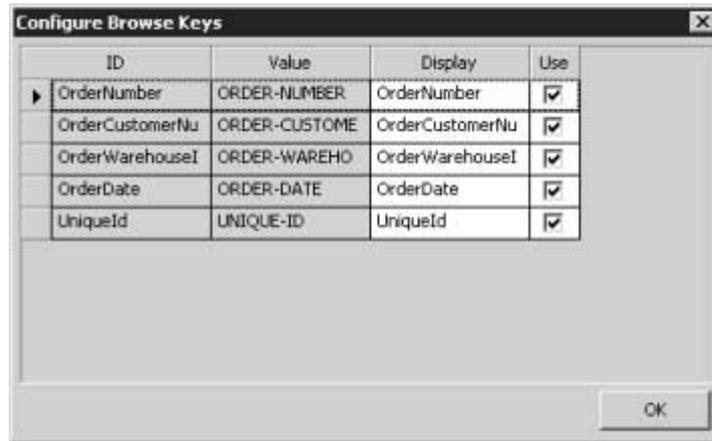
- the line number is 111
- the Prefix is “Product”
- the Suffix is “Row”

The tertiary group heading is:

“Product 111 Row”

## Configure Browse Keys Option

- To configure the browse keys for your web page:
  - 1 Select Configure Browse Keys in the Configure Page window. The Configure Browse Keys window is displayed:



Configure Browse Keys Window

Use this window to define which fields are available to browse on your web page.

## Regenerating a Web Page

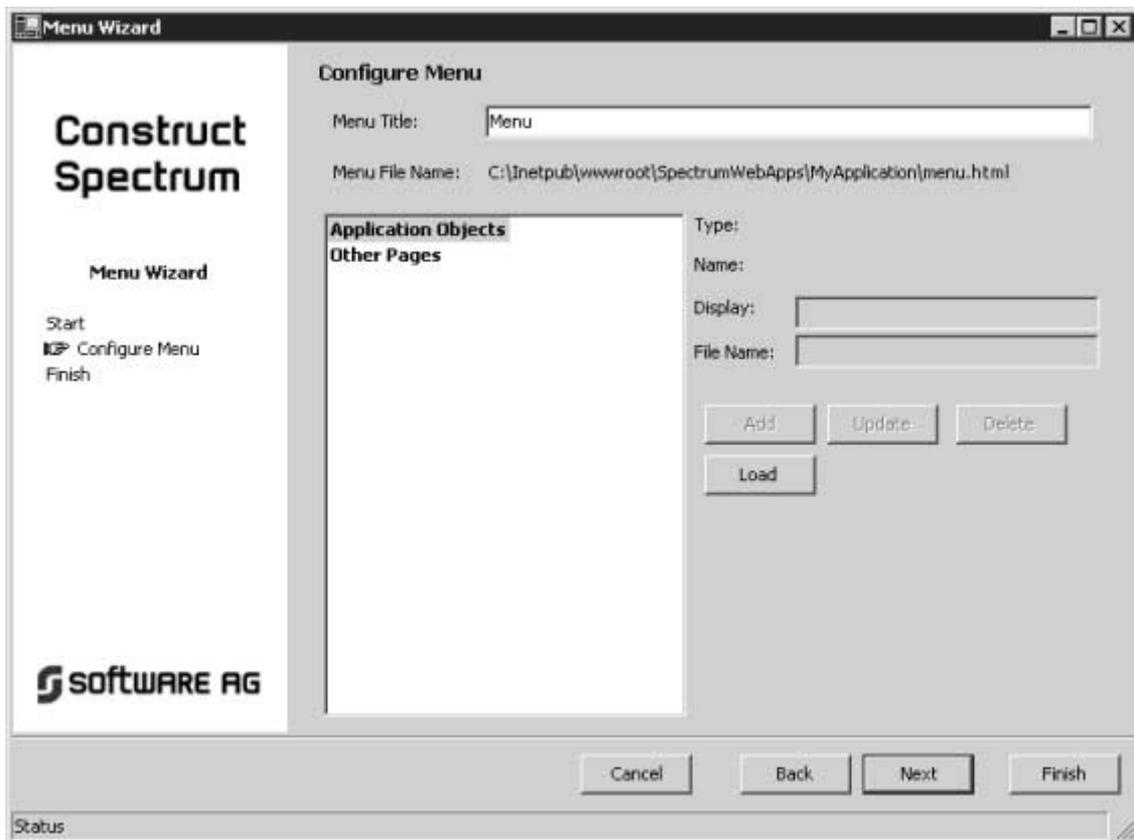
The Web Page wizard supports regeneration.

- To regenerate a web page:
  - 1 Open the Web Application Builder window. For information, see **Step 1: Create the Web Application**, page 48.
  - 2 Open the web application containing the page you want to regenerate.
  - 3 Select the web page from the Explorer window.
  - 4 Right-click the web page and select Show Wizard. The Web Page wizard is displayed, showing the information specified for the selected web page.
  - 5 Modify the page as desired and click Finish. The Generation Status window is displayed, showing which files were changed during regeneration.

## Step 3: Generate a Menu

After generating one or more web pages, use the supplied Menu wizard to generate a menu to tie all the pages together. The following example generates a menu for the web page generated in **Step 2: Generate a Web Page**, page 51.

- To generate a menu for one or more web pages:
  - 1 Right-click Menu in the Explorer view in the Web Application Builder window for your web application.
  - 2 Select Show Wizard.  
The Menu wizard Start window is displayed.
  - 3 Click Next.  
The Configure Menu window is displayed:



Configure Menu Window

Use this window to load application objects and to add, update, or delete web pages.

- 4 Click Load.  
All pages created for the selected application are added to the menu. You can change the names as desired in Display.

---

**Note:** To include web pages that were not generated by the Web Page wizard, click Other Pages and add the full URLs to File Name.

---

- 5 Click Next.  
The Ready to Generate window is displayed.
- 6 Click Finish.  
The Generation Status window is displayed.
- 7 Click Save All to save your menu.  
You have successfully generated a new web application, page, and menu.

## Regenerating a Menu

The Menu wizard supports regeneration.

- To regenerate a menu:
- 1 Open the Web Application Builder window.  
For information, see **Step 1: Create the Web Application**, page 48.
  - 2 Open the web application containing the menu you want to regenerate.
  - 3 Select the menu from the Explorer window.
  - 4 Right-click the menu and select Show Wizard.  
The Menu wizard is displayed, showing the information specified for the selected menu.
  - 5 Modify the menu as desired and click Finish.  
The Generation Status window is displayed, showing which files were changed during regeneration.



---

## CUSTOMIZING

This chapter describes how to modify the Web.config files and how to create a request for a Spectrum security token. It also describes how to add custom code for your Web service and how to create a custom BDT.

The following topics are covered:

- **Modifying the Web.config File**, page 68
- **Creating a Request for a Spectrum Security Token**, page 73
- **Adding Custom Code**, page 74
- **Creating a Custom BDT**, page 79
- **Adding a Web Services Root Directory**, page 82

## Modifying the Web.config File

---

**Note:** Before making any changes to the Web.config file, make a backup copy of the file.

---

---

**Note:** If you make changes to a generated Web service and do not change the Web.config file, the updates will not be accessible to users invoking your service. To make the changes accessible to users, either recycle IIS (Internet Information Server), or unload the root application, or re-save Web.config.

---

During the installation of the Construct Spectrum SDK for Microsoft .NET framework, the Web services root directory is created:

```
C:\inetpub\wwwroot\[SpectrumWebServices]
```

This directory contains the Common and Login directories, as well as the following public files:

- The .xsd files that describe the XML data used by each Web service.
- A .wsdl (Web Service Definition Language) file containing information specific to the SOAP Client. This file describes each Web service and its methods. It also imports the .xsd files.
- A global Web.config file containing information common to all Web services, such as the dispatcher name, security settings, and error and language options.

After generating a Web service, a private folder containing the following files is created in the local folder:

- One .xml file for each PDA used by the Web service; this file contains the mapping between Natural and XML. File names are derived from the PDA name (for example, CDAOBJ2\_PDAMap.xml).
- Metadata .xml file for the Web Page wizard for SWS (if selected).
- A .swsd file containing the Web Service wizard specifications.
- A local Web.config file containing information specific to this Web service, such as the names of the proxy, PDAs, and methods used by the service, as well as any overrides.

With the exception of events, the settings in the local Web.config files override the settings in the global Web.config file. Events in both files are processed.

---

**Note:** You can copy any global Web.config file setting into the Web.config file in the local Web service directory to create an override, but changes will be lost if you regenerate the service. While the global Web.config file remains unchanged during generation, the local files are recreated each time a service is regenerated. Use this file for overrides only (for example, for BDT modifiers).

---

➤ To modify the global Web.config file for your environment:

- 1 Open the global Web.config file.  
This file is located in the Web services root directory (SpectrumWebServices).
- 

**Warning:**

Some settings in the Web.config file are required by the Web Service wizard and should not be modified. For example, configSections contains the IIS (Internet Information Server) settings that associate the .sws files to the Spectrum Web Services Engine.

---

- 2 Scroll to the <Spectrum> node.  
You can define or modify any of the following settings:
    - MessageDatabase  
You can supply the file location for your own messages (for example, mymessages.xml) or you can modify messages or add the text for other languages in the supplied messages.xml file (message numbers 5700–5799).
- 

**Note:** We recommend that you only append information to this file, although you can make minor text changes, such as changing the language number or value.

---

- RetrieveMetaData  
To retrieve metadata for use with the Web Page wizard for SWS, set this option to True; if you are not using the Web Page wizard for SWS, set this option to False.
- 

**Warning:**

If you set this option to True to retrieve the metadata, ensure that you switch it to False before going into production.

---

- **SpectrumSecurity**  
To define Spectrum security options for Web services, set one of the following options:
  - **Token**  
Sends a token with the SOAP message. A token is a unique, system-generated, identification number that allows users to logon the first time with a user ID and password and then request a token for subsequent calls to the mainframe. For information, see **Creating a Request for a Spectrum Security Token**, page 73.
  - **Password**  
Sends the user ID and password with the SOAP message.
  - **None**  
Indicates no Spectrum security.
  - **HardCoded**  
Indicates security on the mainframe, but you do not want to send the token or password with the SOAP message.
- **ErrorOptions**  
You can set the following error options:
  - **Log**  
Set True to log errors to the Windows Event log.
  - **StackTrace**  
Set True to output the call stack.
  - **Email**  
Set True to send errors by email and then specify all email nodes except CC (optional).

---

**Note:** You can also use this option to override all error messages to one generic message (for example, Spectrum error).

---

- **DefaultLanguage**  
Use this option to indicate the default language used for Web services. If the language code is not specified in the Construct Spectrum Administration subsystem for a user, the language code specified here is used. For more information about language codes, refer to the Natural documentation.

- **EventHandler**  
This option identifies the type and location of custom event handlers. You must specify Type to identify the DLL files; FileName is optional.

---

**Note:** Event handlers execute customized code based on a Web service runtime event. For information on how to load the assembly containing your event handlers, see **Modify the Web.config File for Custom Code**, page 78. For information about creating event handlers, see **Create a Custom Event Handler Class**, page 77.

---

- **Dispatcher**  
Use this option to set the name of the dispatcher the Web services will use (as defined in the Spectrum Service Manager), as well as the KillTimeout value and the maximum number of dispatchers in the pool.
- **BDTs**  
If there are no overrides in the private folder for a Web service, this option specifies the BDT defaults. For example, you may want BDTNumeric to suppress zeros for most Web services (ZERO=OFF), but display zeros for one (ZERO=ON).

## Event Hierarchy and Inheritance

The Web.config file settings are used by the Web Service Engine (WSE). Settings specified in the global Web.config file in a parent directory are inherited into the local Web.config files within the .NET framework. For example, the dispatcher used for all Web services is defined in the global Web.config file in the Web services root directory (SpectrumWebServices). Every child directory (directory containing one Web service) in the root directory inherits the global Dispatcher setting. If the dispatcher is specified in a local directory, as well as the SpectrumWebServices directory, the setting in the local directory is used.

Events from the WSE are handled in a slightly different manner. Rather than overwriting the event settings in the global Web.config file, the events specified in the global file are appended to those in the local Web.config file. For example, if an assembly is specified in the global Web.config file to handle the onException event and an assembly is also specified in a local Web.config file, both event handlers are invoked if an exception is raised in the child.

## .SWS Mapping

For the WSE to process \*.sws requests from the internet, both IIS (Internet Information Server) and the .NET framework must be configured:

- Within IIS, the \*.sws files must be mapped to the .NET framework. (aspnet\_isapi.dll). When a .sws request comes to IIS, IIS calls .NET to handle the request. This configuration is done automatically during installation of the SDK.
- Within .NET, .sws requests must be mapped to the WSE.

The following example illustrates this mapping:

```
*.sws Request -> IIS -> .NET framework (aspnet_isapi.dll) -> WSE
```

## Creating a Request for a Spectrum Security Token

You must create a request for a token if the Spectrum Security node in the Web.config file is set to token. A token is a unique, system-generated identification number that allows users to logon to a Web service using their user ID and password and then request a token for subsequent calls.

Use the SOAP Client testing tool to create the request for a security token. For information on using this tool, see **Testing Your Web Service**, page 43.

The following example shows input for the SOAP Client to create the request:

**URL:** http://<machine name>/SpectrumWebServices/login.sws

**SOAP action:** Login

**Parameters:** UserID and Password

**XMLInput:**

```
<?xml version="1.0" encoding="UTF-8"?><soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:spe="http://SoftwareAG.com/Spectrum/SPE451">
  <soap:Body>
    <spe:User>
      <spe:UserID>Drew</spe:UserID>
      <spe:Password>My Password</spe:Password>
    </spe:User>
  </soap:Body></soap:Envelope>
```

**XMLOutput:**

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <spe:Token xmlns:spe="http://SoftwareAG.com/Spectrum/SPE451">f7e759ae-9e00-4097-b04a-aa0726f7ff42</spe:Token>
  </soap:Body>
</soap:Envelope>
```

For more information, refer to the login.wsdl file in the Web services root directory (SpectrumWebServices).

## Adding Custom Code

You can write custom code, called an event handler, to react to events raised within the Web Service Engine (WSE). If an event handler “subscribes” to an event, the custom code will be invoked whenever that event is raised within the WSE.

### Web Service Engine (WSE) Events

The WSE events are:

- ValidateUser
- BeforeCallnat
- AfterCallnat
- ExceptionThrown

The following sections describe these events.

#### ValidateUser Event

This event allows you to add custom validation routines to validate a user ID. If a validation is successful, set the `UserArgs.UserValidated` property to `True` (default is `False`). Unless the `UserValidated` property is manually set to `True`, this event will fail and an exception will be thrown by the WSE.

Argument	Usage
<code>ValidateUserEventArgs</code>	Use this event to authenticate a user against an external source.

#### BeforeCallnat Event

This event is raised after a SOAP message is read into `BeforeCallnatEventArgs.Fields` and before `Natural` is called. The XML document exposed in this event is included for your information only. If you modify this document, the changes will not be sent to the mainframe. To send data to the mainframe, modify the `NatFields` object.

Argument	Usage
<code>BeforeCallnatEventArgs</code>	Use this event to look up data in an external source and then save it in <code>Natural</code> or to “massage” data going to the mainframe. For example, you can access an external source for additional information and then modify the SOAP message accordingly.

## AfterCallnat Event

This event is raised after the call to Natural is returned and the SOAP response has been built and is ready to be sent to the client. As with the BeforeCallnat event, the InputXML and Fields properties are provided for your information only. To send data back to the client, you must modify the OutPutSOAPNode property.

Argument	Usage
AfterCallnatEventArgs	Use this event to look up data in an external source based on data returned from a Natural subprogram proxy. The data is then sent to the client.

## ExceptionThrown Event

This event is raised when the WSE encounters an exception. Subscribing to this event does not stop the exception from being thrown nor allow you to “fix” the exception.

Argument	Usage
ExceptionThrownEventArgs	Use this event to log an exception to an external source.

## Event Arguments

The following sections describe the arguments provided for the WSE events.

**Note:** All of the Event arguments have a Handled property (Boolean), which indicates how the event is handled. If this property is set to True, other subscribers to that event will not be invoked.

## SPEEventArgs

The supplied Spectrum events support the following arguments:

Name	Type	Information only	Description
InputXML	XMLNode	yes	Contains the SOAP message.
InputXMLNameSpaceManager	XMLNameSpaceManager	yes	Contains the namespace manager for InputXML.

Name	Type	Information only	Description (continued)
WebServiceName	String	yes	
Timestamp	DateTime	yes	Contains the date and time from the IIS machine that the event was raised.

### ValidateUserEventArgs

Inherits the SPEEventArgs arguments; supports the following arguments:

Name	Type	Information only	Description
User	UserCredentials	no	Contains user information.
UserValidated	Boolean	no	Indicates whether the user ID is validated.

### BeforeCallnatEventArgs

Inherits the SPEEventArgs arguments; supports the following argument:

Name	Type	Information only	Description
Fields	NatFields	no	Contains the collection of fields passed to the subprogram proxy.

### AfterCallnatEventArgs

Inherits the SPEEventArgs arguments; supports the following arguments:

Name	Type	Information only	Description
OutPutSOAPNode	XMLNode	no	Contains the XML node sent to the client.
Fields	NatFields	no	Contains the collection of fields returned from the subprogram proxy.

## ExceptionThrownEventArgs

Supports the following arguments:

Name	Type	Information only	Description
Exception	Exception	yes	Contains the exception that was thrown.
Handled	Boolean	no	Indicates how the event is handled. See note in <b>Event Arguments</b> , page 75.
ServiceName	String	yes	Contains the name of the Web service being invoked.

## Create a Custom Event Handler Class

If desired, you can create a custom event handler class.

**Note:** The SampleEvent project in your Install directory contains a sample event handler.

- To create a custom event handler class:
  - 1 Create a new class library in Visual Studio.net.
  - 2 Within this library, add a reference to:  
`SoftwareAG.Spectrum.WebServiceEngine.dll` (located in your Construct Spectrum Install directory).
  - 3 Create a class that implements `ISpeEventHandler`.
  - 4 Create the desired event handler methods.
  - 5 In the `ISpeEventHandler.WireUp` method, subscribe the methods to their corresponding events.
  - 6 Modify the `Web.config` file and add the appropriate `EventHandler` node. For information, see **Modify the Web.config File for Custom Code**, page 78.

## Modify the Web.config File for Custom Code

To execute your customized code, the WSE must know how to load the assembly containing your event handlers. To do this, you must make an entry in the EventHandler node in the appropriate Web.config file. There are two methods of doing this.

---

**Note:** The Type attribute format is the same format used to load a custom HTTP handler in Asp.net. For more information, refer to the Microsoft documentation.

---

### Method 1

This method loads the assembly based on the file name specified:

```
<EventHandler FileName="" Type="" />
```

For example:

```
<EventHandler FileName="C:\[Path to  
DLL]\SoftwareAG.Spectrum.TestEvent.dll"  
Type="SoftwareAG.Spectrum.TestEvent.Sample" />
```

### Method 2

This method is more secure than method 1, because the assembly is digitally signed and registered into the GAC. This minimizes the risk of unauthorized users inserting their own assembly or tampering with the original assembly.

```
<EventHandler Type="" />
```

For example:

```
<EventHandler Type="SoftwareAG.Spectrum.TestEvent.Sample,  
SoftwareAG.Spectrum.TestEvent, Version=1.0.1.1, Culture=neutral,  
PublicKeyToken=30e0547b9a8498ae" />
```

---

**Note:** If users are using GAC, they must generate their own tokens.

---

## Creating a Custom BDT

Creating a custom BDT for a Web service is different from creating a custom BDT for a web application. Once created, you can use your custom BDT with any Web service.

- To create a custom BDT:
  - 1 Create a .NET class library project.
  - 2 Add a reference to `SoftwareAG.Spectrum.BusinessDataTypes.dll` in this project.
  - 3 Create a new class that inherits from the BDT (for example, BDT class).
  - 4 Call the BDT constructor with the following parameters:
    - The name of the BDT  
Ensure that the name easily identifies this BDT.
    - A character array of Natural types this BDT will support.  
The characters correspond to those used in Natural (A for Alpha, for example).

In Visual Basic .NET, for example:

```
Mybase.new("ExternalBDT", new Char() {"A"c, "N"c})
```

- 5 Override the `ConvertFromDisplay` and `ConvertToDisplay` functions and add your custom code.
- 6 Compile the class library.

## Use the BDT During Generation

- To use the BDT during generation:
  - 1 Add an entry in the `BDTs.xml` file in your Install directory.  
For a description of this file, see **BDTs.xml File**, page 80.
  - 2 Add an entry for your BDT in the `BDTs` node in the `Web.config` file.  
For example:

```
<BDTFile FileName="[ExternalBDT.dll]">  
  <BDT Type="[ClassName]" SetDefault="true">  
    <Modifier Name="[ModName]">[Value]</Modifier>  
  </BDT>  
</ BDTFile >
```

The user provides substitution values for `ClassName`, `ModName`, and `Value`.

## BDTs.xml File

There are two types of XML nodes in the BDTs.xml file. The first type (BDT) declares a BDT for use within the Web Service wizard. The second type (DataType) matches a Natural data type to a BDT.

### BDT Node

Node Name	Attributes	Parent	Description
BDT		N/A	Parent node used to declare a BDT.
XSDTypes		BDT	Container for Type nodes.
Type	true false (default)	XSDTypes	XSD types used with Web services.
PredictMap			Corresponding Predict keyword.

### *Example of BDT Node Settings*

```
<BDT Name="BDTNumeric">
  <XSDTypes>
    <Type>xsd:string</Type>
    <Type>xsd:decimal</Type>
    <Type Default="true">xsd:double</Type>
    <Type>xsd:float</Type>
    <Type >xsd:int</Type>
    <Type>xsd:short</Type>
    <Type>xsd:long</Type>
  </XSDTypes>
  <PredictMap>BDT_NUMERIC</PredictMap>
</BDT>
```

## Data Type Node

When using the Web Service wizard, the wizard tries to determine which BDT to use for a field based on the Natural format for the field. The `Data Type` node determines the search criteria. The `Type` attribute contains a regular expression that is compared to the Natural format. If the regular expression matches, then that BDT is chosen.

Node Name	Attributes	Parent	Description
<code>Data Type</code>	<code>Type</code> is the regular expression compared to the Natural format.	N/A	
<code>BDT</code>	<code>true false</code> (default) If this BDT is chosen, use it as the default.		Name of a BDT.

### Example of Data Type Node Settings

```
<Data Type Type="[ANP]10$">
  <BDT>BDTPhone</BDT>
</Data Type>
```

If the Natural format is A10, N10, or P10, BDTPhone is used because the “A10”, “N10”, or “P10” string matches the “[ANP]10\$” regular expression.

## Adding a Web Services Root Directory

The Construct Spectrum SDK for Microsoft .NET framework supplies a default Web services root directory during installation. If desired, you can add your own Web services root directory.

- To add an additional Web services root directory:
  - 1 Select Control Panel > Administrative Tools > Internet Services Manager.
  - 2 Scroll to the directory you want to add and right-click the directory name.
  - 3 Select Properties > Configuration > .sws.
  - 4 Click Edit.
  - 5 Copy the name displayed in the Executable field to the clipboard.  
The executable name is used to associate .sws with the .NET runtime environment.
  - 6 Click Cancel to close .sws.
  - 7 Click Add to add another mapping.
  - 8 Paste the executable name in the Executable field.
  - 9 Type “.sws” in the Extension field.
  - 10 Click OK three times to close the Internet Services Manager.

---

## TIPS AND TECHNIQUES

This chapter contains tips and techniques you can consult when using the Construct Spectrum SDK to create Web services. It also contains troubleshooting information for common errors you may encounter while using the SDK.

The following topics are covered:

- **Tips for Data Updates**, page 84
- **Miscellaneous Tips**, page 86
- **Sample SOAP Messages**, page 88
- **Troubleshooting Common Errors**, page 90

## Tips for Data Updates

This section contains helpful tips and techniques when using the UPDATE action.

### Verifying Namespace Names

If data is not returned in your input tags, verify that the namespace name is correct. For example:

```
xmlns="PDA:SIMPLEP.SIMPMSR"
```

### Excluding Attributes During an Update

If attributes are excluded during an update, data may be inadvertently deleted. For example, if delivery lines are not included in the order object XML, it is assumed that the delivery lines are blank. It is also extremely important that the values in the Restricted PDA remain unchanged when the object file is read. If not, data may be duplicated or a modification may not take place.

## Using the CDAOBJ2 Parameter Data Area

The following example shows the structure of the CDAOBJ2 parameter data area when all fields are requested at the lowest level of definition:

```
<CDAOBJ2 xmlns:spe="http://SoftwareAG.com/Spectrum/SPE451"
xmlns="PDA:SIMPLEP.CDAOBJ2">
- <Cdaobj2>
- <Inputs>
  <Function>EXISTS</Function>
  <ClearAfterUpdate>>false</ClearAfterUpdate>
  <ReturnObject>>false</ReturnObject>
  <EtIfSuccessful>>false</EtIfSuccessful>
</Inputs>
- <Outputs>
  <ObjectContainsDerivedData>>false</ObjectContainsDerivedData>
  <Exists>>true</Exists>
</Outputs>
</Cdaobj2>
</CDAOBJ2>
```

- If the outer node (<CDAOBJ2> </CDAOBJ2>) is not defined in the input XML, everything inside the node is ignored as irrelevant data. For example, the following is ignored because it is not within the context of CDAOBJ2:

```
<Cdaobj2>
- <Inputs>
  <Function>EXISTS</Function>
  <ClearAfterUpdate>>false</ClearAfterUpdate>
  <ReturnObject>>false</ReturnObject>
  <EtIfSuccessful>>false</EtIfSuccessful>
</Inputs>
</Cdaobj2>
```

- If you set ReturnObject in Inputs to true, the calculated values from the server are passed back to the client. This option is set in the Web Service wizard by default.
- For a Natural Construct-generated maintenance object to perform maintenance functions, you must include #FUNCTION. The Web service defaults the appropriate #FUNCTION value for the object.

## Miscellaneous Tips

This section describes miscellaneous tips and techniques when using the Construct Spectrum SDK for Microsoft .NET framework.

### Overriding Web Service Wizard BDT Defaults

BDTs are used to do simple data validation. Based on the formats and lengths of the fields being used, the Web Service wizard (WSW) “guesses” which BDT to use. For example, if a field is A6, it assigns a default BDT postal code check.

➤ To override the Web Service wizard BDT defaults:

- 1 Create a backup of the following file:

```
Program Files\Software AG\Spectrum Web Service SDK for .net\BDTs.xml
```

- 2 Modify the original BDTs.xml file.

For example, to remove the default BDT postal code check, comment out the following:

```
<DataType Type="A6$" >  
<BDT>BDTPostalCode</BDT>  
</DataType>
```

## Setting Up Steplibs

### Development Environment

To locate objects in a development environment, allocate steplibs in the Configuration file. For information, see **Using the Configuration Editor**, page 50, *Construct Spectrum SDK Reference*.

### Runtime Environment

To locate objects in a runtime environment, set up the steplibs in the SYSSPEC library on the mainframe.

---

## Changing the Dispatcher

### Development Environment

To change the dispatcher in a development environment, modify the Active Profile in the Configuration file. For information, see **Using the Configuration Editor**, page 50, *Construct Spectrum SDK Reference*.

### Runtime Environment

To change the dispatcher in a runtime environment, modify the dispatcher in the Web.config file in the Web services root directory (SpectrumWebServices). For information, see **Modifying the Web.config File**, page 68.

## Correcting an Invalid Subprogram Method List

If the Web Service wizard (WSW) does not list the correct methods, ensure that your subprogram proxy has the same domain and object name as indicated in the Construct Spectrum Administration subsystem. If they differ, regenerate the subprogram proxy and then ensure that the cache is cleared in the WSW.

---

**Note:** You must manually clear the cache; shutting down the WSW does not clear it. To clear the cache, click the Spectrum Cache icon in the lower, left corner of the window, select the cache to be cleared, and click Delete. For more information, see **Using The Spectrum Cache**, page 68, *Construct Spectrum SDK Reference*.

---

## Supplied Samples

The Construct Spectrum SDK for Microsoft .NET framework supplies samples you can refer to when developing Web services.

- Web service samples are loaded into the SpectrumExamples\Services directory
- Web application samples are loaded into the SpectrumExamples\Applications directory

## Sample SOAP Messages

The following sections contain sample input for each of the SOAP actions for a Natural Construct-generated maintenance object.

### Input for the DELETE Action

```
<?xml version="1.0" encoding="UTF-8"?><soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body><SIMPMSA xmlns:spe="http://SoftwareAG.com/Spectrum/SPE451"
xmlns="PDA:SIMPLEP.SIMPMSA">
<Simpmsa>
<WarehouseTimestamp>2002-09-10T20:53:43.5</WarehouseTimestamp>
</Simpmsa>
<SimpmsaId>217
</SimpmsaId>
</SIMPMSA>
- <SIMPMSR xmlns:spe="http://SoftwareAG.com/Spectrum/SPE451"
xmlns="PDA:SIMPLEP.SIMPMSR">
- <Simpmsr>
<HeldId>217</HeldId> <InterveningUpdFld>2002-09-10T20:53:43.5</
InterveningUpdFld>
</Simpmsr>
</SIMPMSR>
</soap:Body></soap:Envelope>
```

---

**Note:** As with the UPDATE action, the timestamps must match for the DELETE action. For more information, see **Input for the UPDATE Action**, page 89.

---

### Input for the EXISTS, GET, and NEXT Actions

```
<?xml version="1.0" encoding="UTF-8"?><soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body><SIMPMSA xmlns:spe="http://SoftwareAG.com/Spectrum/SPE451"
xmlns="PDA:SIMPLEP.SIMPMSA">
<SimpmsaId>111</SimpmsaId>
</SIMPMSA>
</soap:Body></soap:Envelope>
```

---

**Tip:** To access an object subprogram, include CDAOBJ2 as an output PDA for the EXISTS action and then select the EXISTS flag in the PDA listing so it is included in the XML output.

---

## Input for the INITIALIZE Action

```
<?xml version="1.0" encoding="UTF-8"?><soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
</soap:Body></soap:Envelope>
```

## Input for the STORE Action

```
<?xml version="1.0" encoding="UTF-8"?><soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body><SIMPMSA xmlns:spe="http://SoftwareAG.com/Spectrum/SPE451"
xmlns="PDA:SIMPLEP.SIMPMSA">
<Simpmsa>
  <WarehouseId>217</WarehouseId>
</Simpmsa>
</SIMPMSA>
</soap:Body></soap:Envelope>
```

## Input for the UPDATE Action

```
<?xml version="1.0" encoding="UTF-8"?><soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body><SIMPMSA xmlns:spe="http://SoftwareAG.com/Spectrum/SPE451"
xmlns="PDA:SIMPLEP.SIMPMSA">
<Simpmsa>
<WarehouseId>111</WarehouseId>
<WarehouseTimestamp>2002-09-09T23:18:32.1</WarehouseTimestamp>
</Simpmsa>
  <SimpmsaId >111
  </SimpmsaId >
  </SIMPMSA>
<SIMPMSR xmlns:spe="http://SoftwareAG.com/Spectrum/SPE451"
xmlns="PDA:SIMPLEP.SIMPMSR">
- <Simpmsr>
  <HeldId>111</HeldId>
<InterveningUpdFld>2002-09-09T23:18:32.1</InterveningUpdFld>
  </Simpmsr>
  </SIMPMSR>
</soap:Body></soap:Envelope>
```

For troubleshooting information, see **Correcting Errors During an Update**, page 90.

## Troubleshooting Common Errors

This section describes some of the errors you may encounter and what you can do to resolve them.

### Setting Zero Suppression

At runtime, you may get an error converting a BDT default value. For example:

```
<BDTName>BDTCurrency</BDTName>
  <FieldName>ORD-MSA.UNIT-COST(2)</FieldName>
  <Message>Error converting value: .</Message>
```

This error occurs when the runtime BDT default does not accept zeros as input. To allow zeros, add (or modify) the `BDTCurrency` line to the `Web.config` file in the Web services root directory (`SpectrumWebServices`). Name must be `ZERO` and the setting must be `ON`. For example:

```
<BDTs>
<Modifier BDTName="BDTCurrency" Name="ZERO">ON</Modifier>
</BDTs>
```

---

**Note:** If `BDTCurrency` is `ZERO=ON`, null values are not converted to zeros.

---

### Correcting Errors During an Update

While performing the `UPDATE` action, you may receive the following message:

```
Attempted to update/delete:1:that was not in hold status
Intervening modification, please try again
```

This message indicates that the `MSA-ID` did not match the `HELD-ID`. The timestamp on the database differs from the timestamp in XML. Ensure the `Web.config` file includes the following:

```
<Modifier BDTName="BDTDateTime" Name="XML">true</Modifier>
```

Note the format of the date/time. In addition, retrieve the date from the database to confirm that the timestamp in the XML document is the same as the database. (If there is date in the XML document, it assumes that it is the current date.)

---

**Warning:**

If input parameters are not specified for an update, they are assumed to be blank.

---

## APPENDIX A: GLOSSARY

The following terms are used throughout the Construct Spectrum documentation set. Each term is listed with its meaning:

<b>Term</b>	<b>Definition</b>
active server page (ASP) script	Script that activates the WebApp.cls page handler, which opens the specified web page.
ActiveX business object (ABO)	Visual Basic class that represents a Natural business object on the client. The ABO wraps the Spectrum calls required to communicate with the Natural subprogram exposed by a subprogram proxy.
ActiveX DLL	Dynamic link library (DLL) containing one or more ABOs. It is used to package and deploy web applications.
application library	Natural library containing the server application components of a client/server application.
application service definition	Definition in the Construct Spectrum Administration subsystem that identifies the methods exposed by a subprogram. The definition is created automatically by the Subprogram-Proxy model. You can modify these settings on the Maintain Application Service Definition panel in the Construct Spectrum Administration subsystem.
application services	Natural subprogram implementing methods that can be called as remote services.
architecture	High-level description of the organization of functional responsibilities within a system. The architecture conveys information about the general structure of systems. It defines relationships between system components, but not the implementation of components.
assembly	Logical grouping of functionality in .NET. Can be one or more executable or DLL files.
browse data cache	Area containing database records returned from the server. Records are usually displayed in a browse window.
browse dialog	Generic GUI browse window called to display any browse data residing on a mainframe or PC.

Term	Definition (continued)
browse process	Process by which framework components and generated browse components retrieve data and, optionally, display it in a browse window. For example, a browse process can retrieve rows of data, search for specific values, and then perform calculations and conditional processing. Users can display the results in a browse window, if desired.
browse window	See <b>browse dialog</b> , page 91.
business data types (BDTs)	<p>Type validation on the client that applies business semantics to a field. Typically, BDTs are used to format field data specified by the user.</p> <p>For example, if an application has an input field to enter a phone number, you can associate a BDT with the field to reformat the number with hyphens. A user can enter “7053332112”. When the user moves to the next field or performs another action, the number is automatically reformatted as 705-333-2112.</p> <p>Construct Spectrum supplies standard BDTs, which you can customize, or you can create your own. BDT modifiers are added to the keyword components of a field in Predict.</p>
BDT class	Collection of all BDT procedures.
BDT controller class	Collection of methods available to members of a BDT class. See also <b>BDT class</b> .
BDT controller object	Supplied client framework component that is an instance of the BDT controller class and uses the methods available to that class. Each application declares a BDT controller object, which records and maintains a list of names for each BDT and points to the BDT definition. See also <b>business data types (BDTs)</b> , page 92.
BDT modifier	Additional logic users supply to modify the formatting or validation rules for a BDT. For example, <code>BTD_NUMERIC</code> ensures that only numeric values are entered in a field. You can also add a modifier to round numeric values. To increase flexibility, each BDT defines its own modifiers.
BDT procedure	Code that implements a BDT.
cache	See <b>security cache</b> , page 99.
cardinality	Number of dimensions of information. Information with the same number of dimensions has the same cardinality.

Term	Definition (continued)
child model	Individual model for which a super model (parent model) collects parameters and generates specifications.
client application	Portion of a Construct Spectrum client/server application that runs on a Windows platform.
complex redefine	Redefinition of a data area containing multiple data types, multiple redefinitions of a data field, or multiple levels of redefined fields.
compression	Reduce the byte size required to transmit data to and from the client and server. Data is compressed when it is sent and then decompressed when it reaches its destination. This reduces the size of data transmissions and improves network performance.
Construct Spectrum	Application consisting of a client and server component. The client component is a Construct Spectrum application running in Visual Basic. The server component is a set of subprograms accessed remotely by the client component.
Construct Spectrum Administration subsystem	Mainframe subsystem used to maintain and query tables defining Construct Spectrum application services and security. To access the Administration subsystem, enter “menu” from the SYSPEC library.
<b>Note:</b>	You must have security privileges to access this subsystem.
database record	Logical view of database information. A database record can be comprised of one or more logically related database files or tables. Construct Spectrum represents database information in parameter data areas (PDAs).
DBID	Acronym for database ID, which is the number identifying the server database containing application components.
deployment	Movement of an application from a development environment to a production environment.
dialog	GUI form (window) running on the client.
dispatcher or dispatch service	Server component used to broker communications between server components and client framework components. See also <b>Spectrum dispatch service</b> , page 100.
domain	Entity that defines a collection of related business objects (for example, Test, Admin, and Sales).

<b>Term</b>	<b>Definition (continued)</b>
double-byte character set (DBCS)	Related collection of characters in some non-Latin languages that require two bytes to display.
download data	Transfer (copy) modules from the server to the client.
encapsulation	Technique in object-oriented programming in which the internal implementation details of an object are hidden from users of the object. Methods control how the object data is manipulated. Encapsulation allows internal implementations to change without affecting the way an object is used externally.
encryption	Encoding data so it is unusable for individuals without access to the decryption algorithms. Construct Spectrum allows you to encrypt sensitive data, such as payroll information, during network transmission. Data is decrypted when it reaches its destination.
Entire Broker service settings	Collection of Entire Broker-related parameters, including Entire Broker ID, server class, server name, and service.
Entire Broker stub	Entire Broker DLL on a Windows platform.
event	Action recognized by an object, such as pressing a key or clicking a mouse. You write code to respond to events.
event handler	Custom code that reacts to events raised within the Web Service Engine (WSE).
field	Component of a database record. The term also refers to areas on a panel in which values are entered.
FNR	Acronym for the file number that identifies a specific server database file containing application components.
foreign key	Key field pointing to a record in an external file. For example, the demo application has an Order file containing a foreign key to the Warehouse field in the Warehouse file. Foreign keys can be set up with a browse function, enabling users to search for and select values.

Term	Definition (continued)
form	<p>Dialog (window) that acts as the interface for an application. You add controls and graphics to a form to create the effect you want. Construct Spectrum supplies forms in the client framework and generates form modules for business object maintenance dialogs.</p> <p>When you run a project, forms are compiled into GUI windows that the user interacts with while using the application. Some forms, such as the generic BrowseDialog form, are dynamically configured at runtime by the client framework to alter the look of the form.</p> <p>Form definitions are saved in files with the extension .frm.</p>
form section	Portion of a web page containing a block of related information.
framework template	Structure or container supplied for applications. These customizable templates include header, footer, navigation bar, messages area, and constants.
generate	Process of producing code from specifications.
generated module	Generated component for either the client or server portion of an application. Generated server modules include Natural subprograms, subprogram proxies, and parameter data areas. Generated client modules include object factories, windows, and maintenance objects.
generation data cache	In-memory hierarchical data structure that allows you to quickly retrieve stored generation data.
grid	<p>Displays 2-dimensional data for a client/server application in a table format.</p> <p>One-dimensional data shows one type of data, such as a phone number, name, or quantity. Two-dimensional data shows additional information in a grid or table. For example, the detail lines on a customer order can be displayed in a grid with each grid row corresponding to a unique line item. Each column in the grid corresponds to a discrete piece of information about the line, such as an item name, price, or quantity.</p>

Term	Definition (continued)
grid control	GUI control that displays related information in a table format. For example, purchase order line items can be displayed in a grid. The grid control supplied with Construct Spectrum sizes itself to the minimal width required to display all grid components. You can configure the supplied grid control as desired.
group	Collection of users defined in the Construct Spectrum Administration subsystem.
GUI	Acronym for graphical user interface.
GUI control override	Use Predict keywords to force a GUI control derivation. See also <b>keyword</b> , page 96.
host	See <b>server</b> , page 99.
HTML fragment	Portion of HTML that is not a complete web page.
HTML template	HTML that may contain replacement tags, which are dynamically exchanged for content or nested HTML templates at runtime.
http request	Parameterized list of named value pairs sent by a browser client to a web application.
instantiation	Process of creating an instance of a class. The result is an object.
internationalization	Adapting an application to make it easy to localize. See also <b>localization</b> , page 96.
keyword	Predict metadata type that acts as a label or identifier.
Level 1 data block	Level one field or structure and its subfields in a Natural parameter data area (PDA).
library image file (LIF)	File that defines Natural definitions used by the Spectrum Dispatch Client.
LIF definitions module	BAS module in a Visual Basic project containing the definitions for application services, parameter data areas, and subprograms.
localization	Process of translating and adapting a software product for use in a different language or country.

Term	Definition (continued)
lookups	Return descriptive information when a user requests a browse window or enters a value in a foreign key field in a maintenance window. For example, assume Warehouse Number is a foreign key field in the Order window and Warehouse Name is a descriptive field attached to the foreign key value. When a user enters a valid warehouse number, the lookup returns the name of the warehouse to display.
maintenance dialog	GUI dialog (window) from which a user can perform one or more actions on a business object. For example, a Customer Order object can be represented in a maintenance window. Using this window, an authorized user can add, delete, or update customer order information.
maintenance window	See <b>maintenance dialog</b> above.
MDI child	Dialog (window) opened from an MDI parent dialog in a client/server application. For example, the Order maintenance window in the demo application is an MDI child to the MDI frame window.
MDI frame	Standard Visual Basic MDI frame supplied with the Construct Spectrum client framework.
MDI parent	MDI dialog (window) from which other dialogs are opened and displayed in a client/server application. The MDI frame supplied with the client framework is an MDI parent.
menu	<p>On a mainframe server, a panel or window listing available functions. To access a function, users enter a value in an input field or move the cursor to a value and press Enter.</p> <p>In Windows, a navigation bar listing the available functions. To access a function, users select an option from the menu using the cursor or a keystroke combination.</p>
menu bar	Displays the menus available for user selection. By default, Construct Spectrum client/server applications contain File, Edit, Actions, Window, and Help menus on the menu bar, each containing standard menu commands.
metadata	Information about data. Metadata describes how physical data is formatted and interrelated. It includes descriptions of data elements, data files, and relationships between data entities. Typically, metadata is maintained in a repository known as a data dictionary, such as Predict.

Term	Definition (continued)
method	Procedure that operates on an object and is implemented internally by the object. For example, the Update method updates a Customer Order object after changes to the order information.
model	Template used to generate modules. Each model contains one or more specification panels. Using these panels, you can specify parameters for a desired module and then generate the corresponding code. Natural Construct provides numerous models, including the Object-Maint-Subp and Subprogram-Proxy models.
module	Single application component, such as a hand-coded Natural program, subprogram, or data area or a Natural Construct-generated program, subprogram, data area, or subprogram proxy.
multi-level security	Security you can define at a high level or at a detailed level affecting many objects. For example, you can apply multi-level security to domains, objects, and methods.
multiple-document interface (MDI)	Microsoft Windows paradigm for presenting windows whereby a parent window can encompass one or more child windows. See also <b>MDI child</b> , page 97, and <b>MDI parent</b> , page 97.
Natural Construct nucleus	Sophisticated driver program that invokes the model subprograms at the appropriate time in the generation process and performs functions common to all models, such as opening windows and performing PF-key functions. The nucleus communicates with the model subprograms through standard parameter data areas (PDAs). These PDAs contain fields assigned by Natural Construct, as well as fields required by a model.
Natural Debugging facility	Utility available in a Natural environment to help you locate and analyse logic errors. To access the facility, use the Invoke Proxy function in the Construct Spectrum Administration subsystem. The subprogram proxy sets up an online environment that simulates the client/server environment and allows you to use all the features of the Natural Debugging facility.
navigation bar	Menu bar on a web page containing links to other pages or actions.
node	Individual computer or, occasionally, another type of machine in a network.

Term	Definition (continued)
nucleus	See <b>Natural Construct nucleus</b> , page 98.
object	Any application component, such as a form or record. A business object is a group of services related to a common business entity, such as Customer, Order, or Department.
overflow condition	Situation where there are more fields than can be displayed on a dialog.
parent model	Super model that collects parameters for child models and generates specifications.
ping	Request sent to a service to determine whether the service is running.
platform	Piece of equipment that, together with its operating system, serves as a base on which you can build other systems. For example, an MVS mainframe computer can serve as a platform for a large accounting system.
project	Collection of files used to build an application in Visual Basic.
property	Characteristic of an object, such as size, caption, or color. In Construct Spectrum, it refers to the data settings or attributes for an object in Visual Basic.
regular expression	String-searching criteria to scan for a specified text string and, optionally, substitute another string. For example, a regular expression can search for “Construct Spectrum” and replace each occurrence with “Spectrum” alone.
remote call	Communication with an object residing in a different location, such as a server.
resource	Text or binary value that can be localized. See also <b>localization</b> , page 96.
security cache	File used to store recently-accessed security data.
server	Computer that provides services to another computer (called a client) and responds to requests for services. On multitasking machines, a process that provides services to another process is called a server.
server application	Application that runs on a server machine.
service	Software service that runs on a server. Several services can run on one server.

<b>Term</b>	<b>Definition (continued)</b>
service exit	Exposed exit routine called by the Spectrum dispatch service; it can be replaced by a user-supplied routine.
service log	File used to store service log data.
shutdown	Command sent to a service to terminate the service.
Simple object access protocol (SOAP)	XML-based standard communication protocol to interact with Web services.
software development kit (SDK)	See <b>toolkit</b> , page 101.
Spectrum client/server application	Application created using the Construct Spectrum wizards and add-ins. Users access mainframe business functions and data through a Visual Basic component running on a Windows platform.
Spectrum Control record	Record that is created daily and contains system control and statistic data for a Spectrum dispatch service.
Spectrum Dispatch Client (SDC)	Provides the Construct Spectrum data exchange, which facilitates calls from a client to Natural subprograms running on a server.
Spectrum dispatch service	Middleware component that encapsulates broker calls on the server, provides directory services, enforces security, and invokes backend Natural services.
Spectrum security service	Component of the Construct Spectrum Administration subsystem that controls access to application libraries, objects, and methods.
Spectrum Service Manager	Client tool supplied with Construct Spectrum that allows you to specify which Spectrum services the client uses to communicate with the server.
Spectrum service settings	Collection of parameters used to configure a Spectrum service.
Spectrum web application	Application created using the Construct Spectrum wizards and add-ins. It allows users to access mainframe business functions and data from a web browser.
Spectrum web framework	Group of Visual Basic modules and classes that collaborate to dynamically generate web pages.

Term	Definition (continued)
Status bar	Area that displays status information about a selected item, application, or business object in a client/server application. It contains sections for a message, status indicators, and the current date and time. Status bars are also displayed at the bottom of an MDI form.
steplib chain	Hierarchy of Natural libraries that determines the location from which modules are executed.
subprogram proxy	Natural subprogram called by a Spectrum dispatch service to translate data formats between the client and a Natural subprogram on the server. Each subprogram requires a subprogram proxy, which allows Construct Spectrum to provide a common interface to any subprogram.
target module	See <b>target subprogram</b> , page 101.
target object	See <b>target subprogram</b> , page 101.
target subprogram	Any Natural subprogram.
template parser	Class used to parse HTML or other templates.
token (Spectrum security)	Unique, system-generated, identification number used in place of a user ID and password for multiple calls. Users can logon with their user ID and password and then request a token to use for later calls.
toolbar	Bar that provides quick access to commonly used commands in an application. A user clicks the appropriate toolbar button to perform the action it represents. Any action that can be performed from a toolbar can also be invoked from a menu.
toolbar button	Icon on a toolbar that allows users to perform an action.
toolkit	Set of related and reusable classes that provide general-purpose functionality. An application incorporates classes from one or more toolkits.  Toolkits, or software development kits (SDKs), emphasize code reuse and are the object-oriented equivalent of subroutine libraries. For example, a toolkit can be a collection of classes for lists, associative tables, or stacks.
trace options	Options that specify how to trace messages sent between the client and server.
upload data	Transfer modules from the client to the server.

Term	Definition (continued)
variant	<p>Visual Basic term identifying a late-binding data type. Variants allow Construct Spectrum subroutines or functions to accept different types of data. The exact type is determined when they receive the value in Visual Basic.</p>
verification rule	<p>Predict-defined business rules that are implemented in the object subprogram on the server and the maintenance object on the client. They also provide default values for derived fields represented by GUI controls, such as check boxes, option buttons, or drop-down combo boxes.</p> <p>You can use verification rules to force users to make a selection based on one or more choices. For example, if an application has an input field for the state name, you can attach a verification rule to the field in Predict so that only valid state names are accepted.</p>
Web service	Service to expose data or functionality to the intra/internet.
Web Service Engine (WSE)	Core DLL supplied with Construct Spectrum that handles Spectrum Web service requests from IIS (*.sws requests).
Web services root	<p>Main/root directory (folder) for Spectrum Web services. By default, the name of the Web services root directory is SpectrumWebServices. For example:</p> <pre data-bbox="655 1115 1222 1178">http://machine name/SpectrumWebServices C:\inetpub\wwwroot\SpectrumWebServices</pre>
<b>Note:</b>	For examples used in this documentation, assume the Web services root is SpectrumWebServices.
wildcard	<p>Character or symbol that qualifies a selection, such as “*”, “&lt;”, or “&gt;”. For example, using a value followed by an asterisk (*) indicates a range of file names beginning with that value. To list all modules that begin with “Maint”, enter “Maint*” as the selection criteria.</p>
WSDL (Web Service Definition Language)	XML document used to describe a Web service.
XML (Extensible Markup Language)	Industry-accepted standard language used to transmit data.
XML extract	<p>Extract information from Predict and other sources, which is stored on the client as metadata in XML format. This includes information about business objects, as well as the formatting used by wizards to build application components. See also <b>metadata</b>, page 97.</p>

# INDEX

## Symbols

- .NET
  - mapping .sws requests to WSE, 72
- .sws
  - mapping, 72

## A

- Accessing
  - Configuration editor, 16
- Active server page (ASP) script
  - definition of, 91
- ActiveX business object (ABO)
  - definition of, 91
- ActiveX DLL
  - definition of, 91
- Advanced Options window
  - Linking tab, 58
  - ValueList tab, 60
- AfterCallnat event
  - description, 75
- AfterCallnatEventArgs event
  - arguments, 76
- Application library
  - definition of, 91
- Application service definition
  - definition of, 91
- Application services
  - definition of, 91
- Architecture
  - definition of, 91
- Arguments
  - event, 75
- Assembly
  - definition of, 91
- Attributes
  - excluding during an update, 84

## B

- BDT class
  - definition of, 92
- BDT Configuration File
  - Configuration editor, 17
- BDT controller class
  - definition of, 92
- BDT controller object
  - definition of, 92
- BDT modifier
  - definition of, 92
- BDT modifiers
  - description, 37
  - separating multiple, 37
- BDT node
  - BDTs.xml file, 80
- BDT procedure
  - definition of, 92
- BDTAlpha
  - modifiers, 38
- BDTBoolean
  - modifiers, 39
- BDTCurrency
  - modifiers, 40
- BDTDateTime
  - modifiers, 40
- BDTHexByte
  - modifiers, 40
- BDTNumeric
  - modifiers, 41
- BDTs
  - Web.config file, 71
- BDTs.xml file
  - XML nodes, 80
- BeforeCallnat event
  - description, 74
- BeforeCallnatEventArgs event
  - arguments, 76

- Browse data cache
  - definition of, 91
- Browse dialog
  - definition of, 91
- Browse for Module window
  - description, 26
- Browse process
  - definition of, 92
- Browse window
  - definition of, 92
- Browser
  - requirements
    - Web Application Builder, 47
- Business data types (BDTs)
  - creating custom, 79
  - definition of, 92
  - description, 37
  - overriding defaults
    - Web Service wizard, 86
  - supplied, 38
- C**
- Cache
  - definition of, 92
- Cardinality
  - definition of, 92
- CDAOBJ2 PDA
  - structure, 85
- CheckBox
  - HTML control
    - description, 57
- Child model
  - definition of, 93
- Class
  - event handler
    - creating custom, 77
- Client application
  - definition of, 93
- Code
  - adding custom, 74
  - custom
    - modifying Web.config file, 78
- Code Frame
  - Configuration editor, 17
- Combo Box
  - HTML control
    - description, 57
- Combo box
  - adding list of valid values, 60
- Complex redefine
  - definition of, 93
- Compression
  - definition of, 93
- Configuration editor
  - accessing, 16
  - settings
    - runtime component, 16
- Configure Browse Keys window
  - Web Page wizard, 63
- Configure HTML window
  - Advanced Options window
    - Linking tab, 58
    - ValueList tab, 60
  - Web Page wizard, 56
- Configure Menu window
  - Menu wizard, 64
- Configure Methods window
  - Web Page wizard, 61
  - Web Service wizard, 27
- Configure Page window
  - Web Page wizard, 53
- Configure PDAs window
  - Web Service wizard, 29
- Configure Tertiary Display window
  - Web Page wizard, 62
- Configure Web Service window
  - Web Service wizard, 31
- Configuring
  - runtime component, 16
- Construct Spectrum
  - definition of, 93
- Construct Spectrum Administration subsystem
  - definition of, 93
- Construct Spectrum SDK for Microsoft .NET framework
  - architecture, 3
  - configuration settings, 15
  - overview, 2
  - prerequisite knowledge, 5

- structure of documentation, 6
- supplied wizards, 4
- using the SDK, 19
- Conventions
  - document, 7
  - naming
    - for examples, 8
  - typographical
    - used in this guide, 7
- Create Application window
  - description, 49
- Create New Service window
  - Web Service wizard, 33
- Creating
  - web application, 47–48
- Customizing
  - adding custom code, 74
- D**
- Data updates
  - tips and techniques, 84
- Database record
  - definition of, 93
- DataType node
  - BDTs.xml file, 81
- DBID
  - definition of, 93
- Default Web Service Server
  - Configuration editor, 17
- DefaultLanguage
  - Web.config file, 70
- DELETE action
  - sample input, 88
- Deployment
  - definition of, 93
- Developing Web Applications
  - layout, 6
- Development
  - environment, 14
    - changing dispatcher, 87
    - setting up steplib, 86
- Dialog
  - definition of, 93
- Directory
  - Web Services
    - contents, 68
- Dispatch service
  - definition of, 93
- Dispatcher
  - changing, 87
  - definition of, 93
  - runtime environment
    - setting, 18
  - setting for development environment, 14
  - setting for runtime environment, 14
  - Web.config file, 71
- Display
  - HTML control
    - description, 57
- Display Width
  - Configure HTML window, 56
- Documentation
  - related, 9
- Domain
  - definition of, 93
- Double-byte character set (DBCS)
  - definition of, 94
- Download data
  - definition of, 94
- E**
- Encapsulation
  - definition of, 94
- Encryption
  - definition of, 94
- Entire Broker service settings
  - definition of, 94
- Entire Broker stub
  - definition of, 94
- Environment
  - development, 14
  - runtime, 14
- ErrorOptions
  - Web.config file, 70
- Errors
  - converting BDT, 90
  - correcting during an update, 90
- Event
  - definition of, 94
- Event handler
  - definition of, 94
- Event handler class
  - creating a custom, 77

- EventHandler
  - Web.config file, 71
- Events
  - arguments, 75
  - from WSE
    - how handled, 71
  - hierarchy, 71
  - supplied Spectrum
    - arguments, 75
  - WSE, 74
- Examples
  - directory
    - web applications, 88
    - Web services, 88
- ExceptionThrown event
  - description, 75
- ExceptionThrownEventArgs event
  - arguments, 77
- EXISTS action
  - sample input, 88
- Expression
  - regular
    - definition of, 99

**F**

- Field
  - definition of, 94
- Field Length
  - Configure HTML window, 56
- Field override
  - setting
    - Web Service wizard, 28
- FNR
  - definition of, 94
- Foreign key
  - definition of, 94
- Form
  - definition of, 95
- Form section
  - definition of, 95
- Framework template
  - definition of, 95

**G**

- Generate
  - definition of, 95
- Generated module
  - definition of, 95
- Generating
  - menu, 64
  - subprogram proxy, 20
  - web page, 51
  - Web service, 23
- Generation data cache
  - definition of, 95
- Generation Status window
  - Web Page wizard, 55
  - Web Service wizard, 33
    - after regeneration, 36
- GET action
  - sample input, 88
- Grid
  - definition of, 95
- Grid control
  - definition of, 96
- Group
  - definition of, 96
- GUI
  - definition of, 96
- GUI control override
  - definition of, 96

**H**

- Headings
  - defining for tertiary grid, 62
- Hidden
  - HTML control
    - description, 57
- Host
  - definition of, 96
- HTML controls
  - description, 57
- HTML fragment
  - definition of, 96
- HTML template
  - definition of, 96

HTTP Application Root  
Configuration editor, 17  
http request  
definition of, 96  
HTTP Web Services Root  
Configuration editor, 17

## I

IIS  
mapping .sws files to .NET framework,  
72  
Inheritance  
Web.config settings, 71  
Initialization  
local  
Web Service wizard, 31  
INITIALIZE action  
sample input, 89  
Input  
SOAP actions  
samples, 88  
validating, 42  
Instantiation  
definition of, 96  
Internationalization  
definition of, 96  
Invoking  
SOAP Client, 44  
Web Service wizard, 24

## K

Keyword  
definition of, 96

## L

Level 1 data block  
definition of, 96  
Library image file (LIF)  
definition of, 96  
LIF definitions module  
definition of, 96  
Linking tab  
Advanced Options window, 58  
Listing  
methods, 87

Local Application Root  
Configuration editor, 17  
Local Web Services Root  
Configuration editor, 17  
Localization  
definition of, 96  
Lookups  
definition of, 97

## M

Mainframe  
confirming Spectrum data, 21  
Maintain Application Service Definitions  
panel  
verifying Spectrum data, 22  
Maintenance dialog  
definition of, 97  
Maintenance window  
definition of, 97  
Mapping  
.sws, 72  
MDI child  
definition of, 97  
MDI frame  
definition of, 97  
MDI parent  
definition of, 97  
Menu  
definition of, 97  
generating for a web application, 64  
regenerating, 65  
Menu bar  
definition of, 97  
Menu wizard  
Configure Menu window, 64  
generating a menu, 64  
regenerating a menu, 65  
Message numbers  
using  
Web Service wizard, 29  
MessageDatabase  
Web.config file, 69  
Metadata  
definition of, 97  
Method  
definition of, 98

Methods  
listing, 87  
permitted on the client  
setting, 27

Model  
definition of, 98

Modifiers  
BDT, 37  
CASE, 38  
DEC, 41  
DisplayTimeFormat, 40  
EM  
BDTBoolean, 39  
BDTNumeric, 41  
GS, 41  
HexSuffix, 40  
ISDate, 40  
O/I, 39  
ROUND, 41  
SCIENTIFIC, 41  
SIGN, 41  
TRIM, 38  
XML, 40  
ZERO  
BDTCurrency, 40  
BDTNumeric, 41

Modifying  
Web.config file, 68

Module  
definition of, 98

More column  
Configure HTML window, 59–60

More field  
Configure HTML window, 56

Multi-level security  
definition of, 98

Multilingual  
Web services, 29

Multiple-document interface (MDI)  
definition of, 98

## N

Namespace name  
verifying, 84

Natural Construct nucleus  
definition of, 98

Natural Debugging facility  
definition of, 98

Natural subprogram  
preparing for client/server access, 20

Navigation bar  
definition of, 98

NEXT action  
sample input, 88

Node  
definition of, 98

Nucleus  
definition of, 99

## O

Object  
definition of, 99

Overflow condition  
definition of, 99

Override  
field  
setting for Web Service wizard, 28

## P

Parent model  
definition of, 99

PDAs  
configuring  
Web Service wizard, 29

Ping  
definition of, 99

Platform  
definition of, 99

Predict keywords  
BDTs, 38

Project  
definition of, 99

Project Templates  
Configuration editor, 17

Property  
definition of, 99

Proxy  
subprogram  
generating, 20

**R**

Radio button  
 adding list of valid values, 60

RadioButton  
 HTML control  
 description, 57

Ready to Generate window  
 Web Page wizard, 54  
 Web Service wizard, 32

Regenerating  
 menu, 65  
 web page, 63  
 Web service, 35  
 making changes accessible to users,  
 36

Regular expression  
 definition of, 99

Remote call  
 definition of, 99

Resource  
 definition of, 99

RetrieveMetaData  
 Web.config file, 69

Root directory  
 Web services  
 adding, 82

Runtime  
 environment, 14  
 changing dispatcher, 87  
 setting dispatcher, 18  
 setting up steplibs, 86  
 steplib chain, 14

Runtime component  
 configuring, 16

**S**

Saving  
 Web service, 24

Security cache  
 definition of, 99

Security mode  
 setting for web application, 49

Security token  
 creating request for, 73

Select Subprogram window  
 Web Service wizard, 25

Select Web Service window  
 Web Page wizard, 52

Server  
 definition of, 99

Server application  
 definition of, 99

Service  
 definition of, 99

Service exit  
 definition of, 100

Service log  
 definition of, 100

Setting  
 zero suppression, 90

Shutdown  
 definition of, 100

SOAP  
 definition of, 100

SOAP Client  
 creating a request for security token, 73  
 invoking, 44  
 testing a Web service, 43

SOAP messages  
 sample, 88

Software development kit (SDK)  
 definition of, 100

Spectrum client/server application  
 definition of, 100

Spectrum Control record  
 definition of, 100

Spectrum data  
 confirming on mainframe, 21  
 verifying  
 Maintain Application Service  
 Definitions panel, 22

Spectrum Dispatch Client (SDC)  
 definition of, 100

Spectrum dispatch service  
 definition of, 100

Spectrum security service  
 definition of, 100

Spectrum security token  
 creating a request for, 73  
 definition of, 101

Spectrum Service Manager  
 definition of, 100

- Spectrum service settings
  - definition of, 100
- Spectrum web application
  - definition of, 100
- Spectrum web framework
  - definition of, 100
- Spectrum Web services
  - configuring
    - runtime component, 16
- SpectrumSecurity
  - Web.config file, 70
- SPEEventArgs
  - supplied Spectrum events
    - arguments, 75
- Start window
  - Web Page wizard, 51
  - Web Service wizard, 24
- Status bar
  - definition of, 101
- Steplib chain
  - definition of, 101
  - runtime environment, 14
- Steplibs
  - setting up, 86
- STORE action
  - sample input, 89
- Subprogram proxy
  - definition of, 101
  - generating, 20

## T

- Target module
  - definition of, 101
- Target object
  - definition of, 101
- Target subprogram
  - definition of, 101
- Template parser
  - definition of, 101
- Tertiary grid
  - defining headings, 62
- Testing
  - Web service, 43

- TextArea
  - HTML control
    - description, 57
- TextBox
  - HTML control
    - description, 57
- Token
  - creating request for, 73
  - definition of, 101
- ToolBar
  - definition of, 101
- ToolBar button
  - definition of, 101
- Toolkit
  - definition of, 101
- Trace options
  - definition of, 101
- Troubleshooting
  - tips and techniques, 83
- Typographical conventions
  - used in this guide, 7

## U

- Update
  - correcting errors, 90
- UPDATE action
  - sample input, 89
- Upload data
  - definition of, 101

## V

- ValidateUser event
  - description, 74
- ValidateUserEventArgs event
  - arguments, 76
- Validating
  - input, 42
- ValueList tab
  - Advanced Options window, 60
- Variant
  - definition of, 102
- Verification rule
  - definition of, 102

**W**

## Web application

- architecture, 47
- creating, 47–48
- generating
  - menu, 64
- setting security mode, 49

## Web Application Builder

- browser requirements, 47
- description, 47

## Web Application Builder window

- description, 48

## Web page

- generating, 51
- regenerating, 63

## Web Page wizard

- Configure Browse Keys window, 63
- Configure HTML window, 56
- Configure Methods window, 61
- Configure Page window, 53
- Configure Tertiary Display window, 62
- generating web page, 51
- Generation Status window, 55
- Ready to Generate window, 54
- regenerating a web page, 63
- Select Web Service window, 52
- Start window, 51

## Web service

- adding custom code, 74
- configuring, 31
- contents of Web.config file, 68
- definition of, 102
- generating, 23
- making changes accessible to users, 36
- regenerating, 35
- retrieving previously saved specifications, 25
- saving, 24
- testing, 43

## Web Service Engine (WSE)

- definition of, 102
- events, 74
- handling events, 71

## Web Service wizard

- Configure Methods window, 27
- Configure PDAs window, 29
- Configure Web Service window, 31

Create New Service window, 33

Generation Status window, 33

invoking, 24

Ready to Generate window, 32

Select Subprogram window, 25

Start window, 24

## Web services

- adding a root directory, 82
- creating multilingual, 29

## Web services root

definition of, 102

## Web services root directory

contents, 68

## Web.config file

- global, 68
- inheritance, 71
- inheriting events, 71
- local, 68
- modifying, 68
  - for custom code, 78

## Wildcard

definition of, 102

## Wizards

- supplied with the Construct Spectrum SDK for Microsoft .NET framework, 4

## WSDL

definition of, 102

## WSE

- Web service engine
  - definition of, 102

**X**

## XML

definition of, 102

## XML extract

definition of, 102

## XML nodes

BDTs.xml file, 80

## XML tag name

defining, 30

**Z**

## Zero suppression

setting, 90