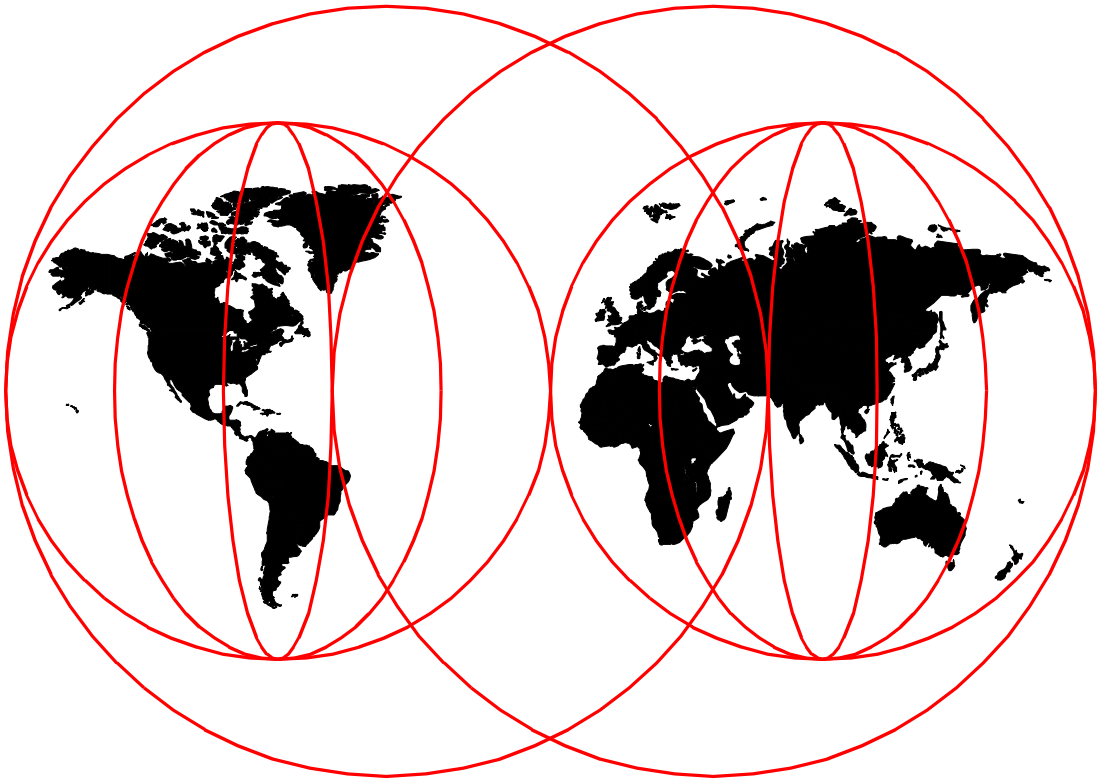
e-business Solutions for VSE/ESA

Annegret Ackel, Roger Brooks, Peter von Hirschfeld, Erhard Hoehn,
G.M. (Jerry) Johnston, Anette Stolvoort, Monika Zimmermann



International Technical Support Organization

www.redbooks.ibm.com

SG24-5662-00

International Technical Support Organization

SG24-5662-00

e-business Solutions for VSE/ESA

May 2000

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix C, “Special notices” on page 171.

First Edition (May 2000)

This edition applies to VSE/ESA Version 2 Release 3, Program Number 5690-VSE.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HYJ Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures vii

Tables xi

Preface xiii
The team that wrote this redbook xiii
Comments welcome xiv

Chapter 1. Introduction 1
1.1 e-business options for VSE customers 1
1.1.1 Two-tier versus 3-tier 1
1.1.2 Two-tier implementations 3
1.1.3 Three-tier implementations on the IBM e-business Application Framework 5

Chapter 2. The Friendly Foods scenario 7
2.1 The Friendly Foods e-business 7
2.2 How Friendly Foods became an e-business 8
2.2.1 Consumer marketing 8
2.2.2 Account self-service and account tracking 9
2.2.3 Integration with service providers 9
2.2.4 Business integration with suppliers 9
2.3 e-business implementation at Friendly Foods 9
2.3.1 WebSphere Application Server 11
2.3.2 VisualAge for Java 12
2.3.3 WebSphere Studio 12
2.4 The ITSO system setup 13

Chapter 3. Consumer marketing 15
3.1 The User Interface 16
3.2 Installation and configuration 18
3.2.1 DB2 Server for VSE & VM V7.1 (beta) 18
3.2.2 DB2 Connect and DB2 Client Application Enabler 19
3.3 The Friendly Foods Web presence 23
3.3.1 The static entry Web page 23
3.3.2 Creating the dynamic parts 24
3.4 Consumer marketing -- summary 33

Chapter 4. Account self-service and account tracking 35
4.1 WebSphere Studio V3 35
4.1.1 Installation and configuration 35
4.2 The user interface -- Java Server Pages (JSP) considerations 35
4.3 Implementation 36
4.3.1 Creating JSPs with WebSphere Studio 37
4.3.2 Final servlet pages created by NetObjects Fusion 50
4.3.3 Integrating the pages with NetObjects Fusion 53
4.4 Account self-service and account tracking -- summary 58

Chapter 5. Integration with service providers 59
5.1 SecureWay Host On-Demand server 59
5.2 Call center implementation 60
5.3 The Friendly Foods Host On-Demand implementation 62

5.4 Integration with service providers -- summary 66

Chapter 6. Business integration with suppliers 67
6.1 Products to be considered 69
6.1.1 MQSeries. 69
6.1.2 MQSeries Integrator 69
6.1.3 CICS Transaction Gateway 70
6.2 MQSeries implementation 70
6.2.1 Definitions for MQSeries objects 71
6.2.2 Implementation at Friendly Foods 81
6.3 Using MQSeries plus MQSeries Integrator 90
6.3.1 Implementing the application with MQSeries Integrator 91
6.4 Implementation of the CICS Transaction Gateway 97
6.4.1 Introduction to the CICS Transaction Gateway. 98
6.4.2 Setting up the connection 100
6.4.3 Installation and customization of the CICS Transaction Gateway . . 112
6.4.4 Establish the APPC connection 115
6.4.5 Implementing the CICS-based application on VSE/ESA. 116
6.4.6 Final application panel created with VisualAge for Java 117
6.5 Business integration with suppliers -- summary 123

Chapter 7. Options for a 2-tier solution 125
7.1 Web/VSE®-Host 125
7.1.1 Installation 126
7.1.2 Customization 126
7.1.3 Automatically generated 3270 Web pages 127
7.1.4 Creating Web pages with HTML templates 128
7.1.5 Identifying 3270 screens and customizing Web pages 130
7.1.6 Security considerations with Web/VSE-Host 134
7.1.7 Web/VSE-Host and Friendly Foods 134
7.2 IpServer. 134
7.2.1 Installation and customization 135
7.2.2 Running the sample CGI programs 135
7.3 Use modern tools to build business Web sites hosted on VSE/ESA. . . . 140
7.3.1 Setup to work with NetObjects Fusion 140
7.3.2 Designing Web pages with NetObjects Fusion 4.0 140
7.3.3 Using dynamic pages with IpServer 141
7.3.4 Getting ready to publish the site to VSE/ESA. 149

Chapter 8. Summary and outlook 153
8.1 Improve network security 153
8.2 Upgrade to VSE/ESA V2.4 and CICS Transaction Server for VSE/ESA . 153
8.2.1 VSE/ESA V2.4 security 154
8.2.2 Exploiting CICS Web Support and 3270 bridge 154
8.2.3 Additional considerations 154
8.3 New business opportunities with e-commerce 155
8.3.1 Business concept. 155
8.3.2 Implementation 155
8.3.3 Additional opportunities 156
8.4 Improved customer service with pervasive computing 156
8.4.1 Business concept. 156
8.4.2 Implementation 157
8.5 Conclusion 157

Appendix A. e-business technologies 159
A.1 User Interface technologies. 159
A.1.1 HTML 159
A.1.2 JavaScript 159
A.1.3 Java applet 159
A.2 Server-side logic 160
A.2.1 Java servlets 160
A.2.2 Java Server Pages (JSPs) 161
A.3 e-business terms and Java-related definitions 161
A.3.1 Java. 161
A.3.2 JavaBeans 161
A.3.3 Enterprise JavaBeans 161
A.3.4 Java Database Connectivity (JDBC) 162

Appendix B. Sample programs 163

Appendix C. Special notices 171

Appendix D. Related publications 175
D.1 IBM Redbooks. 175
D.2 IBM Redbooks collections 175
D.3 Other resources 175
D.4 Referenced Web sites 176

How to get IBM Redbooks 177
IBM Redbooks fax order form 178

Abbreviations and acronyms 179

Index 181

IBM Redbooks review 185

Figures

1. VSE and the IBM e-business Application Framework 3
2. Two-tier implementations 4
3. Connector architecture 5
4. Friendly Foods before they became an e-business 8
5. Friendly Foods IT environment 11
6. WebSphere Application Server editions 12
7. ITSO system setup 14
8. DB2 system configuration 16
9. DB2's JDBC applet implementation 17
10. VSE DB2 startup job 18
11. Console output of DB2 startup 19
12. Columns of DB2 table FFSTORES 19
13. DB2 Connect installation 20
14. DB2 client configuration -- TCP/IP 21
15. DB2 client configuration -- ODBC 21
16. Connect to DB2 Database 22
17. Successful connection message 22
18. The Friendly Foods entry home page 23
19. The Friendly Foods Web site hierarchy 24
20. BeanBuilder entry screen 25
21. BeanBuilder -- customize the database 25
22. BeanBuilder -- specify the database query 27
23. BeanBuilder - select DB2 table 27
24. BeanBuilder -- select columns 28
25. BeanBuilder -- generated SQL code 28
26. BeanBuilder -- workspace with database list 29
27. BeanBuilder - test 30
28. BeanBuilder -- Publish Wizard 30
29. BeanBuilder -- Publish Wizard Bean panel 31
30. BeanBuilder -- Publish Wizard Local panel 31
31. NetObjectsFusion -- include the generated bean 32
32. NetObjectsFusion -- the generated bean is included 32
33. Final Friendly Foods Web page 33
34. Implementation of JSP servlets working with DB2 36
35. WebSphere Studio: create a project 38
36. WebSphere Studio: using the SQL wizard 38
37. WebSphere Studio -- SQL wizard: defining the output colums 39
38. WebSphere Studio -- SQL wizard: defining the conditions for the query 40
39. WebSphere Studio: using the Database wizard 41
40. WebSphere Studio: Project View - focus on input page 42
41. WebSphere Studio: Project View -- focus on output page 43
42. WebSphere Studio-generated servlet control file 43
43. WebSphere Studio: setting the publishing options 44
44. WebSphere Studio: test the generated input page 45
45. WebSphere Studio: viewing the generated result page 46
46. WebSphere Studio: using the Page Designer to modify the result page 47
47. WebSphere Studio Page Designer: final layout of the result page 48
48. WebSphere Studio: Project View - complete account self service project 49
49. Friendly Foods account self-service Login page 50
50. Friendly Foods account self-service Main Menu 51

51. Friendly Foods account self-service Account Status page 52
52. Friendly Foods account self-service update Order Mix page 52
53. Control flow between HTML pages, servlet, Database Bean and JSP pages .53
54. Design for the account self-service Login page with NetObjects Fusion 55
55. Design for the account self-service Main Menu page with NetObjects Fusion .56
56. Customize page names in Fusion. 57
57. Host On-Demand -- configurations 60
58. Initial IPCC Call Center Web page 61
59. Friendly Foods IPCC Call Center page. 61
60. Host on-Demand screen -- change order mix. 62
61. Host On-Demand applet file - HODStart.html 64
62. Linking to HODStart.HTML 66
63. Additional products for integration with suppliers 68
64. MQSeries environment at Friendly Foods. 71
65. MQPECHO sample program - code fragment 81
66. A more detailed look at the application parts 81
67. Visual Composition of the user interface applet part. 83
68. Method to build a URL to call a servlet 84
69. User Interface of the purchasing application. 85
70. Servlet handling MQSeries queues -- 1 86
71. Servlet handling MQSeries queues -- 2 87
72. Excerpt of the sendMessage method in the servlet dealing with MQSeries . . .88
73. Configuring servlets in WebSphere Application Server 89
74. Queue handling without and with MQSeries Integrator. 90
75. MQSeries Integrator -- overall control flow 92
76. MQSeries Integrator Rules 93
77. MQSeries Integrator Format dialog. 94
78. User Interface of the request price application 96
79. Records added to the DB2 database for one request price transaction 97
80. Control flow of the application using CICS Transaction Gateway. 98
81. CICS Transaction Gateway flow (part 1 of 3) 99
82. CICS Transaction Gateway flow (part 2 of 3) 99
83. CICS Transaction Gateway flow (part 3 of 3) 100
84. SecureWay Personal Communications -- installation 101
85. Personal Communications -- node definition (part 1 of 2). 102
86. Personal Communications -- node definition (part 2 of 2). 103
87. Personal Communications -- device definition 104
88. Personal Communications -- LAN connection definition (part 1 of 3) 105
89. Personal Communications -- LAN connection definition (part 2 of 3) 106
90. Personal Communications -- LAN connection definition (part 3 of 3) 106
91. Personal Communications -- partner LU6.2 definition 107
92. Personal Communications -- local LU6.2 definition 108
93. VTAM definitions -- ATCSTR00.B. 109
94. VTAM definitions -- XCA major node 110
95. CICS connection definition 110
96. CICS session definition. 111
97. CICS Universal Client configuration file 114
98. Configuring the CICS Transaction Gateway -- overwrite value for codepage.115
99. Conversion table for CICS 115
100.Establish the APPC connection -- console output 116
101.SecureWay Personal Communications -- LU 6.2 sessions 116
102.Records added to the DB2 database for one purchase transaction. 117
103.Presentation logic of the VSE-based CICS transaction 118

104.Presentation logic of the browser-based interface to the CICS transaction . 119
105.Visual Composition of the user interface applet 121
106.Java code fragment dealing with the CICS ECI. 123
107.Web/VSE-Host - sample configuration file. 127
108.Web/VSE-Host - sample Web page 128
109.Web/VSE-Host - sample template file 129
110.Web/VSE-Host - sample customized Web screen 130
111.Web/VSE-Host - sample WebScreen definition. 131
112.Web/VSE-Host - sample Web page 132
113.Web/VSE-Host - native 3270 screen. 133
114.Web/VSE-Host - 3270 data in a Web browser 133
115.Output from SAMPCGI 136
116.HTML script used to produce the SAMPCGI output 137
117.Output from SAMPCGI4 138
118.HTML script used to produce the SAMPCGI4 output 139
119.Overview of the CGI process in IpServer 141
120.Definition of the return page in the sample CGI used with IpServer 142
121.Data structure in copy book D21IPARC 144
122.Checking the input parameter in sample CGI used with IpServer. 145
123.Writing on CICS TSQ in sample CGI used with IpServer 145
124.Finish process in sample CGI used with IpServer. 146
125.Design of the HTML page to call the sample CGI used with IpServer. 147
126.Setting up the HTML form to call the sample CGI used with IpServer 148
127.Setting up the button to call the sample CGI used with IpServer 148
128.Define the space to display the result of the sample CGI used with IpServer 149
129.Setup for publishing site for IpServer - formatting 150
130.Publishing view ready to be published for IpServer. 151
131.Setup location properties for publishing site for IpServer 152
132.Result of our HTML page calling our sample CGI on IpServer 152
133.Friendly Foods CGI program for use with IpServer (part 1 of 6) 164
134.Friendly Foods CGI program for use with IpServer (part 2 of 6) 165
135.Friendly Foods CGI program for use with IpServer (part 3 of 6) 166
136.Friendly Foods CGI program for use with IpServer (part 4 of 6) 167
137.Friendly Foods CGI program for use with IpServer (part 5 of 6) 168
138.Friendly Foods CGI program for use with IpServer (part 6 of 6) 169

Tables

1. e-business connectors 6
2. Queue names as used by queue managers 72
3. Channel names and types as used by queue managers and applications . . . 72
4. Comparing the actions to be taken for specific events in your environment . . 91

Preface

This redbook describes the e-business options available for VSE customers to open their VSE environment to the e-business world. It gives you a broad understanding of 2-tier and 3-tier solutions and of the products that you need to implement these solutions.

For the 2-tier solution the use of products like IpServer from Data 21, Inc. and Web/VSE-*Host* from IntelliWare Systems, Inc. are described. The 3-tier solution is based on the IBM Application Framework for e-business and is implemented through a set of e-business connectors. These include CICS connectors (CICS Transaction Gateway), MQSeries connectors (MQSeries Client for Java) or DB2 connectors (DB2 for VSE & VM and DB2 Connect).

This redbook will help you plan for an e-business transition. Detailed usage examples will help you install, configure, and setup the required products.

This redbook is based on a business scenario which has been developed to illustrate how the products implementing the IBM Application Framework for e-business can be used to develop e-business applications.You will learn how Friendly Foods - a fictitious company - performed step-by-step the transition from a traditionally operating company to an e-business enterprise. The provided examples can be adopted by many companies of different industries.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working for the International Technical Support Organization Poughkeepsie Center. The project was run in the IBM Germany Development location in Boeblingen.

Annegret Ackel, from the IBM Development Laboratory Boeblingen, was the project leader.

Roger Brooks, from Mirador Inc, Williamsburg, Virginia, U.S.

Peter von Hirschfeld, from IBM South Africa

Erhard Hoehn, from the IBM Development Laboratory Boeblingen, Germany

G. M. (Jerry) Johnston, from IBM United States

Anette Stolvoort, from the IBM Development Laboratory Boeblingen, Germany

Monika Zimmermann, from the IBM Development Laboratory Boeblingen, Germany

Thanks to the following people for their invaluable contributions to this project:

Mike Carver
Data 21, Inc., Torrance, CA, U.S.

Uli Kettner
IBM Development Laboratory Boeblingen, Germany

Wilhelm Mild
IBM Development Laboratory Boeblingen, Germany

Christian Toepsch
IBM Development Laboratory Boeblingen, Germany

Eric Vaughan
IntelliWare Systems, Inc., Arlington, Texas, U.S.

Debbie Yu
IBM Laboratory Toronto, Canada

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Please send us your comments about this or other Redbooks in one of the following ways:

- Fax the evaluation form found in “IBM Redbooks review” on page 185 to the fax number shown on the form.
- Use the online evaluation form found at http://www.redbooks.ibm.com/
- Send your comments in an Internet note to redbook@us.ibm.com

Chapter 1. Introduction

This chapter offers a high-level introduction to the e-business options available for VSE customers to open their VSE environment to the e-business world.

Note: This redbook is not intended to give an introduction to e-business and the motivations why a company should become an e-business. It concentrates on the products and solutions that are available to the VSE customer and how these products are installed and customized. For an explanation of "new" terms related to e-business, refer to Appendix A, "e-business technologies" on page 159.

1.1 e-business options for VSE customers

When VSE customers start thinking about implementing e-business applications, they have several options, which fall into two general categories: the *2-tier* and the *3-tier* model.

Using the 2-tier model, a client on the Internet (typically using a Web browser) can directly connect to applications and data on VSE/ESA. For this reason, a Web server (HTTP server) is needed on VSE.

The 3-tier model relies upon an intermediate Web server (installed outside the VSE platform), which in turn provides access to VSE data and applications.

1.1.1 Two-tier versus 3-tier

There is no right or wrong answer to this question. The answer depends on customer needs, skills in the company, available resources, and so forth, and might therefore be different for each VSE customer.

Experience shows that most customers go through several phases before they reach the final goal of having changed their business to an e-business. This process can be divided into three major phases:

1. *Publish* -- get your information on the Web (normally static data)
2. *Access data and applications* -- integrate the Web with your existing business systems
3. *e-transaction processing*

Phase 1 might be quite short for many customers because they may recognize the need to put dynamic data on the Web and give access to existing applications. This can be done by Web-enabling existing applications and implementing logic to access existing production data. Several 2-tier solutions on VSE are available to Web-enable existing applications and implement an *intermediate* e-business solution.

When going from phase 2 to phase 3, customers experience that they have to expand their existing applications and/or develop new applications. IBM created a framework to ease the development of these new applications: the IBM Application Framework for e-business. This is an integrated and consistent collection of APIs, protocols, services, tools and conventions, providing an open, standards-based, scalable and complete foundation for developing and deploying

e-business solutions. It is designed to integrate existing applications with new Web applications.

The framework describes a logical 3-tier model (which could result in a physical 2-tier implementation) that consists of:

1. Tier 1 -- a *client tier* containing logic related to the presentation information (that is, the graphical user interface) and requests to applications through a browser or Java applet
2. Tier 2 -- *Web application servers* containing the business logic and processes that control the reading and writing of data (business services)
3. Tier 3 -- *Servers* (external services) that provide the data storage and transactional applications used by the Web application server processes

At the heart of the framework is a single, unifying Java-based programming model. A very important part is the Web application server (for example, one of IBM's Web application servers, which could be WebSphere Application Server or Lotus Domino). The Web application server provides the runtime environment for new e-business applications.

For VSE customers, this *logical* 3-tier model also implies a *physical* 3-tier model because the Web application servers do not natively run on the VSE platform. However, the framework is designed to integrate existing applications and data on VSE (external services) with the business services. This integration is implemented by so-called e-business connectors.

For a detailed description of the IBM Application Framework for e-business, refer to the Web site:

```
http://www.ibm.com/software/ebusiness/library.html
```

WebSphere as one of IBM's Web application servers is available (or planned) on OS/390, AIX, Windows NT, Linux, OS/400, plus a variety of platforms including Sun Solaris, HP-UX, OS/2, and Novell.

Figure 1 on page 3 shows how VSE can be integrated into applications based on the IBM Applications Framework for e-business.

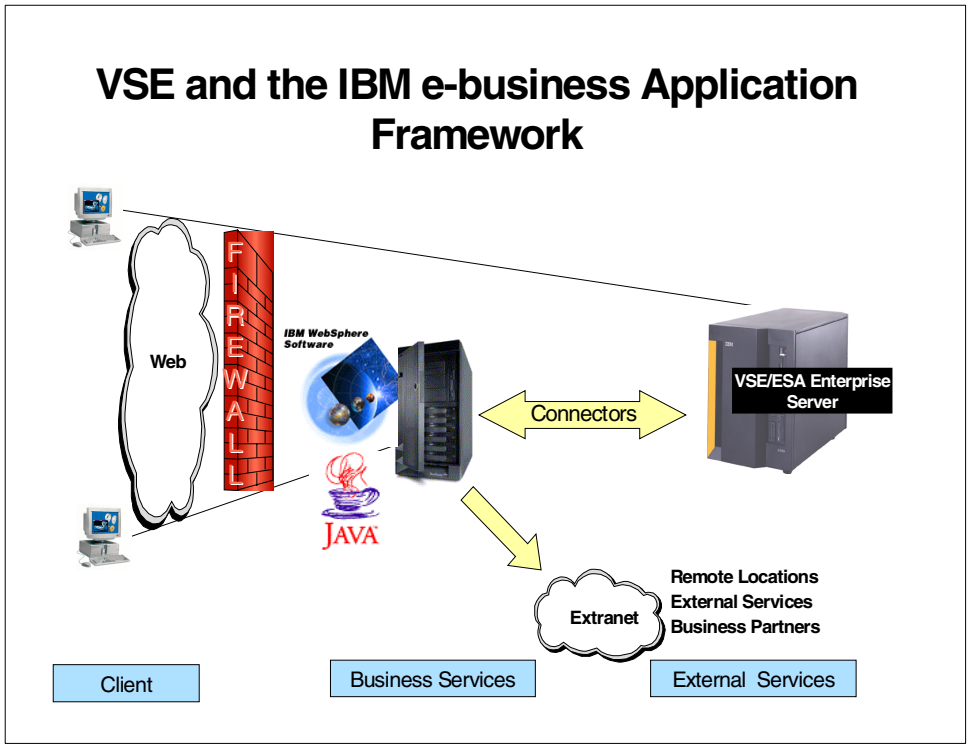


Figure 1. VSE and the IBM e-business Application Framework

You can find a detailed discussion on VSE application options in the paper *VSE Applications: How e-business fits*, GF22-5137.

1.1.2 Two-tier implementations

Two major elements form the base for the 2-tier model:

- TCP/IP support in VSE
- A Web server (HTTP server)

VSE/ESA today meets both requirements. VSE/ESA V2.3 and later deliver native TCP/IP support and open VSE to the Internet. For details on the installation and customization of TCP/IP for VSE/ESA and the related applications, refer to *Getting Started with TCP/IP for VSE/ESA 1.4*, SG24-5626.

In addition to TCP/IP, VSE provides several Web server solutions. IBM TCP/IP for VSE/ESA's Application Pak delivers functions including FTP, Telnet, LPD, LPR in addition to a basic Web server. Another option is IpServer from Data 21, Inc. Both Web servers offer "basic" Web serving capabilities that can be used for an intermediate e-business implementation.

IpServer natively runs in the VSE CICS environment (partition) and allows access to existing CICS applications and data. This access is implemented with the help of Common Gateway Interface (CGI) programs and Server Side Includes (SSIs), which can be written in any programming language supported by the CICS-provided command level interface (C, COBOL, PL/I, HLASM). Existing skills in the CICS environment can be used to start with the e-business implementation.

In addition to the Web servers, there are solutions available from IBM and Independent Software Vendors to Web-enable existing CICS applications without changing them. This enables customers to make CICS as well as 3270 applications accessible from the Internet (intranet or extranet). These solutions are:

- The CICS Web Support together with the 3270 bridge, which are part of CICS Transaction Server for VSE/ESA (available in 2000), allow access to CICS applications (BMS and 3270 terminal control applications). The solution is based on HTML templates, which can be automatically generated from existing BMS maps for BMS applications. For 3270 terminal control applications, HTML pages are dynamically created from the 3270 screens during runtime.
- IpServer 3270-to-HTML is a 3270 screen scraping solution that allows VSE users to easily add a GUI front end to existing CICS applications. An HTML design tool (like NetObjects Fusion) can be used to design the HTML pages. This product is based on the CICS Transaction Server for VSE/ESA and is available on VSE/ESA V2.4 and above.
- Web/VSE-Host® from IntelliWare Systems, Inc. dynamically converts 3270 screens to HTML. It allows access to any 3270 application from a Web browser.

The following graphic summarizes the available 2-tier solutions.

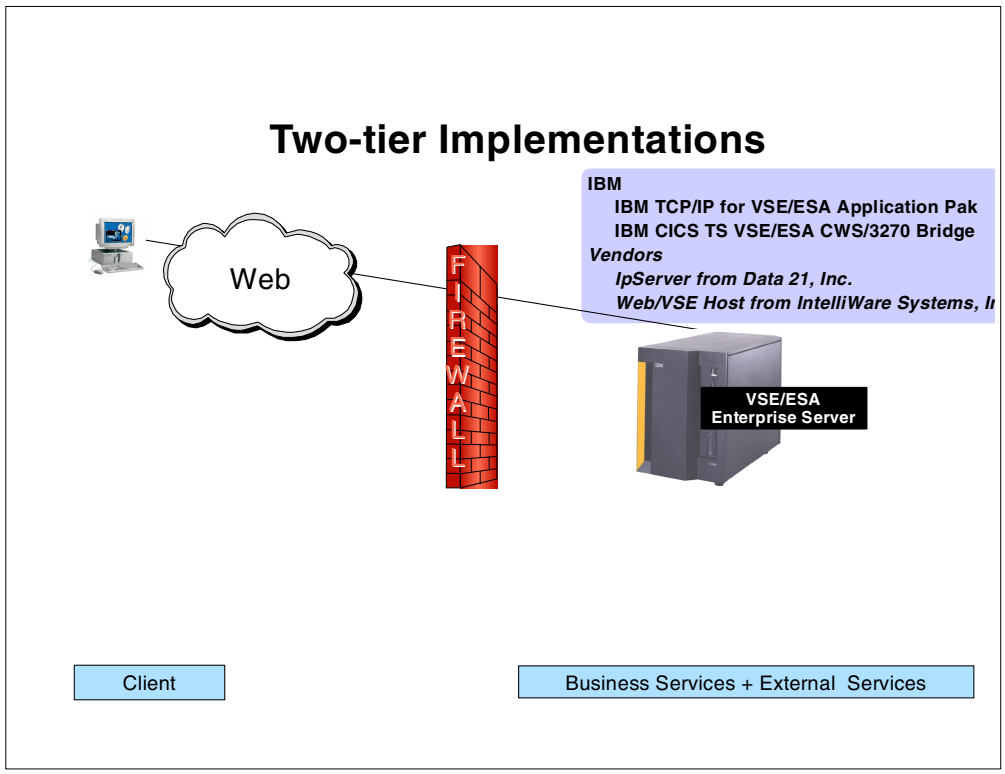


Figure 2. Two-tier implementations

A detailed description of the installation of the 2-tier solutions *IpServer* and *Web/VSE-Host*® is provided in Chapter 7, “Options for a 2-tier solution” on page 125.

1.1.3 Three-tier implementations on the IBM e-business Application Framework

Key to the successful implementation of e-business applications based on the IBM Application Framework for e-business is the availability of connectors to information and assets (external services) on VSE. These resources include CICS, VSE data (for example DB2, VSAM, DL/I), and 3270 (VTAM-based) applications.

The IBM Application Framework for e-business describes the base of the connector architecture in several stages. Only the third stage is described here in more detail. It is referred to as server integration. The following graphic illustrates this stage of the connector architecture.

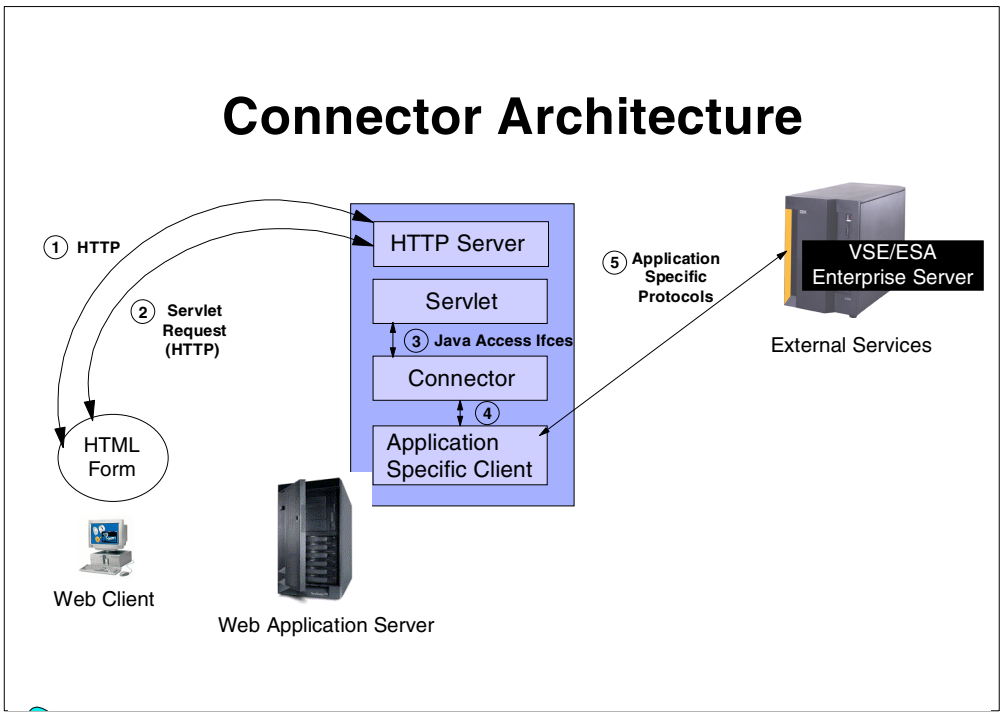


Figure 3. Connector architecture

The flow is as follows:

1. HTML forms, presented to the user via a Web browser, are used to build servlet requests.
2. On completion of the form, an HTTP servlet request is created and sent to the Web application server.
3. As part of the execution of business logic, servlets use Java access libraries to invoke connector functionality.
4. Connectors translate these requests into application-specific client requests.
5. Application-specific protocols are used to communicate with external services (database, transaction system, and so forth) residing on VSE/ESA (see Table 1 on page 6).

Table 1 shows the connectors that are available from IBM and Independent Software Vendors (ISVs) in the VSE environment:

Table 1. e-business connectors

	TCP/IP	**SNA**
Application-to-Application Connectors		
CICS Transaction Gateway	no	yes
IBM SecureWay Host On-Demand	yes	yes
MQSeries Client for Java	yes	yes
Connectors to data stored in		
DB2 for VSE & VM (using DB2 Connect)	yes (in beta)	yes
VSAM	yes (ISVs)	yes (ISVs)
DL/I	yes (ISVs)	yes (ISVs)

Note: ISV products for accessing VSAM and DL/I data resources include the following products:

- CrossAccess from CROSS ACCESS Corporation
- ViaSQL from ViaServ, Inc.

A detailed description on the implementation and customization of the following connectors is provided in the following chapters:

- *DB2 Connect* in Chapter 3, “Consumer marketing” on page 15
- *Host On-Demand* in Chapter 5, “Integration with service providers” on page 59
- *MQSeries Client for Java* in Chapter 6, “Business integration with suppliers” on page 67
- *CICS Transaction Gateway* in Chapter 6, “Business integration with suppliers” on page 67

Chapter 2. The Friendly Foods scenario

This redbook is based on a business scenario that was developed to illustrate how the products implementing the IBM Application Framework for e-business can be used to develop e-business applications. The scenario shows what products the Friendly Foods company used to transform their business into an e-business.

Note: Friendly Foods is a fictitious company. Any similarity to existing companies is coincidental. Friendly Foods was created to illustrate a company's e-business evolution. The Friendly Foods evolution process to an e-business, however, can be adopted by many companies of different industries.

In addition, we describe the setup we used at the ITSO to implement this scenario.

2.1 The Friendly Foods e-business

Friendly Foods is a bakery company based in Vienna, Austria. Some years ago the company started to expand its business by establishing production and distribution centers to supply Friendly Foods bakery stores around the world. For more than 20 years, the company has been using VSE/ESA to run its business applications, including its production and financial control applications. They are currently running on a *VSE/ESA V2.3* system with the goal to upgrade to VSE/ESA V2.4 within the next six months.

Friendly Foods realized that the Internet and e-business could help them to further expand the company. They were looking for a solution that would enable them to:

- Implement the latest Internet technologies and standards in the area of e-business
- Protect their current investment in the VSE system by integrating their existing applications and data
- Be positioned to adopt future technologies

They decided to base their new applications on the IBM Application Framework for e-business while simultaneously integrating as much of their existing data and as many of their existing applications as possible.

The following paragraphs describe how Friendly Foods transformed their company into an e-business. But first, a short journey into history explains how Friendly Foods used to work in the past.

Four major groups had to (and still have to) work together in order to guarantee a smooth process. The main integration element was their office in Vienna, where information from their sales force (stores) and production came together. Employees of the office were entering data coming from the sales force by fax or phone into their central DB2 repository on VSE/ESA. Changes affecting production were directly sent to a printer in the production areas, whereas orders and requests for price information from their suppliers (bids) were sent by fax.

The DB2 database on VSE/ESA included information about their products, store accounts, store order mixes, and so forth. The information was maintained by existing CICS business applications on VSE/ESA.

Information entered via CICS transactions on VSE/ESA included information about their products, store account information, store order mix information, and so forth. With the world-wide expansion of their business, they needed to have people on-site in their Austrian office 24 hours a day in order to serve the requests coming in from the stores and update the production data accordingly.

Figure 4 illustrates how the groups at Friendly Foods used to work together.

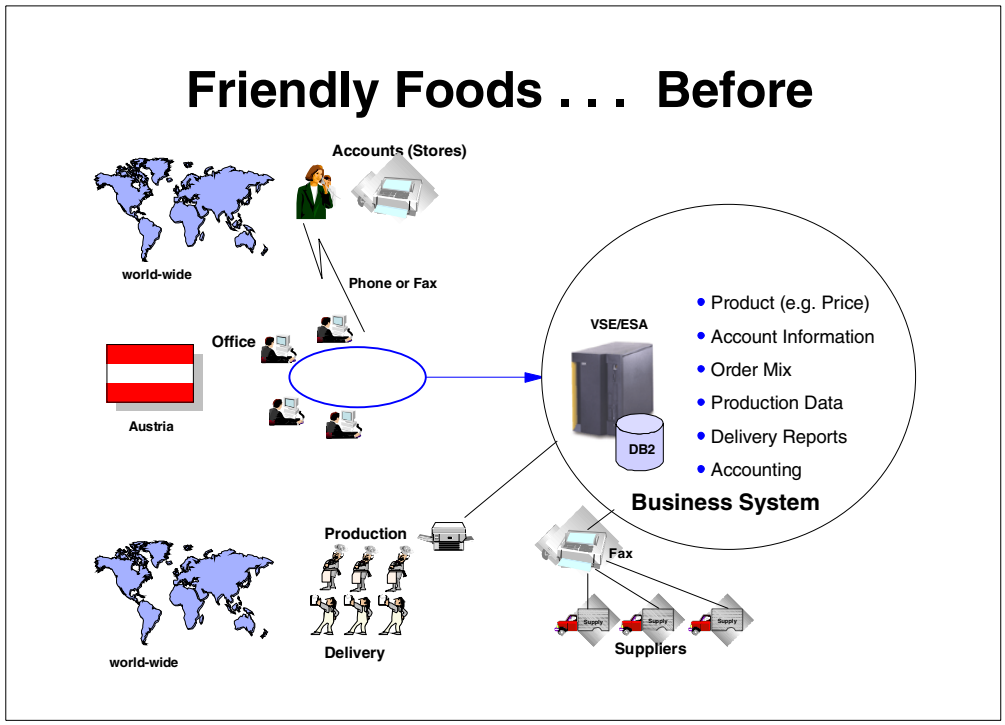


Figure 4. Friendly Foods before they became an e-business

2.2 How Friendly Foods became an e-business

We now describe Friendly Foods' process in becoming an e-business:

1. Consumer marketing
2. Account self-service and account tracking
3. Integration with service providers
4. Business integration with suppliers

Further plans to also offer online order capabilities to their customers do exist today, but have not yet been realized.

2.2.1 Consumer marketing

Friendly Foods' first step towards an e-business was to enhance the relationship with their existing customers by better informing them about their products and services, while at the same time attracting new customers.

To do this, they generated a Friendly Foods Web presence. Their Web page gives general information about the Friendly Foods company and its products, but also offers the customers the possibility to look for the availability of a store in their neighborhood. With this online service, Friendly Foods was able to dramatically enhance customer satisfaction, and they have indications that new customers were attracted to their stores.

2.2.2 Account self-service and account tracking

As Figure 4 on page 8 indicates, the cooperation between the Friendly Foods office in Vienna and its stores required a lot of human interactions. The productivity of these two groups was greatly enhanced by an e-business application that allowed the accounts (stores) to do the following:

- Change their order mix online
- See their actual account balances

With this implementation, Friendly Foods was able to reduce its account care costs to a minimum and at the same time enhance the level of service to those accounts. The accounts no longer have to wait for a person to become available in the account care center, but instead serve themselves on the Internet.

2.2.3 Integration with service providers

In addition to offering its stores this Web self-service possibility, Friendly Foods still wanted to allow them to call in for last minute changes. In order to fully remove the need for people working in the account care center (taking the phone calls and entering the data into the VSE system), they decided to outsource this service to an external call center.

This external call center was given the capability to dial into the Friendly Foods production system and directly update the information in the databases.

This implementation created the infrastructure for the integration of any service provider in case Friendly Foods decides to do so in the future.

2.2.4 Business integration with suppliers

A lot of manual work used to be involved in the cooperation between Friendly Foods and its suppliers. In order to either place an order or request a bid, a fax was sent to the supplier and the response to a price request was returned by fax. The exchange of messages (requests and orders) was stored in the DB2 database on their VSE system.

Here, also, Friendly Foods introduced a new e-business application that allowed it to directly integrate its suppliers into its business process by sending them electronic messages with the appropriate request. The supplier system automatically sends first an acknowledgement and then an electronic answer to the request.

2.3 e-business implementation at Friendly Foods

Before Friendly Foods started with this e-business project, the responsible people met to define the requirements of their e-business implementation as follows:

- Existing assets on VSE should be exploited as much as possible.

- The project should be used to build new skills in areas like Java.
- The implementation should be based on open standards and technologies.
- The network protocol should be TCP/IP whenever possible.

The IBM Application Framework for e-business met all these requirements. Therefore, Friendly Foods used the framework as a guide for its new applications. This resulted in a physical 3-tier implementation, with the Web application server running on an outboard server.

Friendly Foods has several application programmers who developed and still maintain their existing CICS COBOL transactions on VSE. Due to the fact that all Year 2000 and EURO application programming work had been finished, some of the VSE application programmers had spare time which they were able to spend on new tasks: the new e-business applications. The were also supposed to develop the business logic for the new e-business applications.

Friendly Foods decided to hire one *Web designer* who is responsible for the GUI front-end as presented to the end user.

These programmers did not have any experience in Java and the e-business environment. Therefore, easy-to-use visual application development tools were a must for Friendly Foods' application development.

These development tools and the base product setup at Friendly Foods regarding the 3-tier environment are described in this chapter. However, different e-business connectors were used for the implementation steps. These are described in the respective chapters.

Figure 5 on page 11 shows the Friendly Foods system setup and the base products for its e-business implementation.

Note: Although Friendly Foods decided to use a Windows NT server for the installation of the outboard Web server, the configuration with an AIX server would be similar. All products used by Friendly Foods are also available on the AIX platform.

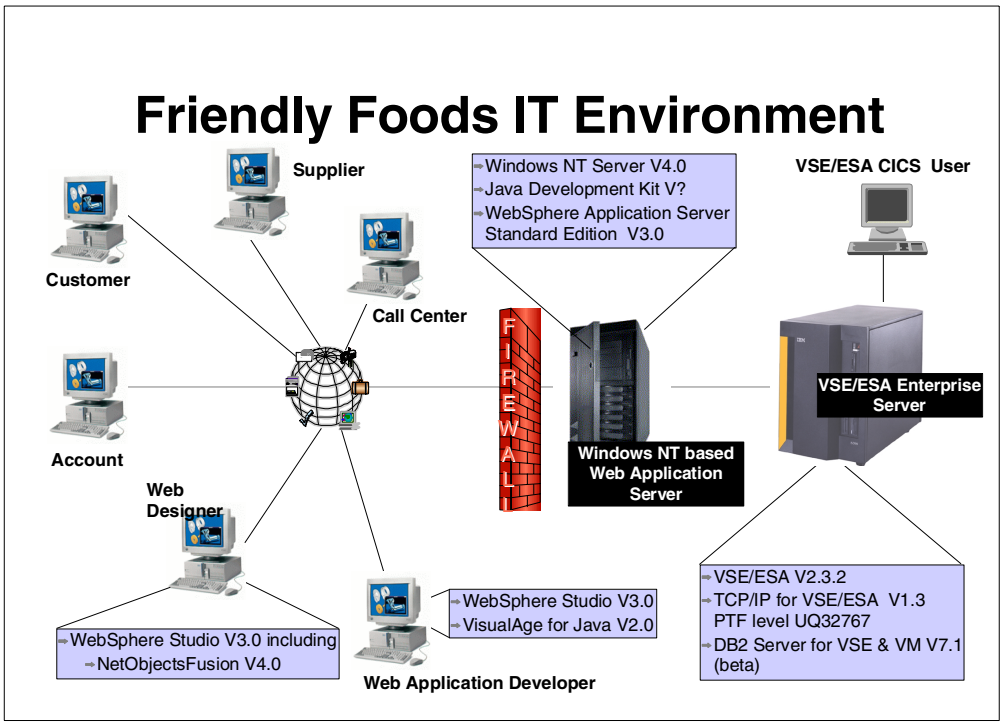


Figure 5. Friendly Foods IT environment

The major new software components of this solution, as illustrated in Figure 5, are:

- WebSphere Application Server Standard Edition V3.0
- WebSphere Studio V3.0 including:
  - NetObjects Fusion V4.0
  - NetObjects BeanBuilder
- VisualAge for Java V2.0

2.3.1 WebSphere Application Server

WebSphere Application Server is a Java-based application server designed to facilitate the management and deployment of Web applications. These deployed applications are typically composed of either Enterprise JavaBeans (EJBs), Java Server Pages (JSPs) and/or Java servlets. They communicate with clients using a Web browser interface. The different types of Java applications that run in the WebSphere environment can also make use of JavaBeans. WebSphere provides the environment (run time) and infrastructure required to install and manage these types of applications.

Websphere does not operate in isolation. It needs to be installed on an HTTP server that handles HTTP requests from browsers and delivers HTML back to them using the HTTP protocol. WebSphere can be installed on a number of HTTP servers. The Windows NT version ships with the IBM HTTP Server, which is based on the Apache Web Server.

Websphere Application Server comes in three editions:

- WebSphere Application Server Standard Edition
- WebSphere Application Server Advanced Edition
- WebSphere Application Server Enterprise Edition

Figure 6 shows the difference between them.

Figure 6. WebSphere Application Server editions

Note: *This redbook does not cover the installation and customization of Websphere itself; we assume that WebSphere is already installed.* For more information on the installation and customization of WebSphere Application Server, refer to *WebSphere Application Servers: Standard and Advanced Editions,* SG24-5460.

For today's Friendly Foods e-business implementation, the WebSphere Application Server Standard Edition met all the requirements.

2.3.2 VisualAge for Java

VisualAge for Java provides a visual programming environment to build Java applications, applets, JavaBeans and Enterprise JavaBean components.

It is mostly used by *software engineers* to write access-to-data code and business logic. It provides everything that is needed to develop an e-business business application.

2.3.3 WebSphere Studio

WebSphere Studio is a tool set for Web site designers and developers deploying interactive e-business applications. It is a *visual* layout tool for *dynamic* Web pages and makes it possible to easily exploit Web and Java standards. WebSphere Studio includes the following major elements:

- A visual page designer to easily create HTML and Java Server Pages (based on technology from IBM NetObjects TopPage)
- A workbench environment that lets Web development teams organize and manage Web development projects
- Wizards that help developers create dynamic interactive Web pages. Wizards can generate Java Server Pages, JavaBeans, SQL statements and servlets
- An applet designer based on the NetObjects BeanBuilder technology
- A web art designer to create masthead images, buttons and other graphics
- An animated GIF designer to easily create animated GIFs
- Integration with IBM VisualAge for Java Professional Edition for building Java applications, applets, servlets, JavaBeans and Enterprise JavaBean components
- NetObjects ScriptBuilder for easier script editing of Extensible Markup Language (XML) and Wireless Markup Language (WML)
- NetObjects Fusion to visually prototype Web sites and site management

WebSphere Studio is mostly used by *Web designers*. They take the JavaBeans (created by the software engineer in VisualAge for Java) and create multimedia applets and Web pages for it. There is some overlap between WebSphere Studio and VisualAge for Java (both can create applets and beans), but a simplified view would be that VisualAge for Java is for developing the business logic and WebSphere Studio is for developing the external interfaces, such as Web page layout and design.

2.4 The ITSO system setup

The Friendly Foods scenario was not only for use as base for this redbook, but also as a demo environment. Therefore, we needed a setup that was portable and mobile, and so a simplified system setup was implemented.

The *Web browser* interfaces for the target groups (Web designer, Web application developer, supplier, customer, store account, call center) were installed on the same physical Thinkpad as the Windows NT-based *server* components. This resulted in the following very simple ITSO system setup:

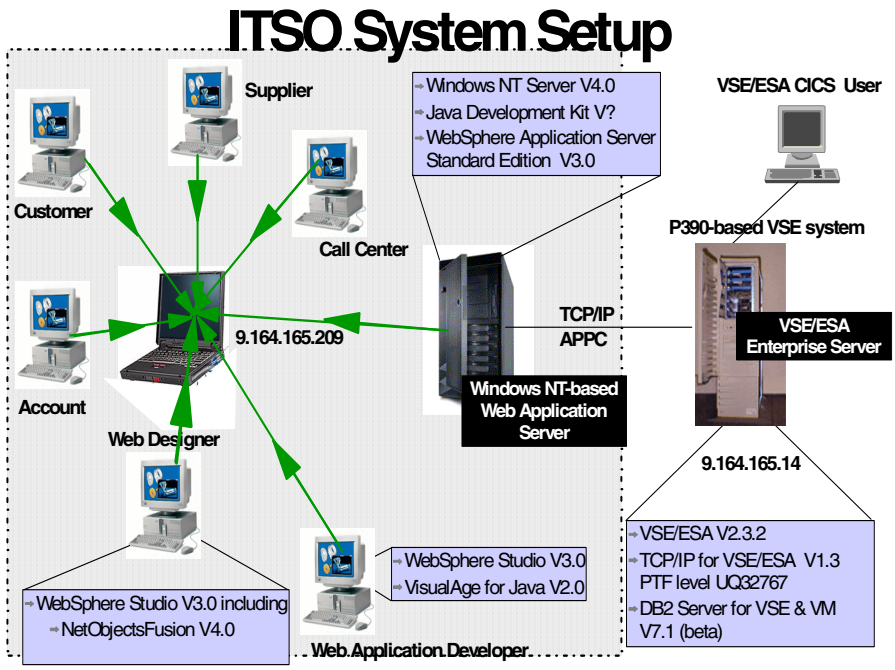


Figure 7. ITSO system setup

All Windows NT-based products are installed on one single IBM Thinkpad model 770. This includes client (Web browser) and server (WebSphere Application Server).

VSE/ESA V2.3 is natively installed (without VM) on a PC Server 330 System/390. The following products are installed in addition to the VSE base operating system:

- CICS/VSE V2.3
- TCP/IP for VSE V1.3
- DB2 Server for VSE & VM V7.1 (beta code)
- COBOL for VSE/ESA

One major goal of the ITSO setup was to use TCP/IP connections to the VSE systems whenever possible. APPC connections have only been set up in cases where TCP/IP connections are not available.

Chapter 3. Consumer marketing

Consumer marketing was Friendly Foods' first step into e-business. Its major goals were the following:

- To enhance the satisfaction of their existing customers by providing additional information about their products and stores
- To attract new customers

With consumer marketing, Friendly Foods provides its information in the form of static text and graphics to its end customers. In addition, Friendly Foods decided to offer a special service to its customers, namely the possibility to get a list of all Friendly Foods stores in a specific country.

The information about its world wide stores is kept in a DB2 database on VSE. This information is needed for billing, account receivables and shipping. The list (DB2 database) is maintained through an existing CICS transaction on VSE. In order to have the most current information on the Web, Friendly Foods decided to directly access the DB2 database on VSE and get the actual list of stores.

The Friendly Foods Web designer was responsible for the static text as well as the visual representation of the dynamic parts. He selected NetObjects Fusion to develop and maintain the Friendly Foods Web pages.

The application programmers are responsible for providing the dynamic parts, that is, the logic for accessing the DB2 database on VSE/ESA. The function they need is to read from the DB2 database on VSE/ESA. One effective way of implementing this is by directly accessing the DB2 database via DB2 Connect. DB2 Connect implements the DB2 DRDA requester function and uses it to enable an application outside VSE to connect to the DB2 DRDA server on VSE. The DB2 DRDA server is part of DB2 Server for VSE & VM.

One of Friendly Foods' requirements was to have a TCP/IP connection between its Web application server platform and VSE. DB2 Server for VSE & VM V7.1 will deliver this function, but is not yet generally available. A beta test program is currently running and Friendly Foods is one of the test sites.

The installation and setup of the DB2 beta code is described in this chapter. *The setup for the DRDA connection over TCP/IP might change in the GA version of DB2 Server for VSE & VM, depending on the results of the official beta test program.*

Figure 8 on page 16 shows the DB2 system configuration as implemented by Friendly Foods.

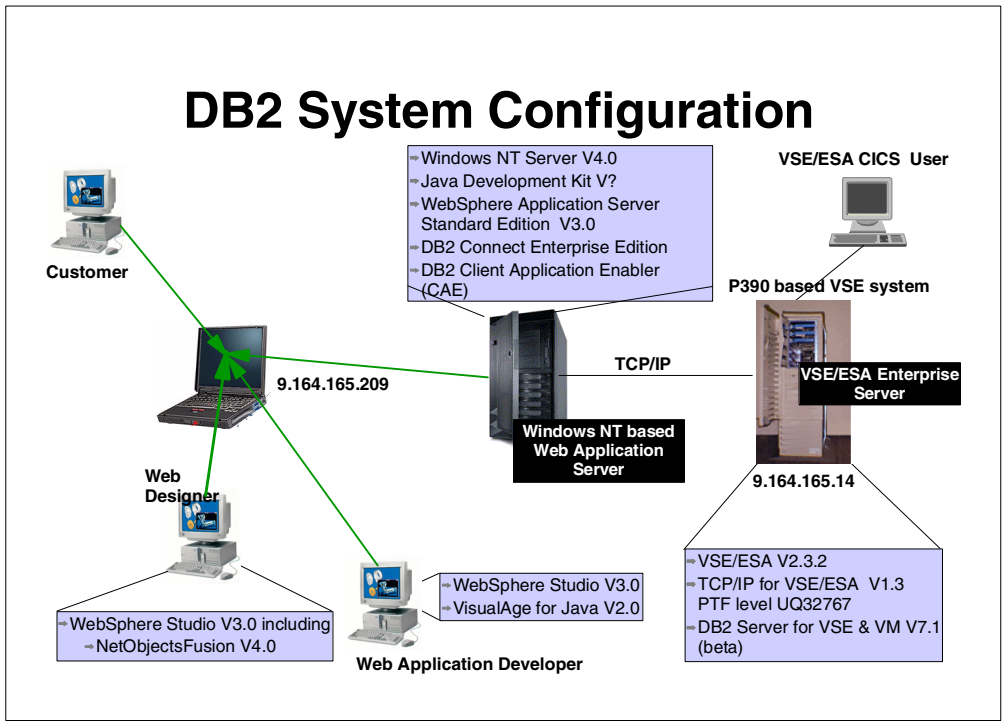


Figure 8. DB2 system configuration

The major components of this configuration are:

- DB2 Server for VSE & VM, which holds the production database and as part of it, the store address database
- DB2 Connect Enterprise Edition, a host access server that provides managed connectivity from desktop and Web applications to DB2 databases
- DB2 Client Application Enabler, which provides the function to allow an application to access DB2 Universal Database servers and DB2 Connect gateways

Friendly Foods decided not to install DB2 Universal Database (UDB) server on their Windows NT server yet because for now they wanted to keep their production data on VSE, and their Web application does not yet require a DB2 UDB server on Windows NT.

3.1 The User Interface

One possibility of providing a dynamic User Interface (UI) that can be run in a Web browser is by implementing a Java applet. Java applets can provide an equivalent for each of the HTML UI elements, as well as an infinite set of UI elements of the Java language.

A disadvantage of using Java applets for UI generation is that the required version of Java must be supported by the Web browser, which means that the UI part of the e-business application will dictate which browsers can be used for the client side e-business application.

A second disadvantage of Java applets is that any class that is not included as part of Java must be loaded from the Web server as it is needed. If these additional classes are large, then the initialization of the applet may take from seconds to minutes, depending upon the speed of the Internet connection.

Friendly Foods decided to implement the customer UI with the Java applet technology because they did not expect the classes to become very large and because they will not include too much business logic.

DB2 provides support for client applications and applets written in Java using JDBC (Java Database Connectivity) to access DB2 databases. Friendly Foods implemented the database access on its database on VSE by providing a Java applet that is loaded to the customer's Web browser on request. This implementation requires a JDBC applet driver, which is installed as part of DB2 Client Application Enabler (CAE).

This driver consists of a JDBC client and a JDBC server. The JDBC client driver is loaded on the Web browser along with the applet. When a connection to a DB2 database is requested by the applet, the client opens a TCP/IP socket to the JDBC server on the machine where the Web server is running. After a connection is set up, the client sends each of the subsequent database access requests from the applet to the JDBC server through the TCP/IP connection. The JDBC server then makes corresponding CLI (ODBC) (call level interface open database connectivity) calls to perform the task. Upon completion, the JDBC server sends the results back to the client through the connection.

Figure 9 illustrates how a DB2 JDBC applet is implemented.

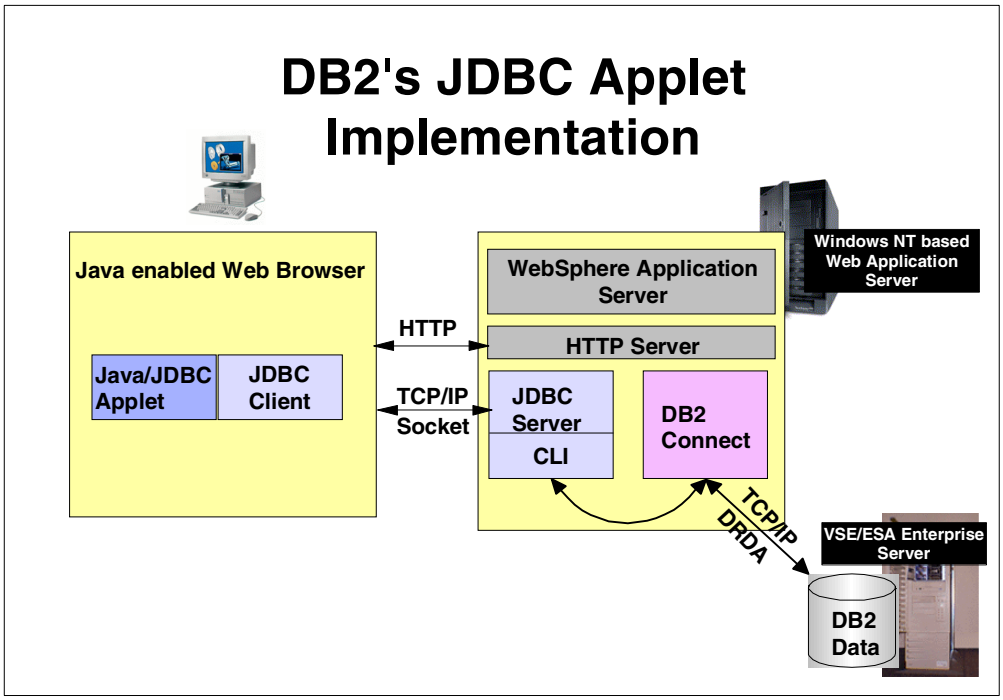


Figure 9. DB2's JDBC applet implementation

3.2 Installation and configuration

In this chapter we describe the steps to install and configure DB2 Server for VSE & VM, DB2 Connect, and DB2 Client Application Enabler.

3.2.1 DB2 Server for VSE & VM V7.1 (beta)

Friendly Foods migrated from DB2 Server for VSE V6.1 to the beta version of DB2 Server for VSE V7.1 because of the DRDA support over TCP/IP. DB2 was already installed and set up at Friendly Foods, therefore the DB2 installation and implementation are not described in this redbook. There are several other redbooks available describing the DB2 usage on VSE. See D.1, “IBM Redbooks” on page 175 for a list of redbooks related to DB2 Server for VSE & VM.

In order to activate this support, Friendly Foods had to change its DB2 startup job to establish the connection to TCP/IP. DB2 is installed in the VSE sublibrary PRD2.DB2710. Figure 10 shows the startup job as it is used on the ITSO system:

```
// JOB DB2START    START DB2 IN MULTIPLE USER MODE
// LIBDEF PROC,SEARCH=(PRD2.DB2710)
// LIBDEF PHASE,SEARCH=(PRD2.DB2FIX,PRD1.BASE,                          X
               PRD2.SCEEBASE,PRD2.DB2710)
// EXEC  PROC=ARIS61DB    *-- DB2 DATABASE ID PROC
// EXEC  ARISQLDS,SIZE=AUTO,PARM='TCPPORT=446,RMTUSERS=20,        X
               NCUSERS=20'
/*
/&
```

Figure 10. VSE DB2 startup job

The parameter TCPPORT=446 indicates that DB2 will establish a connection with TCP/IP for VSE/ESA and listen on port 446.

The parameter RMTUSERS=20 indicates that 20 remote users can access the DB2 database in parallel. You have to think about this parameter quite carefully and make an estimate of how many Internet users will need to access your DB2 Server on VSE in parallel. If the value is too small, it could happen that access to the information is denied because too many parallel users are accessing the DB2 database. The value for RMTUSERS, however, depends on the availability of virtual storage. If the value is too big, you might waste virtual storage on your VSE system.

Note: TCP/IP has to be started before you start DB2 Server on VSE in order to establish a connection between TCP/IP and DB2 at the time DB2 is started.

The following message is displayed on the console after successfully establishing the connection to TCP/IP:

```
F8 0008 ARI4100I TCP/IP service initialized from TCPPORT.
                 TCP/IP Port is 446.
```

Figure 11. Console output of DB2 startup

Disclaimer

This description might change depending on the results of the DB2 Server for VSE & VM beta test program.

Friendly Foods kept all addresses of its worldwide stores in the FFSTORES table as part of its DB2 Server for VSE database. The following columns have been defined in table FFSTORES:

```
STOREID    1-6                  ID of the store account
STORENAME  7-31                 Name of the store
LOCSTREET  32-56                Address of the store
LOCCITY    57-81                City where the store is located
LOCZIP     82-91                Zip code
LOCCOUNTRY 92-116               Country where the store is located
LOCREP     117-126              Sales rep of the store
STOREN1    127-133
STOREN2    134-140
LDATE      141-150
WEBPIC1    151-170              Graphic1 to be displayed on the Web
WEBPIC2    171-190              Graphic2 to be displayed on the Web
ACODE      191-200               Access Code
```

Figure 12. Columns of DB2 table FFSTORES

3.2.2 DB2 Connect and DB2 Client Application Enabler

To establish a connection from the Windows NT server to DB2 on VSE, Friendly Foods installed DB2 Connect. DB2 Client Application Enabler is delivered together with DB2 Connect and can be installed at the same time. A selection panel gives you the choice to install either DB2 Connect, DB2 Client Application Enabler, or both. Select *both* items on the following panel.

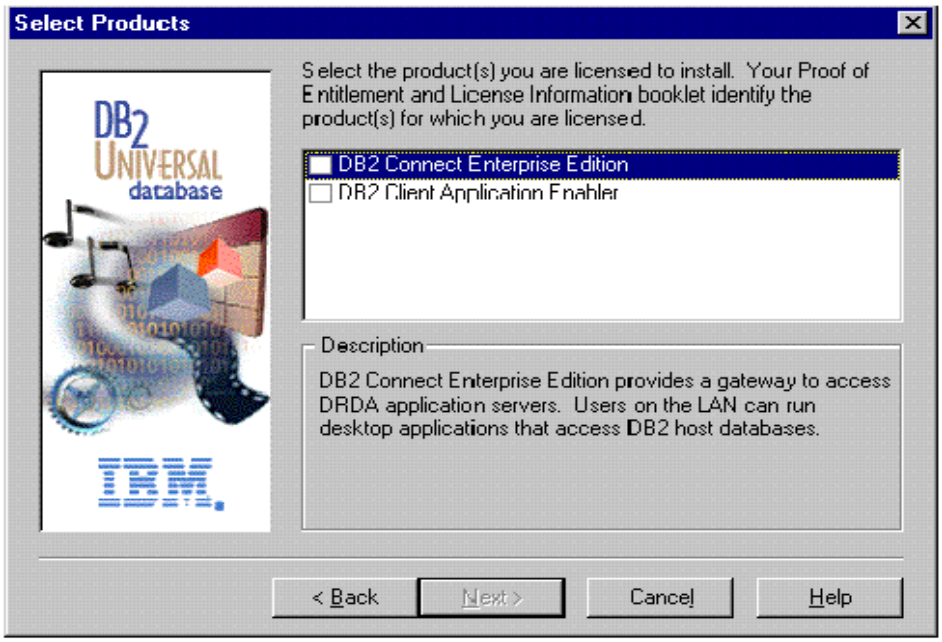


Figure 13. DB2 Connect installation

Follow the default installation path and reboot the Windows NT server. After successful installation, you should see DB2 for Windows NT added on your Windows NT desktop. Select **Client Configuration Assistant** in order to create a connection to your DB2 database on VSE. Perform the following steps in order to specify the parameters for the connection:

1. Click **Add Database** on the first panel. Continue with **Next**.
2. Select **Manual Configuration** on the panel titled 1. Source. Continue with **Next**.
3. Specify TCP/IP on the screen titled 2. Protocol. Continue with **Next**. Specify the values related to your DB2 TCP/IP connection on the screen titled 3. TCP/IP as follows:

   a. The IP address of your VSE system as specified in your VSE TCP/IP configuration file. In our environment, the address is 9.164.165.14.

   b. The port number used to establish a connection between DB2 Server for VSE & VM and TCP/IP for VSE/ESA as specified in your DB2 startup job (Figure 10 on page 18). In our environment it is 446.

   Continue with **Next**.

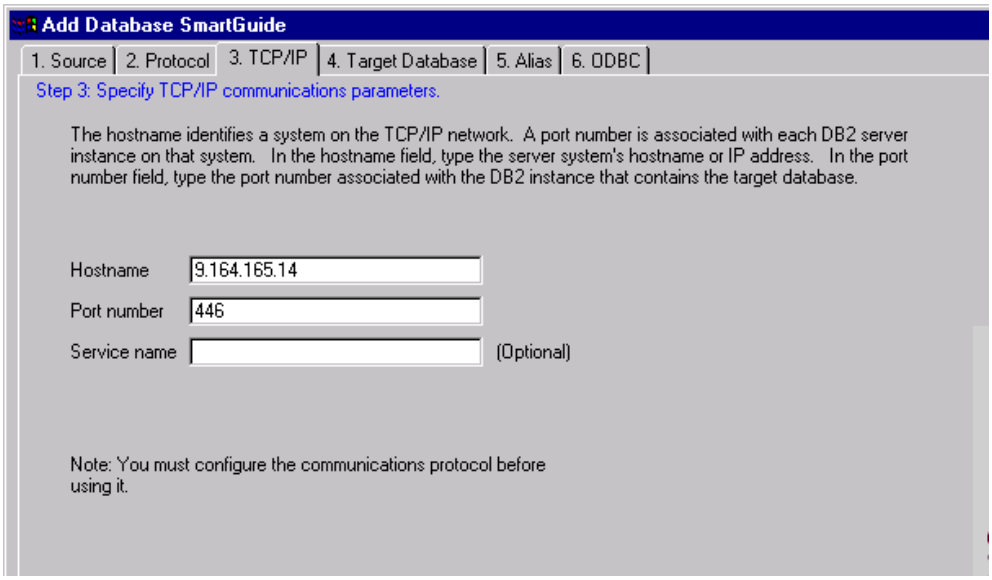


Figure 14. DB2 client configuration -- TCP/IP

4. Specify the name of the DB2 database on VSE on the panel titled 4. Target Database. In our environment this is SQLDS. Continue with **Next**.
5. Change the default alias name to your preferred alias name on the panel titled 5. Alias. We decided to also use SQLDS as the alias name. Continue with **Next**.
6. Use the default values on the panel titled 6. ODBC, shown in Figure 15.

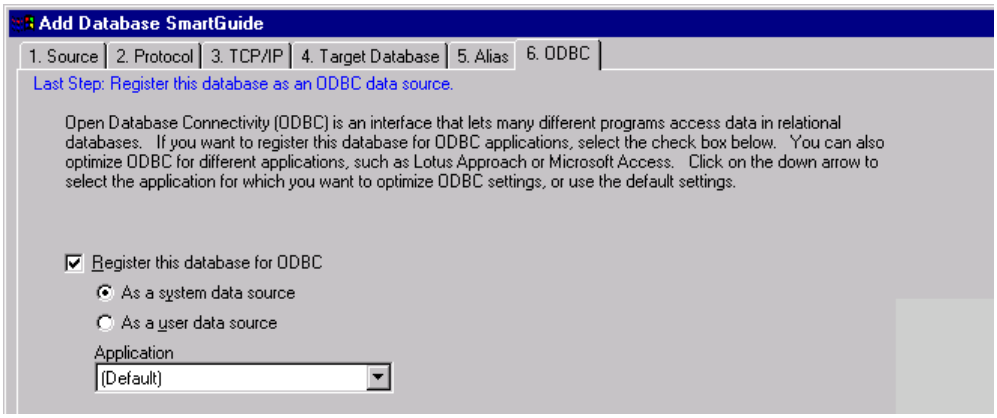


Figure 15. DB2 client configuration -- ODBC

7. Finish by selecting **Done**.
8. You can directly test the connection between your Windows NT server and your VSE system by selecting **Test** on the panel “Confirmation SQLDS.”
9. Specify a user ID of a user who is authorized to access your DB2 database together with his password on the panel “Connect to DB2 Database.” In our case, the user ID is HOEHN, as shown in Figure 16 on page 22.

Figure 16. Connect to DB2 Database

10. The panel shown in Figure 17 will be displayed if the test is successful. If not, the displayed message will help you identify the error.

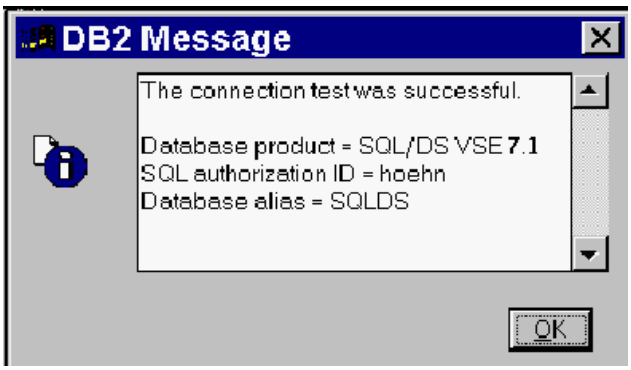


Figure 17. Successful connection message

With this, a connection is established between the DB2 client on Windows NT and the DB2 server on VSE/ESA.

To be able to establish a connection from a Java applet to the DB2 Connect gateway, the JDBC applet driver has to be started on the Windows NT server. It has to be started with a TCP/IP port number that is not yet used in your system.

The command to start the JDBC applet driver with 6789 as the default port is:

```
DB2JSTRT [port no | 6789]
```

In our configuration we used the default port.

With this, the setup for the client and server is complete and the Friendly Foods e-business applications can start to be developed.

3.3 The Friendly Foods Web presence

To generate Friendly Foods' Web presence, the following Web pages had to be created:

- Static Web pages for providing general information about the company
- Dynamic Web pages for providing online information about the stores

This section describes the creation of the graphical layout of the Web pages.

3.3.1 The static entry Web page

As already mentioned, the Web designer was responsible for designing the graphical user interface for the Friendly Foods Web pages. He used NetObjects Fusion, which is part of WebSphere Studio. It is not the intention of this redbook to describe the use of NetObjects Fusion as site management and Web page development tool. Figure 18 and Figure 19 on page 24 show the entry Web page of Friendly Foods and the Web site hierarchy.

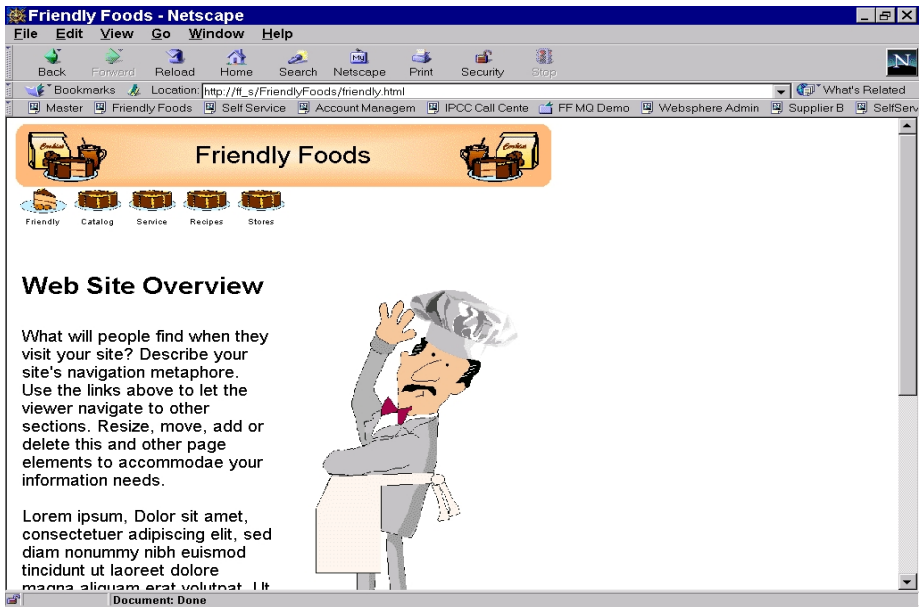


Figure 18. The Friendly Foods entry home page

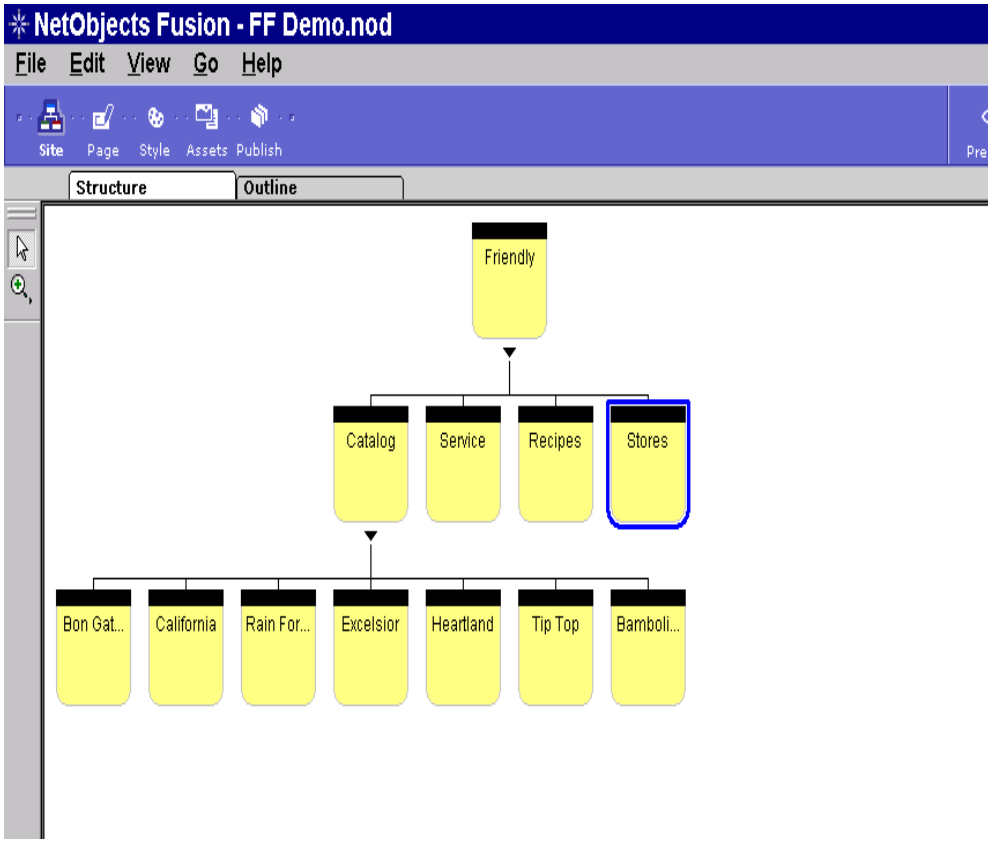


Figure 19. The Friendly Foods Web site hierarchy

The following sections show how Friendly Foods implemented the access to the DB2 database.

3.3.2 Creating the dynamic parts

To create the dynamic parts, the Web designer had to work together with the application developers. The task was to show a list with all worldwide stores retrieved from the DB2 database on VSE. They implemented this by providing a Java applet that is downloaded to the Web browser environment whenever needed.

3.3.2.1 Build the Java applet

The application programmer had to provide the JavaBean, which was then included in the Web page by the Web designer. As development tool he used NetObjects BeanBuilder, which is part of WebSphere Studio. NetObjects BeanBuilder should be installed and started on the workstation that is normally used by the application programmer. In our environment, this is the same Thinkpad that is used as Web server.

In order to create a JavaBean that is able to access the DB2 database on VSE and read the FFSTORES database, perform the following steps:

1. Open a new bean by selecting **File -> New.**
2. Select the white square at the upper left corner and change it to the size of the area you want to show on the Web site by clicking on the lower right corner

and dragging it. Figure 20 shows how your BeanBuilder workspace should look.

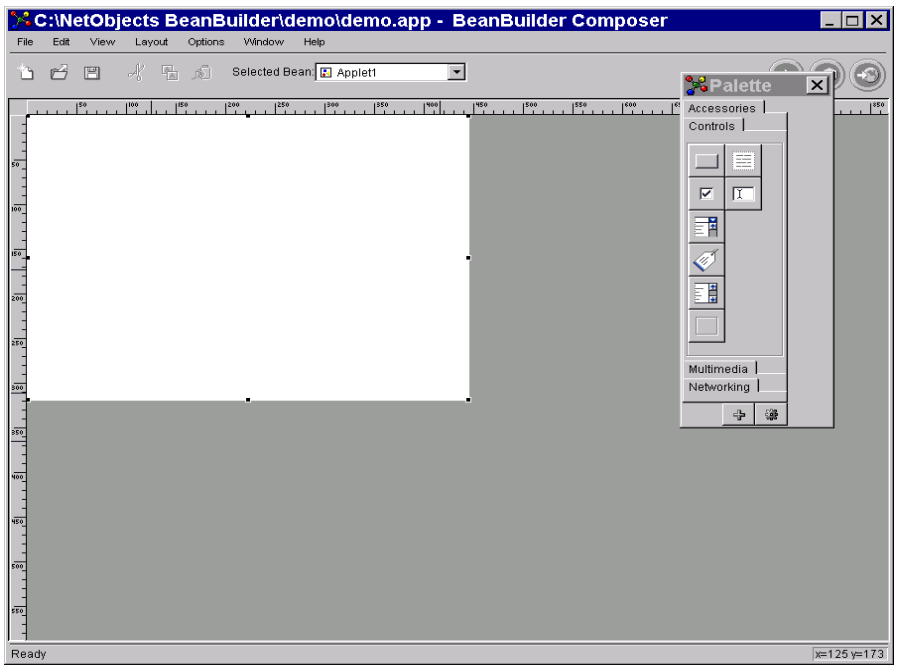


Figure 20. BeanBuilder entry screen

3. Click **Networking** in the Palette window and select the icon for the database, which is the first one in our installation. Figure 21 will be displayed in order to customize the database.

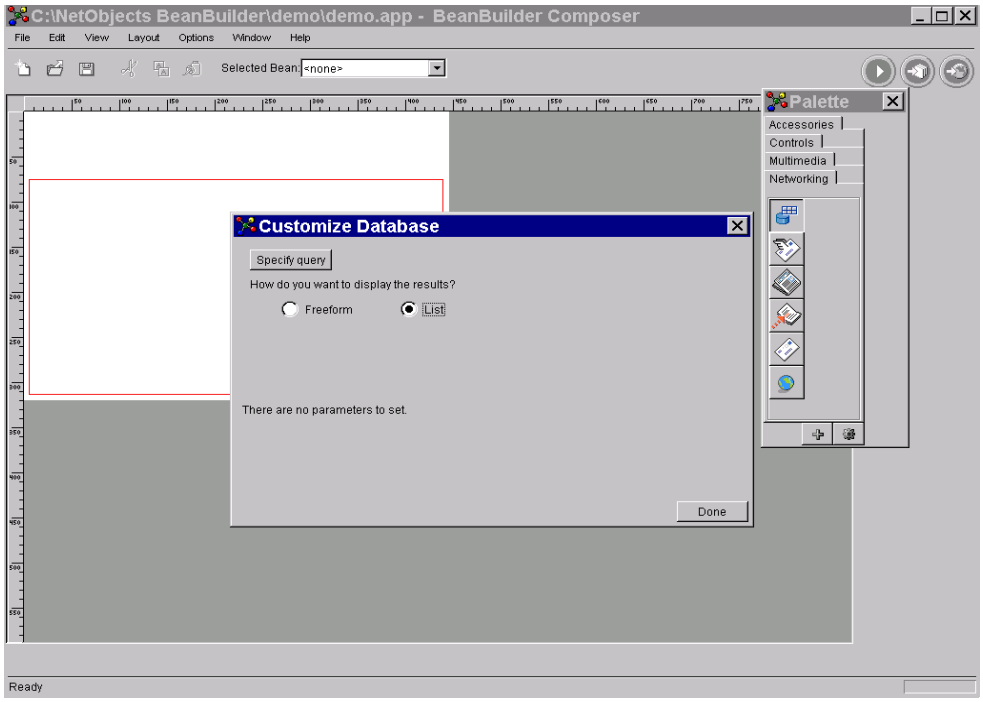


Figure 21. BeanBuilder -- customize the database

We selected **List** to display the results in list form. Click **Specify query** to continue, which starts the database wizard. You can now visually specify the SQL query. *No manual SQL coding is necessary in order to build the JavaBean.*

4. The panel shown in Figure 22 on page 27 is used to connect to your DB2 database on VSE. You have to specify the following values:

   a. The driver to connect to the database

      Select **IBM DB2 UDB remote** from the pull-down menu to indicate that you are accessing a remote database.

   b. The URL to be used to access the database

      The URL has to be specified in the following format: **jdbc:db2://[server]:**[**port_no**]/[**database**]**.** In our installation we had to specify the following values:

      1. **jdbc:db2**: This is the JDBC driver to be used. It is filled in by selecting the IBM DB2 UDB remote. *Do not change this value.*
      2. **server**: Specify either the symbolic name of your Web server or the specific IP address. In our case, it is ff_s, which is the symbolic name of our NT server with the absolute IP address of *9.164.165.209* as specified in the Windows NT Hosts file. Specify the IP address of your Windows NT server here, because the JDBC client will connect to the JDBC server installed on Windows NT.
      3. **port_no**: Specify the port number you used to start the JDBC applet server. If you have not specified a port number, the default 6789 will be used.
      4. **database:** Specify the name of the database as defined in the Client Configuration Assistant. In our configuration this is sqlds (see 3.2.2, “DB2 Connect and DB2 Client Application Enabler” on page 19).

      In our configuration the specified string is:

      jdbc:db2://ff_s:6789/sqlds

   c. The **Userid** and **password** of a user who is defined in your DB2 Server for VSE & VM to be able to access the SQLDS database

      Authorization for the table FFSTORES within the database SQLDS would be sufficient. In our configuration we are using the user ID HOEHN.

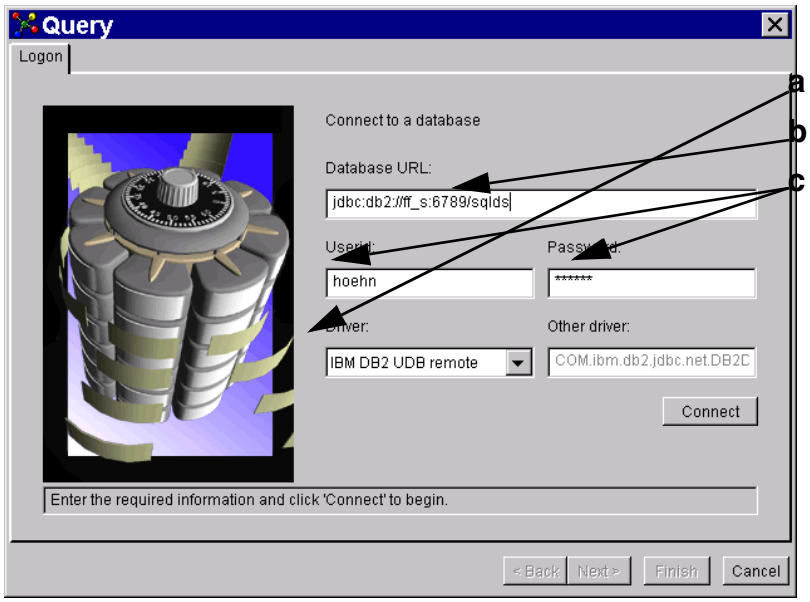


Figure 22. BeanBuilder -- specify the database query

Press **Connect** to establish a connection to your DB2 database on VSE. This step can take some seconds.

5. The next panel, Figure 23, shows all tables that are defined in your DB2 database. Select the table that you want to access from the JavaBean. In our case, this is SQLDS.FFSTORES. Click **Columns**.

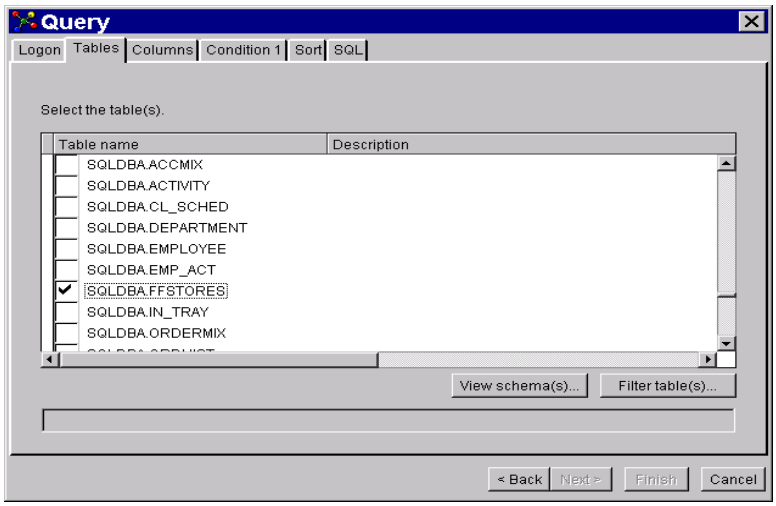


Figure 23. BeanBuilder - select DB2 table

6. Figure 24 on page 28 shows all columns that are defined in your DB2 table on VSE. Select the columns you want to display on your Web site and click **Add**. The right side will then show all selected columns. Click **Sort**.

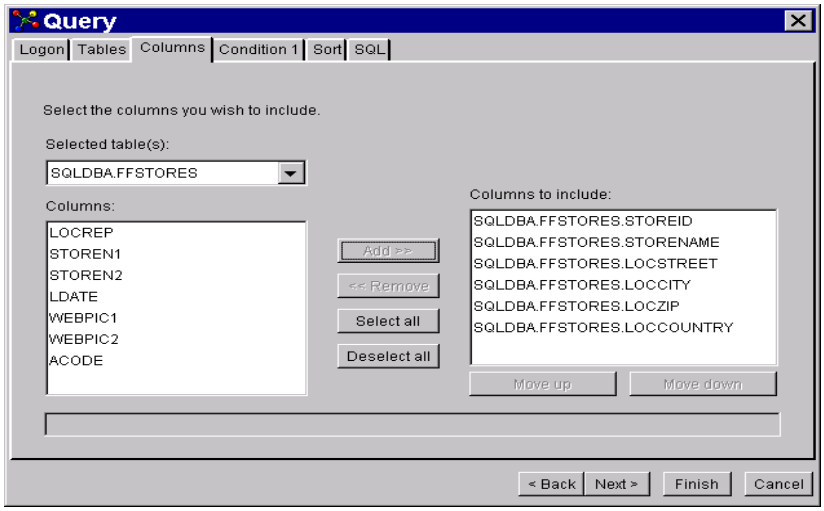


Figure 24. BeanBuilder -- select columns

7. If you want to sort your table according to specific columns, you can do this on this panel. In our example, we will sort the retrieved entries according to country.
8. If you are interested to see the SQL statements which are automatically generated by the wizard, click the **SQL** button.

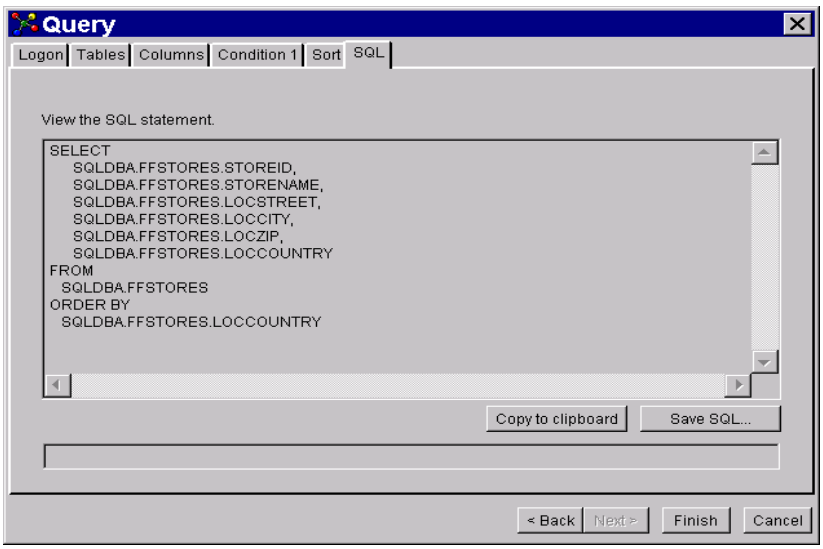


Figure 25. BeanBuilder -- generated SQL code

Click **Finish** to end the query specification. With this you will see a pregenerated table in your BeanBuilder workspace that already shows the titles of the columns (table names). See Figure 26 on page 29.

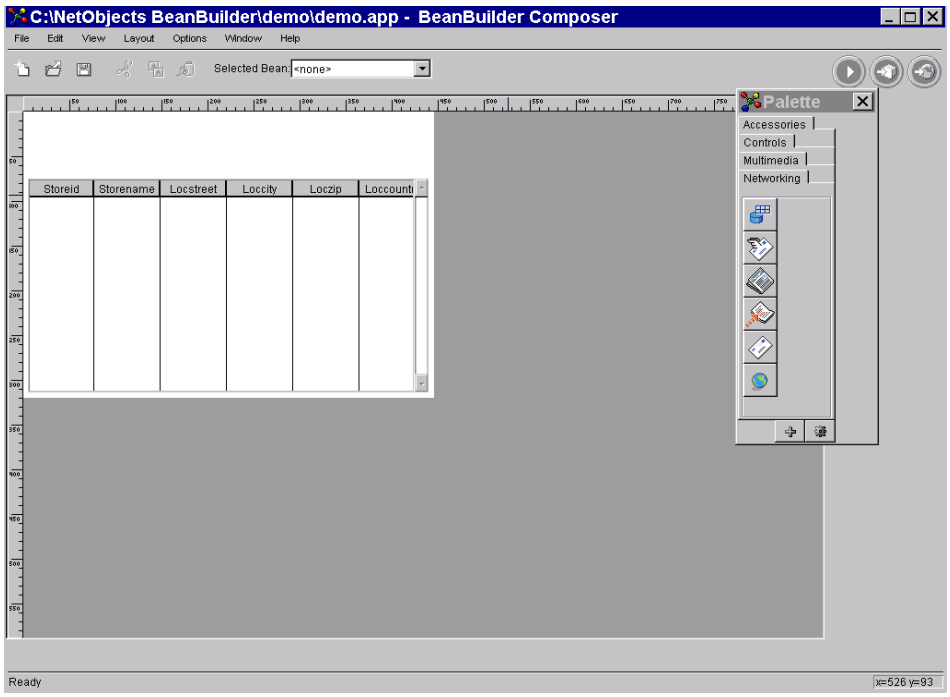


Figure 26. BeanBuilder -- workspace with database list

9. You can now add a button to the applet, which the end user has to click in order to fill the table. You could also decide to always automatically fill the table as soon as the Web page is invoked. We decided to add a button, but this is not described in detail in this redbook. If your choice is to directly fill the table with content as soon as the page is invoked, you are done now and you can save the JavaBean. In our environment, we called it demo.app.

   BeanBuilder also offers you the possibility to directly test your generated bean. Click **File -> Test** to start the test. The Java applet will be built and you will get a panel with the end result. See Figure 27 on page 30.

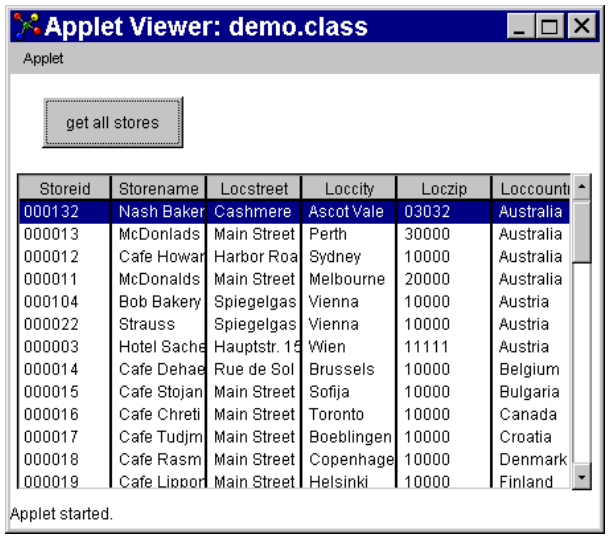


Figure 27. BeanBuilder - test

10. After a successful test you can publish the generated JavaBean to the NetObjects Fusion subdirectory in order to include it in one of your home pages. Select **File -> Publish.** The Publish Wizard will be activated. Specify the name under which you want to publish the applet on this panel. In our case it is demo. Click **Next**.

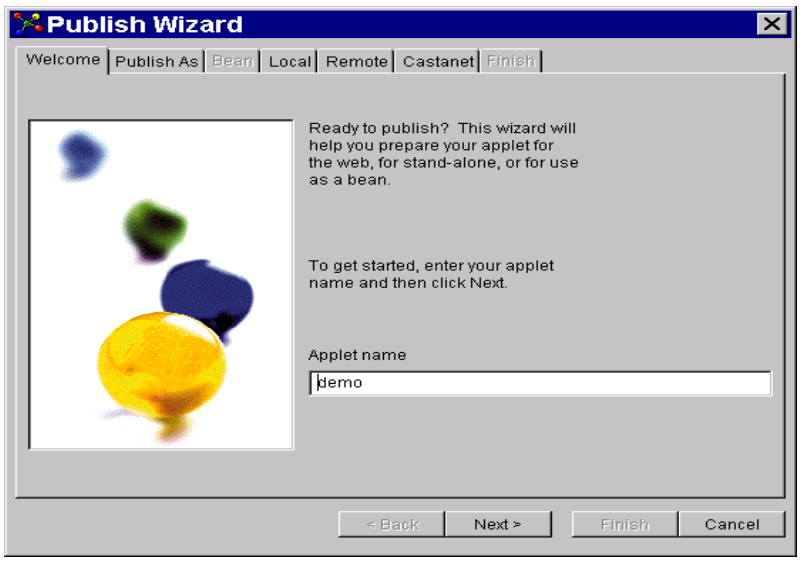


Figure 28. BeanBuilder -- Publish Wizard

11. Specify **As a bean** on the “Publish As” panel and click **Next**.
12. Specify the following values on the Bean panel (see Figure 29 on page 31):
    a. **Bean type**: **Applet** -- you want to publish this bean as a Java applet.
    b. **Description**: Any description you want to associate with the bean.

    Click **Next**.

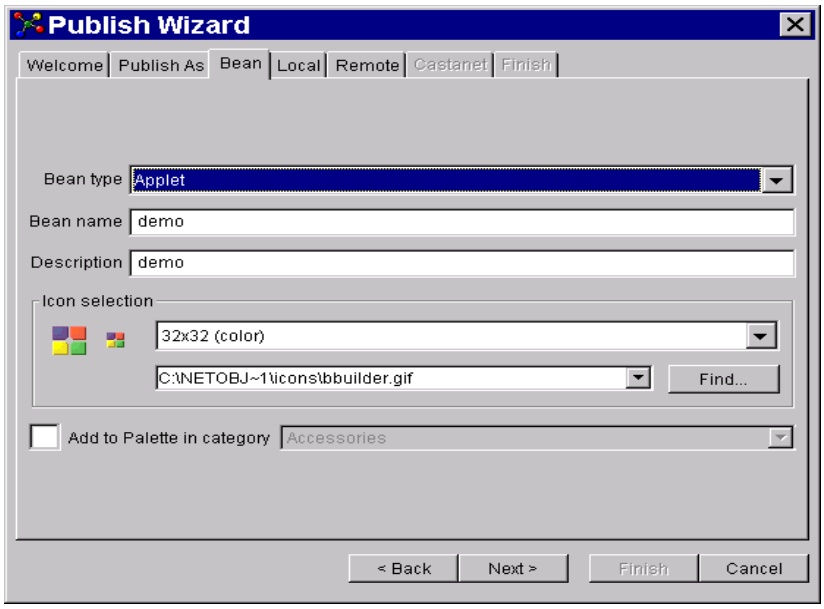


Figure 29. BeanBuilder -- Publish Wizard Bean panel

13. On the Local panel (Figure 30), we had to select that we want to publish locally to a subdirectory so that NetObjects Fusion would know the bean. In your environment, you might decide to publish it to a remote PC, which most probably would be the PC of the Web designer, who would then integrate the applet into a Web page. The next panels therefore are very specific to our environment. Click **Finish** after entering your definitions.

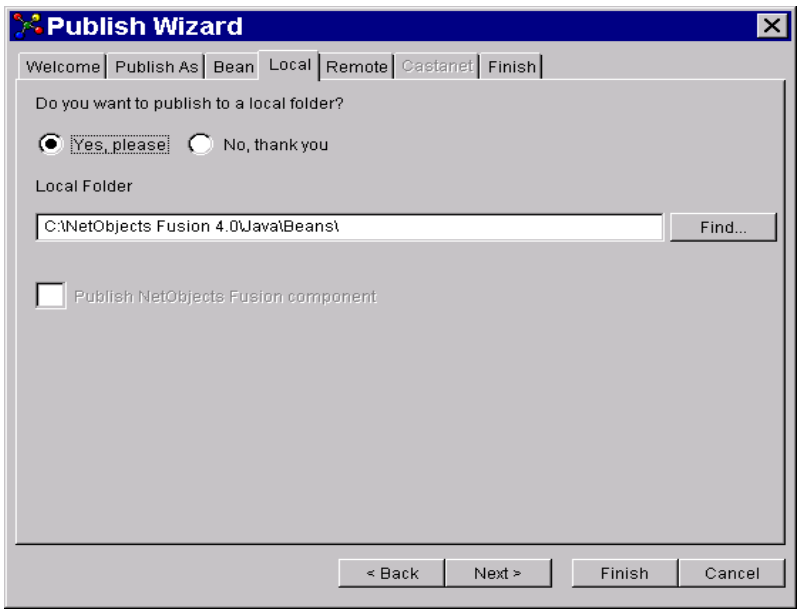


Figure 30. BeanBuilder -- Publish Wizard Local panel

14. After successful completion of this step, you are ready to include the generated applet into one of your Web pages.

3.3.2.2 Adding the generated Java applet to a Web page

The task of the Web designer is now to add the generated applet to one of the Web pages. To do so, go to NetObjects Fusion and select the Web page where you want to have the information displayed. In our case, we added the applet to a new empty Web site.

To add the applet, click the JavaBean icon on the left side of the NetObjects Fusion workspace (see Figure 31), hold the mouse button until you have selected the BeanBuilder icon that will appear in a pull-down selection window. NetObjects Fusion will automatically reserve space for the applet, as you can see in Figure 31. Select the bean that you want to include in the Web page. In our environment, it is demo.jar.

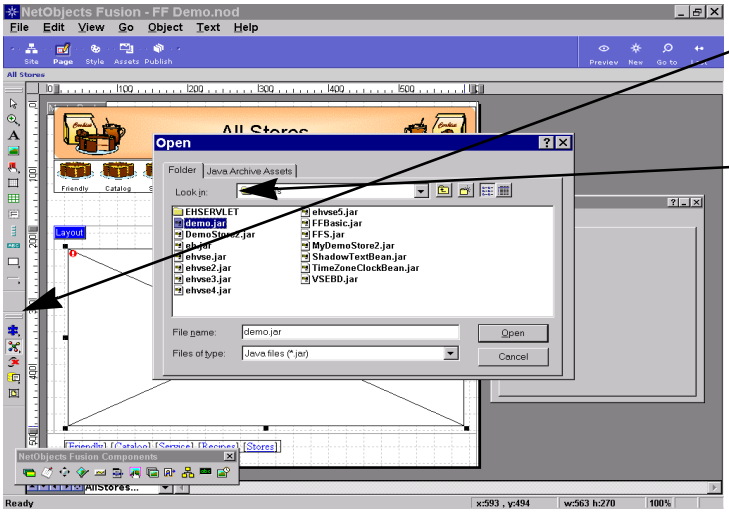


Figure 31. NetObjectsFusion -- include the generated bean

Figure 32 shows that the bean has been included. You can now add static text to the Web page or include another bean. We did not add more content to the Web page.

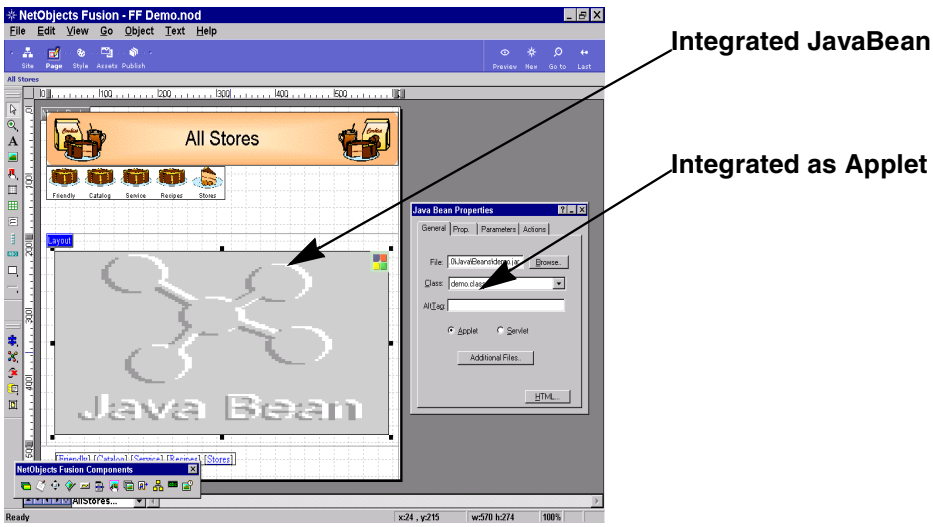


Figure 32. NetObjectsFusion -- the generated bean is included

Now you can publish the generated Web page (after also having it included in the site hierarchy) to your Web server. Again, in our configuration, the publish process has to write it to a local disk. In a real environment, the publish process would send the Web site to the NT server running WebSphere Application Server.

After having published the site, the customer's Web browser displays the panel shown in Figure 33.

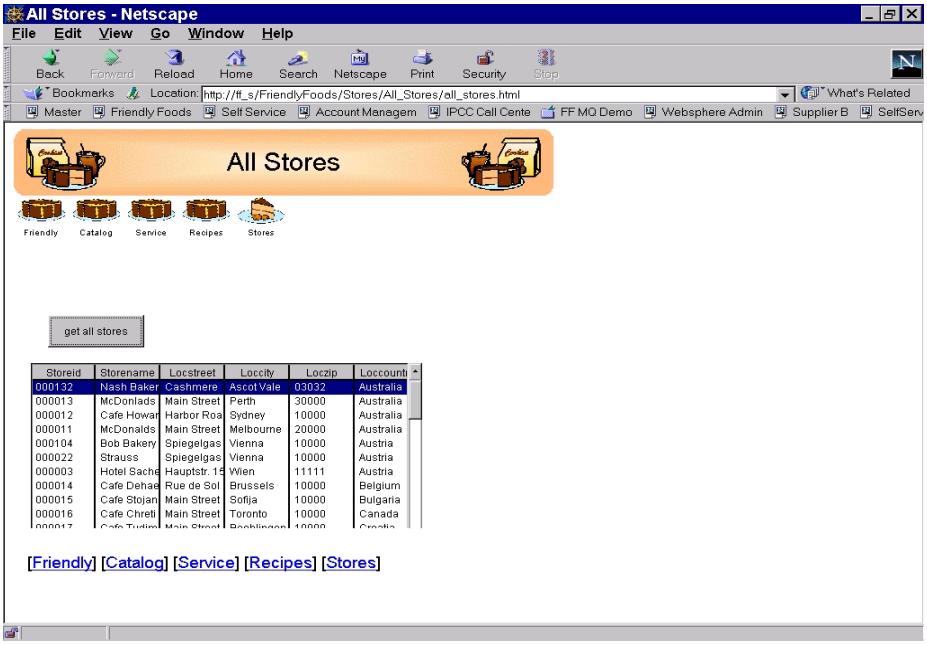


Figure 33. Final Friendly Foods Web page

3.4 Consumer marketing -- summary

Friendly Foods was able to quickly deliver the first part of the project because they successfully used the visual tools. The application programmer quickly got productive and was able to provide the JavaBean to bring dynamic content to the Friendly Foods Web pages. There was no need for the programmer to learn Java because to create this simple JavaBean, the BeanBuilder generated all necessary Java code.

Chapter 4. Account self-service and account tracking

Consumer marketing helped Friendly Foods expand its business. Not only did it increase awareness of their products worldwide, this new application also supported the goal to increase the number of stores and achieve broader market coverage by implementing such services as the store finder.

Because of the fast growth in the number of stores, new, more cost-effective structures were required to support the stores. Therefore, Friendly Foods established a new account self-service and account tracking application with the goal to:

- Reduce the amount of administrative work per store at Friendly Foods headquarters
- Increase the flexibility of the individual stores

The new application is implemented according to the IBM Application Framework for e-business. It enables the stores to review account information, retrieved from the accounting database on VSE, at the most current level. It also enables the stores to view the actual order mix they are receiving on a daily basis and to update quantities to reflect growing demand. This information is also directly retrieved from and updated in the DB2 database of the Friendly Foods production and distribution control system on VSE/ESA.

4.1 WebSphere Studio V3

The Friendly Foods application development department used WebSphere Studio V3 to build this application. More information about WebSphere Studio can be found in 2.3.3, “WebSphere Studio” on page 12.

4.1.1 Installation and configuration

The installation of WebSphere Studio is pretty straightforward and in most cases there is not much customization work (at least for environments like ours where WebSphere Studio and WebSphere Application Server are on the same machine).

We downloaded WebSphere Studio Version 3 beta code from the IBM Software Web site. As default, WebSphere Studio installs in a new subdirectory in the WebSphere folder. During installation some of the settings of the WebSphere Application Server become part of the Studio initial settings. In our case, the document root directory and servlet directory were detected by the install process, which means that no additional customization had to be done when publishing our projects from WebSphere Studio to the local WebSphere Application Server.

4.2 The user interface -- Java Server Pages (JSP) considerations

Consumer marketing was implemented by using Java applets, which are downloaded to the Web browser and executed in the Web browser Java Virtual Machine environment.

For the account self-service and account tracking application, Friendly Foods decided to use a different approach, namely the use of server-side Java logic by implementing Java Server Pages (JSPs). A JSP in its basic form is an HTML Web page that contains additional statements that execute application logic to generate dynamic content. JSPs are an extension of the Java Servlet API and are compiled into Java servlets before they are used. These compiled servlets retrieve the needed data from the DB2 database on VSE and return it to the Java Server Page for formatting. The major advantages of this implementation are the following:

- All Java code is executed on the Web application server.
- It allows a complete separation of business (program) logic and presentation logic.

Figure 34 shows how JSP servlets working with DB2 are implemented. Compared to the implementation of consumer marketing (see Figure 9 on page 17) there is no additional support required on the client.

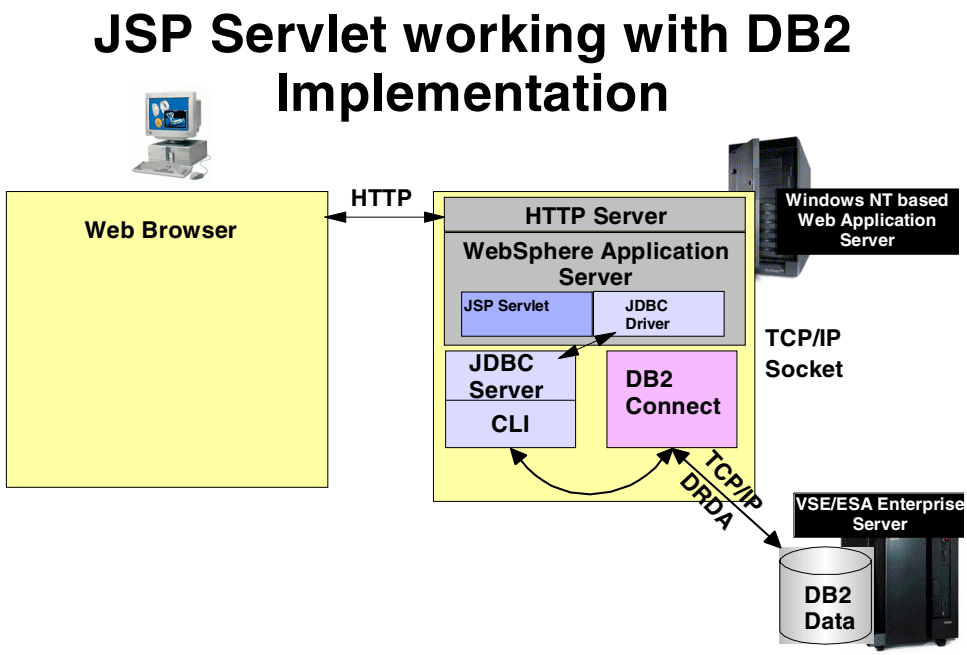


Figure 34. Implementation of JSP servlets working with DB2

4.3 Implementation

We now describe these steps for the creation of the account self-service and account tracking application:

- Create a simple Web page with WebSphere Studio.
- Use NetObjects Fusion for the final page design.
- Integrate the pages with NetObjects Fusion.

4.3.1 Creating JSPs with WebSphere Studio

As mentioned before, Java Server Pages are executed in the secure environment of a Web application server. This implies that the data structures of the involved DB2 databases are not used in code that is executed outside of their well-controlled and secure environment.

The base of the account self-service and account tracking application consists of many database query and database update beans. Each of these beans is linked with some input and output fields.

Rather than explaining in detail how each of the beans and Web pages is created, we focus on the creation of one sample Web page: the logon page. However, we do give you some hints and tips on the specifics of other parts comprising the account self-service and account tracking application.

During the logon process, store users are asked for their user ID and password, which are compared with the values stored in the DB2 store database on VSE. If the given information is valid, a Web site with basic store information is displayed.

4.3.1.1 WebSphere Studio -- preparation for a new project

Within WebSphere Studio we created a new project called Demo1. By generating a new folder with that name (Demo1), we ensured that all generated Web pages (HTML files) are published into a folder with the same name (Demo1) in the directory structure associated with the HTTP server.

All servlet elements are published into a folder with the project name in the directory structure assigned as the default servlet directory to the WebSphere Application Server. Using the same name ensures that all elements related to this project reside in the following two folders with the same name:

- One under the HTTP server
- One under the WebSphere Application Server

As a result we had to move the theme folder (created by default) to the Demo1 folder. The theme folder contains the style sheets that we used for the layout of our Web pages.

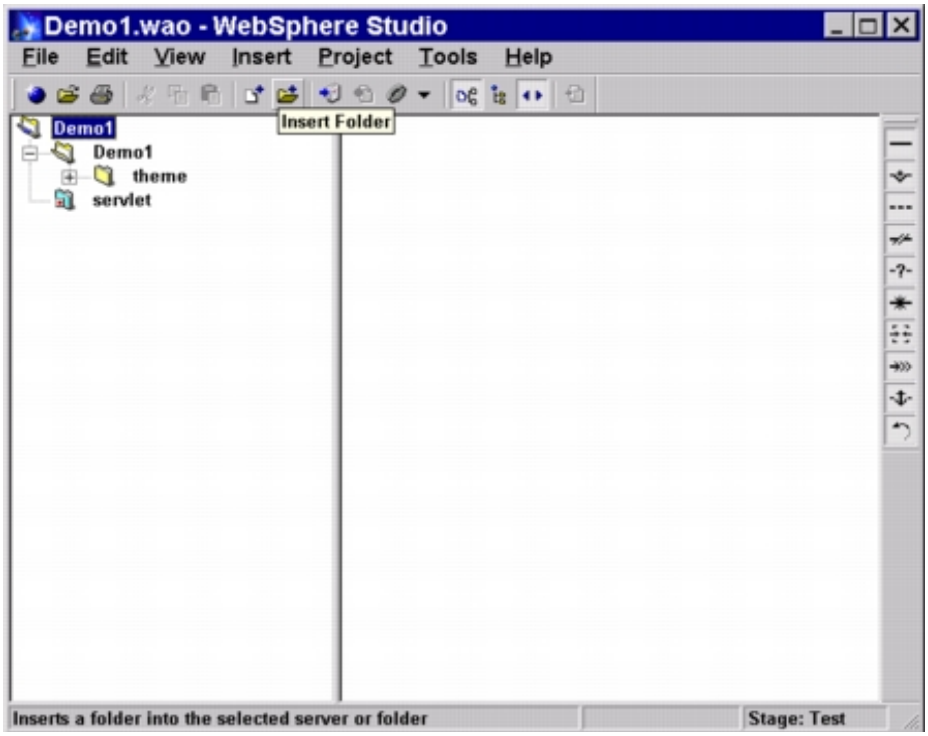


Figure 35. WebSphere Studio: create a project

4.3.1.2 WebSphere Studio -- create the database query bean

For the creation of the database query bean, we used the SQL wizard, which can be found under the Tools/Wizards action menu. Before starting the wizard, make sure that the project folder is selected (see Figure 35).

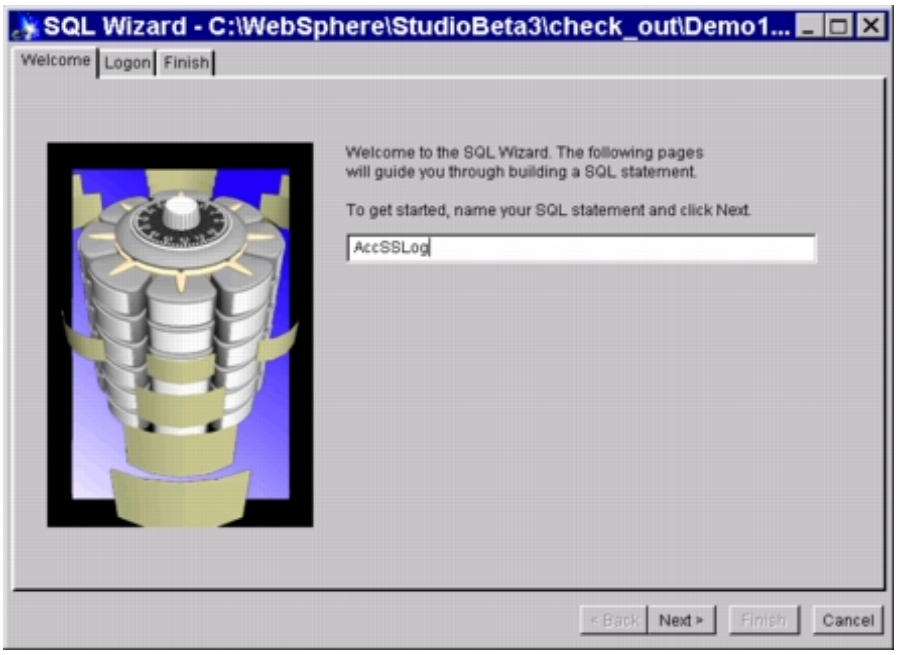


Figure 36. WebSphere Studio: using the SQL wizard

The SQL wizard offers a dialog similar to that of the NetObjects BeanBuilder as described in 3.3.2, “Creating the dynamic parts” on page 24.

To create the database query bean, perform the following steps:

1. Establish a connection to the DB2 database on VSE. Refer to Figure 22 on page 27 for a discussion of the parameters to specify on the logon screen. After pressing **Connect**, the wizard contacts the DB2 database on VSE and provides a list of all tables as defined in the DB2 database. The basic store information is kept in the table SQLDBA.FFSTORES.
2. On the Columns screen, select the columns you want to display on the Web page in addition to all columns you might need for internal comparisons or calculations. We selected the store name and all columns containing the address information of the store. In addition, we needed the STOREID and the fields WEBPIC2 and ACODE (authorization code) later on in the application.

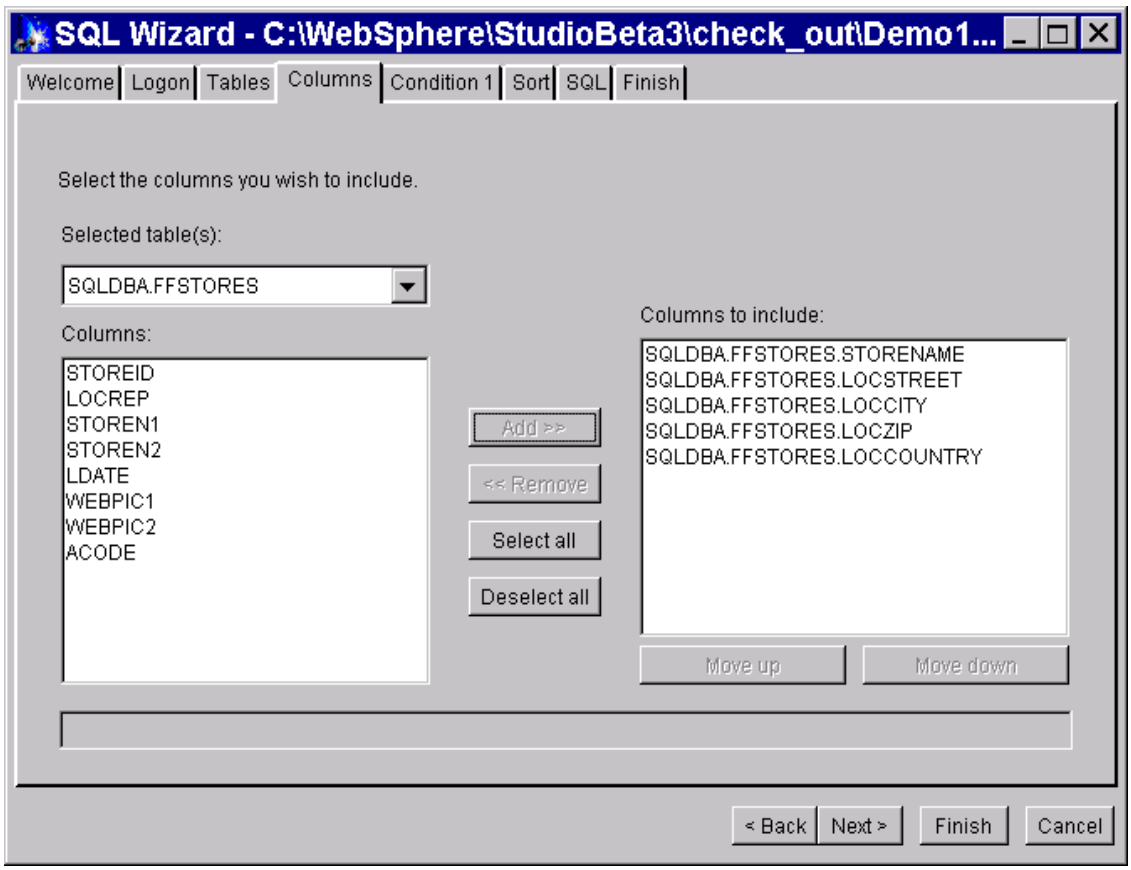


Figure 37. WebSphere Studio -- SQL wizard: defining the output colums

3. Define the parameters that are used as input to this query on the Condition 1 screen. Since we want to verify that a store user is authorized to access the store information, we ask for the store ID and the authorization code (password).
   - STOREID has to be exactly equal to the input parameter. By pressing **Parameter**, you get another window where you can specify the name of the parameter, *inUID* in our case.

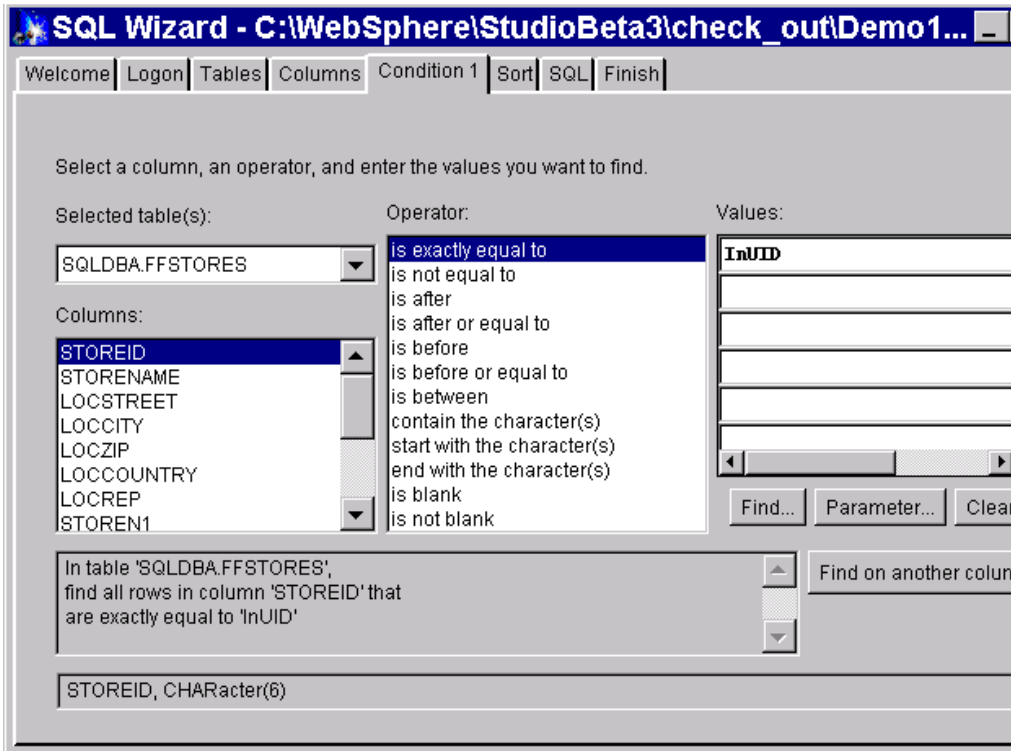


Figure 38. WebSphere Studio -- SQL wizard: defining the conditions for the query

4. Press **Find another Column**, which allows you to do the setup for a second parameter. On the panel for this second parameter we specified that both conditions are AND connected and that the field ACODE (in the DB2 database) has to be exactly equal to the input field InPW.
5. Click **Finish** to generate the database query bean. Back on the project page in WebSphere Studio you will find the SQL Query Bean added to our project under the servlet folder (AccSSLogDBBean).

4.3.1.3 Create the input and output Web pages

With the query bean created, we can now start to specify the input and output Web pages. We used the Database wizard in WebSphere Studio to create the Web pages.

Note: The Database wizard helps you create all input and output Web pages that are needed to work with a specific query. The SQL wizard is used to define the details for a specific database query.

To create the input and output pages, you have to perform the following steps:

1. Start the Database wizard from the Tools/Wizards action menu. You will see a list of SQL queries defined in this project.
2. Select the SQL Query Bean for which you want to build the input and output pages (AccSSLogDBBean).
3. Specify the pages you would like to have generated on the next panel. To keep our example simple, we just select input and result pages.

   However, you should keep in mind that for the final application you should also select the error and no data pages because your definitions for those

pregenerated pages will trigger the structure of the servlet control file (see Figure 42 on page 43).

In our example the “no data returned” condition will most probably mean that the user ID does not exist or the password is wrong. Therefore, you would let Studio generate this page and later replace it with a page according to your overall site design.

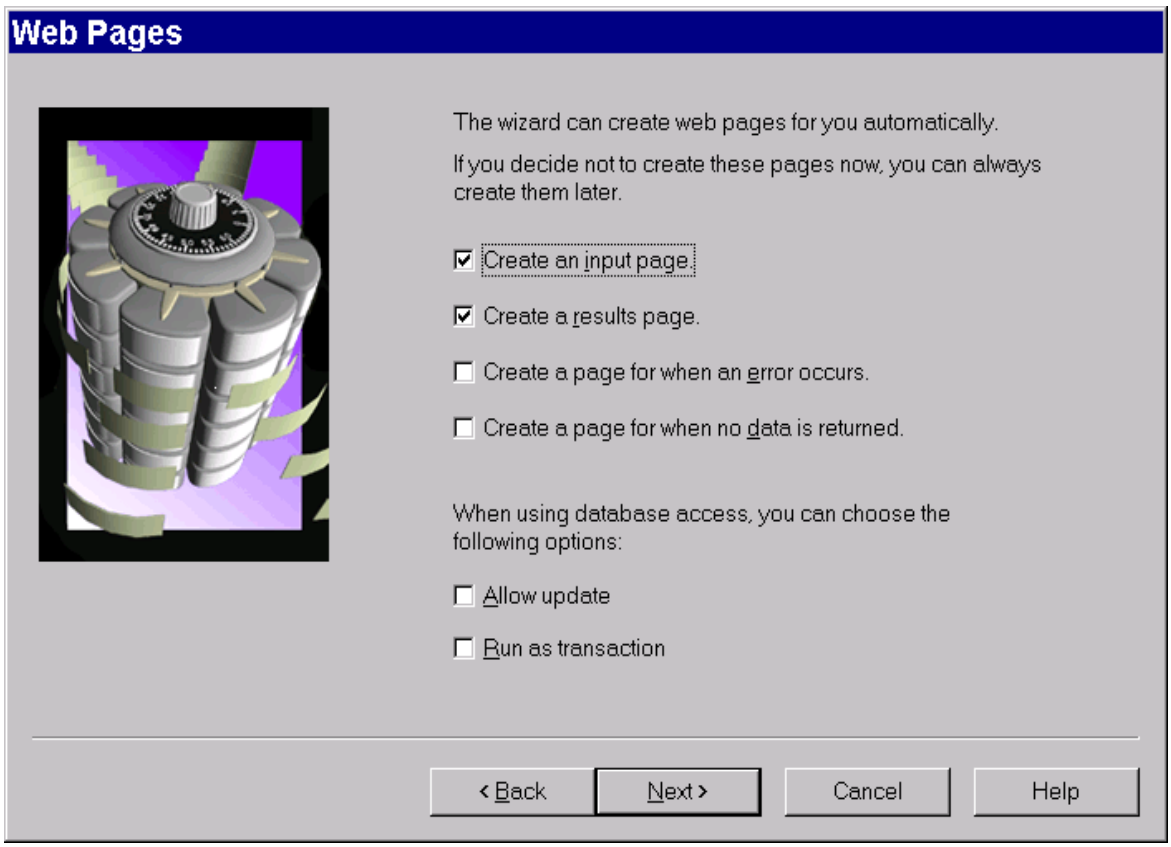


Figure 39. WebSphere Studio: using the Database wizard

4. The next panel displays all the columns selected in the database query and allows you to define the columns that should be displayed on the result page.
5. Follow the Database wizard through some more pages dealing with, for example, the naming of the generated pages.
6. Complete this sequence by clicking **Finish** and return to the WebSphere Studio project view.

The right side of the Project view in WebSphere Studio shows the element in focus (selected in the left side hierarchical view) and how it relates to other elements of this project. For example, in Figure 40 on page 42 we have the input page in focus, and this page has a direct link to the style sheet (Master.css) -- all pages are linked with this control file containing general settings like background color -- and an embedded link to our servlet for this first part of the application (AccSSLog.servlet).

By using the “+” symbol next to an element you can navigate within the project. For example, selecting the “+” next to the servlet puts the servlet in focus, showing all elements that have a relation to the selected element.

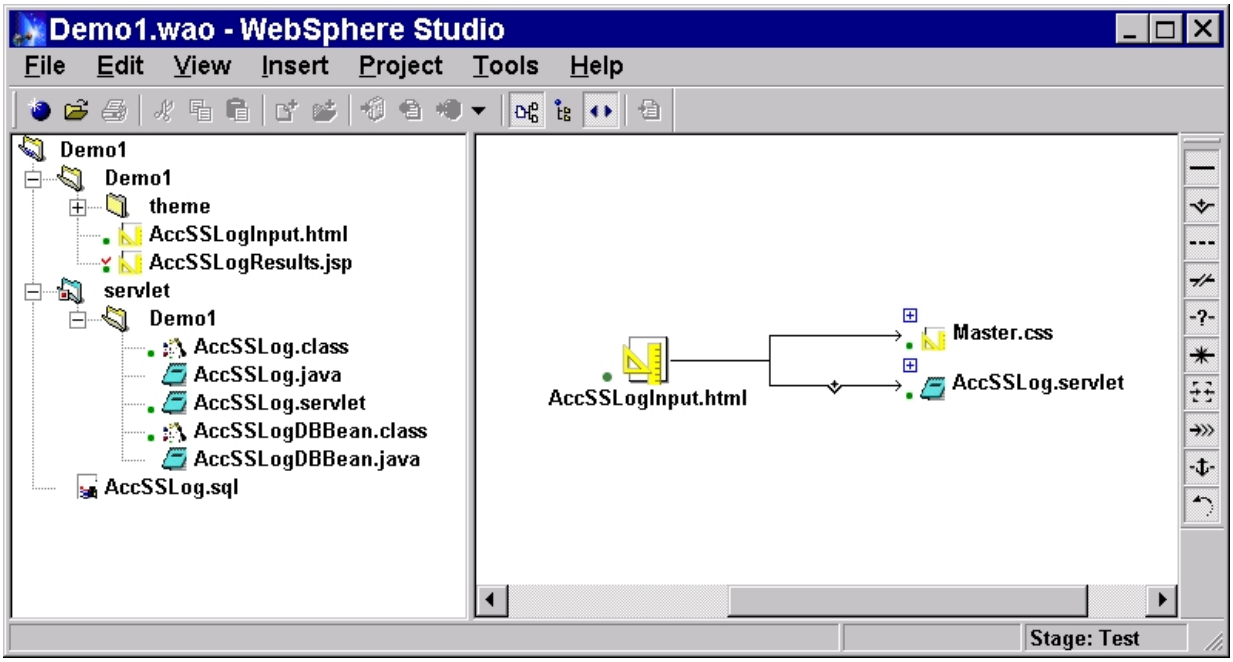


Figure 40. WebSphere Studio: Project View - focus on input page

Selecting the output page (AccSSResults.jsp) will show you that this page is triggered by the servlet (symbol on the left side of the arrow pointing to the output page, not completely shown in this view). The output page also has a link to the style sheet and an embedded link to the database access bean (AccSSLogDBBean.class) used to retrieve the data.

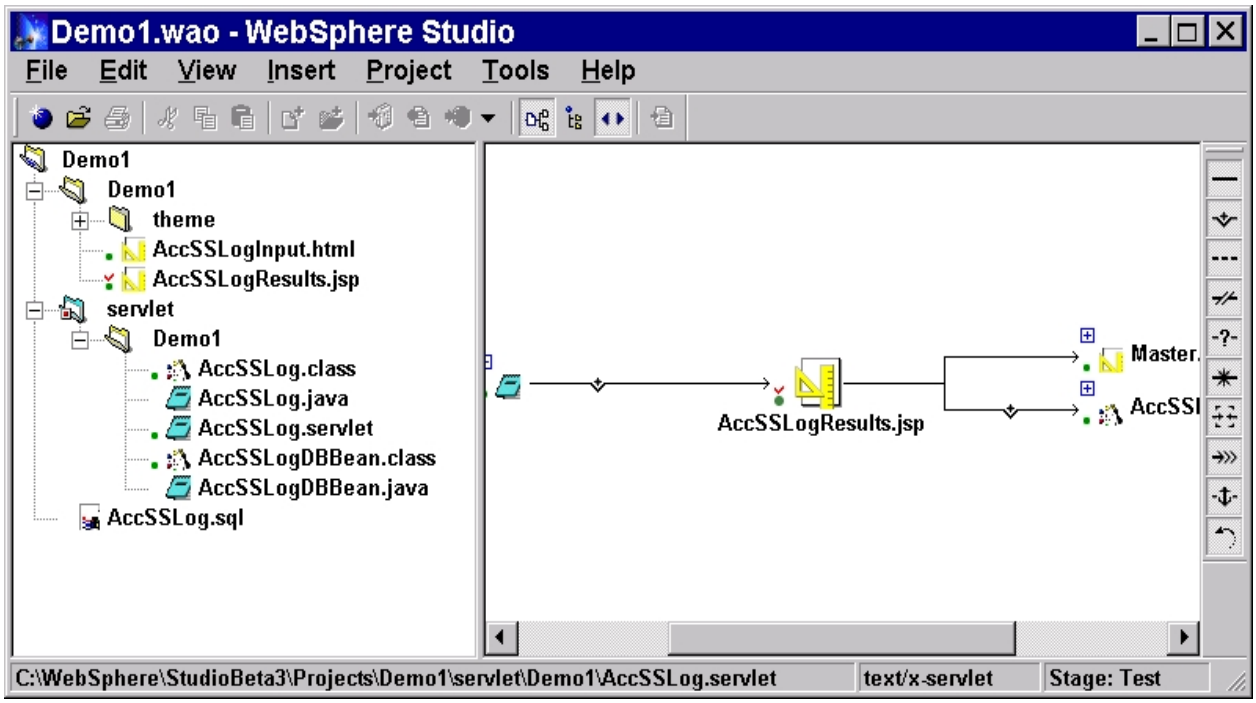


Figure 41. WebSphere Studio: Project View -- focus on output page

Figure 42 shows a servlet control file (AccSSLog.servlet) generated by WebSphere Studio. This version of the servlet also contains the controls to trigger an error page and a “no data returned” page.

```
<?xml version="1.0"?>
<!-- This file was generated by IBM WebSphere Studio 3.0.0 using
C:\WebSphere\StudioBeta3\BIN\GenerationStyleSheets\AppServerV2\JSP0.91\WebSphere\S
ervletConfig.xsl-->
<servlet>
    <page-list>
      <default-page>
        <uri>/Demo1/AccSSLogResults.jsp</uri>
      </default-page>
      <error-page>
        <uri>/Demo1/AccSSLogError.jsp</uri>
      </error-page>
      <page>
        <uri>/Demo1/AccSSLogNoData.jsp</uri>
        <page-name>com.ibm.webtools.runtime.NoDataException</page-name>
      </page>
    </page-list>
    <code>Demo1.AccSSLog</code>
    <init-parameter value="COM.ibm.db2.jdbc.net.DB2Driver" name="driver"/>
    <init-parameter value="dyf390" name="password"/>
    <init-parameter value="jdbc:db2://ff_s:6789/sqlds" name="URL"/>
    <init-parameter value="hoehn" name="userID"/>
</servlet>
```

Figure 42. WebSphere Studio-generated servlet control file

4.3.1.4 Test the generated parts

We now have the database query and a first draft of the Web pages, and we want to test this as part of the application in the WebSphere Application Server environment. Therefore, we have to publish our project with the options set as shown in Figure 43.

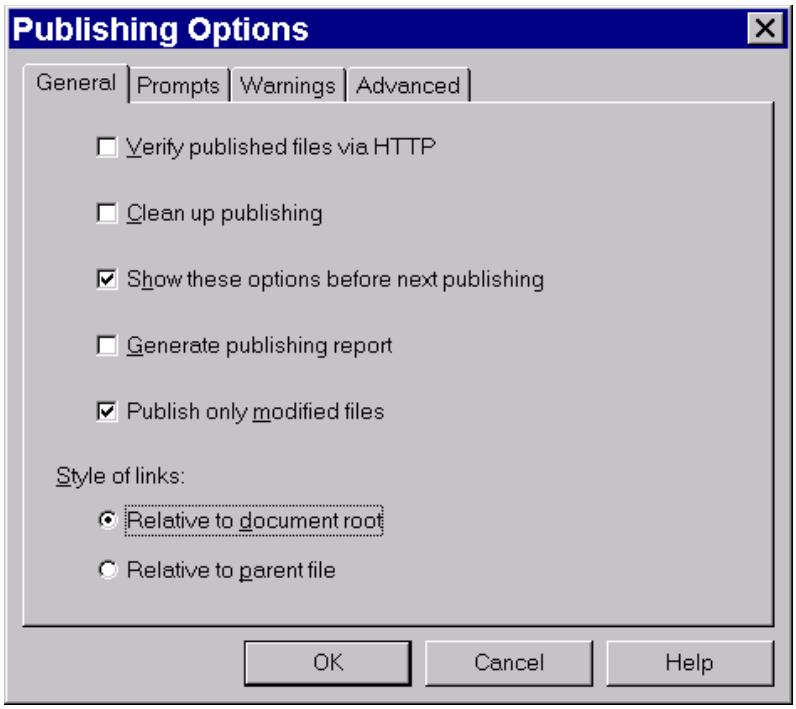


Figure 43. WebSphere Studio: setting the publishing options

After publishing the pages we can invoke them from the Web browser. Figure 44 on page 45 shows the generated input Web page.

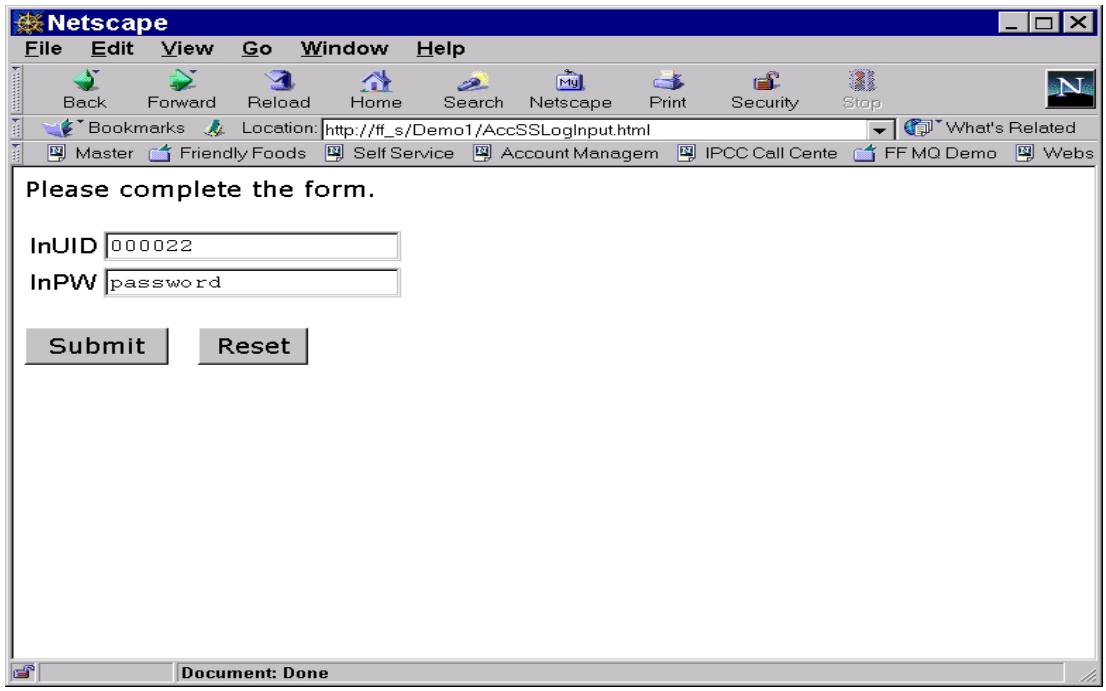


Figure 44. WebSphere Studio: test the generated input page

Using the correct user ID and password for one of the stores in the DB2 table will display the result page shown in Figure 45 on page 46. This is not the final layout for the pages Friendly Foods would like to use as an entry point for the connected stores.

Therefore, we use NetObjects Fusion to give our pages the same look and feel as the consumer marketing pages. The easiest way to achieve this is to imbed the generated page into the NetObjects Fusion page. This, however, does not allow further manipulation of the imbedded section itself, but only of the surrounding sections. Therefore, we first finalize the result page before importing it into NetObjects Fusion.

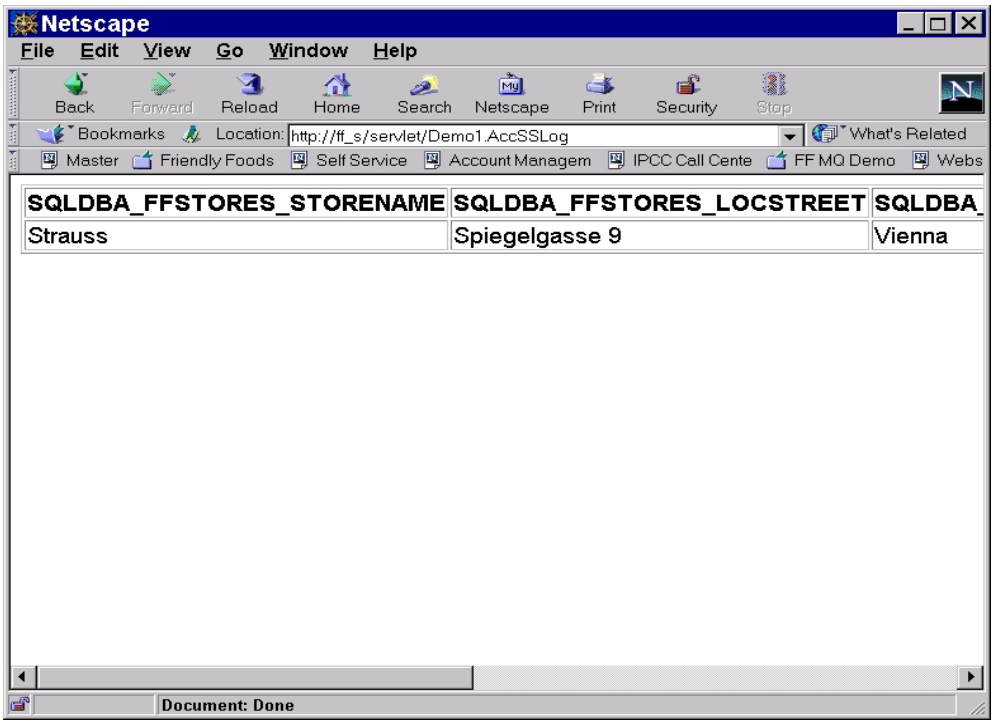


Figure 45. WebSphere Studio: viewing the generated result page

4.3.1.5 Change the layout of the generated table

To change the layout of the generated table we used WebSphere Page Designer, which is part of WebSphere Studio.

WebSphere Page Designer is an advanced-function HTML editor that makes it possible to quickly build complex Web pages, both visually and textually. However, since it is designed to build single pages and does not include site management capabilities like NetObjects Fusion, we recommend to use it primarily for complex pages or page parts that you then integrate with NetObjects Fusion.

Figure 46 on page 47 shows the graphical view of the result page in WebSphere Page Designer.

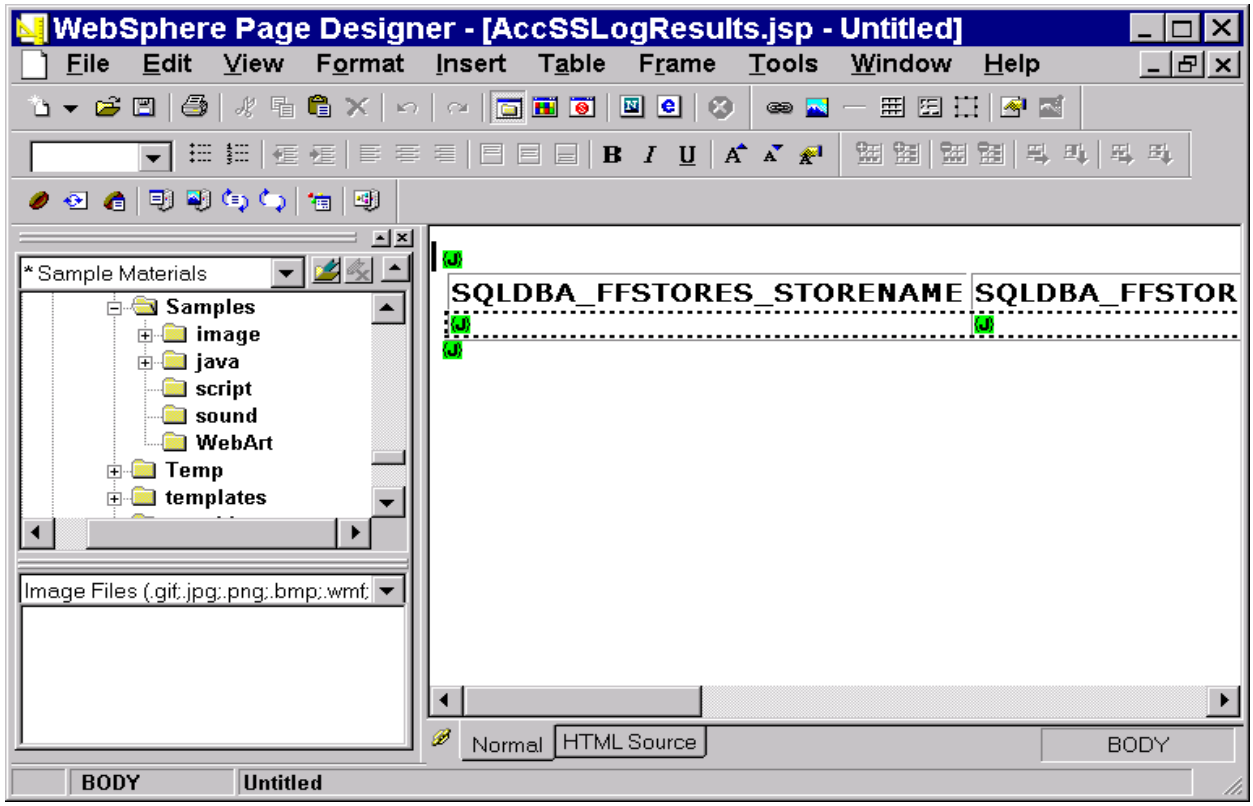


Figure 46. WebSphere Studio: using the Page Designer to modify the result page

The generated table displays the results on one single line and therefore would force the end user to scroll through the result, so we decided to display the result table in the more complex form shown in Figure 47 on page 48.

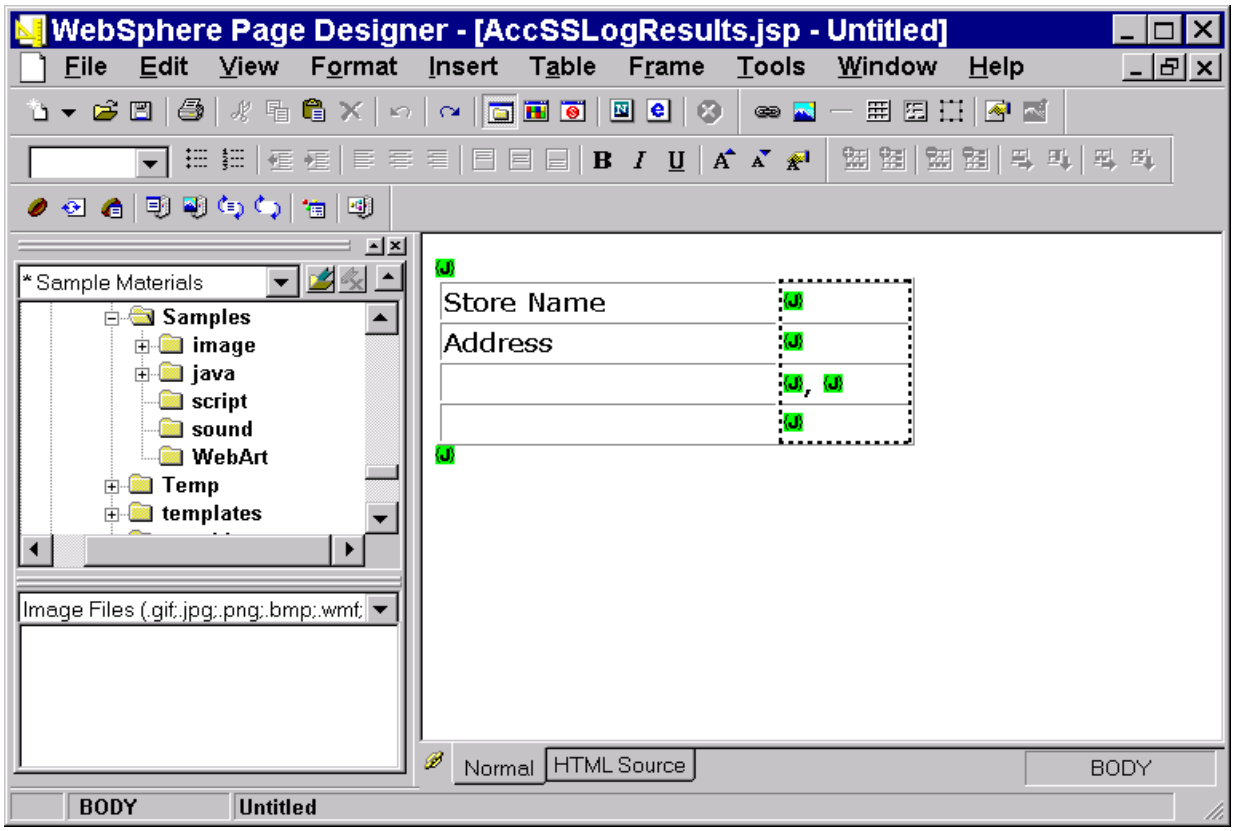


Figure 47. WebSphere Studio Page Designer: final layout of the result page

The small rectangles labled with “{J}” represent small pieces of Java code. The one above the table makes the database bean acccessible, the one below the table closes the result, and the other ones in the table cells will retrieve the value for this particular cell element from the database bean. An expert user might copy one of these cell elements, then delete the entire table, replace it with a more proper table, and reconstruct the necessary cell elements from the one kept. But the easiest way is to work with simple cut and paste. Save all cell elements outside the table, adjust table layout, and move each of the cell elements to the proper space in the modified table.

As mentioned before, this was just an example showing how to create a small part of the application Java Server Pages with Websphere Studio. All other parts were built and tested in a similar way. Figure 48 on page 49 shows the final WebSphere Studio project view for this application:

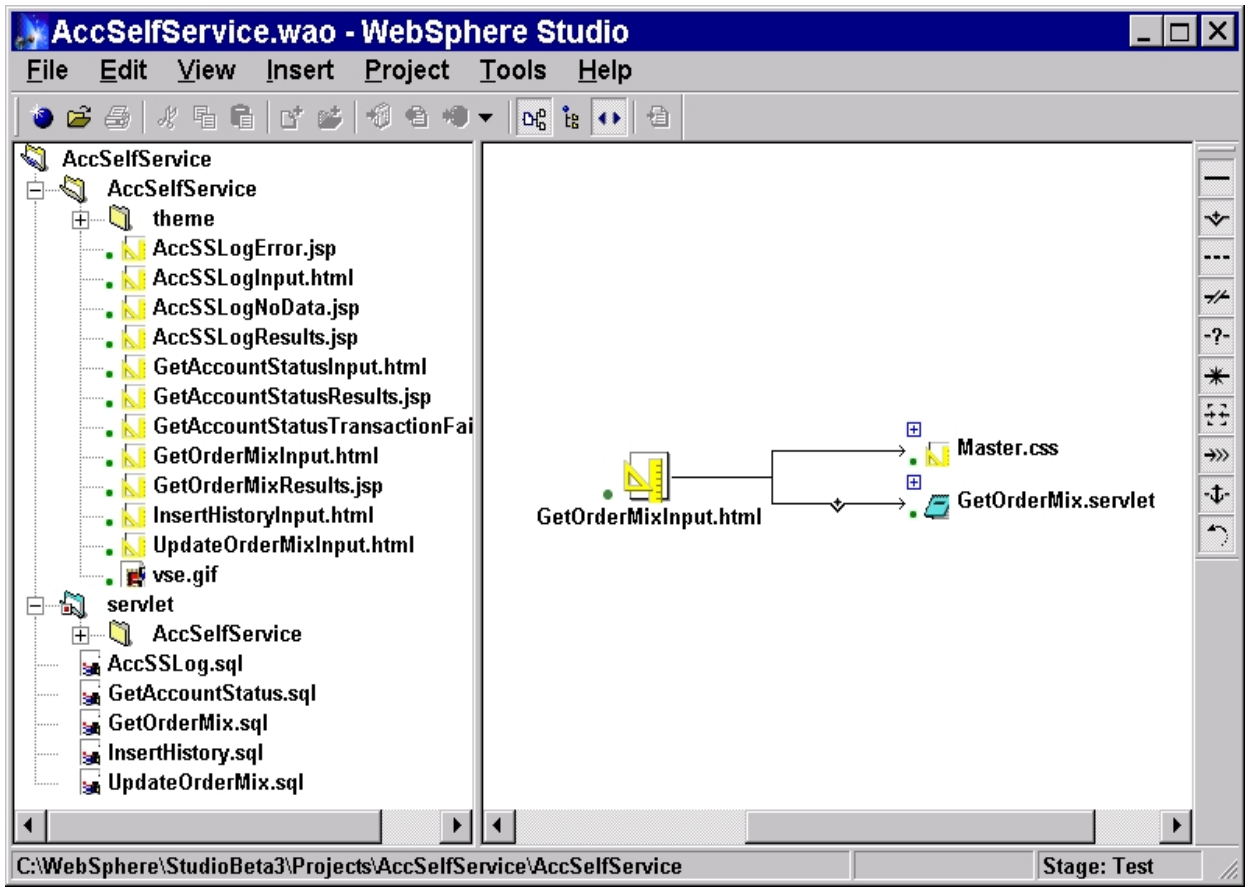


Figure 48. WebSphere Studio: Project View - complete account self service project

Since Friendly Foods used NetObjects Fusion to create the consumer marketing pages, it wanted to have the same look and feel for all other pages including the Java Server Pages for the store users. There are two ways to achieve this:

1. Use WebSphere Studio for the final page design.

   Import the design elements used in the consumer marketing section into the pages created by WebSphere Studio.

2. Use NetObjects Fusion for the final page design.

   Import the pages created by WebSphere Studio as section in a NetObjects Fusion design.

Both alternatives have their advantages and disadvantages. The first alternative might be the right choice if you consider your design as fairly stable with more focus on changing the logic (changes in the query, displayed columns, and so forth).

The second alternative might result in more effort to finish the site the first time, but you will benefit from this work when you have to perform design changes on the Web site. Friendly Foods decided to go with the second alternative to take advantage of the site management capabilities of NetObjects Fusion.

4.3.2 Final servlet pages created by NetObjects Fusion

Figure 49 shows the final input page after the final page design in NetObjects Fusion was completed.

Figure 49. Friendly Foods account self-service Login page

This page is the entry point for a store user. The code for the HTML form section was a page simply imported from the page generated by WebSphere Studio (AccSSLogInput.html).

Entering a valid user ID and password results in the page shown in Figure 50 on page 51.

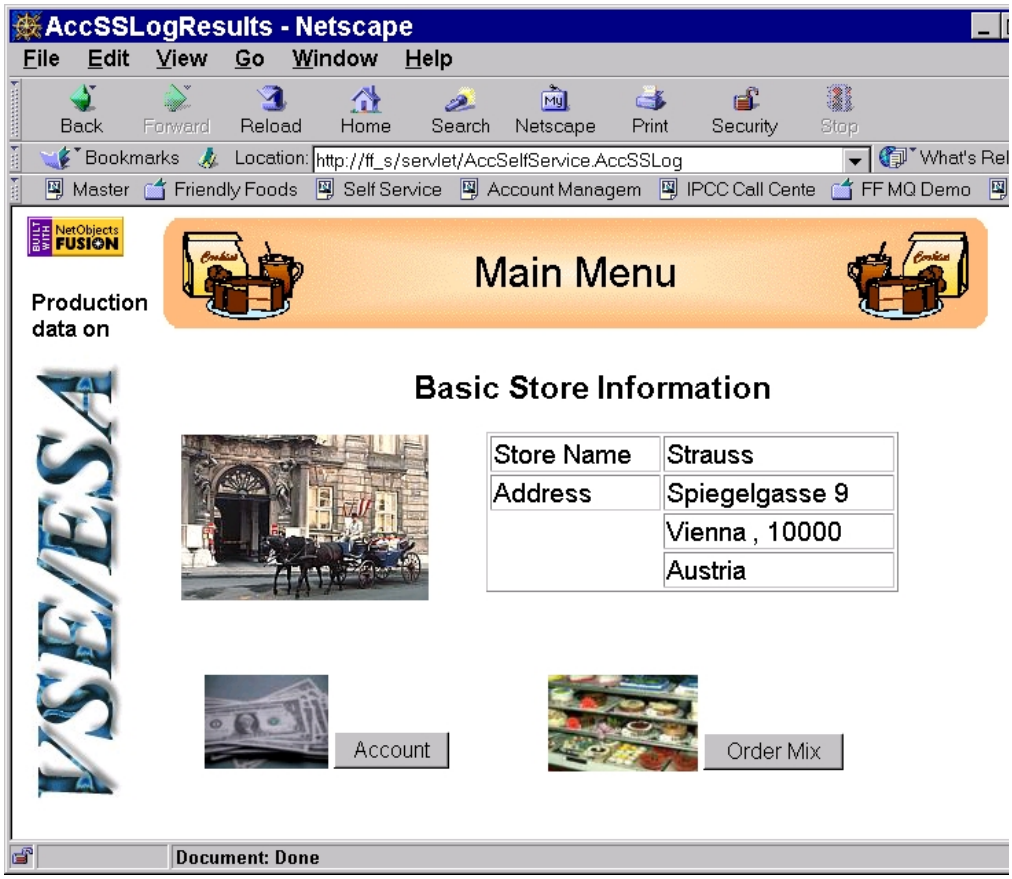


Figure 50. Friendly Foods account self-service Main Menu

This page combines result elements (AccSSLogResults.jsp) from the Logon servlet as well as input elements (for example, GetOrderMixInput.html) calling the other parts of the application in one single page.

The following two pages are not part of the subset we are discussing in all details here in this book. However, they are shown to give you a complete overview and to show additional options you have working with WebSphere Studio and NetObjects Fusion. The first page (Figure 51 on page 52) is the result page of the GetAccountStatus servlet called by the Account button in the page shown in Figure 50. The second page (Figure 52 on page 52) is the result page of the GetOrderMix servlet called by the Order Mix button shown in Figure 50.

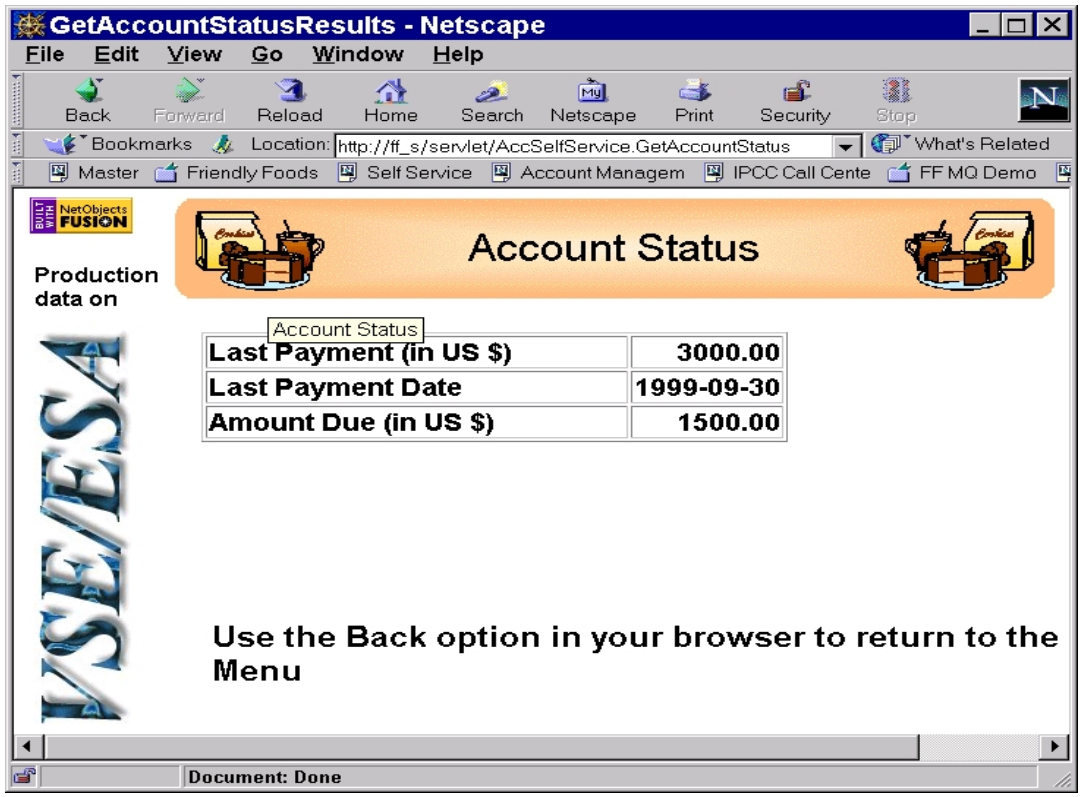


Figure 51. Friendly Foods account self-service Account Status page

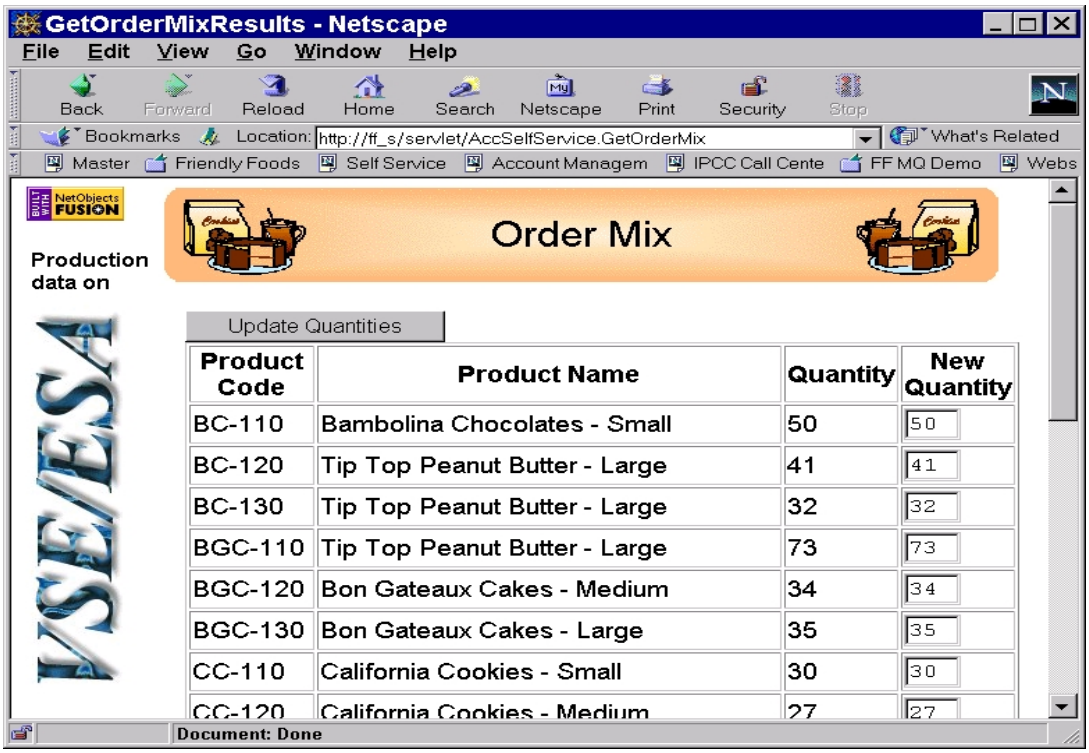


Figure 52. Friendly Foods account self-service update Order Mix page

Except for the Update Quantities button in the page shown in Figure 52 on page 52 all logic around the database queries in this application was implemented using just the visual design capabilities of WebSphere Studio. Not a single line of Java code or Java Script code was added or modified to implement these functions. For the Update Quantities function we had to modify some of the generated Java code. The SQL wizard in WebSphere Studio provides support for generating a database bean plus the related input and output panel to update a table. The database update statement is of the form:

```
UPDATE SQLDBA.ACCMIX SET QYT=? WHERE ((SQLDBA.ACCMIX.STOREID=?) AND
(SQLDBA.ACCMIX.PRNBR=?))
```

For the Update Quantities function we added a loop to check all rows in the table for an update of the quantities for a specific product. For all changed rows we call the database update bean to add a record to a separate table enabling us to keep track of all changes made in the account order mix.

4.3.3 Integrating the pages with NetObjects Fusion

Before we look at the details of how the different parts generated by WebSphere Studio were integrated into NetObjects Fusion, Figure 53 illustrates the general control flow for these servlet pages.

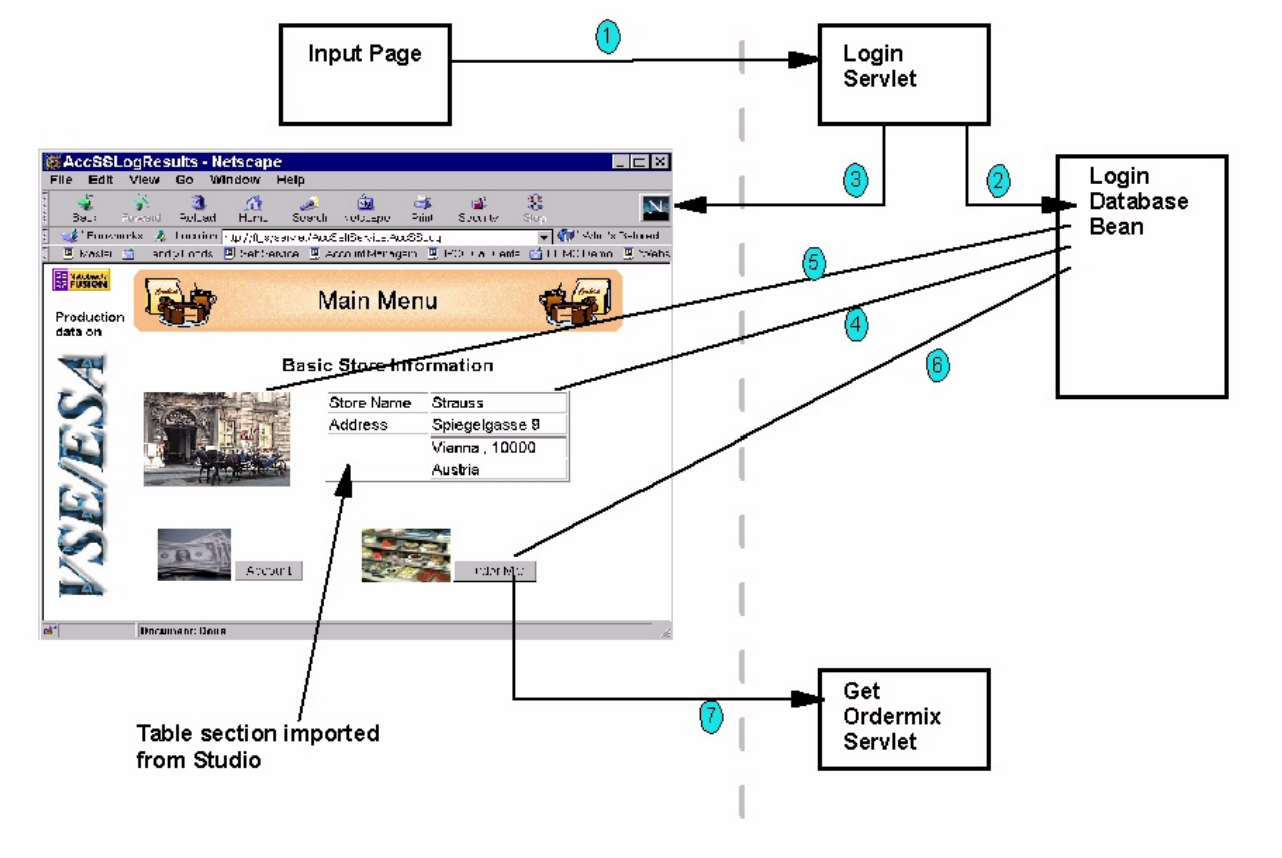


Figure 53. Control flow between HTML pages, servlet, Database Bean and JSP pages

1. The Input Page (HTML page) posts the servlet to run on the WebSphere Application Server (in this case the Login Servlet AccSSLog.servlet).

2. The servlet calls the Database Bean (AccSSLogDBBean). The Database Bean executes the database query and holds the result set, which can then be retrieved by Java code in the result page (JSP page).
3. If the database query creates a result, the result page (AccSSLogResults.jsp) is executed on the server and sent to the browser.
4. In the process of executing the JSP (before sending it to the browser) the Java code in this page builds the dynamic result table and uses methods defined in the Database Bean to retrieve the elements from the result set.
5. One of the elements in the result set (column in the stores table) is meant to hold the name of a picture to be associated with this particular store. The dynamic picture loader function in NetObjects Fusion makes it possible to retrieve this from the result set and translate it into a static HTML image (img) tag, which then results in sending the picture along with the page to the browser.
6. The HTML form to start the next dialog step in this application (for example, the Get Order Mix part) needs to pass the store ID value as a hidden field to the servlet behind this function. As in steps 4 and 5, the Java code in the JSP retrieves this value from the Database Bean.
7. When clicking **Order Mix**, the Get Order Mix servlet (GetOrderMix.servlet) is called.

Before we could start integrating the parts generated by WebSphere Studio, we had to publish all elements that are part of the project within WebSphere Studio. As mentioned before, the HTML parts are published in the AccSelfService subdirectory under the directory root defined to the HTML server, and the servlet parts are published in a directory with the same name under the root defined to WebSphere Application Server to be used with servlets. We do not change the servlet parts but replace the HTML parts with the pages designed by NetObjects Fusion. After we are done with NetObjects Fusion, we publish our design into the same directory we used before with WebSphere Studio. To keep an unchanged version of the pages generated by WebSphere Studio, we copy all these parts into another directory.

In NetObjects Fusion we start with a blank site for which we defined the same site style as for the consumer marketing sites. Since we have a transaction-driven page flow, we do not need a navigation bar.

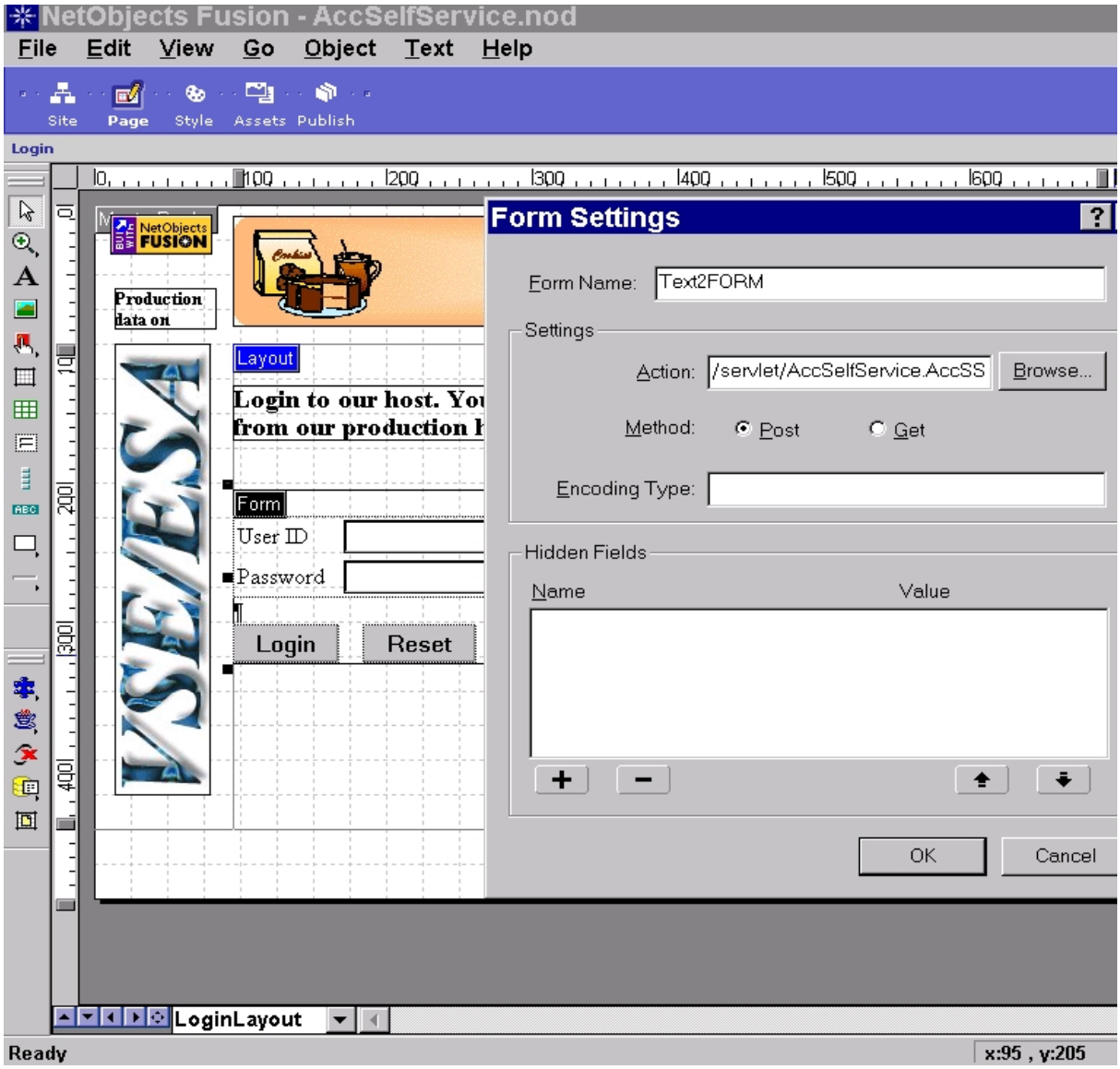


Figure 54. Design for the account self-service Login page with NetObjects Fusion

You have to perform the following steps to make the appropriate definitions in NetObjects Fusion:

1. Define the left sidebar to hold the VSE graphic.
2. Import the HTML page used for the Login input as it was generated by WebSphere Studio. Make sure that the Layout region is in focus when doing the import (**File -> Import HTML Page**). Depending on the defaults, the imported HTML page (basically an HTML form) might be expanded to fill the entire Layout region. In this case, uncheck the “Size to Layout” check box in the Text Box settings and you will be able to size the form and move it around in the Layout region. Since we imported the source for the form we can also modify parts of the form or change its display properties.

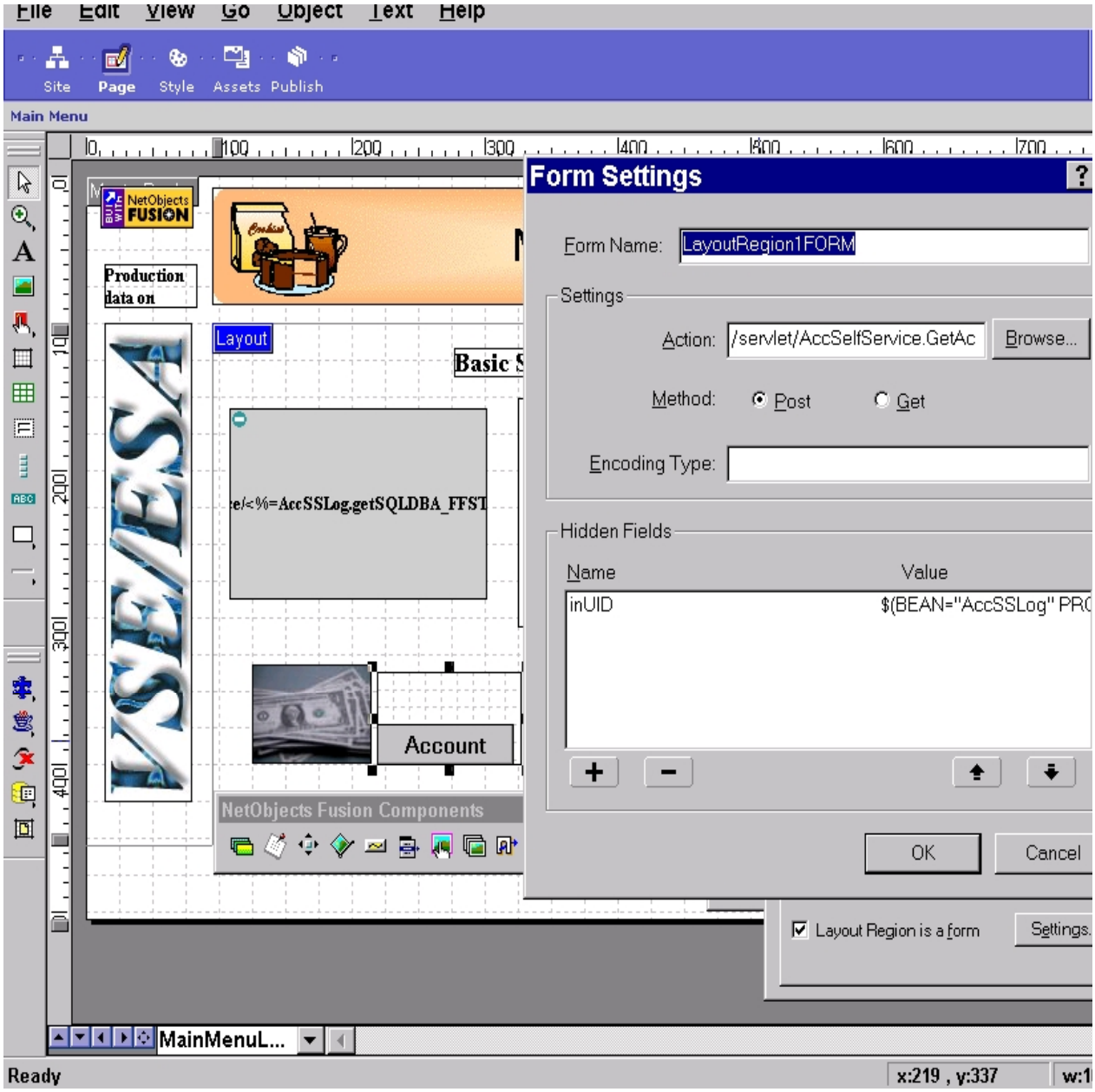


Figure 55. Design for the account self-service Main Menu page with NetObjects Fusion

3. Define the layout for the Main Menu page, which is a combination of the result page from the login step as well as an input page for the next dialog steps by:
   a. Creating a new page (in the site view of Fusion this will appear as a children page to the Login page even if we do not have a direct URL link between these pages).
   b. Importing the result page (AccSSLogResults.jsp) from the Login step onto the blank page. This time we do not directly import the source into NetObjects Fusion but define a reference to the external HTML source file. Both methods (importing the source or using a reference to an external HTML file) will create exactly the same output when the site is published. At publishing time the reference method will also import the source of the

external file and make it a part of the generated page. The only difference is that with the reference method Fusion will not attempt to interpret the code in the external HTML file.

The reference will be defined by **File -> Reference HTML**.

The result table generated by Studio is now integrated and you can use NetObjects Fusion's capabilities to make the site more attractive.

Next we add the HTML forms to call the next steps of the application (like the forms in the input pages generated with Studio). Since these are fairly easy forms we do not import the source but define the forms from scratch. We draw a text area, define the text area as a form, add a submit button, and define the servlet to be called when the form is submitted. Both servlets expect an input variable called inUID containing the store ID on which the next action should be executed. We pass this variable and the value as hidden fields (use the "+" button on the Form Settings panel to add a hidden field). The value of the hidden field should look like the following:

```
$(BEAN="AccSSLog" PROPERTY="SQLDBA_FFSTORES_STOREID(0)")
```

This forces the WebSphere Application Server to fill in the store ID when building the JSP for the Main Menu. It is placed in a hidden field in the form to call the next function (in this case to get the account status).

Now we have completed the design for the first two pages. Before we can publish the pages we have to change the file name of the Main Menu page. Remember that the servlet controls the flow of the panels. When our query delivers a result, the servlet looks for a page named AccSSLogResults.jsp. Therefore, we have to make sure that Fusion publishes the Main Menu page in a file with this name. To do so we switch to the site view in Fusion. The properties panel allows us to customize various names used with the selected page. Figure 56 shows the different options we have in specifying the names. Important for us are the Name field and the File Extension field, both used to construct the file name for the particular page.

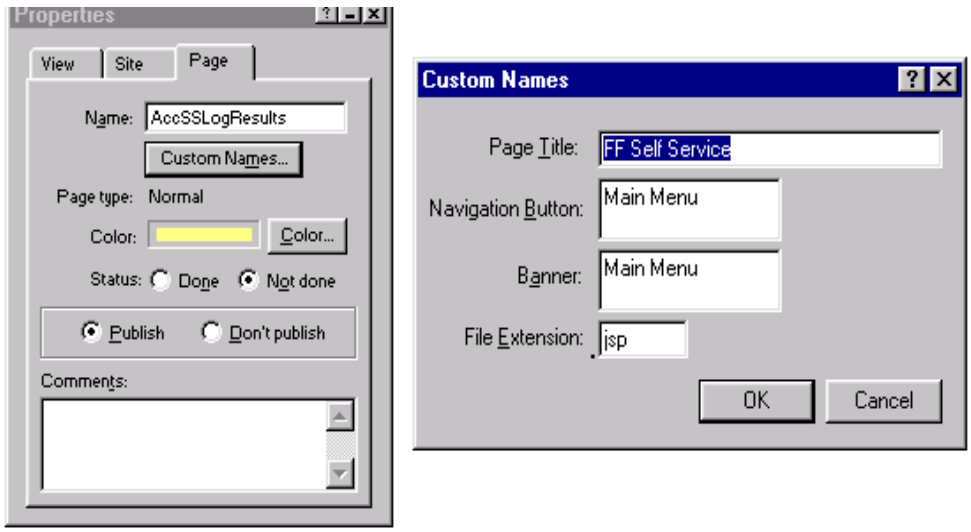


Figure 56. Customize page names in Fusion

As mentioned before, all servlet parts of this application, except the function of the Update Quantities button in the Ordermix dialog, were created with WebSphere Studio without any Java skills.

4.4 Account self-service and account tracking -- summary

WebSphere Studio enabled Friendly Foods to quickly develop a Web site for their stores (accounts), allowing the store users to view account information and manage their actual order mix over the Web. Using the visual design tools and the wizards in WebSphere Studio required only minimal Java skills. Integrating the Java Server Pages with NetObjects Fusion resulted in the same page layout of those pages as for the consumer marketing pages and gave Friendly Foods the flexibility to quickly change the style of the site.

Chapter 5. Integration with service providers

Friendly Foods wanted to serve its accounts mainly by providing the account self-service and account tracking application as described in Chapter 4, “Account self-service and account tracking” on page 35. However, Friendly Foods still wanted to provide the possibility for the accounts to call in for last minute changes. Friendly Foods decided to outsource this service to an external call center, the IPCC Call Center. This allows Friendly Foods to minimize the amount of people working in its own account care center.

The call center should be able to change the order mix of an account and also see the order mix history, which means that Friendly Foods had to open its system to an external service provider. The implementation had to meet two major requirements:

- No special software needs to be installed at the call center.
- Existing VSE/CICS applications will be used (no new development).

Friendly Foods implemented IBM SecureWay Host On-Demand to provide access to its VSE/CICS applications. SecureWay Host On-Demand enables the people in the call center to work from their Web browser interface. It does not require any changes to the existing applications on VSE/ESA. In addition, the employees of the call center did not have to install any additional software. The Host On-Demand code is downloaded from the Friendly Foods Web server whenever it is needed in the call center.

5.1 SecureWay Host On-Demand server

Host On-Demand uses the Java environment, industry standard 3270, SSL and other Internet protocols to provide secure, platform-independent host access through a Web browser. It provides a variety of options for integrating existing applications and data on VSE with intranet and Internet capabilities. Figure 57 on page 60 summarizes the different configuration options.

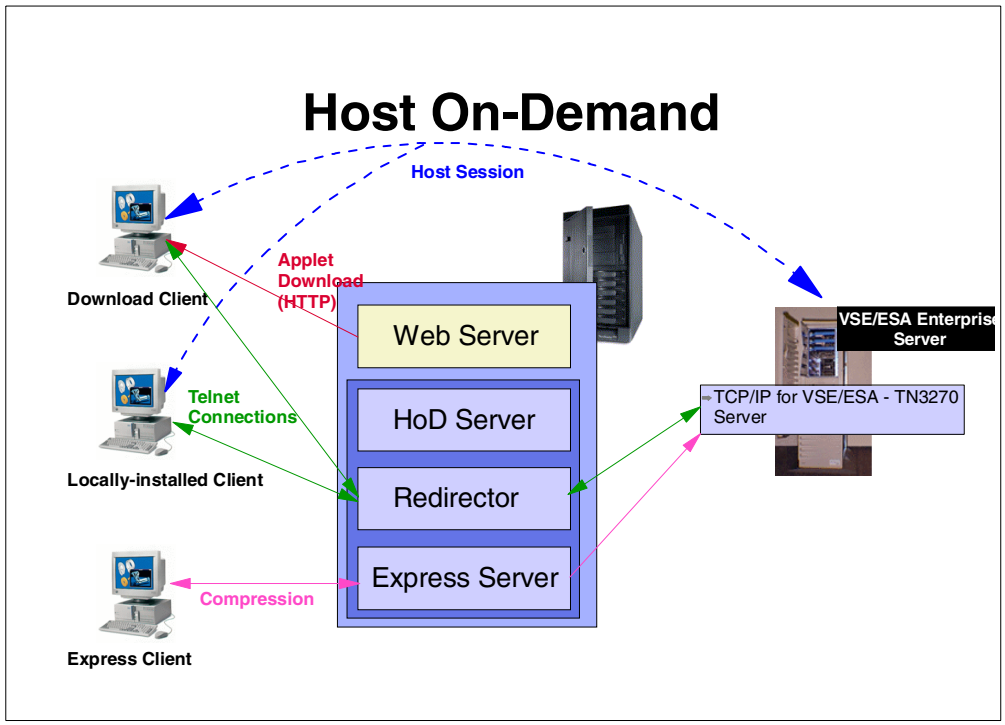


Figure 57. Host On-Demand -- configurations

Friendly Foods implemented the configuration by downloading the Host On-Demand client to the call center Web browser and directly connecting to the TN3270 server running on VSE/ESA. For a more complete discussion of Host On-Demand, refer to the redbook *IBM SecureWay Host On-Demand: Enterprise Communications in the Era of Network Computing,* SG24-2149.

The installation and administration of Host On-Demand is also described in this redbook; it is not be repeated here.

5.2 Call center implementation

The IPCC Call Center is responsible for companies other than Friendly Foods. Therefore, the employees have an entry page on which they select for which company they are actually receiving a call as shown in Figure 58 on page 61.

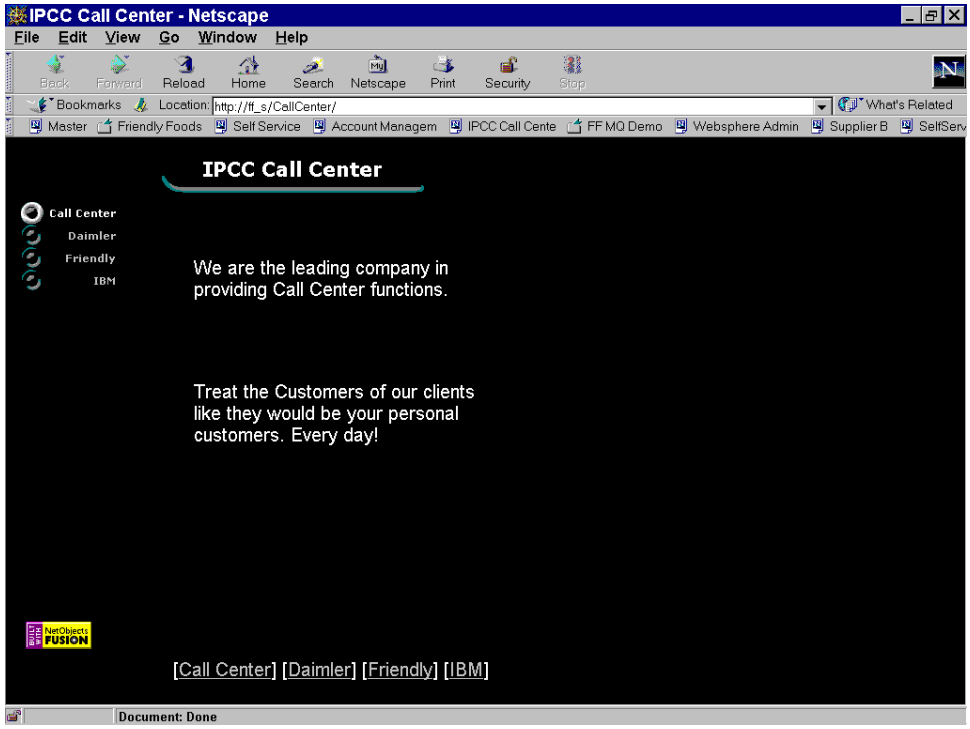


Figure 58. Initial IPCC Call Center Web page

After selecting Friendly Foods, the Web site shown in Figure 59 is displayed.

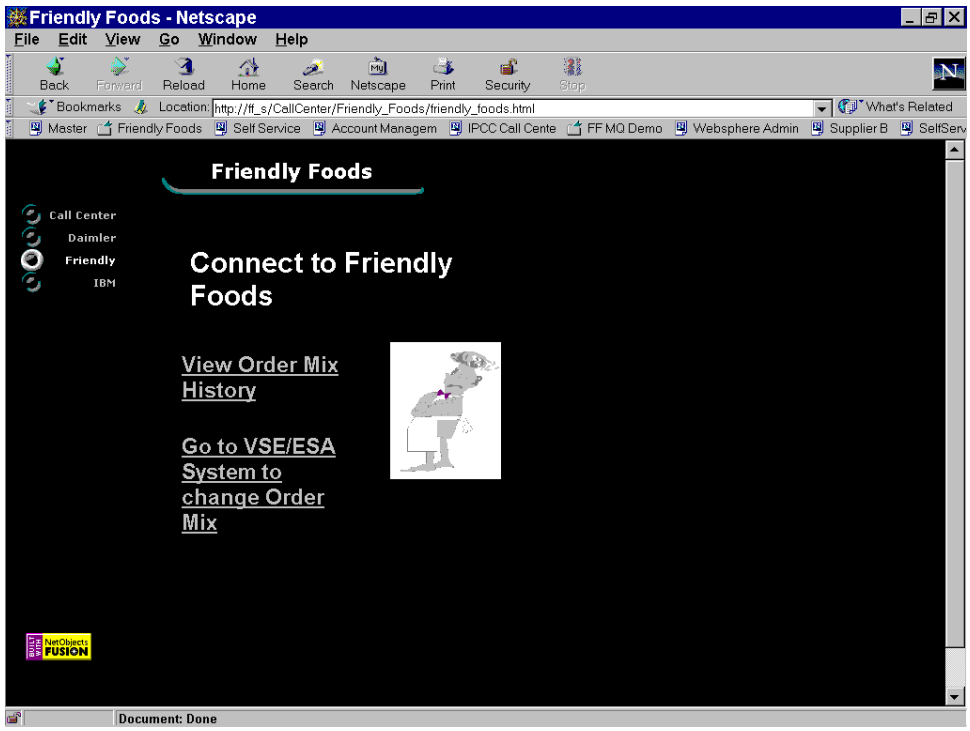


Figure 59. Friendly Foods IPCC Call Center page

After selecting **Go to VSE/ESA System to change Order Mix**, the following Web site (see Figure 60) is displayed, containing access to the order mix application on VSE/ESA.

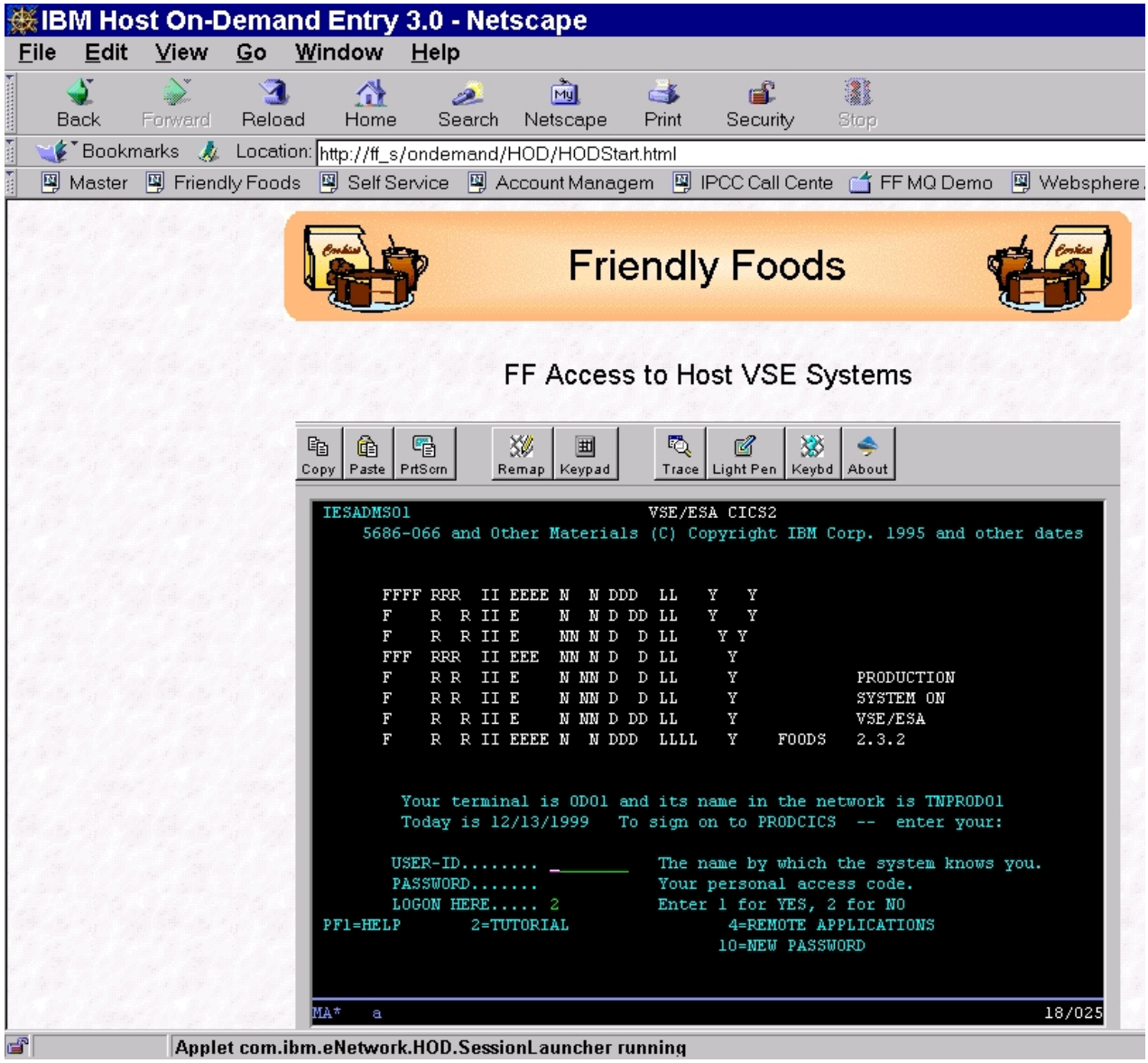


Figure 60. Host on-Demand screen -- change order mix

With this implementation, Friendly Foods is well positioned to allow access to their VSE applications from any external service provider.

5.3 The Friendly Foods Host On-Demand implementation

The Friendly Foods Host On-Demand implementation is based on Host On-Demand V3.

Host On-Demand offers an administration interface to administer the Host On-Demand server and the Host On-Demand clients. When launching sessions through the Host On-Demand clients, however, the end user is asked to enter the configuration values for the session to be established (for example, the IP address of the VSE system). In order to hide the configuration from the end user, Host On-Demand also offers the possibility to create your own applet files.

The applet file session2.html is delivered as part of Host On-Demand in the subdirectory \ondemand\hod\[language]\help. This applet file can be used to configure a session entirely through the applet file itself, then launch it. This bypasses the user administration function. However, a printer session cannot be started using this method.

Figure 61 on page 64 shows our HODStart.html file generated from the session2.html sample.

```
<!doctype html public "//W3//DID HTML 3.2 Final//EN">
<HTML>
<HEAD>
<META content="text/html; charset=iso-8859-1">
<TITLE>IBM Host On-Demand Entry 3.0</TITLE>
</HEAD>
<BODY BACKGROUND="hodbkgnd.gif">
<CENTER>
<IMG id="Banner1" HEIGHT=80 WIDTH=600
SRC="./Friendly_banner1.gif"  BORDER=0  ALT=" Friendly Foods " >
<P>
FF Access to Host VSE Systems
<P>
<APPLET archive="hod30.jar"
CODE="com.ibm.eNetwork.HOD.SessionLauncher.class" WIDTH=584 HEIGHT=450>
<PARAM NAME=CABBASE                VALUE=hod30.cab>
<PARAM NAME="SessionName"     VALUE="Friendly Foods VSE System">
<PARAM NAME="Host"            VALUE="FF_VSE">
<PARAM NAME="SessionType"          VALUE="1">
<PARAM NAME="SessionID"            VALUE="A">
<PARAM NAME="Port"                 VALUE="23">
<PARAM NAME="TNEnhanced"           VALUE="true">
<PARAM NAME="LUName"               VALUE="">
<PARAM NAME="ScreenSize"           VALUE="2">
<PARAM NAME="CodePage"             VALUE="037">
<PARAM NAME="SSL"                  VALUE="false">
<PARAM NAME="SSLAuthentication"    VALUE="false">
<PARAM NAME="AutoConnect"          VALUE="true">
<PARAM NAME="AutoReconnect"        VALUE="true">
<PARAM NAME="StartupApplet"        VALUE="">
<PARAM NAME="OIAVisible"           VALUE="true">
<PARAM NAME="Keypad"               VALUE="false">
<PARAM NAME="Toolbar"              VALUE="true">
<PARAM NAME="ToolbarText"          VALUE="true">
<PARAM NAME="VTTerminalType"       VALUE="1">
<PARAM NAME="VTNewLine"            VALUE="true">
<PARAM NAME="VTBackspace"          VALUE="false">
<PARAM NAME="VTLocalEcho"          VALUE="false">
<PARAM NAME="VTCursor"             VALUE="false">
<PARAM NAME="VTKeypad"             VALUE="false">
<PARAM NAME="VTAutowrap"           VALUE="false">
<PARAM NAME="CICSGWCodePage"       VALUE="000">
<PARAM NAME="CICSGWServerName"     VALUE="">
<PARAM NAME="LightPenMode"         VALUE="false">
<PARAM NAME="Embedded"             VALUE="true">
<PARAM NAME="MacroManager"         VALUE="false">
<p>If you are reading this message, your client platform is not capable of
running IBM Host On-Demand.  To run IBM Host On-Demand, you must have a
Java-enabled webbrowser such as Netscape Navigator or Microsoft Internet
Explorer.
</APPLET>
</CENTER>
</BODY>
</HTML>
```

Figure 61. Host On-Demand applet file - HODStart.html

We changed the highlighted lines in the original session2.html file because we wanted to:

- Include our Friendly Foods banner
- Include some additional text (FF Access to Host VSE Systems)

We also had to change the following configuration parameters, which are specific to our configuration:

- `PARAM NAME="SessionName"  VALUE="Friendly Foods VSE System"`
  to give a unique name to this session.
- `PARAM NAME="Host"  VALUE="FF_VSE"`
  to specify the IP address of the VSE system. `FF_VSE` is the symbolic name as specified in the hosts file on the Windows NT server.
- `PARAM NAME="Embedded"  VALUE="true"`
  to specify that we wanted to get the 3270 screen inside the existing browser window and not started as an extra session.

Please note that these configuration parameters are specific to the client you are selecting. We selected the client that is downloaded from the Web server on demand. Different Java applets exist for each client that is supported by Host On-Demand. We had to specify hod20.jar in the HODStart.html file. For a more detailed discussion on this, refer to the Host On-Demand documentation.

Those were all changes that had to be made in the sample session2.html file. In addition, we had to link to this HTML page from the pages built in NetObjects Fusion. HODStart.html is linked to as an external link as you can see in Figure 62.

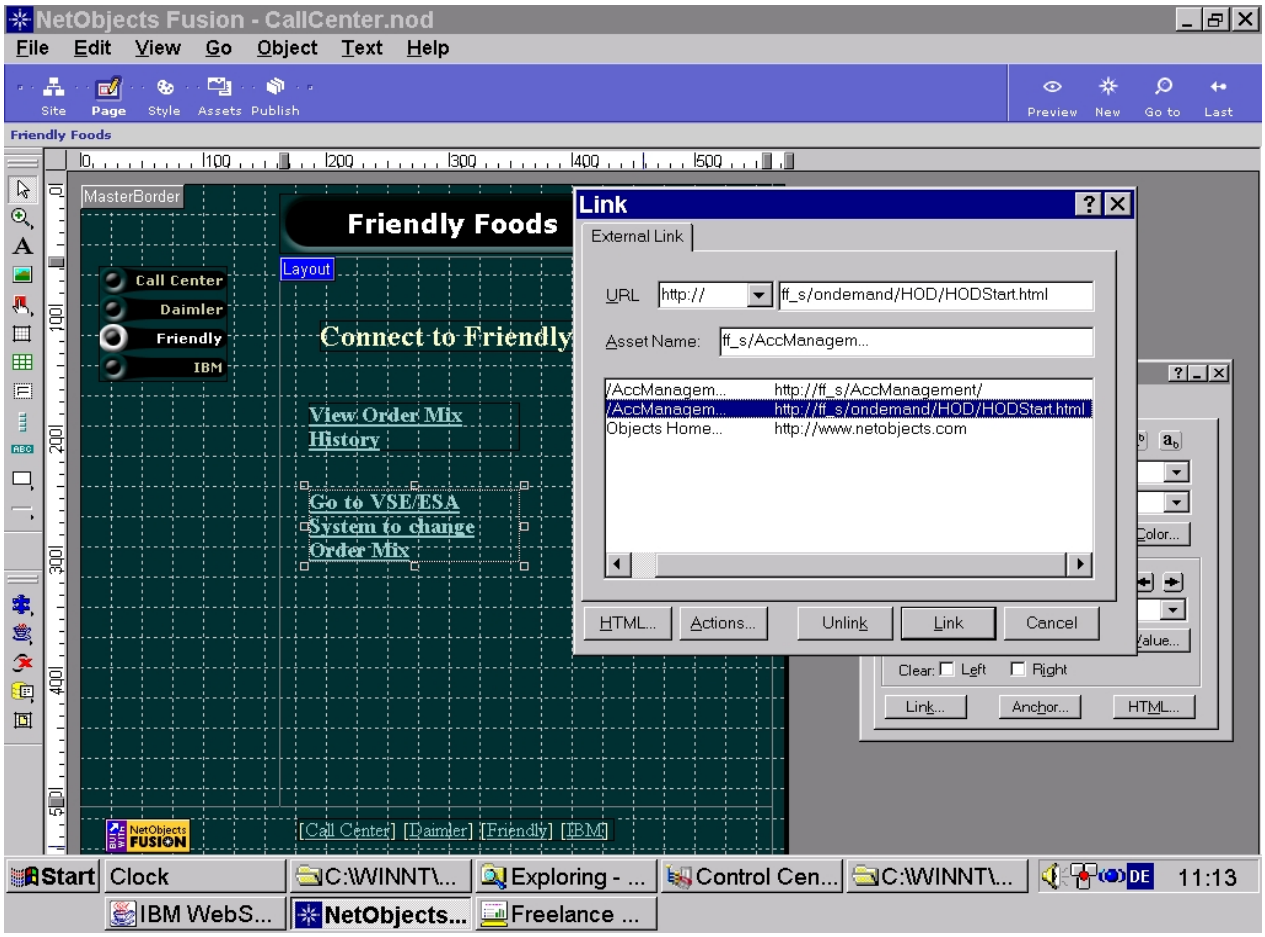


Figure 62. Linking to HODStart.HTML

5.4 Integration with service providers -- summary

Using SecureWay Host On-Demand made it easy for Friendly Foods to integrate a call center. The implementation on its Web server did not require much effort, and changes to its existing VSE/CICS applications were not necessary.

Chapter 6. Business integration with suppliers

Before Friendly Foods started this project, it did not have an application that supported the integration with its various suppliers. Requests for bids and orders were sent either by phone or fax. Answers to bids or order confirmations were sent the same way. After manually sending the letters to the suppliers, a record of the communication was kept in a DB2 database on VSE via the supply chain management (SCM) application.

Friendly Foods decided to modernize this application and at the same time make it more efficient. This was especially important because the number of suppliers was growing and more and more of them were offering interfaces based on e-business technologies. Therefore, Friendly Foods introduced a new e-business application on its Windows NT server, which allowed them to electronically exchange messages with its suppliers. Friendly Foods wanted to be able to exploit the rich functionality of its current SCM application in order to protect its investment, and therefore was looking for a solution that would allow it to integrate the new e-business application with the existing SCM application.

The suppliers of Friendly Foods work on different platforms, using different applications, resulting in a variety of data formats. This required a highly flexible approach that resulted in an architecture based on IBM MQSeries and MQSeries Integrator. This enabled Friendly Foods to integrate with the majority of its suppliers as well as with its existing SCM application on VSE.

The SCM application works with a 3270-based interface. Friendly Foods decided to design a new user interface for this existing CICS application by directly invoking the business logic with help of the CICS Transaction Gateway.

In our scenario, Friendly Foods as well as its suppliers use a Windows NT server for their e-business applications. TCP/IP is the underlying protocol for all communications.

Figure 63 on page 68 shows the MQSeries and CICS Transaction Gateway configuration as implemented by Friendly Foods. The implementation itself is the subject of this chapter.

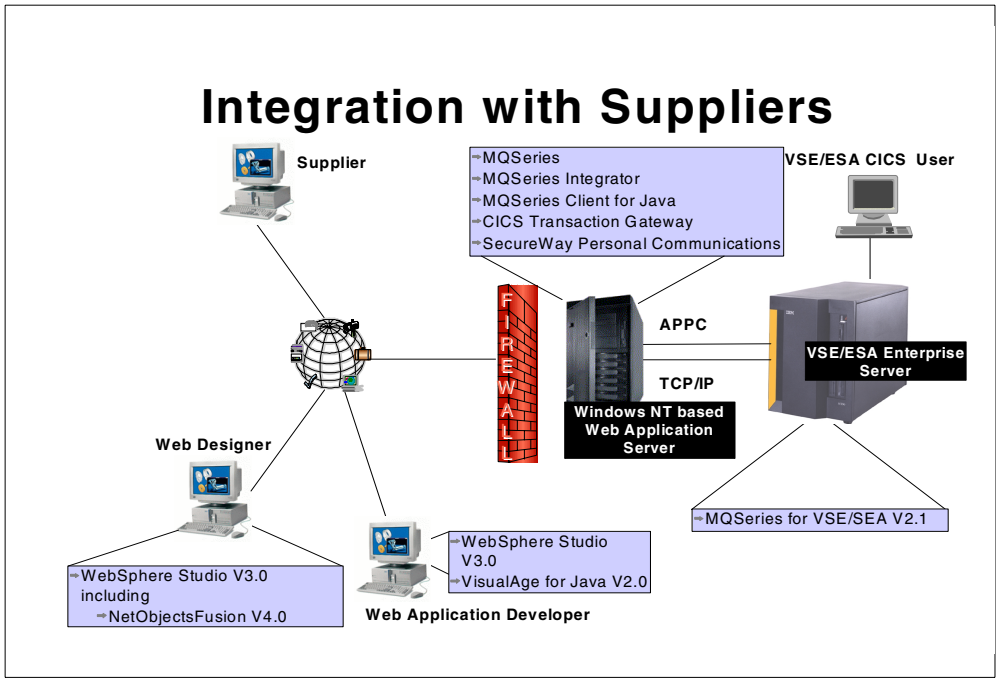


Figure 63. Additional products for integration with suppliers

The major components of this configuration are:

- On the Friendly Foods Windows NT server:
  - MQSeries for Windows NT, which functions as the MQSeries server. It has a server-to-server connection to MQSeries on VSE as well as to MQSeries on the suppliers system. It also accepts client connections from the Java-based e-business MQ applications.
  - MQSeries Integrator, which makes it possible to distribute incoming messages to different target queues, depending on the originator of the message.
  - MQSeries Client for Java, which is an MQSeries client written in the Java programming language. You can use this client to write both Java applets and Java applications. It is a set of Java classes that enable Java applications to include the MQSeries message queue interface (MQI). The MQI establishes a client connection to the MQSeries server.
  - CICS Transaction Gateway, which incorporates the CICS Universal Client and is used to access CICS transactions on VSE. The CICS Universal Client supports APPC (LU 6.2) connections to VSE only.
  - SecureWay Personal Communications, which is used to establish an SNA connection between the NT server and VTAM on VSE/ESA. This is needed to support the CICS Transaction Gateway.
- On the VSE/ESA host system:
  - MQSeries for VSE/ESA V2.1, which has a server-to-server connection to the MQSeries server on the Friendly Foods Windows NT system.

6.1 Products to be considered

The following sections offer a short overview of the products that were used by Friendly Foods for its new e-business application.

6.1.1 MQSeries

MQSeries helped Friendly Foods to link up its supply chain to deliver greater efficiency, fewer errors, lower operating costs and, ultimately, better service to its customers by offering the ability to move information reliably and quickly across multiple, disparate systems.

MQSeries offers the following key features:

Assured delivery

Information between systems and partners has to get through 100 % of the time, once and once only to avoid commercial exposure -- unstocked warehouses and dissatisfied customers. MQSeries ensures that messages get to where they need to go accurately and without duplication, even across unreliable or impermanent connections.

Fast, continuous delivery

MQSeries message-based technology is the ideal way to implement real-time, trickle-feed supply chain information, enabling continuous replenishment or just-in-time inventory management. Capable of handling large data volumes with equal efficiency, MQSeries enables you to update a data warehouse quickly, reliably and accurately.

Compatibility

Linking multiple systems is perhaps the biggest obstacle for Web-enabling and integrating supply chains. Once information passes outside your business, it has to cross all manner of technical and geographical boundaries, with the potential for information loss and errors. But MQSeries connects every major platform in use in supply chains today, with simplicity and speed.

6.1.2 MQSeries Integrator

Supply chains are dynamic environments where applications, relationships and information change daily. In this context, MQSeries Integrator is a powerful message brokering software solution that ensures that multiple business-critical applications and processes up and down the supply chain can understand each other.

Based on the MQSeries messaging system, MQSeries Integrator is a real-time, rules-based message broker, routing and dynamically transforming messages between different business systems. It allows very rapid integration of all types of applications and systems, with minimal rework or recoding. It is particularly useful for supply chain integration projects, due to its ability to act as an information hub transforming information formats between different partner systems.

6.1.3 CICS Transaction Gateway

The CICS Transaction Gateway (CTG) provides a comprehensive set of Java-based Web server facilities for access to CICS applications from a Web browser. These include Java classes and Java beans for writing application-specific server programs (servlets) and browser programs (applets). The CICS Transaction Gateway is part of the CICS Transaction Server for VSE/ESA.

6.2 MQSeries implementation

We assume that the MQSeries products have already been installed on the various systems. For the installation procedure, refer to the appropriate documentation and the redbooks that are available for MQSeries implementations on the different platforms. For a list of related redbooks, refer to D.1, “IBM Redbooks” on page 175.

After installing the products on Windows NT and VSE, we defined the MQSeries objects such as queue managers, queues and channels as described in 6.2.1, “Definitions for MQSeries objects” on page 71.

The implementation of the e-business application is described in 6.2.2, “Implementation at Friendly Foods” on page 81.

Figure 64 on page 71 gives you an overview of the MQSeries environment. In order to illustrate the overall process, the figure also shows the e-business applications dealing with the MQSeries queues, including the data flow between the queues.

For convenience, we refer to the VSE/ESA system as VSE1, the Friendly Foods Windows NT system as FF_S and supplier A’s Windows NT system as SUPPA_S. The terms are also used for the appropriate MQSeries queue managers. This chapter only describes the implementation for supplier A. Similar definitions have to be used for all other suppliers.

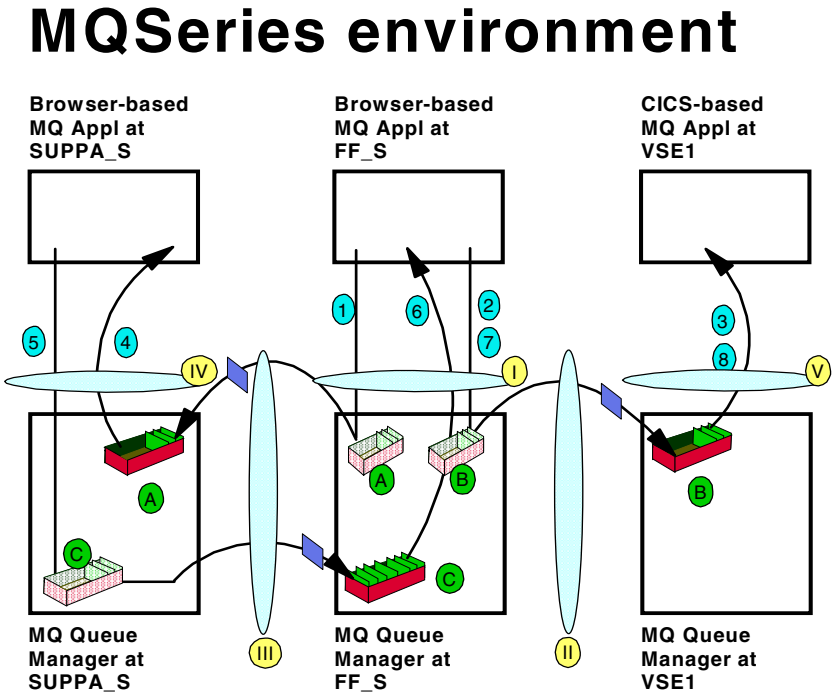


Figure 64. MQSeries environment at Friendly Foods

The parts marked with I to V in Figure 64 refer to the channel names shown in Table 3 on page 72.

1. The browser-based purchasing application puts an order record on the request queue of supplier A (or supplier B, depending on the user's input) using its local queue manager.
2. The purchasing application also puts a record on the request queue on VSE to document the process in the DB2 database.
3. Whenever a record is received on the VSE request queue, a CICS transaction is triggered that receives the record from the queue and puts a corresponding record into the SCM database on VSE.
4. Supplier A has an application in place receiving records from its local request queue.
5. After the request has been processed, supplier A puts an answer back on the sender's response queue.
6. The purchasing application is monitoring the response queue; the answer from supplier A is received and displayed on the Web browser window.
7. The same application also forwards the answer to VSE to document this process in the DB2 database.
8. Same as 3.

6.2.1 Definitions for MQSeries objects

To implement the above scenario we need to do the following:

- Define an MQSeries server-to-server connection between FF_S and VSE1.
- Define an MQSeries server-to-server connection between FF_S and SUPPA_S.
- Configure FF_S to accept a client connection from the e-business application.

Table 2 and Table 3 summarize the naming conventions that we used for the queue names, channel names and channel types as defined in the sections that follow.

Table 2. Queue names as used by queue managers

Queue / QMgr	FF_S	VSE1	SUPPA_S
A	Remote queue: SUPPA_S.REMOTE.QUEUE Corresponding xmit queue: SUPPA_S		Local queue SUPPA_S.REQUEST.QUEUE
B	Remote queue: VSE1.REMOTE.QUEUE Corresponding xmit queue: VSE1	Local queue VSE1.REQUEST.QUEUE	
C	Local queue: FF_S.RESPONSE.QUEUE		Remote queue FF_S.REMOTE.QUEUE Xmit queue FF_S

Table 3. Channel names and types as used by queue managers and applications

Channel	
I	JAVA.CHANNEL Server connection channel defined on FF_S to accept client connections from the application
II	FF_S.TO.VSE1 Sender channel between FF_S and VSE1 defined on FF_S Corresponding receiver channel defined on VSE1
III	FF_S.TO.SUPPA_S Sender channel between FF_S and SUPPA_S defined on FF_S Corresponding receiver channel defined on SUPPA_S
IV	JAVA.CHANNEL Server connection channel defined on SUPPA_S to accept client connections from the application
V	No explicit channel is needed since MQSeries is also a CICS application running in the same partition.

6.2.1.1 Windows NT definition of FF_S

Queue manager creation

Create a queue manager named FF_S.

You can create the queue manager under Windows NT as follows:

1. Open a command prompt session.
2. Enter the command:

   ```
   crtmqm -q FF_S
   ```

   This command creates a directory structure for this queue manager. You may have multiple queue managers on the same Windows NT system.

MQSeries object definitions to communicate with SUPPA_S

We need to define the following five objects:

- One sender channel named FF_S.TO.SUPPA_S
- One receiver channel named SUPPA_S.TO.FF_S
- A transmission queue named SUPPA_S
- A local queue to receive incoming messages from SUPPA_S named FF_S.RESPONSE.QUEUE

  Messages from other suppliers are also received from this queue.
- A remote queue (local for FF_S) to send messages to SUPPA_S named SUPPA_S.REMOTE.QUEUE

MQSeries object definitions to communicate with VSE1

Since no messages are received from VSE1, we neither need a receiver channel nor a local queue. We need to define the following three objects:

- One sender channel named FF_S.TO.VSE1
- A transmission queue named VSE1
- A remote queue (local for FF_S) named VSE1.REMOTE.QUEUE to send messages to VSE1

MQSeries object definitions to accept client connections

We define a server-connection channel named JAVA.CHANNEL.

MQSeries object definition to work with MQSeries Integrator

We define a local queue named FF_S.REQUEST.QUEUE where the applications will put their requests, which will be monitored by Integrator for further processing.

MQSeries object definition file for FF_S

We used WordPad to create a file named defffs.txt with our object definitions.

```
************************************************************
*
* Definitions of Queues and Channels for FF_S Manager
*
************************************************************

************************************************************
** LOCAL QUEUES - Requests
************************************************************

*
* Input Queue for request to be handled by Integrator
*

DEFINE QLOCAL(FF_S.REQUEST.QUEUE)                + (1)
```

```
       MAXDEPTH(100000)                          +
       REPLACE

************************************************************
*  LOCAL QUEUES - Responses
************************************************************

*
* Input queue to receive messages from SUPPA_S and other suppliers
*
DEFINE QLOCAL(FF_S.RESPONSE.QUEUE)                + (2)
       MAXDEPTH(100000)                          +
       REPLACE

************************************************************
*  LOCAL QUEUES - INITIATION
************************************************************

*
*  Queue to hold trigger messages related to queue VSE1
*

DEFINE QLOCAL(VSE1.INIT.QUEUE)                    + (3)
       REPLACE

*
*  Queue to hold trigger messages related to queue SUPPA_S
*

DEFINE QLOCAL(SUPPA_S.INIT.QUEUE)                 +
       REPLACE

************************************************************
*  LOCAL QUEUES - TRANSMISSION QUEUES
************************************************************

*
*  Transmission queue to send messages to VSE1
*

DEFINE QLOCAL(VSE1)                               + (4)
       DESCR('Q for transmission to VSE1')        +
       USAGE(XMITQ)                               + (5)
       INITQ(VSE1.INIT.QUEUE)                     + (6)
       TRIGGER                                    + (7)
       TRIGDATA(FF_S.TO.VSE1)                     + (8)
       TRIGTYPE(EVERY)                            + (9)
       REPLACE

*
*  Transmission queue to send messages to SUPPA_S
*

DEFINE QLOCAL(SUPPA_S)                            +
       DESCR('Q for transmission to SUPPA_S')     +
       USAGE(XMITQ)                               +
       INITQ(SUPPA_C.INIT.QUEUE)                  +
       TRIGGER                                    +
```

```
       TRIGDATA(FF_S.TO.SUPPA_S)                  +
       TRIGTYPE(EVERY)                            +
       REPLACE

*******************************************************
*  REMOTE QUEUES
*******************************************************

*
*  Remote queue for outgoing messages to VSE1
*

DEFINE QREMOTE(VSE1.REMOTE.QUEUE)                 + (10)
       RNAME(VSE1.REQUEST.QUEUE)                  + (11)
       RQMNAME(VSE1)                              + (12)
       XMITQ(VSE1)                                + (13)
       REPLACE

*
*  Remote queue for outgoing messages to SUPPA_S
*

DEFINE QREMOTE(SUPPA_S.REMOTE.QUEUE)              +
       RNAME(SUPPA_S.REQUEST.QUEUE)               +
       RQMNAME(SUPPA_S)                           +
       XMITQ(SUPPA_S)                             +
       REPLACE

************************************************************
* CHANNELS - TO OTHER SERVERS
************************************************************

*
*  Sender channel to VSE1
*

DEFINE CHANNEL(FF_S.TO.VSE1)                      + (14)
       CHLTYPE(SDR)                               + (15)
       DESCR('MESSAGE CHANNEL: FF_S ==> VSE1')    +
       CONNAME(FF_VSE)                            + (16)
       TRPTYPE(TCP)                               + (17)
       SEQWRAP(100)                               +
       NPMSPEED(NORMAL)                           +
       CONVERT(YES)                               + (18)
       XMITQ(VSE1)                                + (19)
       REPLACE
*
*  Sender channel to SUPPA_S
*

DEFINE CHANNEL(FF_S.TO.SUPPA_S)                   +
       CHLTYPE(SDR)                               +
       DESCR('MESSAGE CHANNEL: FF_S ==> SUPPA_S') +
       CONNAME('9.164.165.209(1415)')             + (20)
       TRPTYPE(TCP)                               +
       DISCINT(0)                                 +
       HBINT(1)                                   +
       NPMSPEED(NORMAL)                           +
```

```
       SHORTRTY(10000)                          +
       SHORTTMR(1)                              +
       XMITQ(SUPPA_S)                           +
       REPLACE

*******************************************************
*  CHANNELS - FROM OTHER SERVERS
*******************************************************

*
*  Receiver channel from VSE1
*

DEFINE CHANNEL(VSE1.TO.FF_S)                    + (21)
       CHLTYPE(RCVR)                            + (22)
       SEQWRAP(100)                             +
       TRPTYPE(TCP)                             +
       NPMSPEED(NORMAL)                         +
       REPLACE

*
*  Receiver channel from SUPPA_S
*

DEFINE CHANNEL(SUPPA_S.TO.FF_S)                 +
       CHLTYPE(RCVR)                            +
       TRPTYPE(TCP)                             +
       NPMSPEED(NORMAL)                         +
       REPLACE

************************************************************************
* CHANNELS - FROM CLIENTS
************************************************************************

*
*  Server - connection channel
*
DEFINE CHANNEL(JAVA.CHANNEL)                    + (23)
       CHLTYPE(SVRCONN)                         + (24)
       DESCR('MQI CHANNEL: JAVA CLIENTS')       +
       TRPTYPE(TCP)                             +
       REPLACE
```

Please refer to the MQSeries documentation for a detailed explanation of these parameters. The following points correspond to the numbers shown in parenthesis on the right of the definitions:

1. Name of the local queue to which messages that are to be handled by MQSeries Integrator are written.
2. Name of the local queue to which messages coming from SUPPA_S and other suppliers are written.
3. Name of the initiation queue, to which trigger messages are written.
4. Name of the transmission queue to hold messages destined for the remote queue manager VSE1.
5. Queue is a transmission queue.

6. Name of the initiation queue, to which trigger messages relating to the transmission queue are written.
7. Trigger messages are written to the initiation queue VSE1.INIT.QUEUE.
8. Data that is inserted in the trigger message.
9. Specifies how often a trigger message is written.
10. Name of the remote queue to which outgoing messages for VSE1 are written.
11. Name of the corresponding local queue on VSE receiving outgoing messages.
12. Name of the VSE1 queue manager.
13. Name of the transmission queue.
14. Name of the sender channel. It must be the same name as the receiver channel on VSE1.
15. Channel type. Must be SDR for a sender.
16. The connection name of the partner. For TCP/IP it is either the hostname or the network address of the remote machine.
17. Specifies the protocol, TCP/IP.
18. The sender will convert the message data to the receiver code page.
19. The name of the queue from which messages are retrieved.
20. The network address of the SUPPA_S queue manager. Since the SUPPA_S queue manager is defined on the same system as the FF_S queue manager, it has to listen to a different port.
21. Receiver channel from VSE1.
22. Channel type. Must be RCVR for receiver.
23. Name of channel used in MQSeries client for Java application.
24. Channel type. Must be SRVCONN for server-connection channel.

Create and configure MQSeries objects for FF_S

After all these definitions are specified in the file, the objects need to be created and configured. First you need to start the FF_S queue manager from a command window, as follows:

```
strmqm FF_S
```

You can then create your MQSeries objects using the RUNMQSC utility. You can pipe your definitions to this utility using the command:

```
runmqsc FF_S < defffs.txt
```

Remember, FF_S is the Friendly Foods queue manager name. This command creates the objects defined in defffs.txt.

Now all objects are created and you need to start your sender channels and the listener program with the following commands:

```
runmqchl -c FF_S.TO.VSE1 -m FF_S

runmqchl -c FF_S.TO.SUPPA_S -m FF_S

runmqlsr -t tcp -m FF_S
```

We did not specify a port number when starting the TCP/IP listener program because MQSeries uses its default port 1414.

6.2.1.2 Windows NT definition of SUPPA_S

We omitted the detailed definition because it is done the same way as for FF_S.

Queue manager creation

Create a queue manager named SUPPA_S.

MQSeries object definitions to communicate with FF_S

We need to define five objects, the counterparts of the FF_S objects:

- One sender channel named SUPPA_S.TO.FF_S
- One receiver channel named FF_S.TO.SUPPA_S
- A transmission queue named FF_S
- A local queue to receive incoming messages from FF_S named SUPPA_S.REQUEST.QUEUE
- A remote queue (local for SUPPA_S) to send messages to FF_S named FF_S.REMOTE.QUEUE

Create and configure MQSeries objects for SUPPA_S

Start the SUPPA_S queue manager from a command window:

```
strmqm SUPPA_S
```

Create your MQSeries objects (we called our definition file defsuppa.txt):

```
runmqsc SUPPA_S < defsuppa.txt
```

Start the sender channel to FF_S:

```
runmqchl -c SUPPA_S.TO.FF_S -m SUPPA_S
```

We have to specify the queue manager SUPPA_S explicitly, since all queue managers are created on the same system.

Start the listener program:

```
runmqlsr -t tcp -m SUPPA_S -p 1415
```

When starting the TCP/IP listener program, we have to specify a port (we used 1415). This is so because we created the queue managers SUPPA_S and FF_S on the same system, and the default port 1414 is already used by FF_S.

6.2.1.3 VSE/ESA definition of VSE1

Definitions on VSE/ESA are dialog-driven. Refer to the redbook *MQSeries for VSE/ESA*, SG24-5647, for a detailed description.

Queue manager creation

Create a queue manager named VSE1.

MQSeries object definitions to communicate with FF_S

Since we do not send messages from VSE1 to FF_S, we neither need a sender channel nor a transmission queue nor a remote queue. We define two objects:

- One receiver channel named FF_S.TO.VSE1
- A local queue named VSE1.REQUEST.QUEUE to receive incoming messages from FF_S

Since we want to start a CICS program whenever a message arrives, we specify the following trigger information in the queue definition panel:

- Trigger Enable - Y
- Trigger Type - E
- Maximum Trigger Starts - 0001
- Allow Restart of Trigger - Y
- Trans ID - <blanks>
- Term ID - <blanks>
- Channel Name - <blanks>
- Program ID - MQPECHO

MQPECHO application

Our program is a CICS program written in COBOL/VSE.

The trigger API handler does a CICS LINK to the application program. To return to the API handler, the program should issue an EXEC CICS RETURN.

MQPECHO compile consideration

The program contains SQL statements as well as CICS statements. Therefore, the compile job has to call the following:

1. The DB2 preprocessor
2. The CICS preprocessor
3. The COBOL/VSE compiler

You can generate this job by using the Interactive User Interface (IUI) panel Compile Job Generation (specify 8 (Compile) at the source member in the ICCF library to get the panel). In the panel, specify:

- SOURCE TYPE - 1 (1=ONLINE PROGRAM)
- LANGUAGE - 6 (6=COBOL VSE)
- DB2 SERVER - 1 (1=YES)

This will generate the compile job. Change the LIBDEFs according to your system configuration.

MQPECHO CICS definition

Add the definition for this program to a group known to CICS. We used group MQM, which is the default group for CICS resource definitions for MQSeries.

```
CEDA DEF PROG(MQPECHO) GROU(MQM)
CEDA INST GROUP(MQM)
```

MQPECHO sample program -- code fragment

The base for our MQPECHO program is the MQPECHO sample program provided with MQSeries for VSE/ESA. We adapted this program to our needs. Basically, we connect to the queue manager, open the queue, and get a message from the queue. The message contains information about the communication of Friendly Foods with the suppliers. This information is stored in the DB2 data repository.

```
*-------------------------------------------------------------*
* CONNECT -----------------------------------------------------*
*-------------------------------------------------------------*
```

```
*--MQCONNECT TO QM
          MOVE  'CONNECT' TO   WS-FUNCTION.
          MOVE  SPACES    TO   WS-QM-NAME-CONNECT.
          MOVE MQCC-OK                      TO   WS-CCODE-VALUE.
          MOVE MQRC-NONE                    TO   WS-RCODE-VALUE.
          SET WS-HCONN-VALUE  TO NULL.
          CALL 'MQCONN'  USING  WS-QM-NAME
                                WS-HCONN-VALUE
                                WS-CCODE-VALUE
                                WS-RCODE-VALUE.
     *
          IF WS-CCODE-VALUE   NOT EQUAL  ZERO
             GO TO  9900-ERR-DISPLAY
          END-IF.
     *

*-----------------------------------------------------------------*
* OPEN ------------------------------------------------------------*
*-----------------------------------------------------------------*
*--MQOPEN   QUEUE TO QM
          MOVE  'OPEN'    TO   WS-FUNCTION.
          MOVE MQOO-INPUT-SHARED TO   WS-Q-OPEN-OPTIONS-VALUE.
          MOVE SPACES            TO   MQOD-OBJECTQMGRNAME.
          MOVE MQI-PROC-LOCAL-QUEUE-NAME
                                 TO   MQOD-OBJECTNAME.
          MOVE MQCC-OK           TO   WS-CCODE-VALUE.
          MOVE MQRC-NONE         TO   WS-RCODE-VALUE.
          SET WS-HOBJ-VALUE  TO NULL.
          CALL 'MQOPEN'  USING  WS-HCONN-VALUE
                                WS-Q-NAME-AREA
                                WS-Q-OPEN-OPTIONS-VALUE
                                WS-HOBJ-VALUE
                                WS-CCODE-VALUE
                                 WS-RCODE-VALUE.

   *
          IF WS-CCODE-VALUE   NOT EQUAL  ZERO
             GO TO  9900-ERR-DISPLAY
          END-IF.


*-----------------------------------------------------------------*
* GET-MESSAGES ----------------------------------------------------*
*-----------------------------------------------------------------*
*--MQGET TO QUEUE TO QM
          MOVE  'GET'     TO   WS-FUNCTION.
          MOVE MQCC-OK    TO   WS-CCODE-VALUE.
          MOVE MQRC-NONE  TO   WS-RCODE-VALUE.
          MOVE LOW-VALUES TO  MQMD-MSGID
                              MQMD-CORRELID.
          MOVE LENGTH OF WS-BUFFER-AREA  TO   WS-BUFFER-LENGTH.
          MOVE MQGMO-ACCEPT-TRUNCATED-MSG
                                    TO  MQGMO-OPTIONS.

          CALL 'MQGET'   USING  WS-HCONN-VALUE
                                WS-HOBJ-VALUE
                                WS-MSG-DESCRIPTOR
```

```
                                    WS-GET-OPTIONS
                                    WS-BUFFER-LENGTH
                                    WS-BUFFER-AREA
                                    WS-DATA-LENGTH
                                    WS-CCODE-VALUE
                                    WS-RCODE-VALUE.
*
*
     IF  (WS-CCODE-VALUE   NOT EQUAL  ZERO)
     AND (WS-RCODE-VALUE   NOT EQUAL MQRC-TRUNCATED-MSG-ACCEPTED)
         IF WS-RCODE-VALUE EQUAL MQRC-NO-MSG-AVAILABLE
            SET WS-NOMORE-DATA  TO TRUE
         ELSE
            GO TO  9900-ERR-DISPLAY
         END-IF
     END-IF.
```

Figure 65. MQPECHO sample program - code fragment

6.2.2 Implementation at Friendly Foods

At the beginning of 6.2, “MQSeries implementation” on page 70 we gave an overview of the Friendly Foods MQSeries environment. Refer back to Figure 64 on page 71 for a full description of the MQSeries environment before we start explaining the different parts of the purchasing application and how these interact with MQSeries.

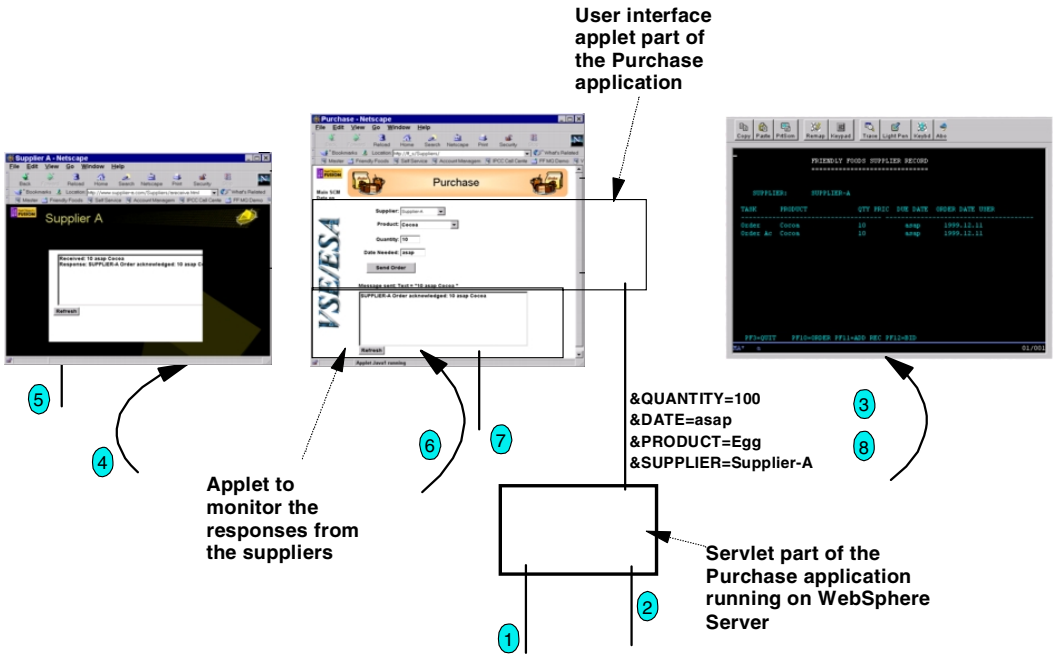


Figure 66. A more detailed look at the application parts

The browser-based part of the application running on the NT server consists of three major parts:

- A Java applet comprising the user interface to order the products from the suppliers

  This part calls a servlet on the WebSphere Application Server passing on the information about the supplier, product, quantity, and order date.

- A Java applet monitoring the response queue to display the answer from the supplier and to forward this answer to VSE for adding to the database

- A Java servlet, running on the WebSphere Application Server

  This handles all details about queue manager names, queue names, and channels, sending the records to the requested supplier and to VSE.

6.2.2.1 Implementing the user interface applet with VisualAge for Java

This redbook is not meant to provide an introduction to programming with VisualAge for Java. Refer to D.1, “IBM Redbooks” on page 175 for additional information.

Therefore, this redbook does not describe all the details for creating a Java program or applet but focuses on aspects specific to this environment. This means, for example, that for an applet we do not describe all the details on how to create the user interface, but we show how to call a servlet.

Let us have a look at the Visual Composition of the applet in VisualAge for Java.

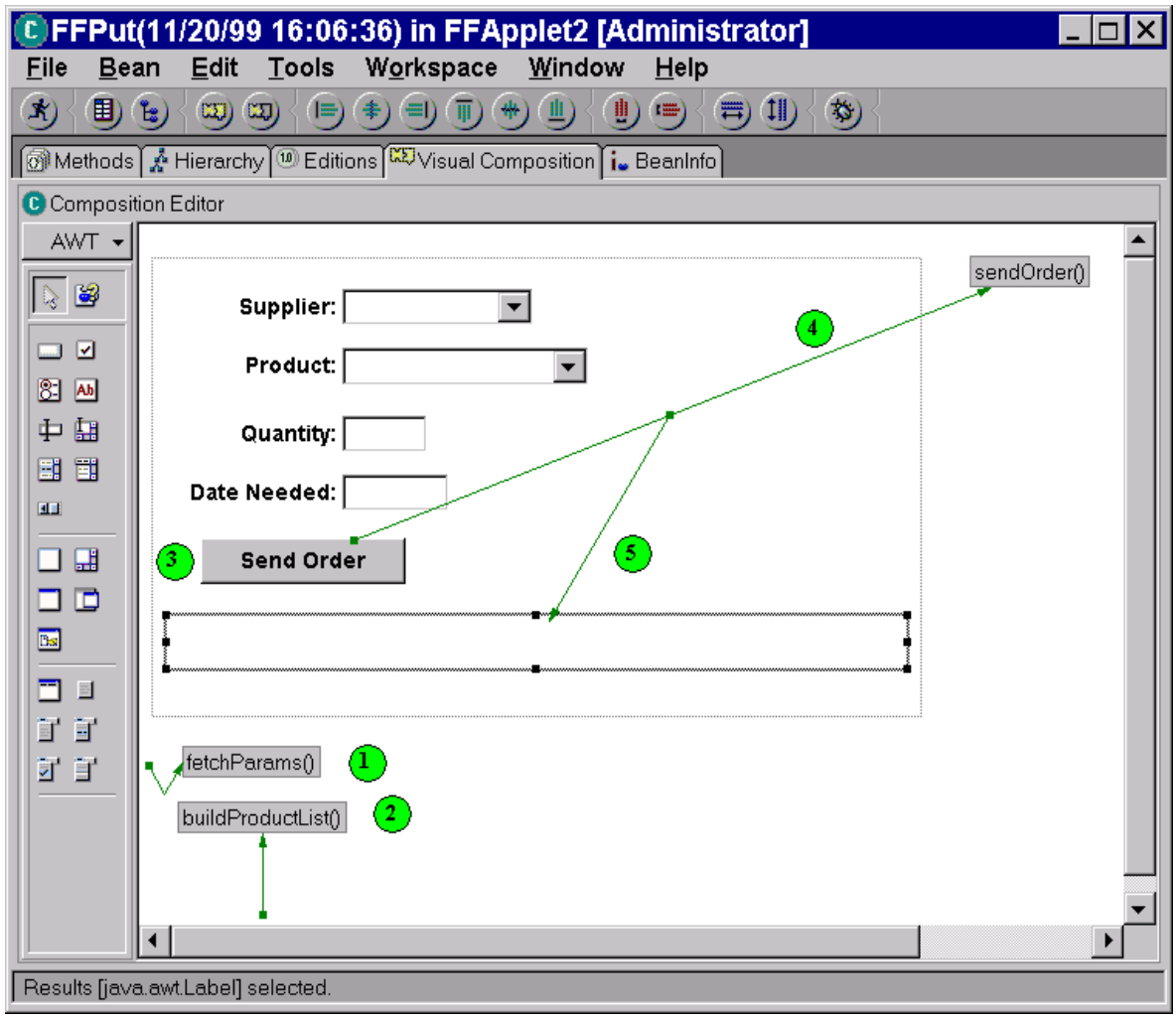


Figure 67. Visual Composition of the user interface applet part

Following is an explanation of the marked fields in Figure 67:

1. The fetchParams method retrieves the parameters passed from the HTML page calling the applet (basically used for testing different paths).
2. The buildProductList method is used in the initialization phase to fill pull-down choice controls with the list of orderable products and available suppliers.
3. When **Send Order** is clicked, the sendOrder method is triggered.
4. The sendOrder method does some preparation and then triggers the sendOrderServlet method, which builds the URL to call the servlet on the WebSphere server.
5. The result passed back from the servlet is displayed in the text area in the bottom part of the applet.

Let us now have a look at the sendOrderServlet part:

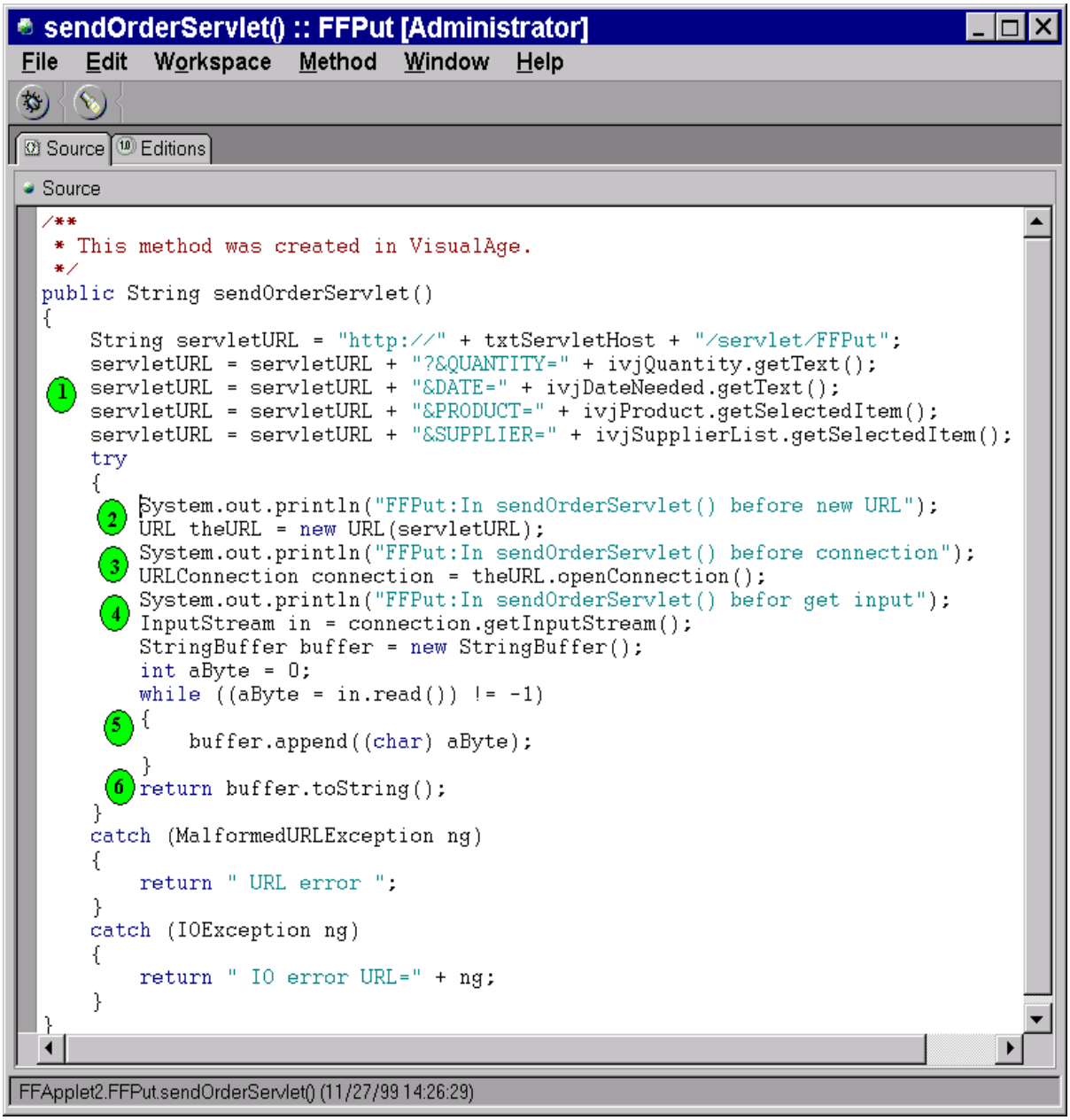


Figure 68. Method to build a URL to call a servlet

1. First we build up a text string called servletURL with the URL of the servlet we are going to call plus all the parameters we are passing. If we order 10 cocoa from supplier A to be delivered as soon as possible (asap), the string will be:
   http://ff_s/servlet/FFPut?&QUANTITY=10&DATE=asap&PRODUCT=Cocoa&SUPPLIER=Supplier-A
2. We establish a new instance of the class URL (part of java.net) with the value of our servletURL.
3. This connects to the servlet.
4. Retrieves the string returned by the servlet.
5. Moves the returned string into a StringBuffer.

6. Returns the string, which will then be displayed in the message area.

Figure 69 shows how the user interface part is displayed in the browser.

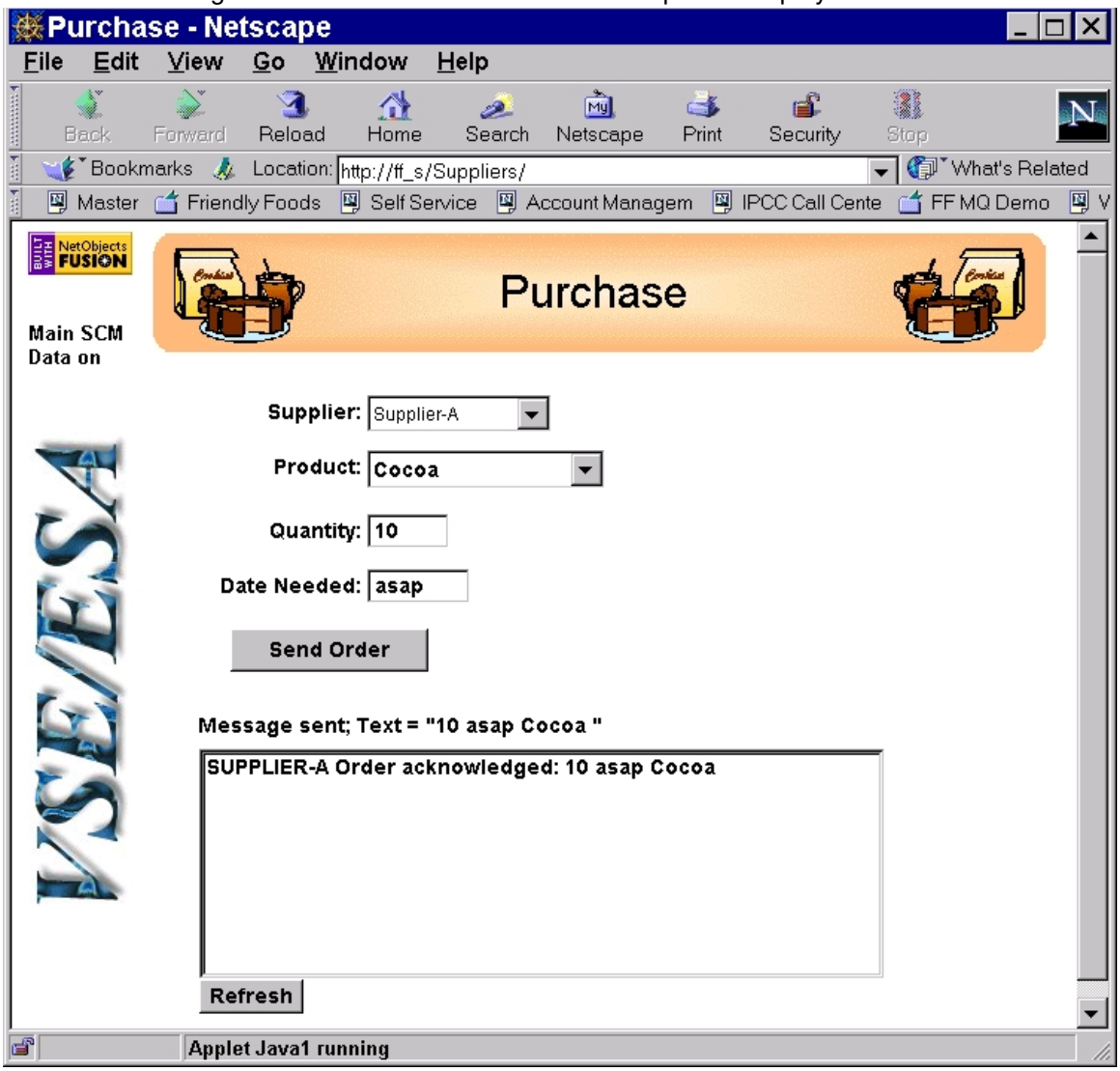


Figure 69. User Interface of the purchasing application

As described in the overview picture (Figure 64 on page 71), the upper part in the browser window is the user interface applet working together with the WebSphere hosted servlet to process an order. The lower part in the browser window (the text box with the Refresh button) is an applet monitoring the responses received from the supplier.

We created this entire page again with NetObjects Fusion by first exporting the packages from VisualAge for Java into a directory under Fusion/Java/. We then designed the page by dragging two Java applet spaces on the page surface and linking them to the corresponding package and class file. Lastly we published the page in the same way as previously described.

6.2.2.2 Implementing the servlet part with VisualAge for Java

Figure 70 shows the servlet part:

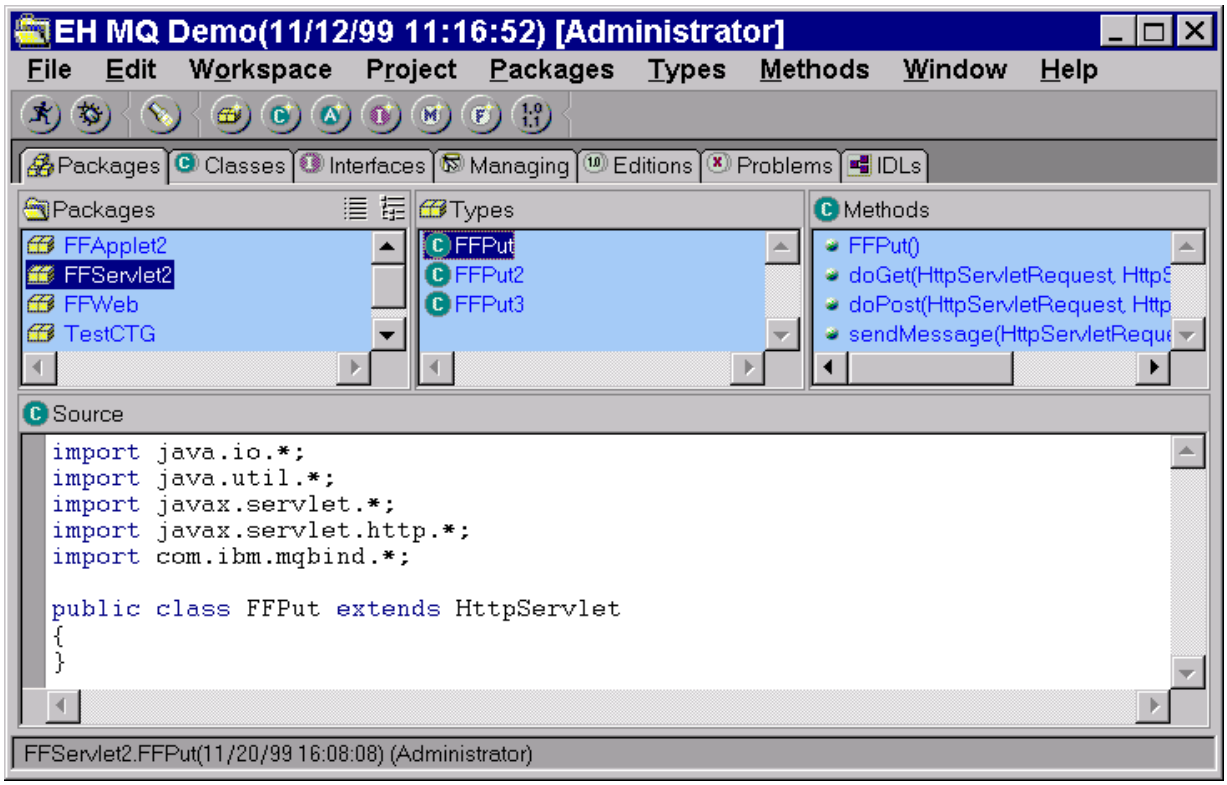


Figure 70. Servlet handling MQSeries queues -- 1

The basic classes to create servlets (javax.servlet.) are part of VisualAge for Java.

Both the doGet and the doPost method trigger the sendMessage method.

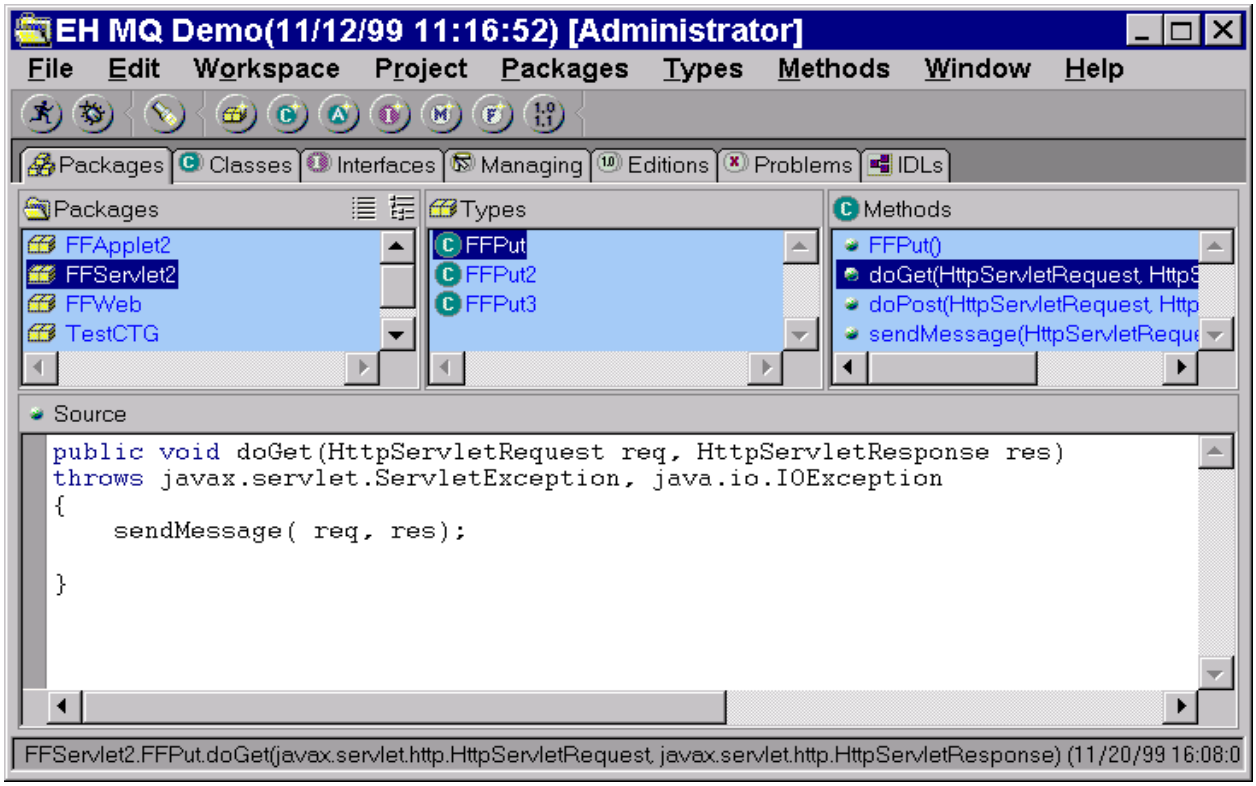


Figure 71. Servlet handling MQSeries queues -- 2

Figure 72 on page 88 shows the main parts of the sendMessage Java code handling the interface to MQSeries.

```
public void sendMessage(HttpServletRequest req, HttpServletResponse res) throws


ServletException, java.io.IOException
{
// MQ objects
MQQueueManager qMgr;
MQQueue local_queue = null;

// Text strings arriving as parameters
...
// Define MQ resources
txtQMgr = "FF_S";
txtQName = "SUPPA_S.REMOTE.QUEUE";
txtQMgr2 = "FF_S";
txtQName2 = "SUPPB_S.REMOTE.QUEUE";
txtQMgr3 = "FF_S";
txtQName3 = "VSE1.REMOTE.QUEUE";
// Get parameters
txtProduct = req.getParameterValues("PRODUCT");
// ...
// setup the message text for Supp A + B
txtMessage = txtQuantity[0] + " " + txtDate[0] + " " + txtProduct[0] + " ";
// setup message for VSE and adjust positions for the different parms
txtTemp1 = txtQuantity[0] + "     ";
txtMessage2 = "Order   " + txtTemp1.substring(0,4) + ...
// Supp A: Establish connection to Queue manager and open Queue
try
{
qMgr = new MQQueueManager(txtQMgr);
// Set up the options on the queue we wish to use
int openOptions = MQC.MQOO_OUTPUT;
// Now specify the queue that we wish to open
local_queue = qMgr.accessQueue(txtQName, openOptions, null, null, null);
}
// If an error ...
// Was it an MQ error?
catch (MQException ex)
{
// error message and handle error
}
// same for a Java buffer space error
catch (java.io.IOException ex)
{
// error message and handle error
}
try
{
MQMessage message = new MQMessage();
message.format = "MQSTR   ";
message.writeString(txtMessage);
MQPutMessageOptions pmo = new MQPutMessageOptions();
message.messageType = MQC.MQMT_DATAGRAM; // default
message.characterSet = 819;
// put the message on the queue
local_queue.put(message, pmo);
// Close the queue
local_queue.close();
}
// If an error ...
// Was it an MQ error?
catch (MQException ex)
{
// error message and handle error
}
// same for a Java buffer space error
catch (java.io.IOException ex)
{
// error message and handle error
}
// same for "Supplier-B" if needed and for VSE in any case
// ...
}
```

Figure 72. Excerpt of the sendMessage method in the servlet dealing with MQSeries

Before we can call the servlet we have to publish it to the SERVLETS directory used by WebSphere Application Server. This is done by exporting the package from VisualAge for Java into the target directory.

6.2.2.3 Configuring servlets in WebSphere Application Server

We call the servlet without the full package name:

```
http://ff_s/servlet/FFPut
```

This requires to have the servlet configured within WebSphere from the servlet configuration page in WebSphere. You will reach this page by logging on to the WebSphere Administration function (default user ID and password is admin) and selecting **Servlet -> Configuration**.

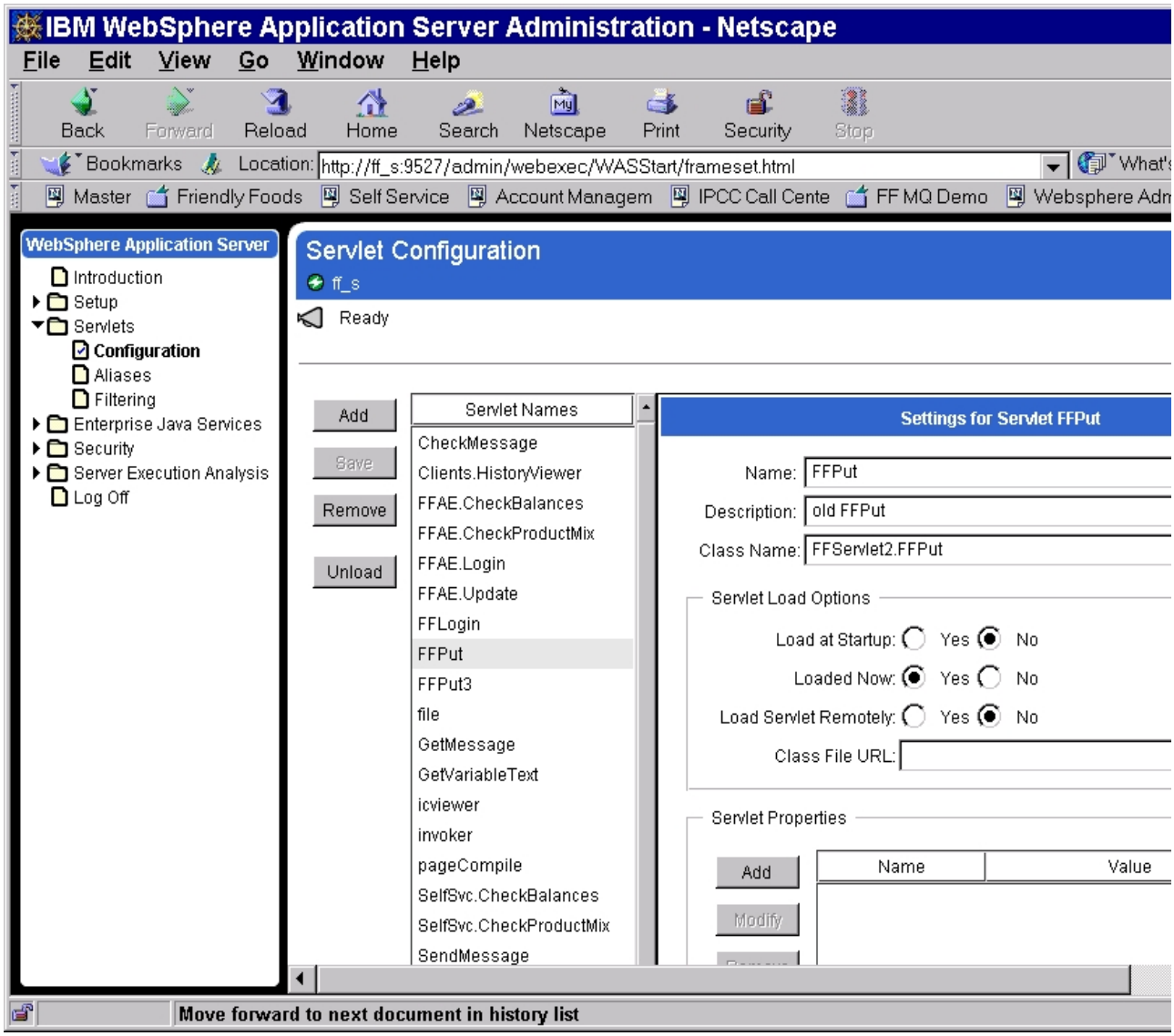


Figure 73. Configuring servlets in WebSphere Application Server

In addition to the name under which the servlet is called, you have to specify the full class name under which WebSphere will find the servlet. Please note that this

name is now identical to the package name we used when we developed the servlet under VisualAge for Java. After completing the servlet we published the package into the servlet directory that is known to Websphere. This created a subdirectory with the package name and placed the class files into this subdirectory.

WepSphere offers multiple options to load the servlet. One would be to preload the servlet whenever WebSphere is started, another would be to load it even from a remote WebSphere server. Lastly we could supply the servlet with some default/initial properties.

6.3 Using MQSeries plus MQSeries Integrator

In the first step we discussed how MQSeries could help start a business-to-business type of e-business transformation by integrating with your suppliers. We demonstrated how easy it is to develop a Web-based application that sends ordering records in MQSeries message format to suppliers and at the same time feeds these records into an existing application datastore on VSE/ESA. Since all the logic to deal with the suppliers (with their MQSeries queue managers) is integrated in the servlet part of our application, this works fine as long as we are dealing with a small number of suppliers and do not expect frequent changes in the message formats we have to deliver to them.

But what could we do to get the flexibility to add a new supplier, who might require a different message format, without changing the logic in our application or in the supplier's application?

The answer is: MQSeries Integrator (MQI). Figure 74 shows the queue handling without and with MQI.

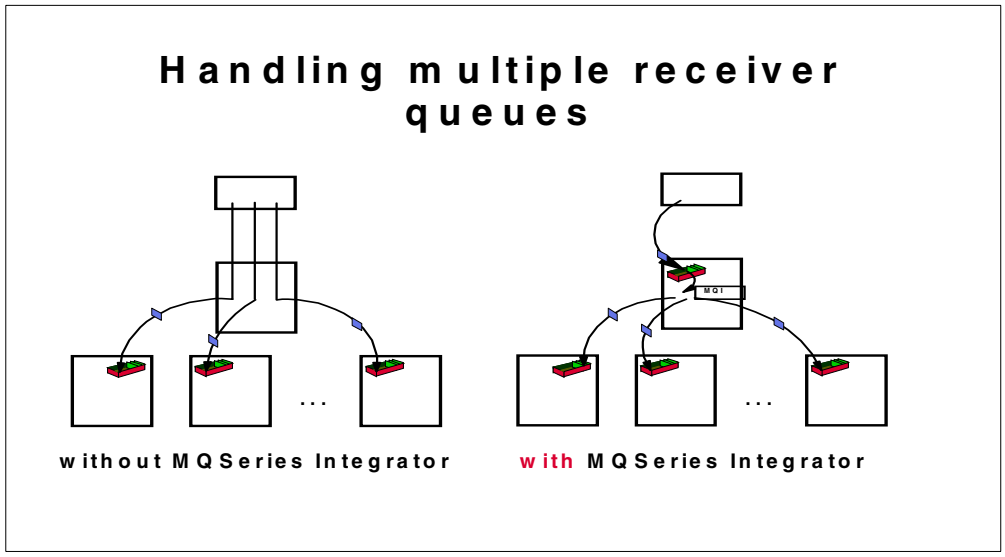


Figure 74. Queue handling without and with MQSeries Integrator

MQSeries Integrator takes environment-specific information out of your application logic, which then enables you to easily do the following with its own user interface dialogs:

- Change distribution lists

- Change record formats

Furthermore, MQSeries Integrator allows you to implement your own internal business rules (in this case, your rules for dealing with suppliers) on the same interface, making all further changes just an update of a control value in the MQSeries Integrator rules database.

Table 4 shows the actions you have to take in case there are changes to your environment:

Table 4. Comparing the actions to be taken for specific events in your environment

Event	Action without MQSeries Integrator	Action with MQSeries Integrator
Adding a new supplier	Your Application: Add all the logic to handle an additional queue and to select in which case a record should be sent to this queue.	MQSeries Integrator Rules Dialog: Add a new supplier to the distribution list by copying an existing one and just modifying the receiving queue name.
Changing the record format for a supplier	Your Application: Modify the code to set up the appropriate record.	MQSeries Integrator Formats Dialog: Modify the appropriate record format by using Integrator controls.
Changes in your internal business rules (for example large quantity handling)	Your Application: Find and modify the code that deals with this rule.	MQSeries Integrator Rules Dialog: Modify the appropriate control value triggering this rule.

6.3.1 Implementing the application with MQSeries Integrator

After the successful implementation of the first part (integration with suppliers based on MQSeries), Friendly Foods decided to use MQSeries Integrator on top of MQSeries for the second step of implementing its supply chain management (SCM) application.

This second part deals with the process of sending a request for a price proposal (bid) to the suppliers.

Figure 75 on page 92 shows the overall control flow for this part of the application.

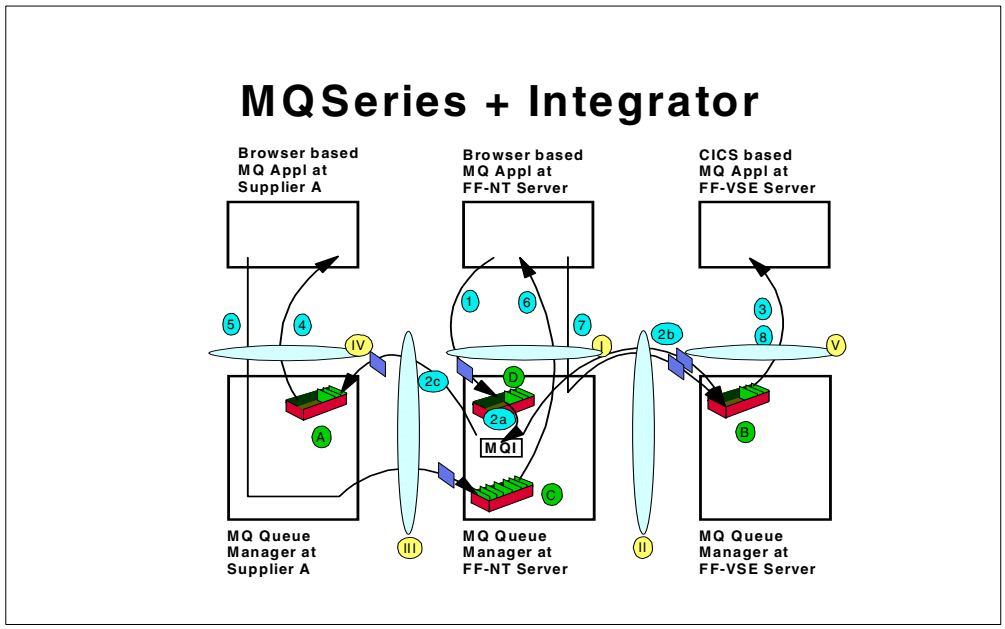


Figure 75. MQSeries Integrator -- overall control flow

This part of the application is fully based on the implementation of the first part, which means that we only modified the user interface applet for the new process and the servlet dealing with the queues. This had been the most complicated part of the application, but with MQSeries Integrator it became a fairly straightforward process. We only had to put one message on an MQSeries queue and let MQSeries deal with the rest.

Since the changes to the Java programs (from part 1 to part 2 of the implementation) were minor, we do not discuss these here in more detail. Instead, we focus on the way we did the setup for MQSeries Integrator so that messages could be read from an input queue and distributed to the target queues according to our rules and definitions.

The requirements for implementing this application with MQSeries Integrator are as follows:

- Small orders should go only to supplier A for requesting a price proposal.
- For large orders we need two proposals, so the request should be sent to suppliers A *and* B.
- “Large orders” means quantities > 100 (our internal business rule).
- All records sent to the suppliers should be stored in the DB2 database on VSE/ESA.

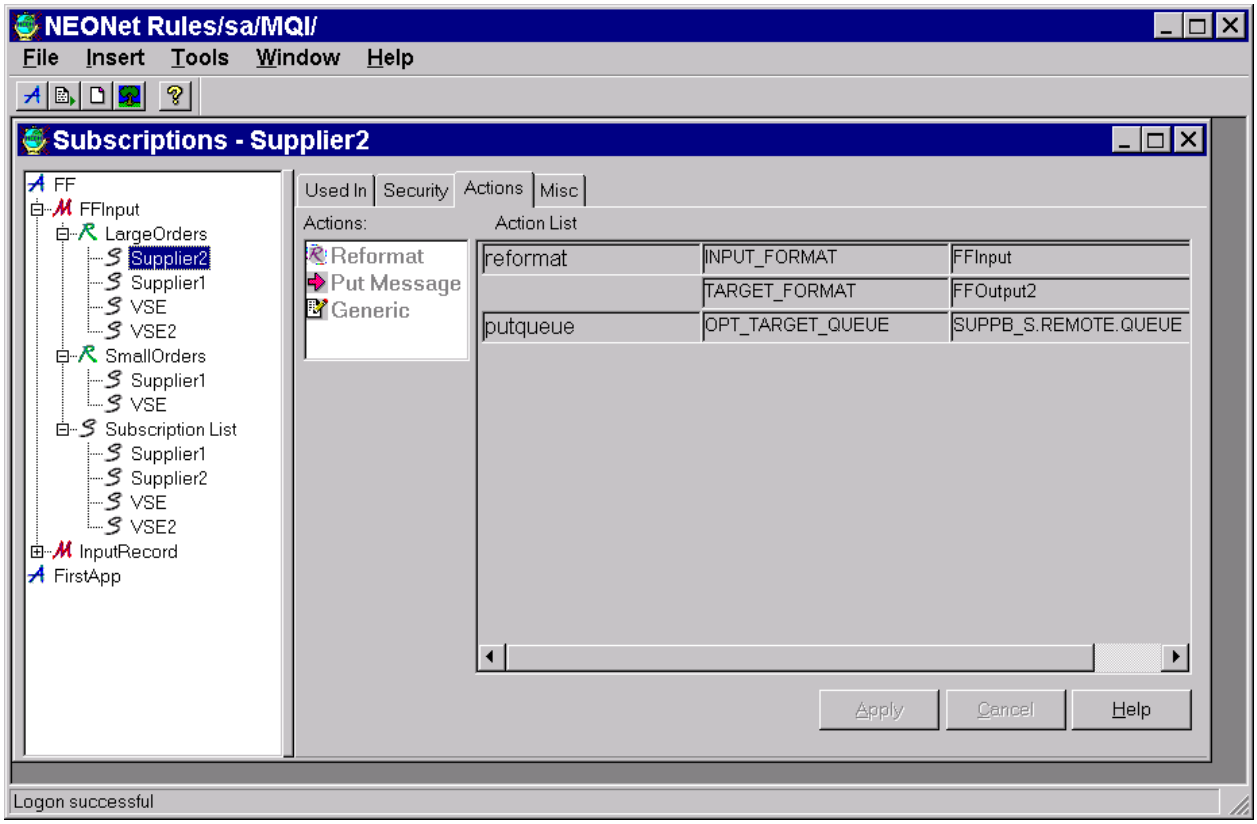


Figure 76. MQSeries Integrator Rules

In the Rules dialog shown in Figure 76 we first defined the subscription list by adding Supplier1, Supplier2, VSE and VSE2 (you will see later why we have two entries for VSE), with all definitions for each entry: Input Format, Target Format, and Target Queue name. Then we defined the two rules Large Orders and Small Orders according to our business rule (large order for quantities > 100). Finally we dragged Supplier1 and VSE on the Small Orders rule and Supplier1, Supplier2, VSE, and VSE2 on the Large Orders rule.

Since our DB2 supplier database and our SCM application are focused on supplier records (not on tasks), we have to get two records into this database for the Large Orders case. Instead of passing one record to the MQSeries queue on VSE and leaving it up to the MQSeries application on VSE, we decided to let MQSeries Integrator handle this requirement. Therefore, we defined two entries (VSE and VSE2) in the Subscription List, both targeting the same VSE queue but using different Target Formats, which then construct different messages with the names of the suppliers in addition to the basic transaction data. After dragging both entries on the Large Orders rule, VSE puts a record with the name of supplier A, and VSE2 puts a record with the name of supplier B on the target queue.

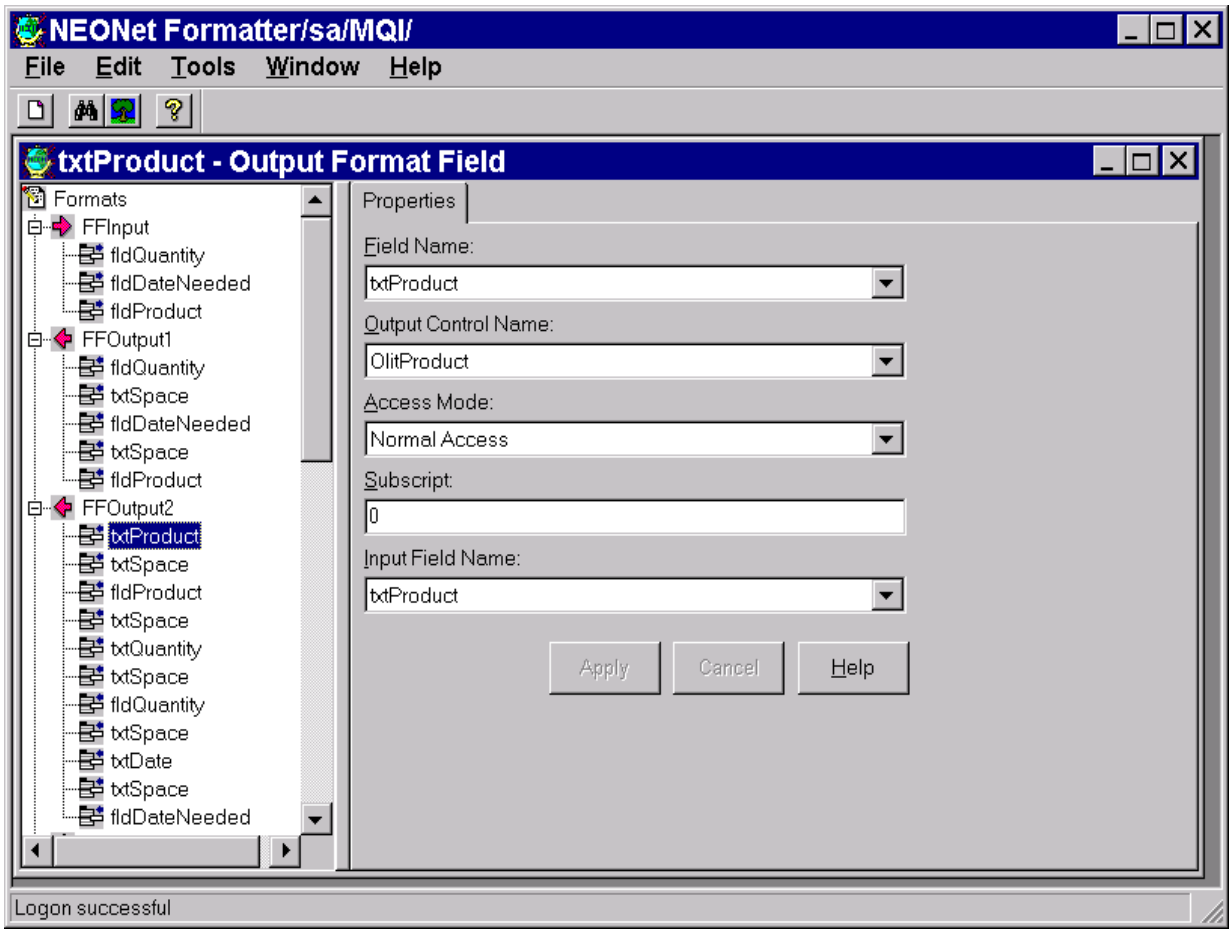


Figure 77. MQSeries Integrator Format dialog

The MQSeries Integrator Format dialog allows you to define different output record formats that are then used by MQSeries Integrator when building the final records to be put on different target queues. Figure 77 shows the definitions we made for the input record as well as for the output records to be put on the supplier A and supplier B queues.

FFInput This record will be parsed into three fields: Quantity, DateNeeded, and Product. Sample record: 150 asap Cocoa.

FFOutput1 This format, used to build the record for the supplier A queue, will basically generate the same record that was received. It contains the same fields in the same sequence, just separated by a text constant containing a space. Sample record: 150 asap Cocoa.

FFOutput2 This format is used to build the record for the supplier B queue. It contains the fields in a different sequence, plus a text constant defining the token. Sample record: Product=Cocoa Quantity=150 Date=asap.

The following definitions have to be made in the MQSeries Integrator startup file to connect the Integrator Engine with the queue manager and the input queue.

```
# MQSeries queue manager name
QueueManagerName = FF_S

# retry limit before replayed message is sent to failure queue (zero
# indicates no replays allowed)
MaxBackoutCount= 0

# these three queue names are mandatory!
InputQueueName   = FF_S.REQUEST.QUEUE
NoHitQueueName   = MQI.DISCARD.QUEUE
FailureQueueName = MQI.DISCARD.QUEUE

# rules default application group and message type values (mandatory)
DefaultAppGroup = FF
DefaultMsgType  = FFInput

[Rules Database Connection]
#
# NEONet connection information for the database (all fields mandatory)
# Note that "DatabaseInstance" is not required for use with Oracle databases.
#
ServerName       = MQI
UserId           = SA
Password         = password
DatabaseInstance = MQI
```

To start the MQSeries Integrator Engine, the following command is used in the NT command prompt (or in a command file starting the entire MQ environment):

```
start "MQI" /min mqsiruleng -p mqsiruleng.mpf
```

The design and definition for the second part of the SCM application are done, so let us now look at the result.

Figure 78 on page 96 shows the user interface of the Request Price part of the SCM application at Friendly Foods. The formats of the reply messages from the two suppliers differ because they receive our message in a different format.

Since the message from supplier B is longer than the message box, this control will automatically get a scroll bar for horizontal scrolling. The price was outside the visible area of the message box when the screen shot was taken. However, you see all the fields of the reply message in Figure 79 on page 97, which shows the records that had been added to the DB2 database on VSE, exploited by the existing SCM application to keep track of the supplier records.

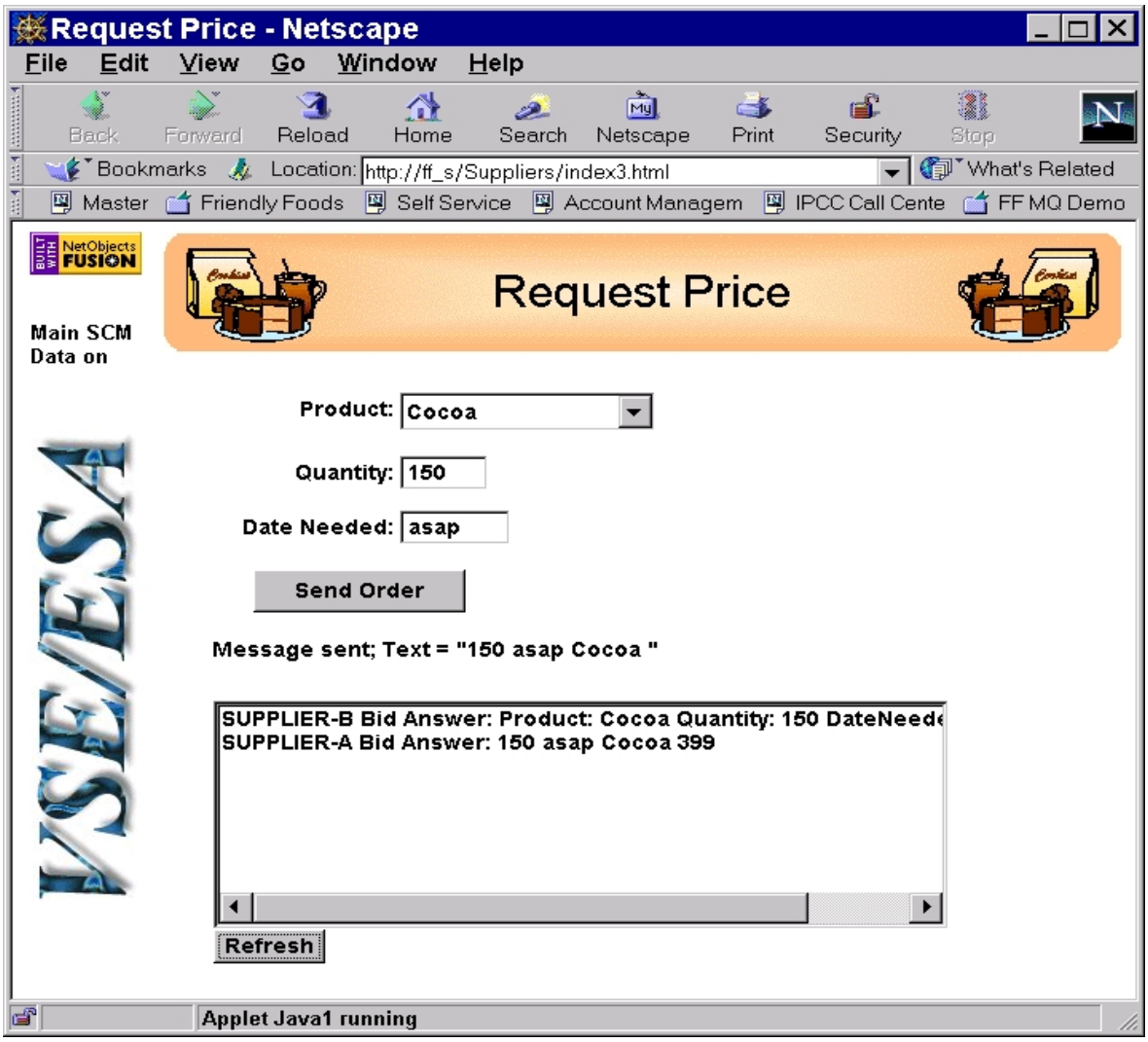


Figure 78. User Interface of the request price application

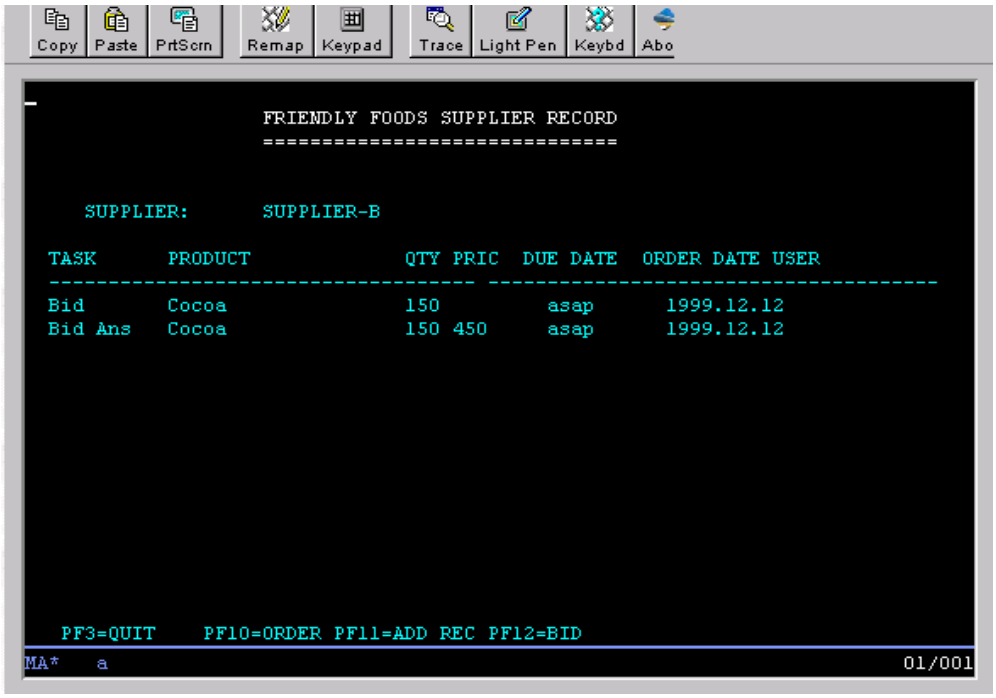


Figure 79. Records added to the DB2 database for one request price transaction

6.4 Implementation of the CICS Transaction Gateway

The first two parts of the new SCM application provided Friendly Foods with the business advantage of faster turnaround cycles in transactions with its suppliers.

For many reasons the browser-based user interface had also been well accepted by the Friendly Foods employees. However, the “older” CICS-based part of the application was still providing much additional value in analyzing completed or pending transactions with suppliers. Since Friendly Foods did not want its employees to switch from a browser-based interface to a 3270 interface and vice versa, and in addition wanted to protect its investment in the CICS-based part of the application, the decision was made to use the CICS Transaction Gateway to connect the CICS-based part of the application with a browser user interface.

The CICS transaction on VSE was designed with a strict separation of presentation logic and business logic. This made this transition easy because there was no need to change a single line of code in the existing CICS transaction.

Figure 80 on page 98 shows the control flow for this application.

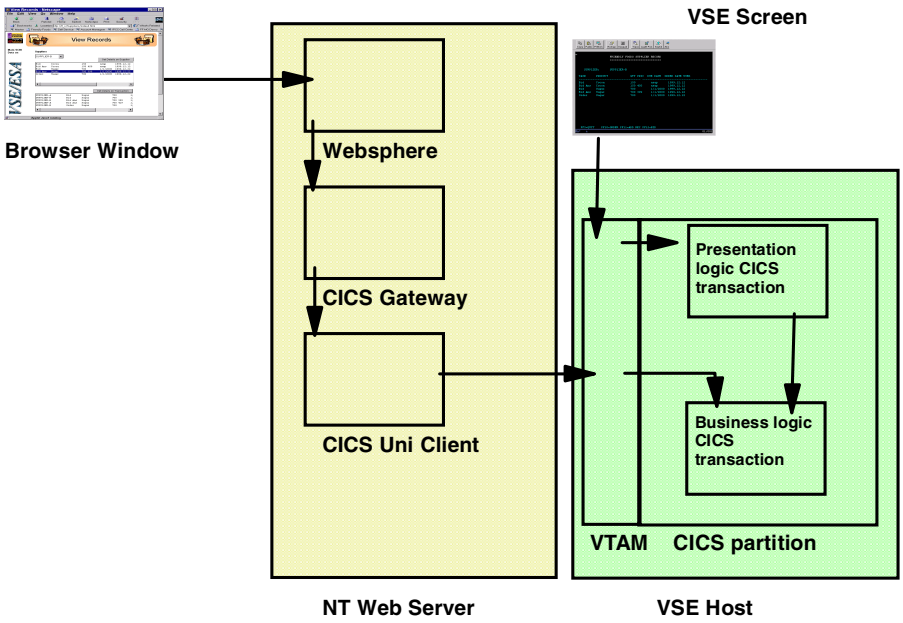


Figure 80. Control flow of the application using CICS Transaction Gateway

The CICS Transaction Gateway implementation uses the ECI (external call interface) in order to communicate between the CICS Universal Client on Windows NT and CICS/VSE. This means that in our case, since we already had a strict separation of presentation and business logic, there was no need to change the code in our CICS transaction programs on VSE. This also allowed us to design a new flow of the dialog steps exploiting the capabilities of the browser-based user interface.

Using CICS ECI meant that we had to develop a new presentation logic. Friendly Foods used VisualAge for Java to implement the presentation logic for this application.

6.4.1 Introduction to the CICS Transaction Gateway

As mentioned before, the CICS Transaction Gateway is a set of server-based software components that enables a Java applet running in a Web browser, or a Java application or servlet running on a Web server, to access programs or transactions running in a CICS environment. The CICS Transaction Gateway can be connected to CICS/VSE V2.3, as well as to CICS Transaction Server for VSE/ESA. The CICS Universal Client supports the following three external interfaces:

- External call interface (ECI)
- External presentation interface (EPI)
- External security interface (ESI)

Refer to the appropriate CICS publication for a more complete discussion of the CICS Transaction Gateway and the CICS Universal Client.

The general flow of the CICS Transaction Gateway, including the CICS Universal Client, is shown in Figure 81.

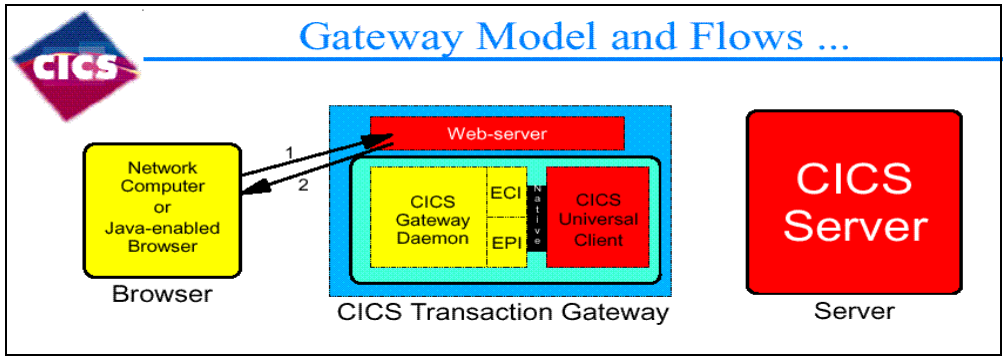


Figure 81. CICS Transaction Gateway flow (part 1 of 3)

The flow is as follows:

1. A Web browser asks for an HTML page from a Web server.
2. The Web server returns the HTML page, including an applet tag identifying a CICS-enabled applet to run.

The browser then requests the relevant Java classes from the Web server.

Once all the necessary classes are downloaded, the browser begins to execute the applet.

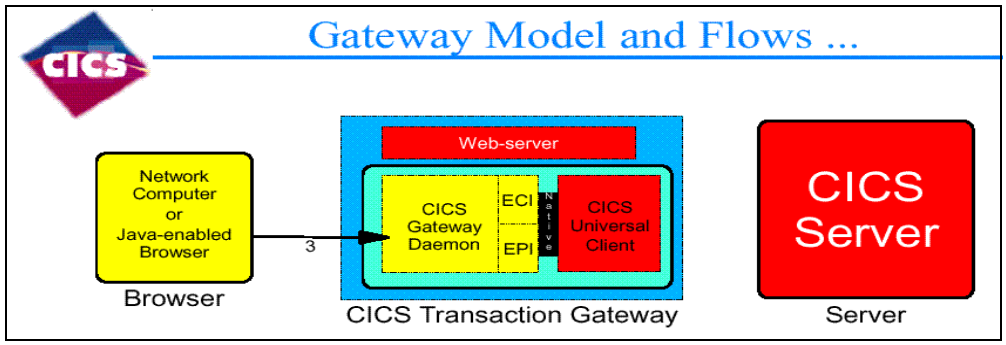


Figure 82. CICS Transaction Gateway flow (part 2 of 3)

3. The applet creates a Java Gateway object to connect to the Gateway. See Figure 82.

   It then creates a request object, for example an ECI request object, containing its CICS request. It flows this request through the Java Gateway, causing the request to be packed up and sent down the connection.

   A small native shared library is used to call the CICS Universal Client interface.

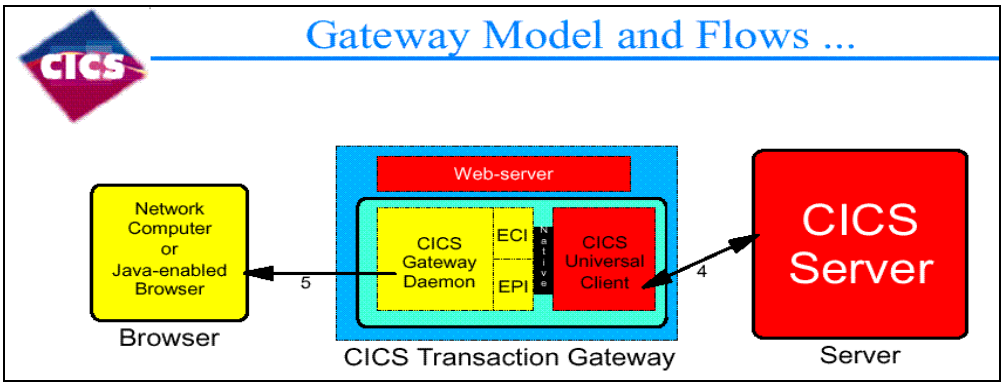


Figure 83. CICS Transaction Gateway flow (part 3 of 3)

4. The Gateway receives the request, unpacks it and makes the relevant calls to the CICS Universal Client, which then calls the CICS server. See Figure 83.
5. On returning from the native call to the CICS Universal Client, the Gateway packs up the results and sends them back to the browser.

Installing the CICS Transaction Gateway allowed Friendly Foods to write new non-CICS applications that gain access to CICS facilities and data using the ECI programming interface. There is no need for the new application to run on VSE/ESA, but it can run on any other operating system that is supported by the CICS Universal Client.

6.4.2 Setting up the connection

The CICS Universal Client supports an SNA LU 6.2 (APPC) connection to the CICS server running on VSE/ESA.

In order to establish this connection we needed a program package on Windows NT that supports this protocol. We used SecureWay Personal Communications. For the APPC connection, we had to make definitions in VTAM on VSE, CICS/VSE, CICS Transaction Gateway and SecureWay Personal Communications.

6.4.2.1 Installing and setting up SecureWay Personal Communications

We performed a server installation of SecureWay Personal Communications as shown in Figure 84 on page 101.

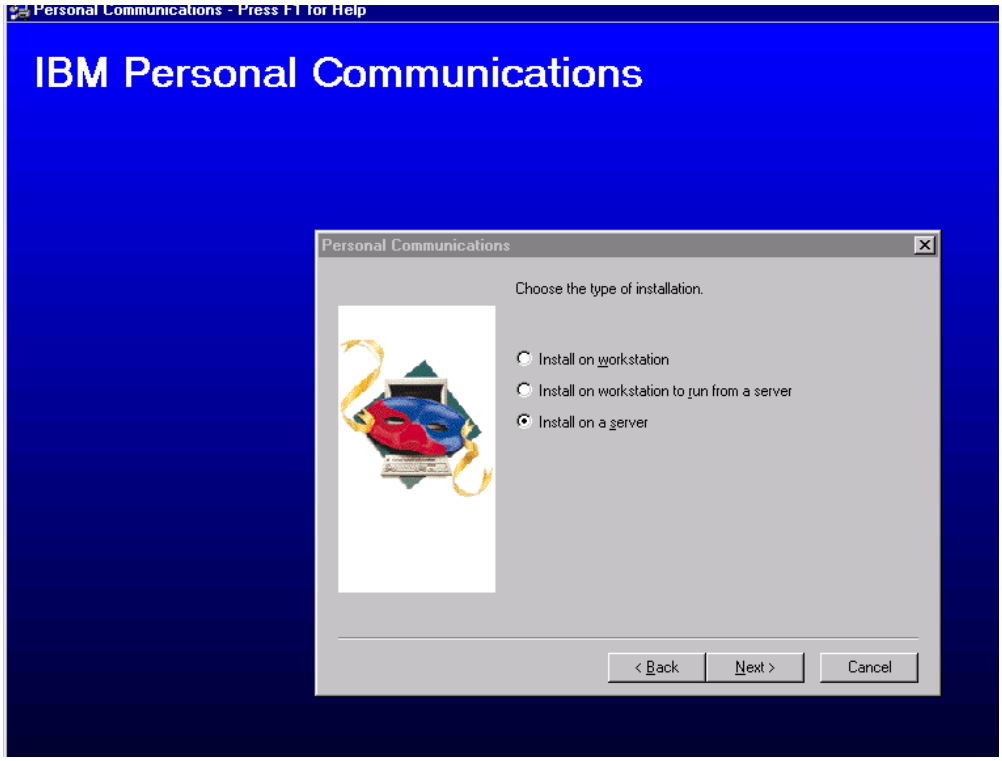


Figure 84. SecureWay Personal Communications -- installation

We selected that we want to install the Personal Communications SNA protocols on the next screen, which allowed us later on to make the appropriate SNA definitions for our APPC connection to VSE. After providing the install path, we were done with the installation of Personal Communications.

The next steps describe the necessary configuration within Personal Communications. Note that the definitions you make in SecureWay Personal Communications have to match the definitions you make in VTAM and CICS on VSE as described in 6.4.2.2, “Definitions on VSE/ESA” on page 108. Also note that we used VTAM APPN support in order to keep this configuration simple, and we did not have to explicitly define each PU and LU to VTAM. The PU and LU definitions are dynamically created by VTAM when a connection is established.

As already mentioned, VSE/ESA is installed on a P/390 system, which implies that we are using the Token-Ring adapter of the PC server for the connection with the Thinkpad acting as a Web server and client. VTAM sees the Token-Ring adapter as a 3172. The VTAM definitions on VSE/ESA have to reflect this hardware configuration.

Enter the SecureWay Personal Communications configuration dialog by calling the executable pcscfg. This will guide you through the following steps of the Personal Communication’s configuration dialog:

1. Configure the node

   Figure 85 on page 102 shows the node configuration panel. Select **Configure Node** and click **New** to define the node. This leads you to the panel shown in Figure 86 on page 103, which lets you define all necessary node parameters.

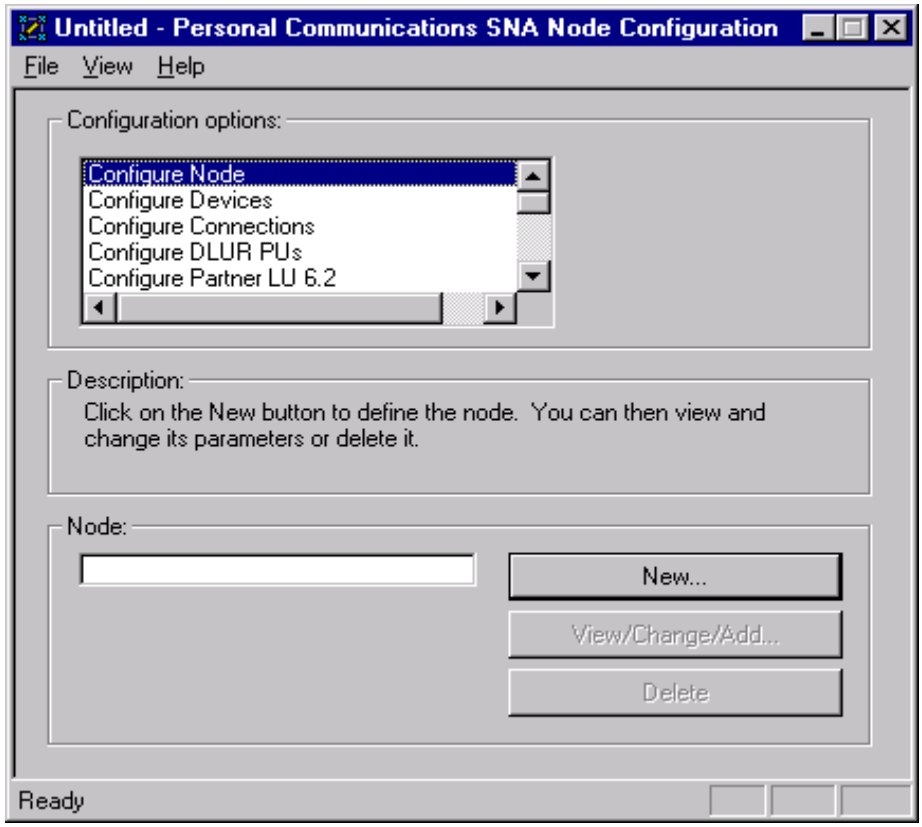


Figure 85. Personal Communications -- node definition (part 1 of 2)

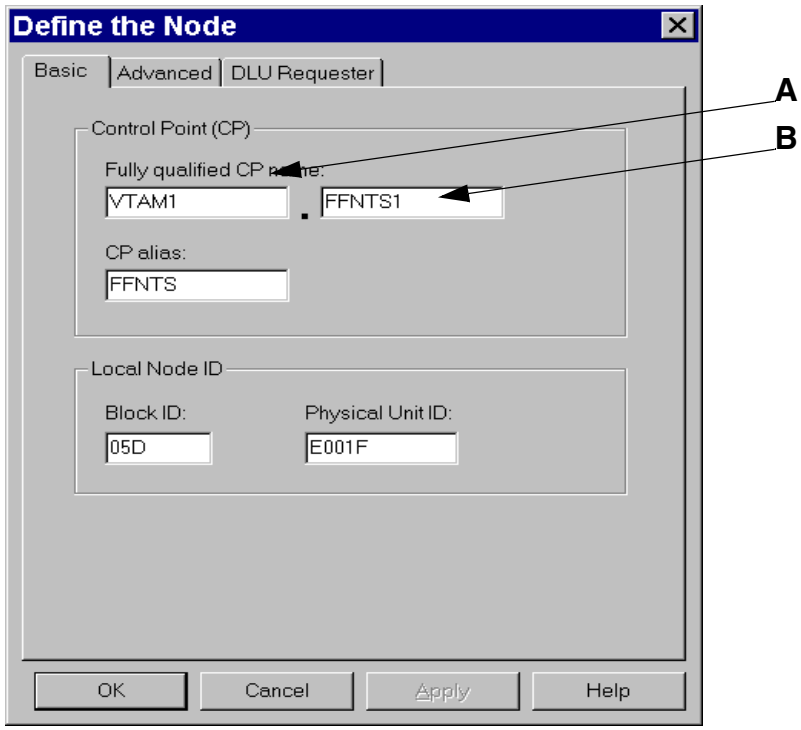


Figure 86. Personal Communications -- node definition (part 2 of 2)

The screen in Figure 86 is used to define the local node, as follows:

- *Field A* has to be specified according to the VTAM NETID parameter in member ATCSTR00.B, as shown in Figure 93 on page 109.
- *Field B* has to be specified but does not have a counterpart in VTAM. You can specify anything for the local CP name.
- The Block ID identifies the device or product type in the network. You have to specify 05D when using SecureWay Personal Communications.
- The Physical Unit ID identifies the physical device in the network. This definition must be unique for each workstation within the same network.

2. Define a LAN device

   The device definition specifies the connection between the local adapter in the PC Server and another system in the network (VSE in our case). We are using a Local Area Network (LAN) and accepted all values that were specified as defaults on the panels.

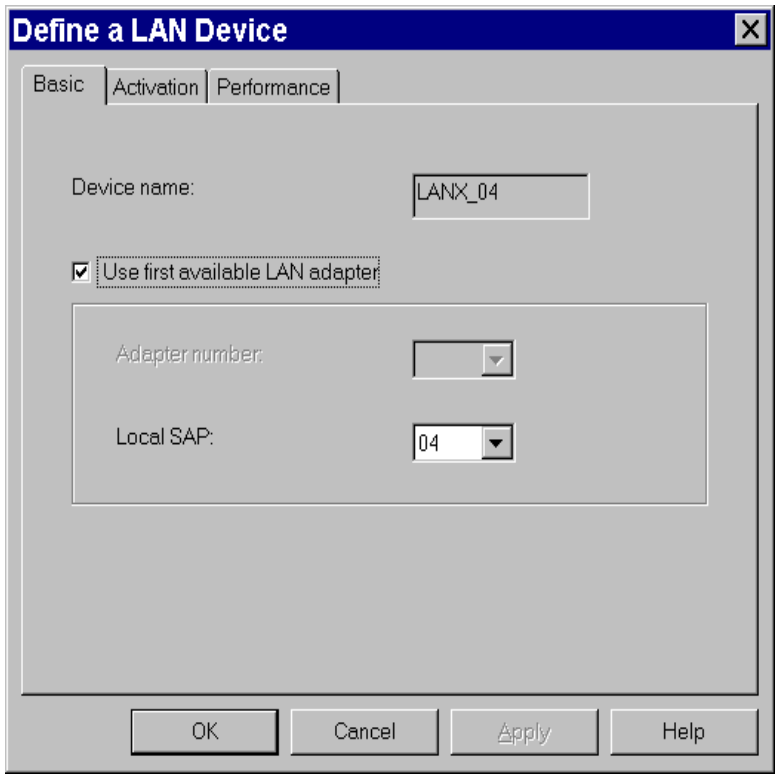


Figure 87. Personal Communications -- device definition

3. Define the LAN connection

   The following parameters have to be defined to specify the LAN connection:

- The Device name as specified in Figure 87.
- The Destination address is the address of the LAN adapter in the destination device. In our case, this is the address of the Token-Ring adapter (as defined in MPTS) in the P/390 system. It is either defined through MPTS or it is the burnt-in address of the Token-Ring adapter.
- For the Remote SAP specify the value as defined in your VTAM major node for your communication controller. In our case, it is defined in the XCA major node (see Figure 94 on page 110).
- Specify whether your connection is Token-Ring or Ethernet.

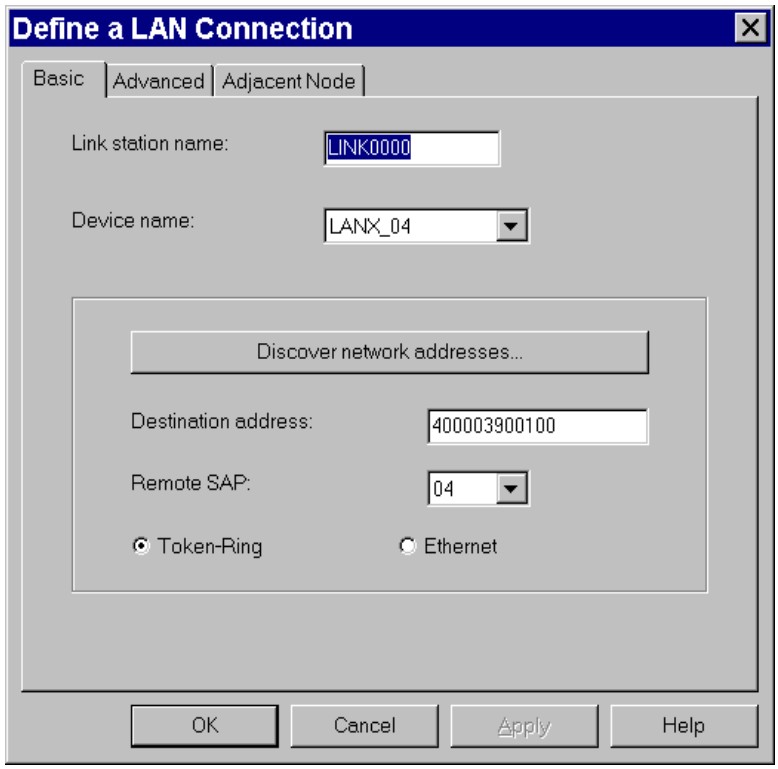


Figure 88. Personal Communications -- LAN connection definition (part 1 of 3)

The following definitions have to be made on two additional panels that you invoke by selecting **Advanced** and **Adjacent Node,** respectively.

Check the following boxes on the Advanced panel:

- Activate Link at start
- APPN support
- Auto-activate support
- Solicit SSCP session

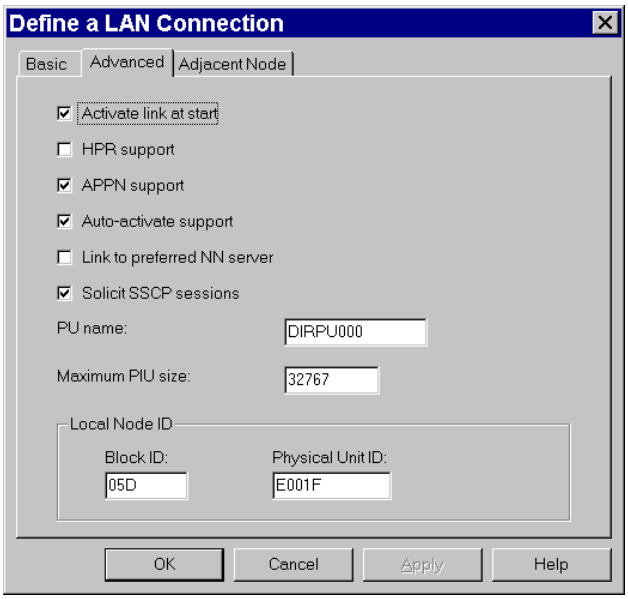


Figure 89. Personal Communications -- LAN connection definition (part 2 of 3)

Specify the following values on the Adjacent Node panel, which is used to define the connection in VTAM on VSE/ESA:

- The Adjacent CP name is VSE's CP name, which consists of the network name as specified in the VTAM startup parameters shown in Figure 93 on page 109 by the NETID parameter, and the adjacent CP name as specified by SSCPNAME in the same VTAM configuration book.
- Specify Network Node for Adjacent CP type.

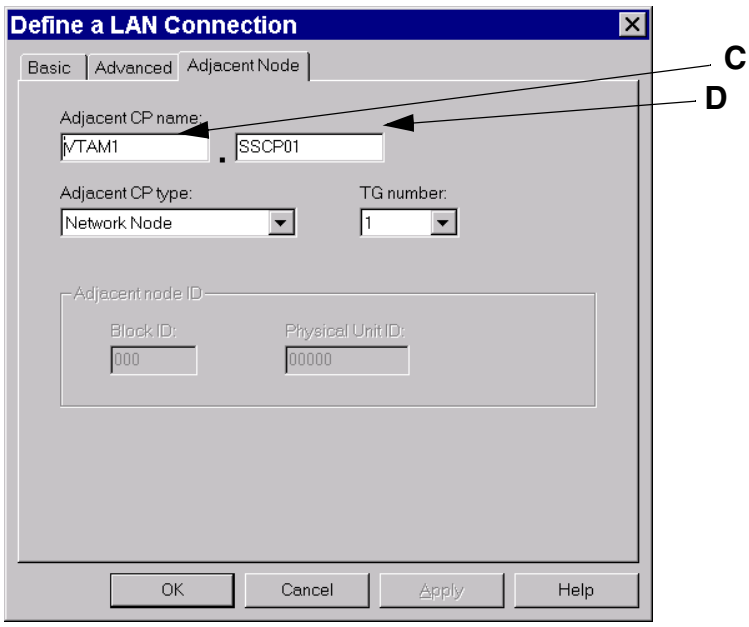


Figure 90. Personal Communications -- LAN connection definition (part 3 of 3)

4. Define the partner LU6.2

   The partner LU name consists of two parts (see Figure 91), the *network name* (Field E) and the *LU name* (Field F). In our case, the network name is VTAM1 as specified in the VTAM startup book by the NETID parameter (see Figure 93 on page 109) and the LU name is PRODCICS, the name of our production CICS.

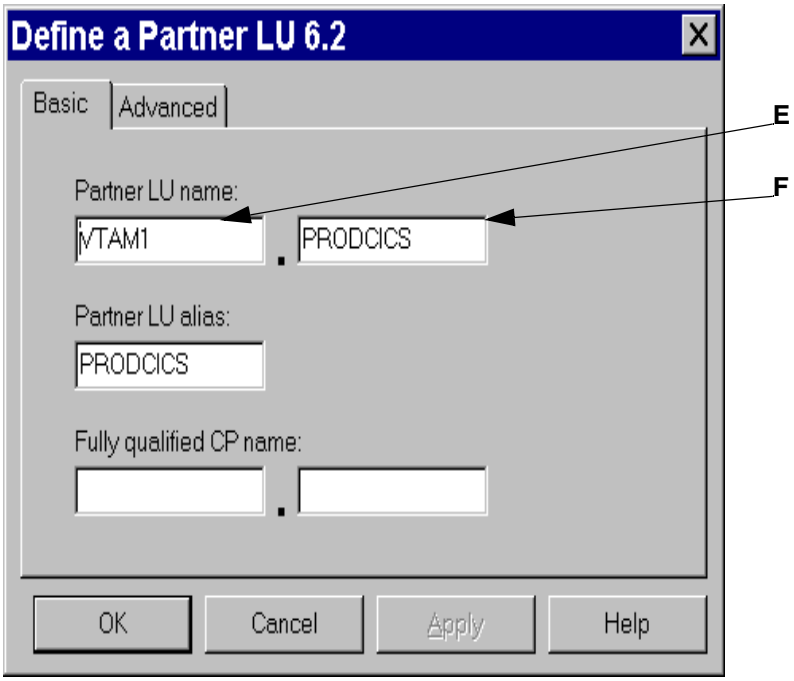


Figure 91. Personal Communications -- partner LU6.2 definition

5. Define the local LU6.2

   The local LU6.2 is specified by the Local LU name. This can be any 8-character name, but it has to match the name of the LU as defined in the VTAM switched major node shown in Figure 94 on page 110. In our case, it is FFNTS.

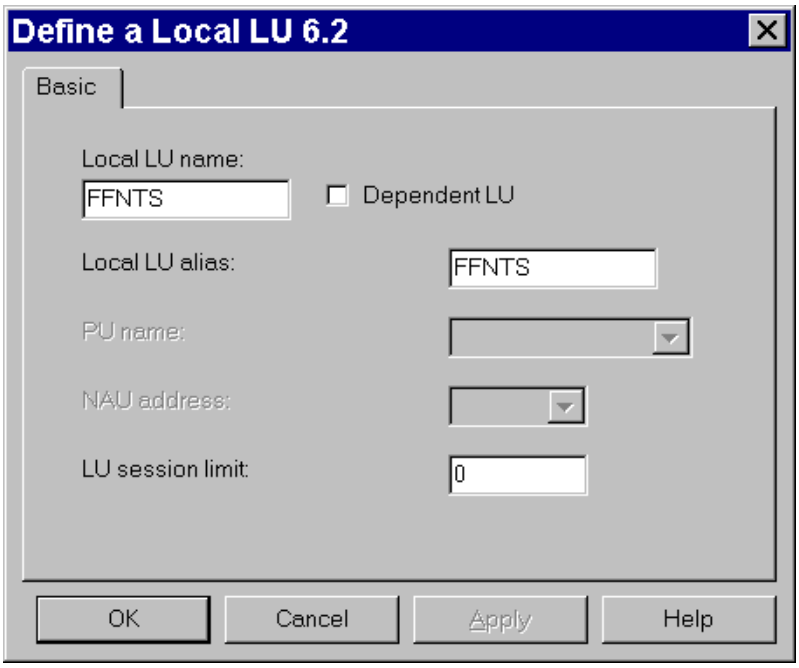


Figure 92. Personal Communications -- local LU6.2 definition

6.4.2.2 Definitions on VSE/ESA

The following definitions have to be made on VSE/ESA:

1. VTAM startup parameters defined in ATCSTR00.B

   Figure 93 on page 109 shows the settings used in our VTAM startup book. It is based on ATCSTR00, which is shipped in ICCF library 51.

```
CATALOG ATCSTR00.B
SSCPID=1,
SSCPNAME=SSCP01,
NETID=VTAM1,
HOSTSA=12,
DYNLU=YES,
NODETYPE=NN,
APPNCOS=#INTER,
HOSTPU=NODE01,
MAXSUBA=255,
CONFIG=00,
NOPROMPT,
IOINT=0,
SGALIMIT=0,
BSBUF=(28,,,,1),
CRPLBUF=(60,,,,1),
LFBUF=(70,,,,11),
IOBUF=(70,288,,,11),
LPBUF=(12,,,,6),
SFBUF=(20,,,,20),
SPBUF=(210,,,,32),
XDBUF=(6,,,,1)
```

Figure 93. VTAM definitions -- ATCSTR00.B

NETID=VTAM1

This parameter is very basic. The same name has to be used by all partners (host and workstations) that are using the same network. This name is used in various places during SecureWay Personal Communications configuration and in the CICS Universal Client setup.

NODETYPE=NN

When Nodetype=NN and HOSTSA are coded, the node operates as an interchange node. An interchange node combines the functions of a subarea node and a network node. It controls resources and functions as a network node in the APPN network and as an SSCP and a cross-domain resource manager (CDRM) in the subarea network.

DYNLU=YES

This parameter is required to have your independent logical units dynamically defined. Independent LUs attaching to VTAM through a type 2.1 PU can be dynamically defined as cross-domain resources. No system definition of the independent LU is required.

This means that if this parameter is coded, you are not required to predefine all resources that can be accessed over adjacent link stations. The LUs of the workstations need no VTAM definition in VSE/ESA.

APPNCOS=#INTER

This is an APPN class of service for LU-LU sessions. It specifies a general, interactive class of services that uses a high transmission priority, and for which short delay is considered more important than high bandwith and low cost.

2. VTAM XCA major node definition

```
CATALOG VTM3172C.B
VTM3172C VBUILD TYPE=XCA
PORT3172 PORT CUADDR=600,                                              X
       MEDIUM=RING,ADAPNO=0,TIMER=60,SAPADDR=4
GRP3172 GROUP DIAL=YES,DYNPU=YES,DYNPUPFX=PX,AUTOGEN=(16,L,P),         X
       ANSWER=ON
```

Figure 94. VTAM definitions -- XCA major node

DYNPU =YES

This specifies that a PU is to be dynamically allocated during a switched call-in operation.

3. CICS definitions

   We used the following CICS connection definitions in our configuration:

```
  OBJECT CHARACTERISTICS
   CEDA View
    Connection :   CCTG
    Group :        CICSNT
   CONNECTION IDENTIFIERS
    Netname:       FFNTS
    INDsys:
   REMOTE ATTRIBUTES
    REMOTESystem :
    REMOTEName :
   CONNECTION PROPERTIES
    ACcesmethod:   Vtam            Vtam | IRc | INdirect
    Protocol:      Appc            Appc | Lu61
    Singlesess :   No              No | Yes
    Datastream :   User            User | 3270 | SCs | STrfield | Lms
    RECordformat : U               U | Vb
   OPERATIONAL PROPERTIES
    AUtoconnect :  Yes             No | Yes | All
  + INService :    Yes             Yes | No
```

Figure 95. CICS connection definition

The key parameters are:

- The value specified for `Connection` must match the Connection name in the CICS/VSE sessions definition shown in Figure 95.
- Specify the name of the `Group` this definition belongs to.
- `Netname` must match the LU name and Alias name in the local LU definition as specified in the SecureWay Personal Communications configuration in Figure 92 on page 108.
- Define `VTAM` for `Accessmethod`.

– Define APPC for Protocol.

– Define No for Singlesess in order to use APPC parallel sessions.

Figure 96 shows our definitions for the CICS session.

```
OBJECT CHARACTERISTICS
 CEDA View
  Sessions : SCTG
  Group: CICSNT
 SESSION IDENTIFIERS
  Connection :CCTG
  SESSName :
  NETnameq :
  MOdename: #INTER
 SESSION PROPERTIES
  Protocol:      Appc            Appc | Lu61
  MAximum :      00008,00004     0-32767
  RECEIVEPfx :
  RECEIVECount : No              No | 1-999
  SENDPfx :
  SENDCount :     No             No | 1-999
  SENDSize :      04096          1-30720
  RECEIVESize:    04096          1-30720
 OPERATOR DEFAULTS
  OPERid :
  OPERPriority :  000            0-255
........
```

Figure 96. CICS session definition

The key parameters you have to specify are as follows:

– Sessions represents the session ID as it is known within the group definition.

– Group is the name of the group to which this definition belongs.

– Connection defines the name of the connection associated with this session and must match the Connection name in CICS/VSE's connection definition in Figure 95 on page 110.

– Modename specifies the name of the mode used for this session. This name has to match the modename specified in SecureWay Personal Communications (we are using #INTER).

– APPC has to be specified for an APPN connection for Protocol.

– Maximum indicates the number of sessions that are to be supported for the modeset.

- *value1* is the maximum number of sessions in the group.
- *value2* is the maximum number of sessions that are to be supported as contention winners.

6.4.3 Installation and customization of the CICS Transaction Gateway

As already mentioned, the CICS Transaction Gateway includes the CICS Universal Client. When installing the CICS Transaction Gateway, the Universal Client is automatically installed. We did a standard installation of the CICS Transaction Gateway.

After the installation, the CICS Universal Client as well as CICS Transaction Gateway have to be customized.

1. Customization of the CICS Universal Client

   The customization of the CICS Universal Client is done in a configuration file named CICSCLI.INI. This file can be found in the directory you defined into which to install the CICS Transaction Gateway. Our configuration file looks like this:

```
;*********************************************************************
;* IBM CICS Universal Client - Initialization File                    *
;*********************************************************************

; Format:
;   Section = Name
;      Parameter = Value
;      Parameter = Value
;      ...
;
; "Section" must be either Client, Server or Driver.  There must be just
; one Client section but there may be several Server or Driver sections.
; For each type of section a set of Parameters and associated Values may
; be defined.  Often these may be omitted and will then assume sensible
; default values.  This sample INI file lists only those parameters that
; typically may required changing.
;
; Refer to the "IBM CICS Universal Clients Administration" book for full
; details of available parameters and values for each section.

;---------------------------------------------------------------------
; Client section - This section defines the local CICS client.  There
;                  should only be one Client section.

Client = *                      ; Auto-install client on the server
   MaxServers = 1               ; Only allow one server connection
   MaxRequests = 256            ; Limit the maximum server interaction
   MaxBufferSize = 32           ; Allow for a 32K maximum COMMAREA
   LogFile = CICSCLI.LOG        ; Set the error log file name
   TraceFile = CICSCLI.TRC      ; Set the trace log file name
   DumpFile = CICSCLI.DMP       ; Set the memory trace dump file name
   DumpMemSize = 16             ; Allow for 16k of trace in memory
;  CPName = ABCD1234.EFGH5678   ; The TCP62 client's fully qualified CP name
;  DomainNameSuffix = cicstcp.ibm.com ; Domain name suffix for TCP62 server
```

```
;---------------------------------------------------------------------
; Server section - This section defines a server to which the client may
;                  connect.  There may be several Server sections.
;
; The default example is for TCP/IP communications.  Further examples
; for other protocols are shown but are commented out.
; Beware that these are examples only, for successful communications
; parameters may need changes from their default or illustrated values.

; Server = CICSTCP                ; Arbitrary name for the server
;   Description = TCP/IP Server   ; Arbitrary description for the server
;   Protocol = TCPIP              ; Matches with a Driver section below
;   NetName = cicstcp.ibm.com     ; The server's TCP/IP address
;   Port = 0                      ; Use the default TCP/IP CICS port

;Server = CICSNETB                ; Arbitrary name for the server
;   Description = NetBIOS Server ; Arbitrary description for the server
;   Protocol = NETBIOS           ; Matches with a Driver section below
;   NetName = CICSOS2            ; The server's NetBIOS name
;   Adapter = 0                  ; Use NetBIOS on LAN adapter 0
;   UpperCaseSecurity = Y        ; Fold Userid and Password to uppercase
;   InitialTransid = CLOG        ; Initial terminal transaction name

Server = FFNTS1                 ; Arbitrary name for the server
   Description = SNA Server     ; Arbitrary description for the server
   Protocol = SNA               ; Matches with a Driver section below
   NetName = VTAM1.PRODCICS     ; The server's fully qualified LU name
   LocalLUName = FFNTS          ; The client's local LU name
   ModeName = #INTER            ; The SNA communications mode name

;Server = CICST62                ; Arbitrary name for the server
;   Description = TCP62 Server   ; Arbitrary description for the server
;   Protocol = TCP62             ; Matches with a Driver section below
;   NetName = ABCD1234.EFGH5678  ; The server's fully qualified LU name
;   LocalLUName = WXYZ9999       ; The client's local LU name
;   ModeName = TN62              ; The SNA communications mode name
```

```
;--------------------------------------------------------------------
; Driver section - This section defines a communications protocol DLL
;                  used to communicate with a server.  There may be
;                  several Driver sections.
;
; The default example is for TCP/IP communications.  Further examples
; for other communications protocols are shown but are commented out.

;Driver = TCPIP                 ; Matches the Server's Protocol value
;   DriverName = CCLWNTIP       ; Use the WinNT TCP/IP communications DLL

;Driver = NETBIOS               ; Matches the Server's Protocol value
;   DriverName = CCLWNTNB       ; Use the WinNT NetBIOS communications DLL

Driver = SNA                   ; Matches the Server's Protocol value
   DriverName = CCLWNTSN       ; Use the WinNT/IBM SNA communications DLL

;Driver = TCP62                 ; Matches the Server's Protocol value
;   DriverName = CCLTCP62       ; Use the IBM TCP62 communications DLL

;********************************************************************
;* End of file                                                      *
;********************************************************************
```

Figure 97. CICS Universal Client configuration file

In the configuration file, we changed the following:

– Added a server section by using the `Server=CICSSNA` example. The values used have to match the definitions you made in SecureWay Personal Communications.

– Removed the semicolon (;) in front of `Driver=SNA` to activate the SNA driver.

2. Customization for the CICS Transaction Gateway

   The CICS Transaction Gateway offers a graphical user interface to customize the parameters. In this dialog, we had to change the code page that is used by the CICS Universal Client.

   Even though the Windows NT server was claiming to use codepage 437, we got the error message that our translation table did not cover the involved combination of codepages (on NT and VSE), or that this combination is not supported. Since we defined the combination in the translation table (437 on the ASCII side and 037 on the EBCDIC side), the assumption was made that NT was providing another codepage to the CICS Transaction Gateway. We had to overwrite the codepage to be used to fix this problem. This is shown in Figure 98 on page 115.

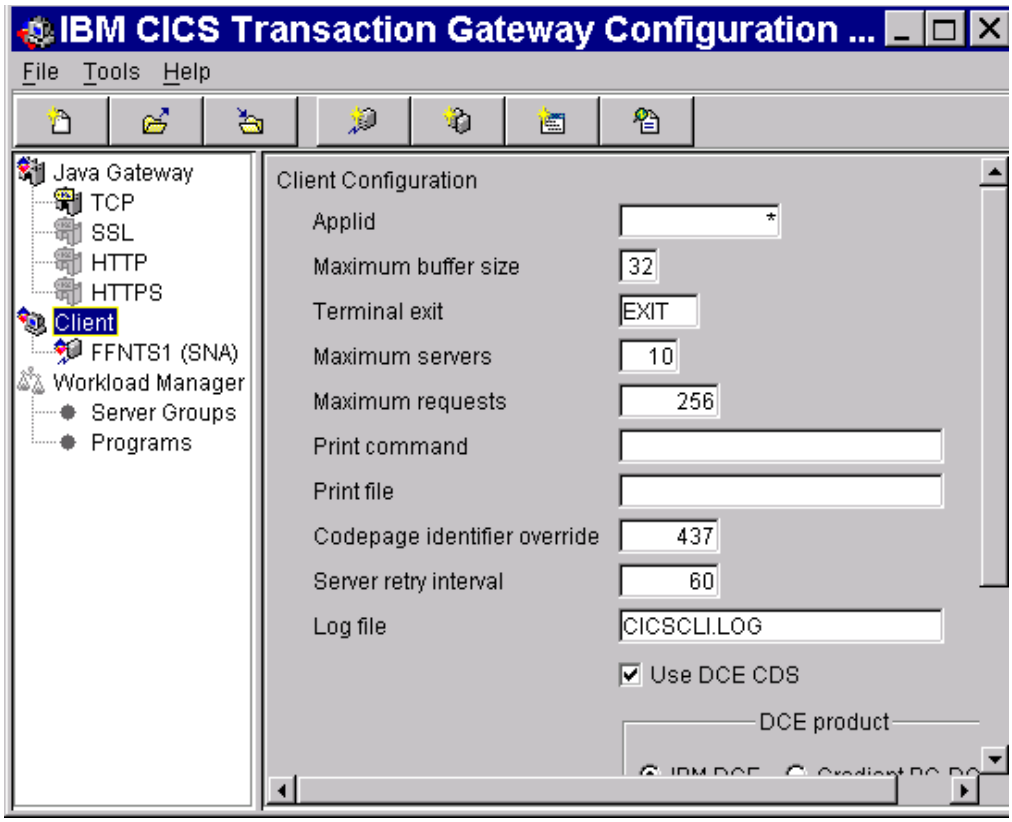


Figure 98. Configuring the CICS Transaction Gateway -- overwrite value for codepage

The corresponding change on the VSE side is in the CICS conversion table DFHCNV. We had to add an entry for the program we call from the client side specifying the codepages involved in the ASCII-to-EBCDIC conversion as shown in Figure 99. The name of the program that is called from the CICS Transaction Gateway is FFSBL.

```
 DFHCNV TYPE=INITIAL,CLINTCP=437,SRVERCP=037

DFHCNV TYPE=ENTRY,RTYPE=PC,RNAME=FFSBL

DFHCNV TYPE=SELECT,OPTION=DEFAULT

DFHCNV TYPE=FIELD,OFFSET=0,DATATYPE=CHARACTER,DATALEN=32767,LAST=YES

DFHCNV TYPE=FINAL

END     DFHCNVBA
```

Figure 99. Conversion table for CICS

6.4.4 Establish the APPC connection

For establishing the APPC connection between SecureWay Personal Communications and VSE, the VTAM and CICS definitions on VSE first have to

be activated. After starting the Personal Communications session, you should see the following message on the VSE console:

```
ISTI059I CONNECTIN ESTABLISHED FOR PU PX000005 ON LINE L0600000
ISTI086I APPN CONNECTION FOR VTAM1.FFNTS1 is ACTIVE - TGN = 1
ISTI097I CP-CP SESSION WITH VTAM1.FFNTS1 ACTIVATED
```

Figure 100. Establish the APPC connection -- console output

The messages indicate that VTAM created dynamic definitions for the incoming PU.

Note that it might be necessary that you manually acquire the connection in CICS.

If all your definitions are correct, you should also be able to see the LU6.2 sessions in your SecureWay Personal Communications status window.

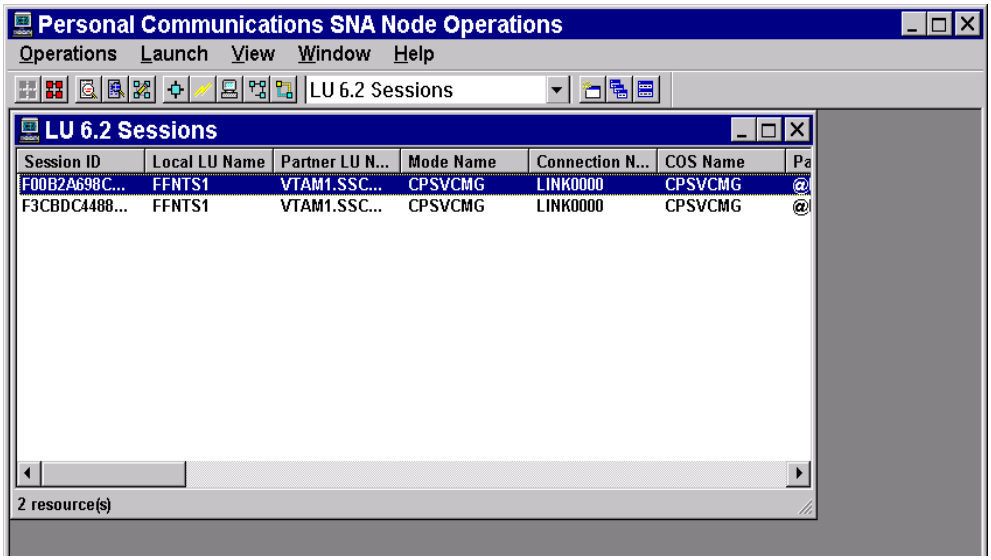


Figure 101. SecureWay Personal Communications -- LU 6.2 sessions

6.4.5 Implementing the CICS-based application on VSE/ESA

The base for the MQSeries application program for receiving the messages on VSE/ESA was the sample program MQPECHO provided with the MQSeries code (parts of this program are listed in Figure 65 on page 81). We just added some code to analyze the records and add them to DB2.

.

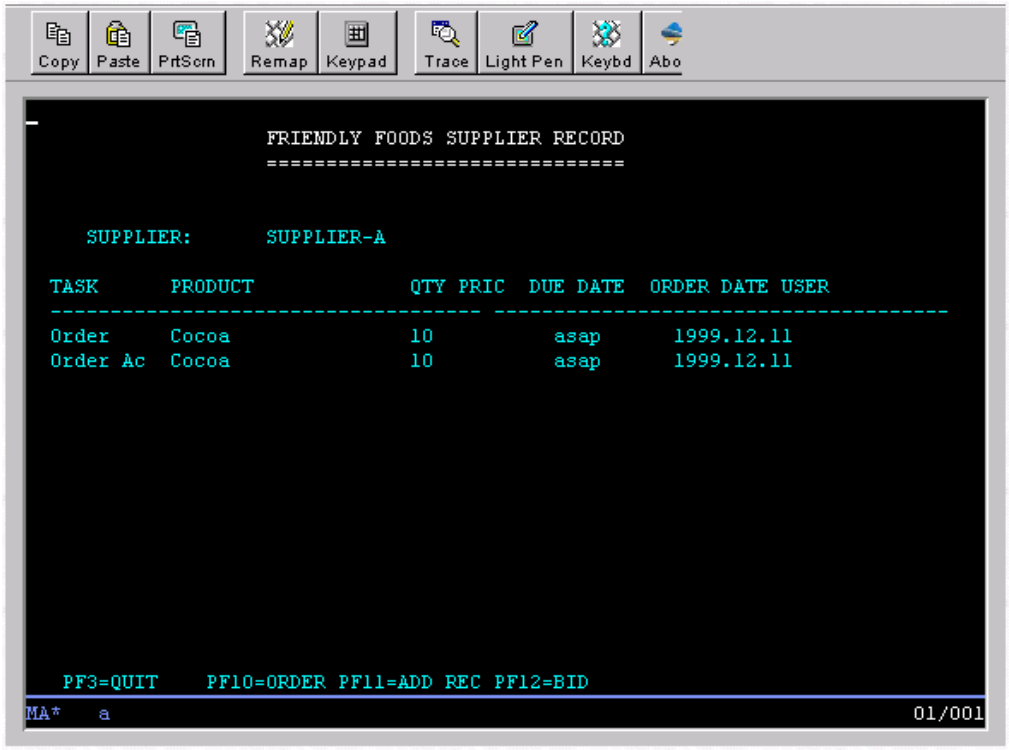


Figure 102. Records added to the DB2 database for one purchase transaction

The screen in Figure 102 shows the records that were added to the DB2 database by the previously discussed COBOL program, which monitors the MQSeries queue on VSE. The existing SCM application (COBOL) is used to keep track of the supplier records.

6.4.6 Final application panel created with VisualAge for Java

Before we discuss how the application was developed with VisualAge for Java, let us have a look at the final result and compare it with the panels used in the existing CICS transaction.

Figure 103 on page 118 shows one of the 3270 panels driven by the presentation logic part of the existing CICS application. To get all the details about one particular supplier and one specific transaction, there are three dialog steps:

1. A panel showing all suppliers -- selecting the supplier
2. A panel showing all transactions with the selected supplier (shown in Figure 103 on page 118) -- selecting the transaction in question
3. A panel showing all details for the selected transaction

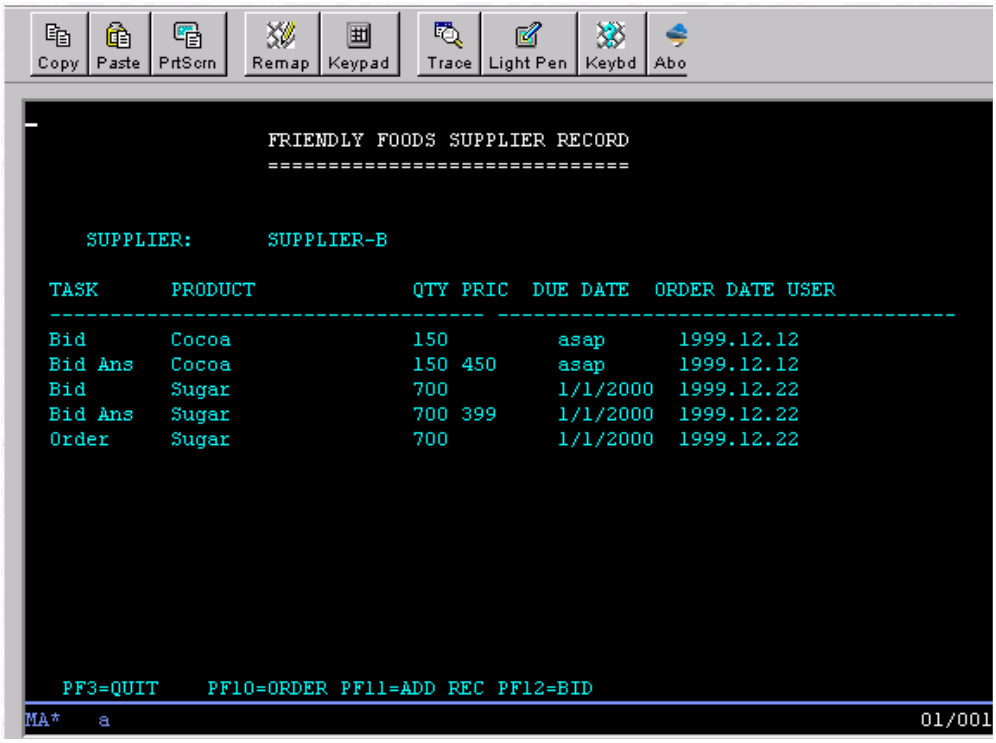


Figure 103. Presentation logic of the VSE-based CICS transaction

Figure 104 on page 119 shows the browser-based user interface of the new application, accessing the same business logic on VSE. Of course, the new application consists of the same dialog steps, but the results are combined in a single screen.

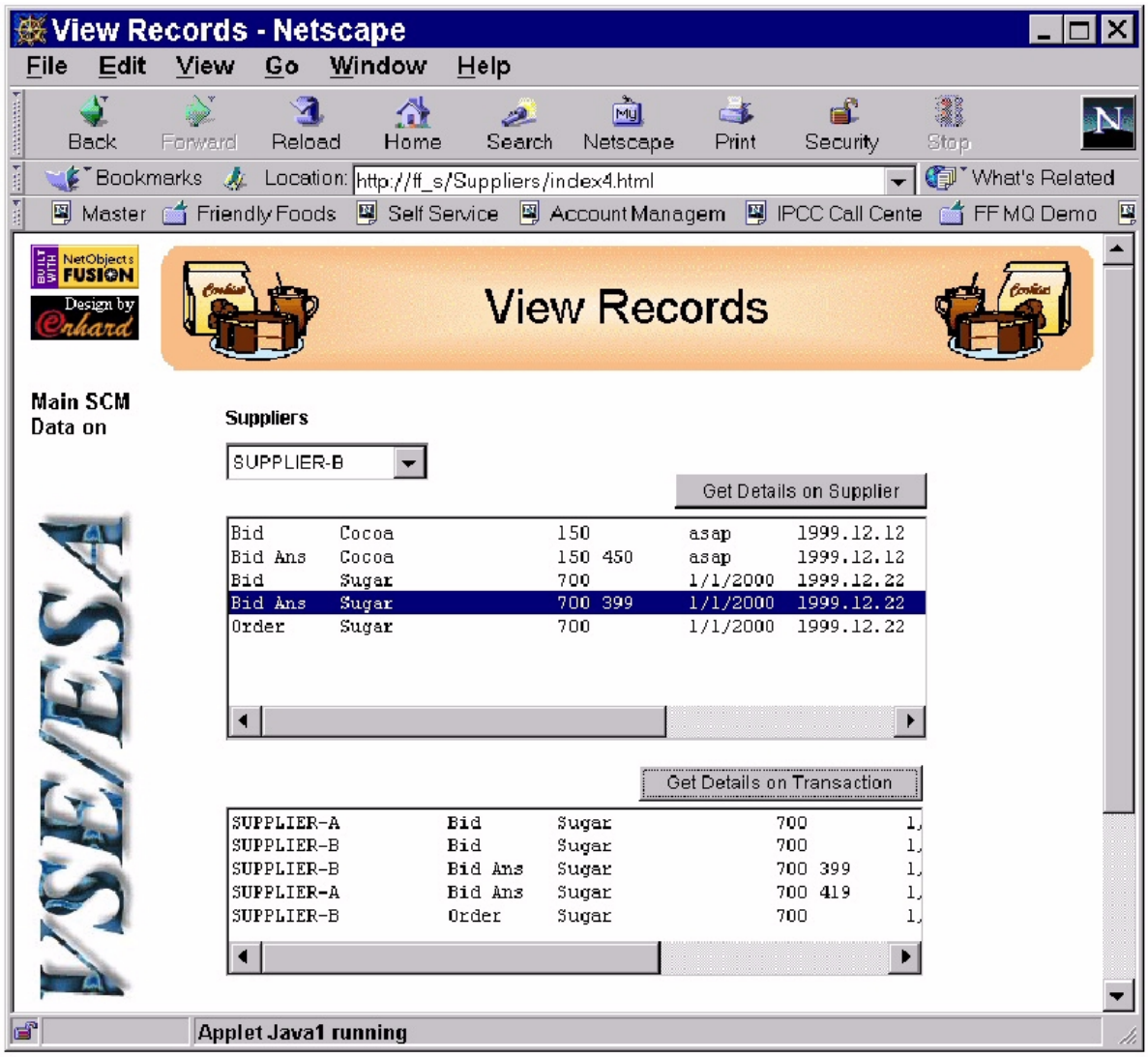


Figure 104. Presentation logic of the browser-based interface to the CICS transaction

Let us have a closer look at this example:

1. When the user starts the application (link to this page), the business logic on VSE is called the first time to get a list of all suppliers stored in the database. This list is added to the pull-down list in the upper left corner of the page.
2. The user selects the supplier he is interested in and clicks **Get Details on Supplier**.
3. The business logic on VSE is called again to get a list of all transactions executed with the selected supplier. This list is added to the upper list panel. This list panel is scrollable horizontally and vertically, so the number of transactions that could be displayed in one single dialog step is not limited as with the 3270 panel used in the presentation logic in the CICS transaction. To avoid changing the business logic (to supply more than the 10 transactions fitting on the 3270 panel), the Java program calls the business logic multiple times until all transactions are added to the list panel.

4. In this example, the user might want to know why, for the last transaction (700 kg of sugar), the order had finally been sent to supplier B. He selects the transaction in the upper list panel and clicks **Get Details on Transaction**.
5. The business logic on VSE is called to get the details on the selected transaction. The result is added to the lower list panel.
6. In our example, the result shows that supplier B was selected because it provided Friendly Foods with the better price.
7. The user might now want to check all transactions with supplier A to see whether there is a trend that supplier A often has the less attractive offering.

This example shows how a new browser-based user interface could provide additional value to the user by protecting the investment made in the business logic on VSE.

Almost all the code of the Java applet for the user interface of this application was developed using the Visual Composition tool in VisualAge for Java. Only the code sections dealing with the CICS ECI interface were added to the respective methods. We will not discuss the development aspects in great detail but focus instead on the integration with applications on a VSE/ESA host.

After adding all the controls to the applet, we started to make the connections to the methods dealing with the CICS ECI. See Figure 105.

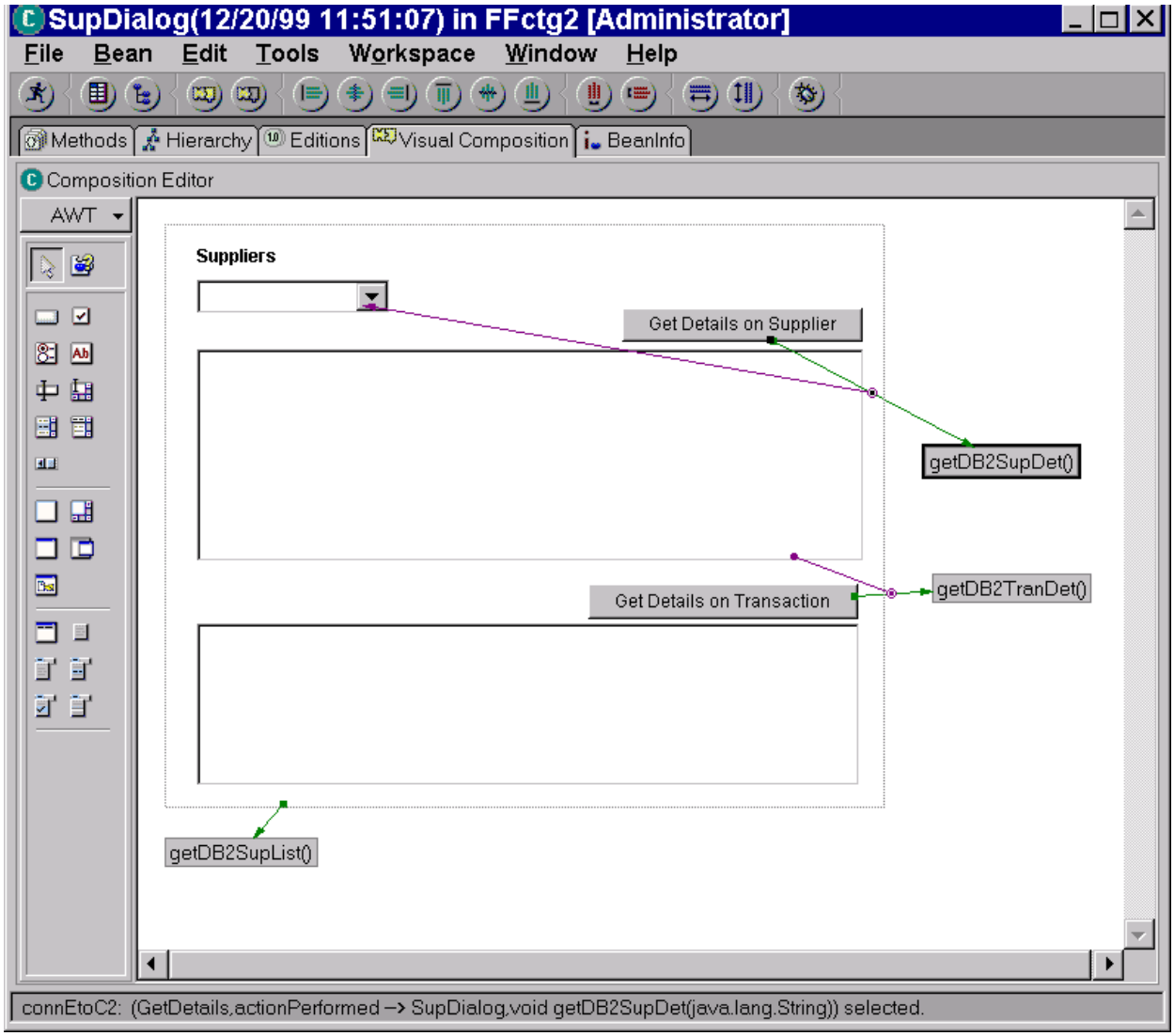


Figure 105. Visual Composition of the user interface applet

1. When the applet is initialized, the getDB2SupList method is triggered.
2. Clicking **Get Details on Supplier** triggers the getDB2SupDet method, taking the selected supplier as a parameter.
3. Clicking **Get Details on Transaction** triggers the getDB2TranDet method, taking the selected transaction as a parameter.

So far we used only Visual Composition in VisualAge. Now we have to add some Java code to each of the three methods calling the CICS ECI. These pieces of code will fill in the information needed to deal with the interface, fill in the information expected by our business logic transaction on VSE (passed via COMMAREA), and receive the results to put in the controls in our applet. We did this without deep skills in Java programming by copying the appropriate code pieces from the example program provided with the CICS Transaction Gateway (the name of the program is TestECI).

The following Java code fragment, used in GetDB2SupList, will give you information on what to consider when using this interface.

```
/**
 * This method was created in VisualAge.
 */
public void getDB2SupList() {
System.out.println( "Start getDB2SupList");
strCommarea = "GETSUPLIST"; // (1)

try
{
/* Create a new JavaGateway to use (2) */
strJGateName = "FF_S";
int   iJGatePort = 2006; // JGate port
jgaConnection = new JavaGateway();
jgaConnection.setURL(strJGateName);
jgaConnection.setPort(iJGatePort);
jgaConnection.setSecurity(strClientSecurity, strServerSecurity);
jgaConnection.open() // (3);

/* Convert commarea from string to array of bytes for CICS. (4) */
byte abCommarea[] = null;
abCommarea = new byte[ iCommareaLength ];
System.arraycopy( strCommarea.getBytes(),
       0,
       abCommarea,
       0,
       Math.min(abCommarea.length, strCommarea.length()));
eciRequest =
new ECIRequest(strServerName, // CICS Server (5)
       strUserId,  // UserId, null for none
       strPassword,// Password, null for none
       null,// Program name
       abCommarea,// Commarea
       ECIRequest.ECI_NO_EXTEND,
       ECIRequest.ECI_LUW_NEW);
eciRequest.Cics_Rc = 0;

/* Set the program name in the eciRequest (6) */
eciRequest.Program = "FFSBL";

/* Flow the request via the JGate to CICS (7) */
jgaConnection.flow(eciRequest);

/* Done handle returend Commarea  */
String xy = new String(eciRequest.Commarea); // (8)
...
}
/* Catch any exceptions  */

catch (IOException e) (9)
{
/* IF jgaConnection is non-null then we have a valid connection
*  attempt to a back-out of any in progress LUW */

if ((jgaConnection != null) && (eciRequest != null))
{
try
```

```
{
/*
* We use the existing eciRequest since it contains
* any relevant Luw_Token      */

eciRequest.Extend_Mode = ECIRequest.ECI_BACKOUT;
jgaConnection.flow(eciRequest);
}
catch (IOException eBack)
...
    }
}
/* Break our connection to the Gateway  */

finally
{
try
{
if (jgaConnection != null)
{
jgaConnection.close();
}
}
catch (IOException eClose)
...
}
}
```

Figure 106. Java code fragment dealing with the CICS ECI

The following points correspond to the numbers shown in parenthesis on the right side of the statements in Figure 106:

1. Move the string GETSUPLIST to the COMMAREA. This is the function code expected by our business logic transaction when it is called to provide the list of suppliers stored in the database.
2. Create a new Java Gateway interface instance and set the required parameters.
3. Open the connection to the Gateway.
4. The COMMAREA has to be converted from string to an array of bytes.
5. Create a new ECIRequest instance and set the required parameters.
6. Fill in the program name of our business logic program on VSE.
7. Execute the request.
8. Receive the returned COMMAREA in a new string; after this there should be the code interpreting the returned COMMAREA and filling the controls in the applet.
9. Handle the exceptions that might occur; if we are in the middle of executing multiple units of work, we have to back out the steps already performed.

6.5 Business integration with suppliers -- summary

In this chapter we demonstrated how Friendly Foods used a new Web-based application to gain business advantage by integrating its business processes with

the business processes of its suppliers. It used modern Web technology to add value to its existing CICS transaction-based SCM application running on VSE/ESA while protecting its investment in this environment.

All products used, including application development tools as well as middleware products, are described by the IBM Application Framework for e-business. This does not only mean that Friendly Foods used the latest development technology, but also that it is perfectly positioned for extending the applications to meet future needs by providing interoperability and portability.

Chapter 7. Options for a 2-tier solution

This chapter describes 2-tier solutions as an alternative to the 3-tier solutions described in the other chapters of this book. As described in 1.1.2, “Two-tier implementations” on page 3, there are two major Solution Developers (formerly known as Independent Software Vendors) offering 2-tier solutions to the VSE customer:

- Web/VSE®-*Host* from IntelliWare Systems, Inc. (`http://www.intelliware.com`)
- IpServer from Data 21, Inc. (`http://www.data21.com`)

We used Web/VSE-*Host* to show how existing 3270-based VSE/ESA applications could be Web-enabled, that is accessed from a Web browser without modifications to the screen definitions or application code.

We used IpServer to show how a 2-tier e-business application can be built which is implemented as a Common Gateway Interface (CGI) program running in the CICS partition on VSE/ESA.

7.1 Web/VSE®-*Host*

Web/VSE®-*Host* from IntelliWare Systems, Inc. is a complete Web-to-*Host* e-business solution that provides full access to any 3270 application from a standard Web browser. By providing dynamic, two-way translation between 3270 and HTML data, Web/VSE-*Host* provides the benefit of instant access to any 3270 application without any changes in the existing applications. Once installed on a VSE/ESA system, users can immediately access their applications from any client that supports a standard Web browser. Because Web/VSE-*Host* automatically transforms the 3270 screen into a more intuitive, browser-based interface, new users with no previous mainframe experience can navigate through the applications in a way that they understand. This enhances and extends existing mainframe investment by increasing the number of potential users of the system, a key e-business objective.

Web/VSE-*Host* includes the WebScreen™ facility that allows users to fully customize the appearance and navigation of the generated screen. With WebScreen, users can easily add or change colors, fonts, add graphics, even change fields into hyperlinks for simple navigation. WebScreen can be used with any 3270 environment including VSE system screens, user applications, and even CICS BMS panels, without any restrictions.

Web/VSE-*Host* also provides a robust, multitasking VSE-based Web server to ensure the high performance and availability that is required for dependable e-business transactions. Web/VSE-*Host* can also be used as a gateway to communicate with any TN3270 server, providing immediate access to applications on other VSE/ESA, VM/ESA, or OS/390 systems. Since Web/VSE-*Host* uses a TN3270 implementation, access is not limited to CICS; for example, even native VTAM applications can be Web-enabled.

Web/VSE-*Host* requires VSE/ESA Version 1.4 or higher and TCP/IP for VSE/ESA. Any workstation that has access to TCP/IP on the VSE/ESA system can use all 3270 applications that are accessible using the TN3270 protocol, including applications running on VM/ESA or OS/390. No other gateway or client

software is required. Web/VSE-*Host* supports and preserves the characteristics of the entire 3270 family of devices including models 2 to 5, and full support for extended data streams.

7.1.1 Installation

Web/VSE-*Host* is supplied either electronically or on diskette. It can be downloaded from the IntelliWare Web site at:

`http://www.vseesa.com` or `http://www.vsebusiness.com`

The software is automatically enabled upon download to run without a license code for up to 10 users for 30 days for evaluation purposes. Additional time and users are obtained through a license code from IntelliWare.

Installation is very simple and can be accomplished in less than one hour. To install the product you will need to perform the following steps:

- Unzip the supplied files to a hard disk on a PC workstation.
- Transfer the installation job to the VSE/POWER reader queue via TCP/IP using an FTP client to send the job (which is the simplest method), or via SNA LU2 File Transfer if an SNA 3270 emulator, for example, IBM Personal Communications, is in use. The job is in binary format and you must set the file transfer option to binary prior to the transfer. The job is transferred to VSE/POWER with a disposition of hold (DISP=H).
- Release the VSE/POWER job. You will be prompted for a SETPARM statement to define the VSE library and sublibrary into which the software will be installed.

A checklist of steps to be executed is supplied with the documentation of the product that is also available for download from the IntelliWare Web site. The User's Guide supplied with the product contains complete information for quick setup and initialization.

Web/VSE-*Host* runs in its own batch partition, static or dynamic, on VSE/ESA and creates a data space where its internal control blocks, buffers, and data are stored. Note that the VSE/ESA system should be configured to support the creation of a 2 MB data space for use by Web/VSE-*Host*.

7.1.2 Customization

Web/VSE-*Host* loads a configuration file at system startup to establish various parameters for use. Many of these options can be modified dynamically through simple console commands to load new profiles and other settings. This allows for easy updates and changes without restarting the system and affecting production users.

Sample configuration files containing default user options and settings are shipped as part of Web/VSE-*Host*. The configuration files can be tailored according to each user's system requirements.

Within the sample configuration, one parameter must be changed to reflect the TCP/IP port number on which the Web/VSE-*Host* Web server “listens” for connections. The configuration file is also where the license key for the product is included after the initial 30-day installation period.

The following sample configuration files are provided:

WEBHOST.CONFIG00 contains a set of basic configuration values, without any WebScreen customizations.

WEBHOST.CONFIGZZ contains a full set of WebScreen customizations designed to Web-enable the VSE/ESA Interactive User Interface (IUI).

Figure 107 shows the content of a sample configuration file.

```
::GLOBAL:: Options Section Start
*
Body text="#0000FF" bgcolor="#C0C0C0"
Initial_Receive_Wait 1500
Max_Receive_Wait   75
Method  Post
Suppress_Unprot    _
*
DataSpace_Size  2
Max_Tasks          5
*
Map_Color   Blue      0000FF
Map_Color   Red       FF0000
Map_Color   Pink      FF00FF
Map_Color   Green   008000
Map_Color   Turq     40E0D0
Map_Color   Yellow  FFFF00
Map_Color   White   FFFFFF
High_Intensity          White
*
Protect_Long_Unprot_Field  Yes
Long_Unprot_Field               158
*
HTTP_Listen_Port   80
HTTP_Cache          On
*
Disc_Page         WEBHOST.HTML
Error_Template  WEBHOST.HTML
*
Disc_Label         Disc
Refresh_Label    Refresh
*
Log Connections On
Log HTTP           On
*
Inactive_Limit  540000   270000
```

Figure 107. Web/VSE-Host - sample configuration file

7.1.3 Automatically generated 3270 Web pages

Immediately after installation, Web/VSE-*Host* is ready to be used to dynamically convert 3270 screens into Web pages. In this mode, no customization or other changes are required. Web/VSE-*Host* uses the system options contained in the configuration file to create each Web page. The user is not required to code any HTML or screen definitions. In this mode the benefits of access are immediately

available to any user who is granted access via Internet, intranet, or extranet connectivity.

To enhance the appearance of the Web pages however, the configuration file options can be used to define the Web page background color, background image, even the text color. In addition, the colors used on the originating 3270 screen may be mapped to different colors on the Web page. For example, if a user chooses a white background color for the Web page, the 3270 white text color should be mapped to another color to ensure that it will be visible on the Web page.

Figure 108 shows a sample Web page that was created by Web/VSE-*Host* using the distributed configuration file WEBHOST.CONFIG00.

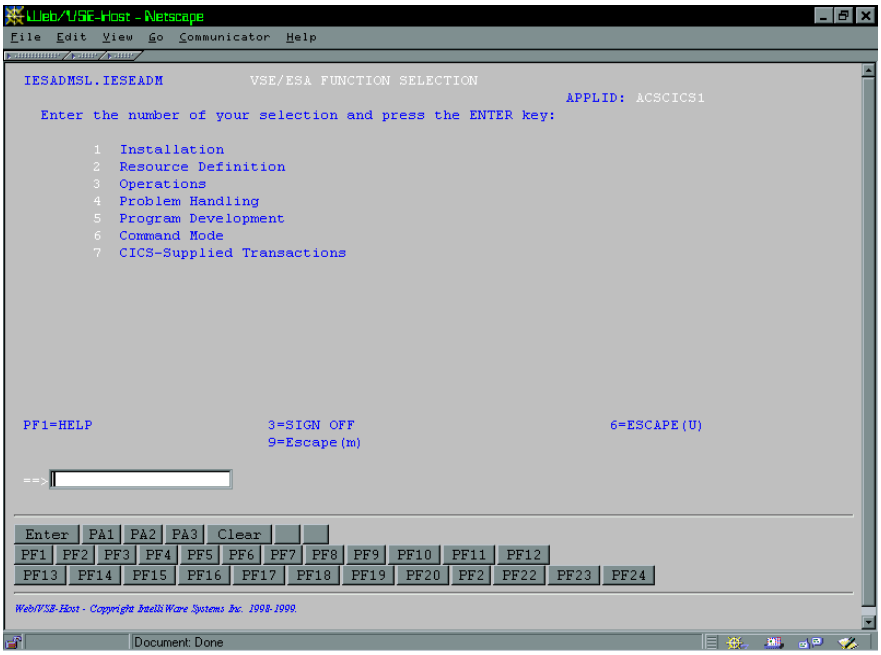


Figure 108. Web/VSE-Host - sample Web page

7.1.4 Creating Web pages with HTML templates

The next step you may choose in Web-enabling your 3270 applications is to create your own Web page design that Web/VSE-*Host* can use as a template. Using this facility, a single HTML template can be defined for use with all 3270 screens to add a common look and feel that is customized for your organization. The template file includes HTML to display a Web page the way you want it to appear, including your desired colors and graphics. Web/VSE-*Host* merges the template definitions when the Web page is dynamically created from your 3270 data, resulting in a custom look and feel.

The template file includes all of the HTML required to display the Web page, and also contains an HTML comment statement, called a “trigger statement”, that tells Web/VSE-*Host* where to insert the 3270 data.

The template file is stored in a VSE library and a statement is added to the configuration file to tell Web/VSE-*Host* the name of the template file to be used. For example:

```
HTML_Template  myhtml.template
```

This configuration file statement will cause Web/VSE-*Host* to use the library member myhtml.template to create each 3270 Web screen. Figure 109 shows a sample of a Web/VSE-*Host* template file.

```
<html>
<! Web/VSE-Host-Host Sample HTML Template  IntelliWare Systems Inc. />
<head><title>Web/VSE-Host-Host Sample HTML Template</title>
<style>input {font-family:courier}</style>
</head>
<body bgcolor="silver" text="blue" onload="InitCursor()">
<img src=vsegif1.gif height=25 width=175 align=middle>
<font color=black><h7><i>Web-enabled by Web/VSE-Host-Host</i></h7></font>
          
<img src=openvse.jpg align=top>
<br clear=right>
<!-- WEB/VSE-HOST-Host SCREEN -->   <!-- The 3270 screen will be put here
-->
<hr><font size="-1" color="red">
Powered by VSE/ESA!
</font></body></html>
```

Figure 109. Web/VSE-Host - sample template file

The bold HTML comment line in Figure 109 is the Web/VSE-*Host* template trigger statement that defines where the 3270 data is inserted. Given the above template example, the VSE/ESA IUI function selection screen processed by Web/VSE-*Host* would create a Web page that looks like Figure 110 on page 130.

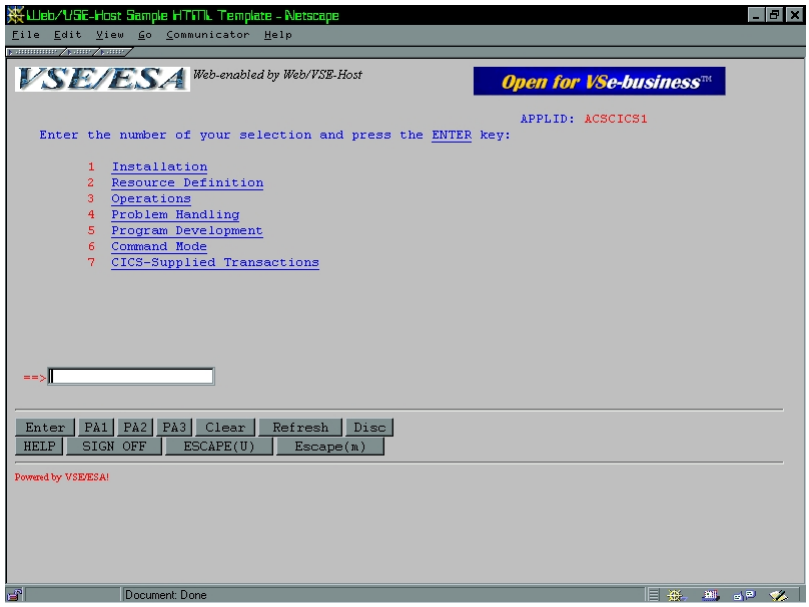


Figure 110. Web/VSE-Host - sample customized Web screen

7.1.5 Identifying 3270 screens and customizing Web pages

Web/VSE-*Host* provides immediate access to any 3270 application as its primary benefit. The second benefit of the product is the ability to customize the look and feel of the Web-based screens. Web/VSE-*Host*'s WebScreen facility offers the ability to provide a much more intuitive and customized appearance, thereby taking host access through Web pages to the next level with a minimum of effort, and even improving functionality. Without any programming or learning any new language facility, the Web/VSE-*Host* WebScreen feature can be used to identify 3270 screens and apply additional customization options to the identified screens.

With WebScreen, users can customize individual screens, groups of screens, and even an entire application. When merged with the results of a Web/VSE-*Host* template definition, a host-based application can easily be transformed into the look and feel of any other Web-based application.

The WebScreen facility provides support for:

- Hypertext links for menu system navigation
- Hiding areas of the 3270 screen from the Web page
- Removing entire screen rows from the Web page
- Use of specific HTML template files on a screen-by-screen basis
- Cursor location detection on protected screen fields
- Screen unique color mapping
- Screen display suppression
- Screen unique virtual keyboard

The screen definitions are supplied using free form control statements in ::SCREEN:: sections of the Web/VSE-*Host* configuration file. One section is coded for each 3270 screen that is to be uniquely processed, but one definition can apply to multiple screens. The options specified in the ::SCREEN:: definition are applied to the creation of the Web page for the identified screen.

The 3270 screen is identified by the presence of one or more character strings on the screen at specified locations. The first screen definition that is found to match the 3270 screen data is used to create the Web page. An entire application or even a subset can be handled with one definition if a common character string can be used to “trigger” the WebScreen. The sample WebScreen definition shown in Figure 111 identifies the IUI Function Selection menu screen and provides a number of customizations for the Web page.

```
*
::SCREEN:: Screen Definition for VSE Function Selection Menu
*
  Screen_Name  VSE_Primary_Menu
  Screen_DATA  Row 1  Column 2  String IESADMSL
  Screen_DATA  Row 1  Column 11 String IESEADM
*
  HTML_Template  vsepri.template
*
  High_Intensity     red
*
  Key_Disable pf1  pf2  pf3  pf4  pf5  pf6  pf7  pf8  pf9  pf10 pf11 pf12
  Key_Disable pf13 pf14 pf15 pf16 pf17 pf18 pf19 pf20 pf21 pf22 pf23 pf24
*
  Cut_Region   1    1
  Cut_Region  13   16
  Cut_Region  21   22
*
  Input_Field 24 6
  Link  3 53  3 57 Enter Data   31
  Link  5 13  5 24 Enter Screen  5 10  1
  Link  6 13  6 31 Enter Screen  6 10  1
  Link  7 13  7 22 Enter Screen  7 10  1
  Link  8 13  8 28 Enter Screen  8 10  1
  Link  9 13  9 31 Enter Screen  9 10  1
  Link 10 13 10 24 Enter Screen 10 10  1
  Link 11 13 11 38 Enter Screen 11 10  1
*
```

Figure 111. Web/VSE-Host - sample WebScreen definition

This WebScreen definition, along with the HTML template stored in the library member vsepri.template, is used to create the Web page shown in Figure 112 on page 132 from the IUI VSE Function Selection 3270 screen.

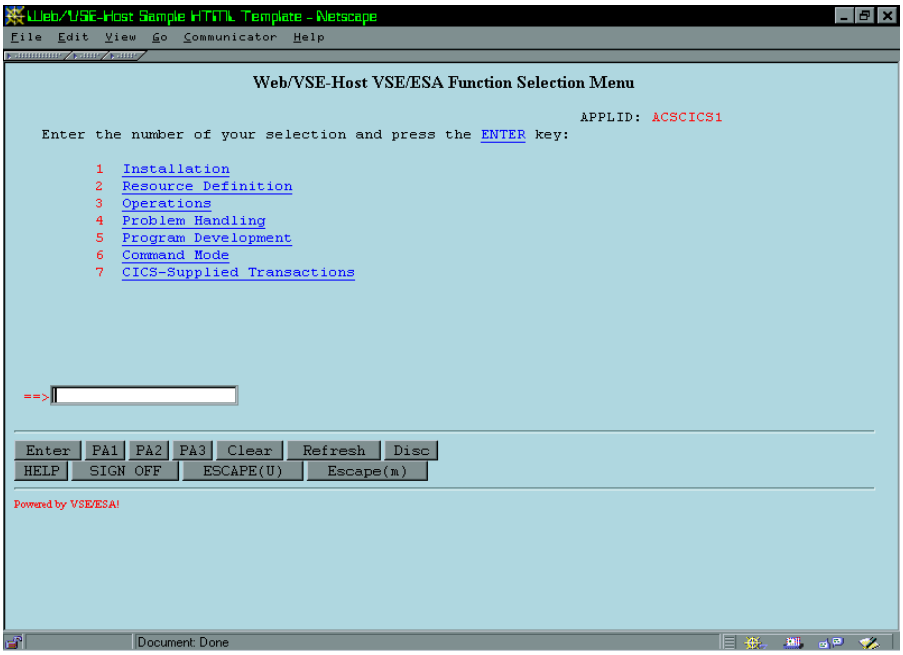


Figure 112. Web/VSE-Host - sample Web page

This VSE/ESA screen transformed to a Web/VSE-*Host* Web page exploits the product's features. By combining the Template facility with the WebScreen facility, several changes can be noted on the resulting screen:

- The HTML template for this specific screen provides the selected colors.
- Intuitive hypertext links may be clicked to navigate to the desired function.
- The original screen heading is removed and a new heading provided.
- Highlighted text is displayed in red.
- Unused lines are removed to shorten the Web page and eliminate scrolling.
- The PF key description lines are removed since the keyboard buttons contain the description of their function.
- Dynamic keyboard processing has automatically enabled the active PF keys using the key description that was on the screen.

The WebScreen and Template facility can be used to completely transform a user-based application, turning PF-Keys into graphical buttons, adding logos and other attributes that make an application inviting to even non-mainframe users. Figure 113 and Figure 114 on page 133 illustrate the dramatic difference a small amount of customization can make.

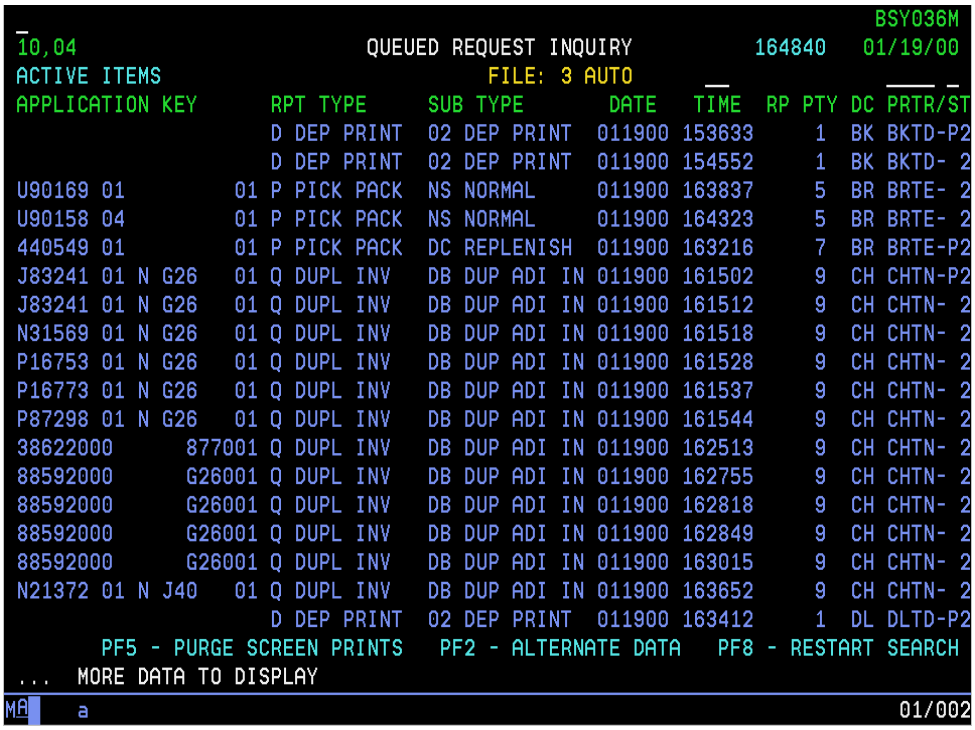


Figure 113. Web/VSE-Host - native 3270 screen

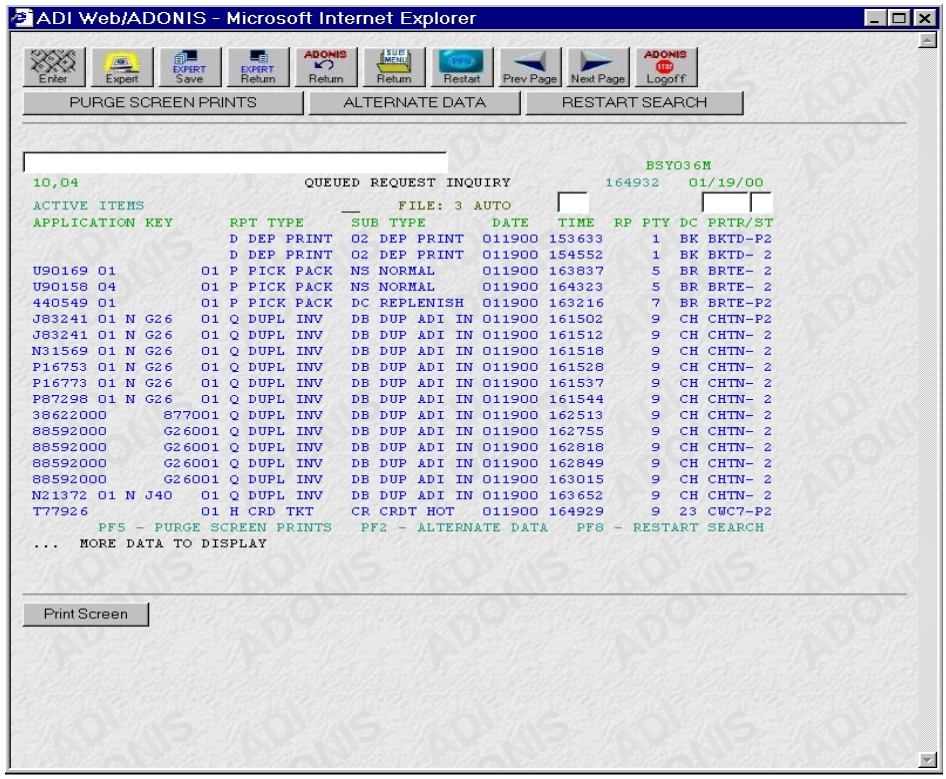


Figure 114. Web/VSE-Host - 3270 data in a Web browser

7.1.6 Security considerations with Web/VSE-*Host*

At the current time, Transport Layer Security (TLS) and Secured Socket Layer (SSL) support are not available in the TCP/IP for VSE/ESA product, so it is not possible to use this level of security in a 2-tier Web-enabled solution. Many organizations, however, do not require this level of security, particularly if their Web/VSE-*Host* implementation is restricted to either an intranet or extranet implementation. For those that do require this level of security, several methods can be used to ensure secure communications in a 3-tier solution with Web/VSE-*Host*.

One option is to use a Virtual Private Network (VPN) connection. These connections use the IPSec, L2TP or PPTP protocols to provide encrypted data transmissions over the Internet. VPN solutions are relatively simple to implement in a 3-tier solution. They are easier to set up than SSL solutions because they do not use a public key infrastructure, and therefore the service of a Public Key Authority is not required.

Another 3-tier solution is the use of a proxy server that is configured as a “reverse proxy”. This is a solution where the browser connects to a proxy server instead of directly to a Web server. The browser and proxy server communicate using a SSL connection, so the data is encrypted while traveling over the Internet. The proxy server, which is behind physically secure walls, relays the connection to Web/VSE-*Host* using a non-SSL connection. When the Web page is returned to the proxy server, it transmits the page to the browser over the SSL connection, therefore providing completely secure access.

In addition to configuring a reverse proxy server, other software vendors are providing solutions that run on a middle-tier machine to provide the same relay facility. They handle incoming SSL connections, relay the request to the mainframe, and send the result back to the browser.

7.1.7 Web/VSE-*Host* and Friendly Foods

In the Friendly Foods scenario, Web/VSE-*Host* could have been used instead of SecureWay Host On-Demand to provide integration of service providers. Web/VSE-*Host* would allow call center personnel to directly start a 3270 screen within their Web browser, without the need to install additional software on the client Web browser workstation. It would also eliminate time-consuming Java-applet downloads that slow down user access to applications.

Web/VSE-*Host* could fully replace the Friendly Foods' call center implementation.

7.2 IpServer

IpServer is a VSE Web server implementation running in the CICS partition on VSE/ESA. This implementation enables you to write CGI programs which run under control of CICS/VSE, and with this you can benefit from CICS as a transaction monitor and develop e-business applications. In addition, you can Web-enable your existing applications by making minor changes to them.

The CGI programs are called by IpServer when the appropriate statement is coded in an HTML page. IpServer requires VSE/ESA Version 1.4 or later, CICS/VSE 2.1, and uses TCP/IP for VSE/ESA to communicate with a user’s Web browser.

7.2.1 Installation and customization

IpServer is supplied in electronic form and you can directly download it from Data 21's Web page at:

`http://www.data21.com`

The installation process consists of running a series of steps to be executed, as documented in a checklist supplied with the documentation of the product. Basically these are:

- Defining the VSE library and sublibraries you want to install the product in
- Running a self-extracting zip file to create a number of binary files
- Using an FTP client (either line-based or GUI-based) to transfer these files to the VSE sublibraries
- Running a link-edit job to create various VSE phases
- Updating CICS definitions by running a supplied RDO batch job
- Compiling the supplied sample CGI programs

Refer to the readvse.txt file supplied with the product for further information.

It is recommended that you create at least four sublibraries. These should be used to contain the following members:

1. lib.IPSERVER contains the Data 21-supplied code and sample CGI programs.
2. lib.IPARMS contains the configuration file INITIAL.TXT.
3. lib.WWW contains various sample GIF files and HTML files.
4. lib.D21I contains Data 21 supplied routines and various HTML files.

It is also recommended that you create a separate sublibrary with user-written HTML pages and CGI programs.

A configuration file containing user options and settings for various parameters must be tailored according to the requirements of the customer site. This includes a license key for the product.

7.2.2 Running the sample CGI programs

Four sample CGI programs and the necessary HTML files are provided with IpServer to show the non-experienced user how a CICS CGI program can be executed from a Web browser, displaying various sample outputs on the browser workstation. You need to go through the following steps to get the supplied samples to work:

1. Compile the source code supplied in lib.IPSERVER. The sample program names are SAMPCGI1, SAMPCGI2, SAMPCGI3, and SAMPCGI4. Catalog the compiled output into a sublibrary in your CICS startup JCL LIBDEF chain.
2. Start IpServer from a CICS screen using the CICS transaction ID IPON.
3. Point your Web browser to the IP address of your VSE/ESA host, making sure that you add the port address to be used for the Web server. (We used port number 8080 in our example - this must be defined in the WWWPORT parameter in the INITIAL.TXT member of lib.IPARMS).

4. You will now see the default HTML page of IpServer. Click **IPSERVER CGI SAMPLES**.
5. The window shown in Figure 115 appears; from here you can select the different sample CGI programs.

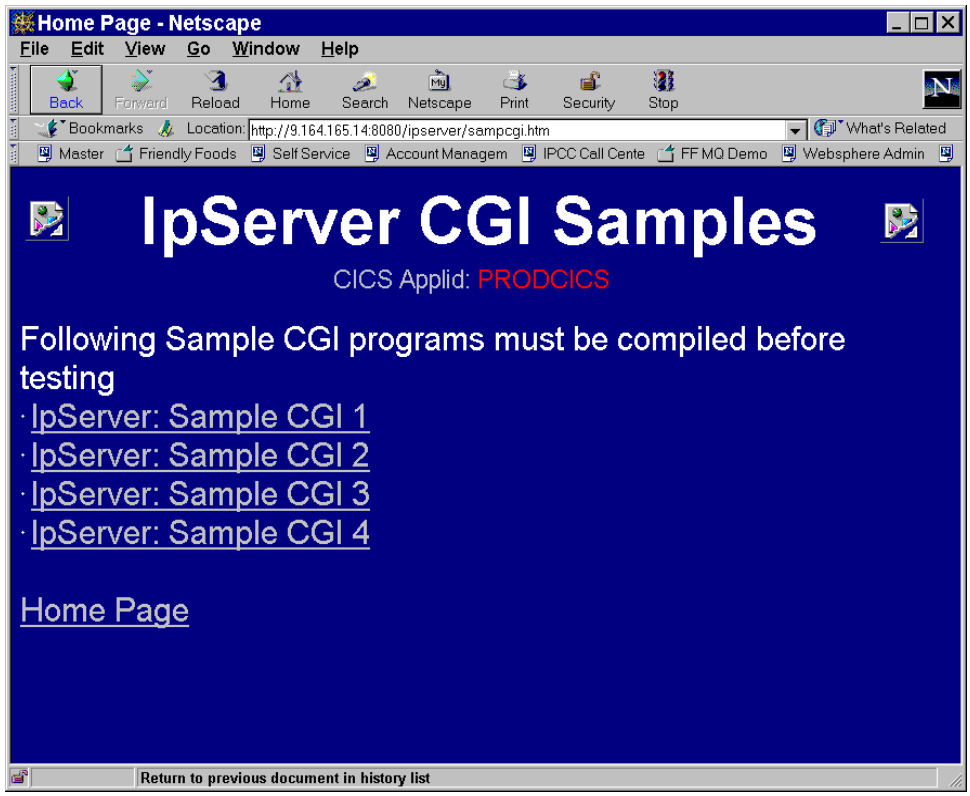


Figure 115. Output from SAMPCGI

Figure 116 on page 137 shows the HTML script of this entry page.

```
<html>
<head>
<title>Home Page</title>
<STYLE TYPE="text/css">
        BODY {color:white; BACKGROUND: darkblue}
       DIV {position:relative; width:100%; font-size:16pt;
           height:40px}
        H1 {text-align:center; font-size:18pt; }
     </STYLE>
</head>
<body text="lightgreen" link="lightgreen" vlink="lightgreen" alink="lime"
topmargin="25">
<table align="center"><tr>
<TD><IMG SRC="/ipserver.gif"> </TD>
<TD WIDTH="55"</TD>
<TD><font size=7 color="white">
<b>IpServer CGI Samples</b></font></TD>
<TD WIDTH="55"</TD>
<TD><IMG SRC="/power390.gif"></TD>
</tr></table>
<center>
<font size=4 color="silver">CICS Applid: </font>
<font size=4 color="red"><!--#echo var="CICS_APPLID"--></font>
</center>
<br>
<font size=5>
Following Sample CGI programs must be compiled before testing
<li><a href="/SAMPCGI1">IpServer: Sample CGI 1</a></li>
<li><a href="/ipserver/sampcgi2.htm">IpServer: Sample CGI 2</a></li>
<li><a href="/ipserver/sampcgi3.htm">IpServer: Sample CGI 3</a></li>
<li><a href="/ipserver/sampcgi4.htm">IpServer: Sample CGI 4</a></li>
<p><a href="/default.htm">Home Page</a></p>
</body>
</html>
```

Figure 116. HTML script used to produce the SAMPCGI output

Figure 117 on page 138 shows the output from SAMPCG14.

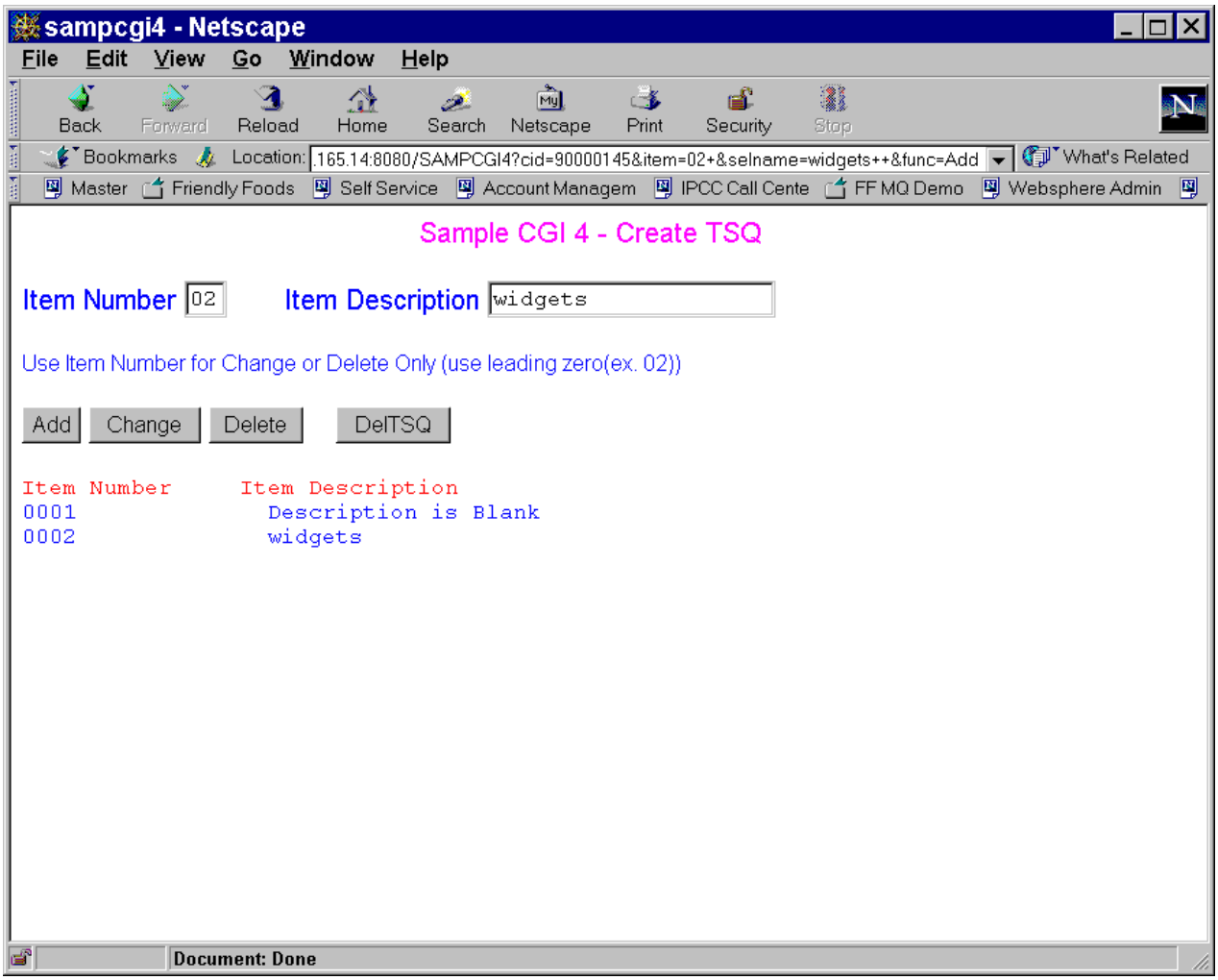


Figure 117. Output from SAMPCGI4

Figure 118 on page 139 shows the HTML script used to drive the Web browser page.

```
<html>
<head>
<title>sampcgi4</title>
</head>

<body bgcolor="#FFFFFF" text="#0000FF"
onLoad="document.form1.selname.focus()">

<p align="center"><font color="#FF0080">
Sample CGI 4 - Create TSQ
</font></p>

<form action="/SAMPCGI4" method="GET" name="form1">
<input type="hidden" name="cid" value="<!--#echo
var="CGI_BUFFER2(CID)"-->">
<p>
Item Number
<input name="item" size="2" value=" ">
     
Item Description
<input name="selname" size="20" value=" ">
</p>
<font size=2>
Use Item Number for Change or Delete Only (use leading zero(ex. 02))
</font>
<p>
<input type="submit" name="func" value="Add">
<input type="submit" name="func" value="Change">
<input type="submit" name="func" value="Delete">
  
<input type="submit" name="func" value="DelTSQ">
</p>
</form>
<pre>
<font color="#FF0000" size=3>Item Number    Item Description</font>
<!--#echo var="CGI_TSQ2"-->
</pre>
<p> </p>
</body>
</html>
```

Figure 118. HTML script used to produce the SAMPCGI4 output

A good way to start using CGI programs and the corresponding HTML Web pages might be to follow through the HTML scripts and the source of the sample CGI programs as supplied in lib.IPSERVER. Once you are familiar with the concepts you should be able to work through the logic of how a CGI program interacts with the IpServer code itself.

In 7.3.3, “Using dynamic pages with IpServer” on page 141 we will create a CGI program for Friendly Foods in the 2-tier environment. This program will allow a user at a Web browser to inquire on data from a VSE/ESA host running IpServer.

7.3 Use modern tools to build business Web sites hosted on VSE/ESA

With IpServer from Data 21 you can host your business Web site directly on VSE. The fact that IpServer runs in the CICS partition provides you with an easy way to access your business data which are maintained by your existing VSE/CICS transactions.

Modern tools to build your business Web site can be used as long as you keep in mind that there are some limitations caused by VSE/ESA. An example of a modern Site Management tool is NetObjects Fusion 4.0 which is also used in other parts of this redbook.

The following restriction in publishing onto and serving from a VSE/ESA-based Web server exists:

- All of the elements comprising your pages should be addressed by a file name of <= 8 characters.

7.3.1 Setup to work with NetObjects Fusion

No special initial setup in NetObjects Fusion is required. As soon as we want to publish our created Web site, we have to make some definitions to indicate that we are publishing to VSE/ESA.

The TCP/IP startup (in our case member IPINIT00.L) on VSE/ESA has to be changed so that the default FTP daemon acts in UNIX mode (receiving commands in UNIX syntax). This is achieved with the following command:

```
DEFINE FTPD,ID=FTP,PORT=21,COUNT=4,UNIX=YES
```

The values for the parameters ID and COUNT are your choice. We used port 21 for this FTP connection. If you have problems with permanently running your FTP daemon in UNIX mode, you should stop and redefine the daemon just for the publishing stage with NetObjects Fusion.

7.3.2 Designing Web pages with NetObjects Fusion 4.0

In the design phase of your Web site you have to keep the limitation of the VSE/ESA system in mind. However, this still leaves you with the many advantages NetObjects Fusion offers you for creating appealing business Web sites. We cover these advantages in 4.3.3, “Integrating the pages with NetObjects Fusion” on page 53, where NetObjects Fusion is discussed in more detail.

The consequences of the VSE/ESA restriction is:

You cannot use most of the Java-based built-in NetObjects Fusion add-ons, such as the Ticker Tape Java applet provided with NetObjects Fusion. Ticker Tape requires you to publish a Java class that is identified by a file name longer than eight characters.

In our example we used the same site we created previously to be published on the WebSphere application server (as described in Chapter 3., “Consumer marketing” on page 15) for publishing now on VSE/ESA. This demonstrates the rich set of NetObjects Fusion functions you can also exploit when working with IpServer on VSE.

If we wanted to set up a site with only static pages, we would now be ready to go into the publishing stage--but since we instead want to populate our Friendly Foods pages with business data stored in the DB2 database on VSE/ESA, we first need to examine the CGI concept of working with IpServer and see how these CGI calls could be integrated into HTML pages published by Fusion.

7.3.3 Using dynamic pages with IpServer

Figure 119 illustrates the concept of calling CGI programs in IpServer and the process flow while a CGI program is called.

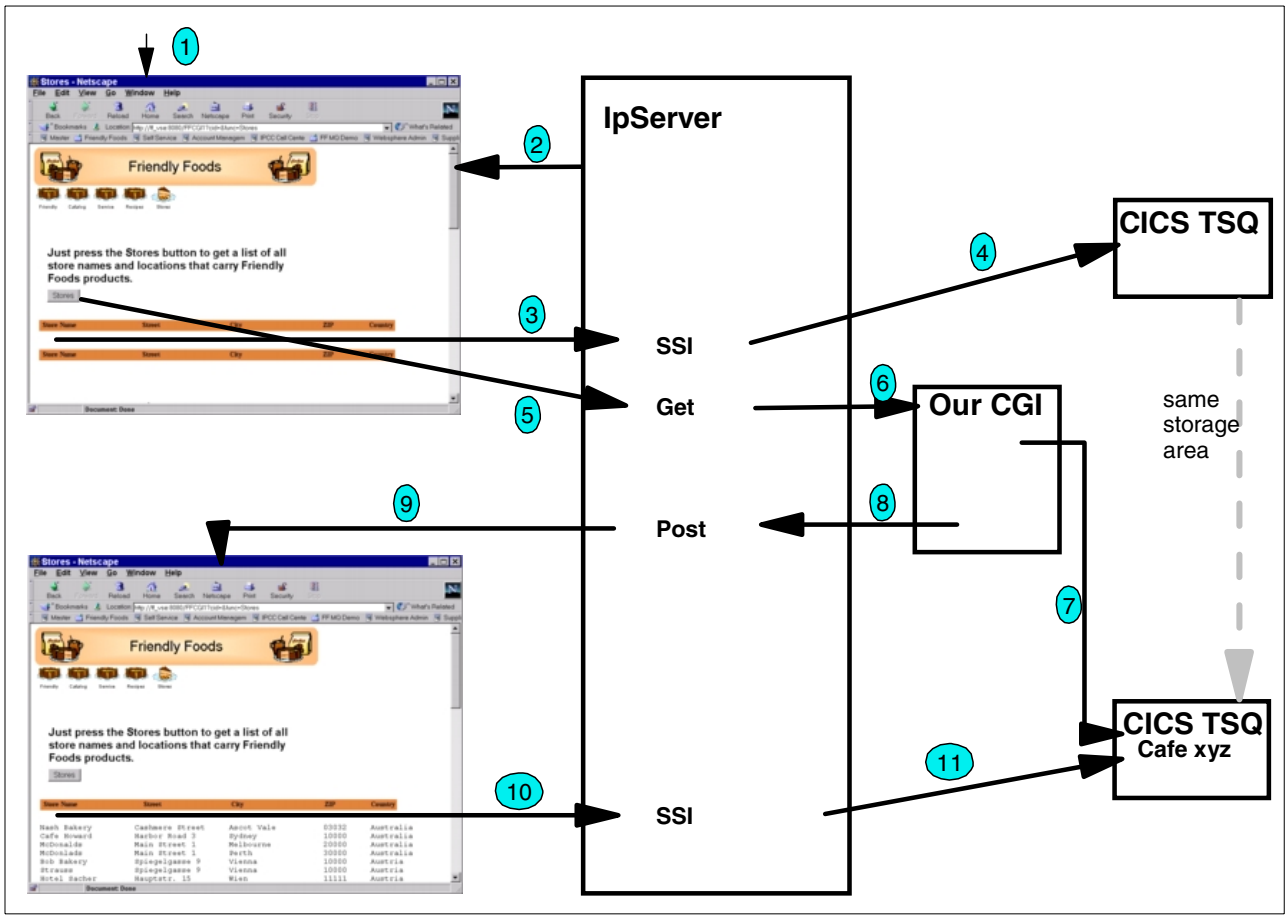


Figure 119. Overview of the CGI process in IpServer

During this process:

1. An HTML page containing a form to call a CGI program is requested.
2. IpServer delivers the page to the browser.
3. The page contains a type of a Server Side Include (SSI) statement, forcing IpServer to look for an additional section to be delivered with the page.
4. IpServer checks the CICS Temporary Storage Queue (TSQ) specified in the SSI. Since this is empty, there is nothing to include in the page.
5. The user presses the button, forcing the browser to submit the form to the Web server. The form contains a get request for our CGI program.
6. IpServer calls our CGI program.

7. Our CGI program reads the records out of the DB2 database and stores the result lines in the TSQ.
8. Our CGI returns control back to IpServer with the URL of the page to be called (in our case this is the same page).
9. IpServer submits this page to the browser.
10. IpServer again detects an SSI statement asking for an additional section to be delivered with the page.
11. IpServer checks the CICS TSQ and this time finds the result lines created by our CGI program and sends these to the browser.

Basically the CGI interface provided by IpServer is an easy-to-handle interface. There are only a few things we have to consider when creating the COBOL program to read the list of Friendly Foods stores out of the DB2 database.

In the following paragraph we discuss the sections in the COBOL program that are related to this interface. For a better understanding of how the pieces come together and how the real program logic gets the records out of DB2, refer to Appendix B, “Sample programs” on page 163. This appendix contains the entire COBOL program FFCGI1, which we derived from one of the sample programs provided by Data 21, Inc.

In the COBOL LINKAGE SECTION we have to link to the CICS COMMAREA which is used to pass the parameters into the program and back to IpServer. The copy book provided by Data 21 defines the structure of the COMMAREA. There is also a second area (BUFFER-DATA-RETURN) for passing information back to IpServer. This allows you to change parameters in the received input parameters (however, this is not done in this example).

```
LINKAGE SECTION.

01  DFHCOMMAREA.
   COPY D21ICGIC.
* LENGTH MUST BE = PARSE2-TEMP-LENGTH
   01  BUFFER-DATA-RETURN       PIC X(512).
```

In the WORKING-STORAGE SECTION there is also a copy book (D21IPARC) included, which is provided by Data 21. This defines the data structure of the parameters passed in from the form. We also have the return page defined, which will get posted after the CGI is completed.

```
      * RETURN-PAGE MUST BE IN UPPER CASE
           05  RETURN-PAGE  PIC X(44) VALUE '/FF/STORES.HTML'.
           05  PARSE2-LENGTH        PIC S9(4)   VALUE +0  COMP.
      * PARSE2-TEMP-LENGTH VALUE MUST BE => PARSE2(SEE INSIDE D21IPARC)
           05  PARSE2-TEMP-LENGTH   PIC S9(8)   VALUE +512 COMP.
.
.
       COPY D21IPARC.
.
.
      * DISPLAY SAME PAGE WITH DATA
           MOVE RETURN-PAGE         TO CGI-RETURN-PAGE-NAME.
```

Figure 120. Definition of the return page in the sample CGI used with IpServer

In the initialization phase of the program, we move the name of the return page in the field expected by IpServer. Then we call a Data 21-provided program (D21IPARS), which parses the string of input parameters into three tokens (PARSE1 to PARSE3) that are defined in the copy book D21IPARC.

Figure 121 on page 144 shows this copy book.

```
*-----------------------------------------------------------------*
* Max field size is 255 and PARSE-DATA-WALEN/PARSE-DATA-WORKAREA
*     must NOT be smaller than largest field.
* This limit may be removed in a future release.
*-----------------------------------------------------------*
*
* To call parsing sub-routing:
* CALL D21IPARS USING DFHCOMMAREA, PARSE1, PARSE2, PARSE3.
*
* Calling prog must be running with the same AMODE= as 'D21I00'
*
*--------- IPSERVER (D21IPARC) COPY BOOK ----------------------*
 01  PARSE.
     05  PARSE1.
         10  PARSE-FIELD-ID-LENGTH PIC S9(4)  VALUE +16  COMP.
         10  PARSE-RETCD           PIC X(1)   VALUE '0'.
             88  PARSE-INVALID-STRUCTURE      VALUE '1'.
         10  PARSE-TR-UPPERCASE    PIC X(1)   VALUE 'Y'.
             88  PARSE-TR-UPPERCASE-YES       VALUE 'Y'.
             88  PARSE-TR-UPPERCASE-NO        VALUE 'N'.
         10  FILLER                PIC X(16)  VALUE ' '.
         10  PARSE-DATA-WALEN      PIC S9(4)  VALUE +255 COMP.
         10  PARSE-DATA-WORKAREA   PIC X(255) VALUE LOW-VALUES.
*-----------------------------------------------------------*
*
* PLACE YOUR FIELDS HERE **** THESE ARE USED FOR SAMPCGI1-?
*
     05  PARSE2.
         10  PARSE-CID-ID          PIC X(16)  VALUE 'CID'.
         10  PARSE-CID-LEN         PIC S9(4)  VALUE +8  COMP.
         10  FILLER                PIC X(4)   VALUE ' '.
         10  PARSE-CID             PIC X(8)   VALUE ' '.
*
         10  PARSE-FUNCTION-ID     PIC X(16)  VALUE 'FUNC'.
         10  PARSE-FUNCTION-LEN    PIC S9(4)  VALUE +8  COMP.
         10  FILLER                PIC X(4)   VALUE ' '.
         10  PARSE-FUNCTION        PIC X(8)   VALUE ' '.
*
         10  PARSE-ITEM-ID         PIC X(16)  VALUE 'ITEM'.
         10  PARSE-ITEM-LEN        PIC S9(4)  VALUE +2  COMP.
         10  FILLER                PIC X(4)   VALUE ' '.
         10  PARSE-ITEM            PIC X(2)   VALUE ' '.
*
         10  PARSE-SELNUM-ID       PIC X(16)  VALUE 'SELNUM'.
         10  PARSE-SELNUM-LEN      PIC S9(4)  VALUE +9  COMP.
         10  FILLER                PIC X(4)   VALUE ' '.
         10  PARSE-SELNUM          PIC X(9)   VALUE ' '.
*
         10  PARSE-SELNAME-ID      PIC X(16)  VALUE 'SELNAME'.
         10  PARSE-SELNAME-LEN     PIC S9(4)  VALUE +20 COMP.
         10  FILLER                PIC X(4)   VALUE ' '.
         10  PARSE-SELNAME         PIC X(20)  VALUE ' '.
*-----------------------------------------------------------*
* MUST ALWAYS BE LAST !!!!!!
     05  PARSE3.
         10  FILLER                PIC X(1)   VALUE LOW-VALUES.
*-----------------------------------------------------------*
```

Figure 121. Data structure in copy book D21IPARC

The PARSE1 section is used to define some input options for the parsing process. We turned uppercase translation off as follows:

```
SET PARSE-TR-UPPERCASE-NO  TO TRUE
```

PARSE2 defines the names of the input parameters the parsing function is looking for in the input stream. In our example we did not change this copy book; we worked with the same names as the Data 21-provided CGI example because func was a suitable parameter name for the function we implemented. In your implementation, you can change these names and add additional parameters.

In our example we expect the input parameter func with the value Stores. We evaluate this in the following statement:

```
EVALUATE PARSE-FUNCTION
    WHEN 'Stores' PERFORM 3000-GET THRU 3100-EXIT
END-EVALUATE
```

Figure 122. Checking the input parameter in sample CGI used with IpServer

We also take the required steps to read the records in the DB2 database. Within this piece of business logic, we read record by record out of the DB2 table and put these records on a CICS temporary storage queue.

```
EXEC CICS WRITEQ TS QUEUE(CID)
          FROM(DB2RECORD) LENGTH(88) END-EXEC.
```

Figure 123. Writing on CICS TSQ in sample CGI used with IpServer

Then we set up the return parameters, move all records into the final CICS temporary storage queue, and return control back to IpServer as shown in Figure 124 on page 146.

```
            PERFORM 8000-PARSE2-IN-BUFFER2
           EXEC CICS DELETEQ TS QUEUE(CGI-RETURN-TSQID2) END-EXEC
            EXEC CICS SYNCPOINT END-EXEC
            MOVE '9' TO CID-POS1
            MOVE 88  TO TSLEN
            PERFORM 9000-COPY-TO-TSQ2 WITH TEST AFTER
                    VARYING TSITEM FROM 1 BY 1
                    UNTIL EIBRCODE NOT EQUAL LOW-VALUES
        END-IF.
    0000-EXIT.
        EXEC CICS RETURN END-EXEC.
.
.
.
    8000-PARSE2-IN-BUFFER2.
        EXEC CICS GETMAIN SET(CGI-RETURN-BUFFER2)
             FLENGTH(PARSE2-TEMP-LENGTH) INITIMG(BLANKS)
             END-EXEC.
        SET  ADDRESS OF BUFFER-DATA-RETURN TO CGI-RETURN-BUFFER2.
        MOVE LENGTH OF PARSE2              TO CGI-RETURN-LENGTH2.
        MOVE PARSE2                        TO BUFFER-DATA-RETURN.

    9000-COPY-TO-TSQ2.
        EXEC CICS READQ TS QUEUE(CID) ITEM(TSITEM)
                  INTO(ITEM-DESC) LENGTH(TSLEN) END-EXEC.
        IF EIBRCODE = LOW-VALUES
           EXEC CICS WRITEQ TS QUEUE(CGI-RETURN-TSQID2)
                     FROM(TSDATA) LENGTH(90) END-EXEC
        END-IF.
```

Figure 124. Finish process in sample CGI used with IpServer

As mentioned, we do not return parameters other than those we received, so we just move this area into the return area expected by IpServer. When moving all records into the final CICS temporary storage queue, these records will be placed in the queue identified by the cid parameter we passed in with a hidden value.

The next step is to set up the HTML page with NetObjects Fusion to call the CGI program.

Figure 125 on page 147 shows the basic design of the page:

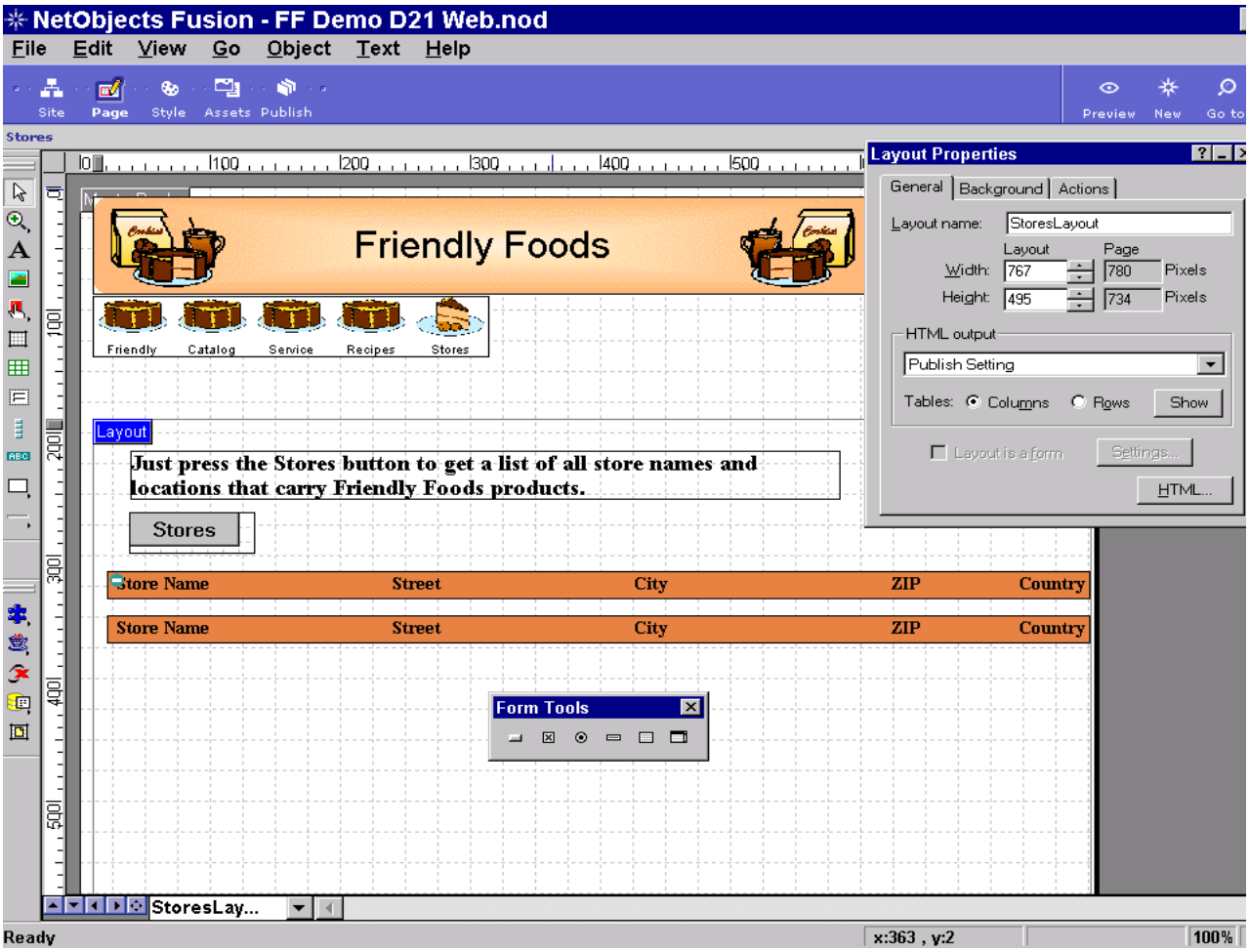


Figure 125. Design of the HTML page to call the sample CGI used with IpServer

First we define the parameters in the HTML form used to call the CGI program. The highlighted HTML form area is created by selecting the **Form Area** button (which is the eighth button from the top) in Fusion and drawing the space for it on the page.

In the action field we have to specify the name of our CGI program.Then we select the GET method for it and add one hidden parameter used to specify the identification of the CICS temporary storage queue; note the name cid and the special format of the value.

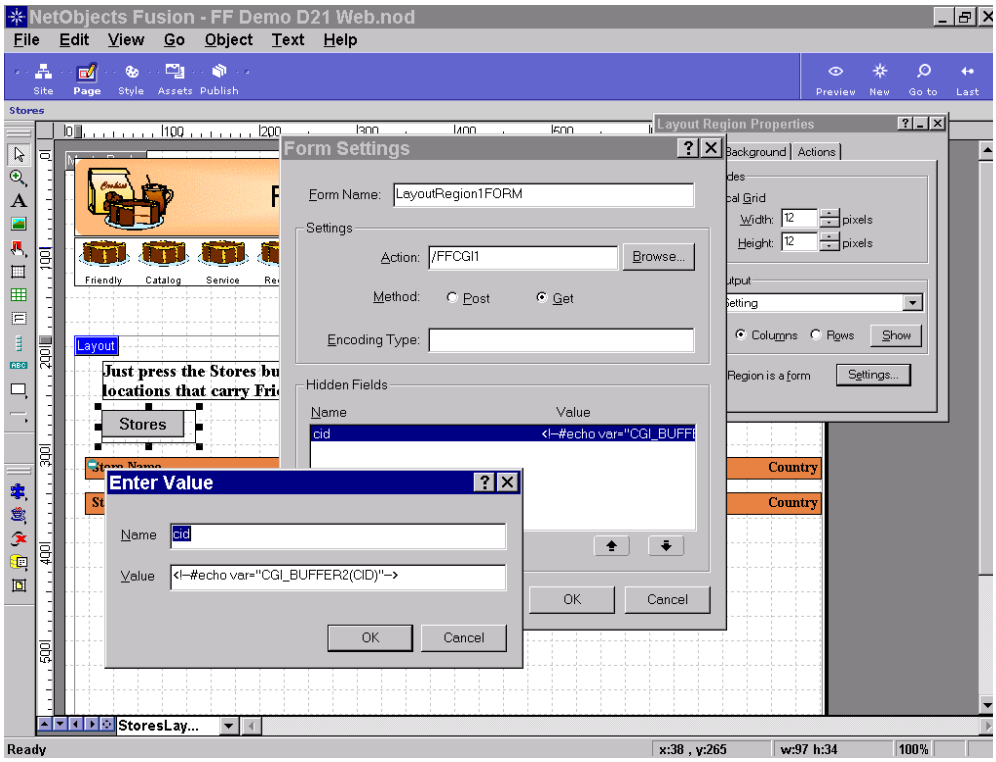


Figure 126. Setting up the HTML form to call the sample CGI used with IpServer

Now we define the pushbutton within the HTML form area with the values shown in Figure 127:

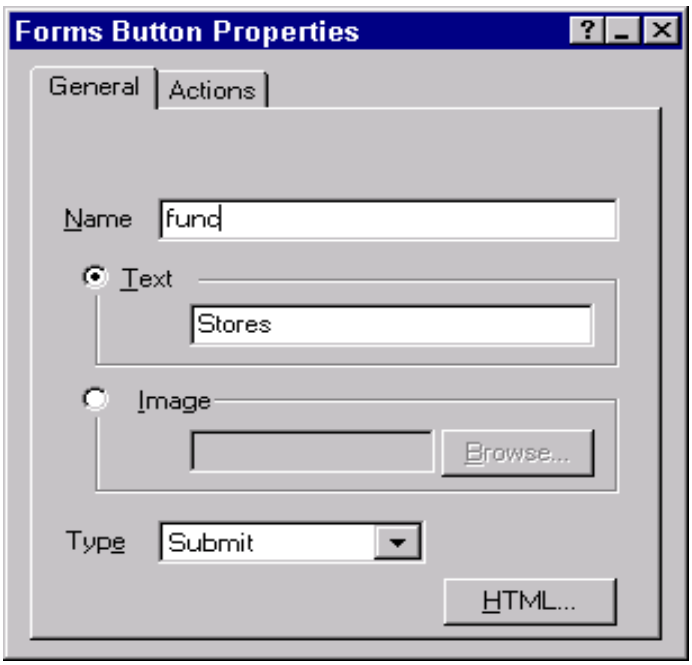


Figure 127. Setting up the button to call the sample CGI used with IpServer

Name identifies the name of the parameter passed. The Text field specifies the text displayed in the button as well as the value passed with this parameter.

Now we have to define the space used to display the result. Since this HTML type is not directly supported in Fusion, we have to attach this as extra code to another HTML object defined with Fusion. In our example we have two graphics acting as table header and footer describing the columns of our DB2 query result created by our CGI program. We attach the extra HTML code to the end of the first graphic. Note the special format, as shown in Figure 128.

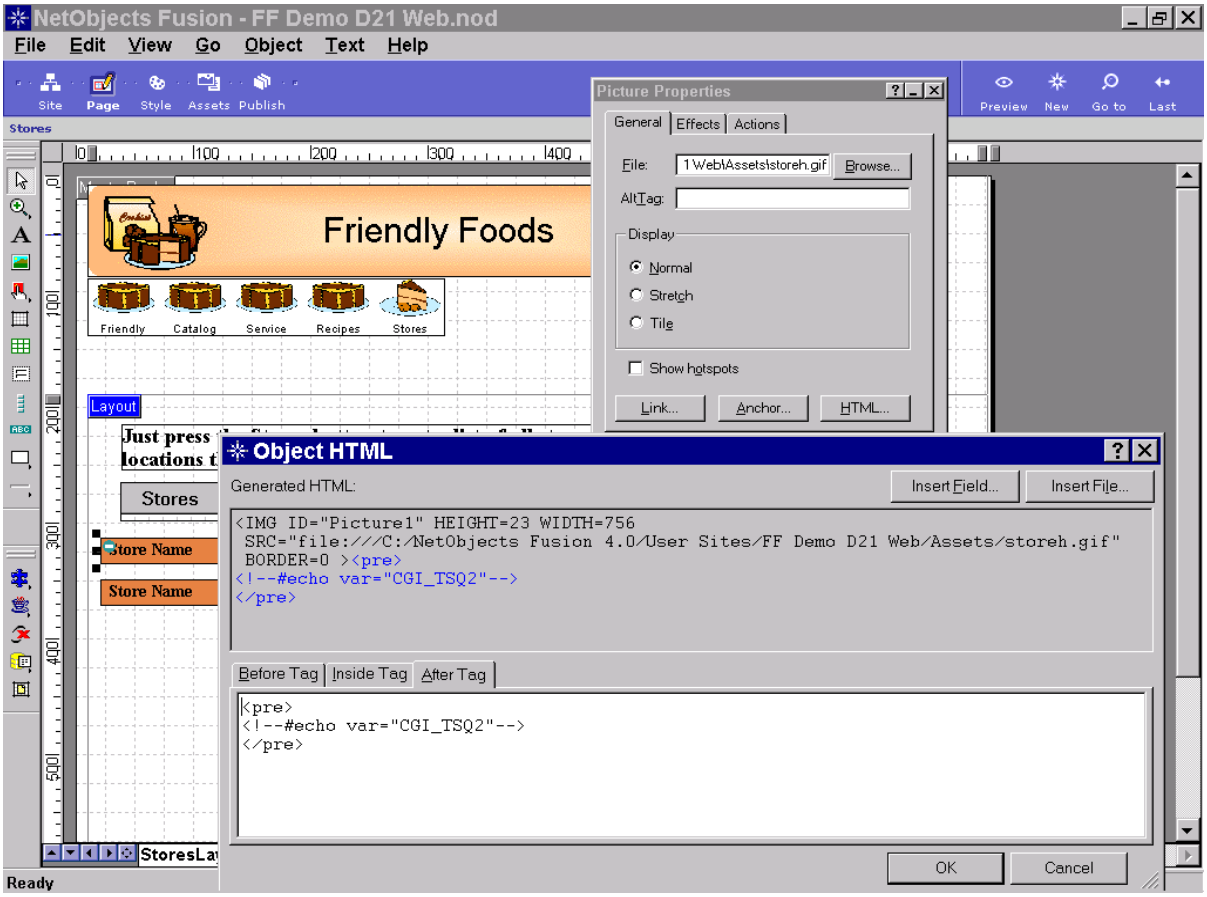


Figure 128. Define the space to display the result of the sample CGI used with IpServer

7.3.4 Getting ready to publish the site to VSE/ESA

After saving the Fusion site repository, we prepare for publishing on VSE. As a first step, we take care of the eight-character restriction for the element names by selecting the Publishing View in Fusion and doing the following.

We select the **Setup** button to make some minor changes required to publish on VSE. This will bring up a notebook with four tabs as shown in Figure 129 on page 150. On the first tab, Directory Structure, we select **Flat**. This means we want to publish all files into one directory and not generate deeper directory levels as we go deeper in our site structure.

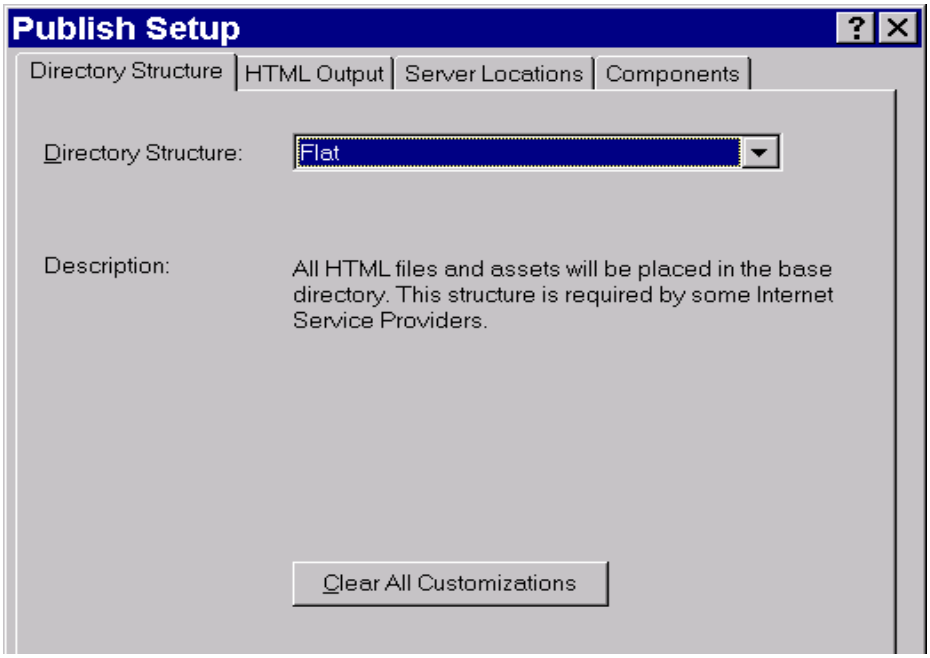


Figure 129. Setup for publishing site for IpServer - formatting

On the second tab, the HTML Output basic settings remain the same.

After we commit these changes, the Publishing View will change so that all elements appear on the same directory level.

Now we have to change the names of the elements so that they are no longer than eight characters and yet still unique. We can rename an element by selecting it and clicking it again. This will open up the name for the overwrite. We do this for each of the elements where the name is longer than eight characters.

The big advantage of using NetObjects Fusion is that we do not have to perform these changes in all places where these elements are referenced. Instead, Fusion will consider our changes later when generating the HTML output files for publishing.

Now we are ready to publish our site; see Figure 130 on page 151.

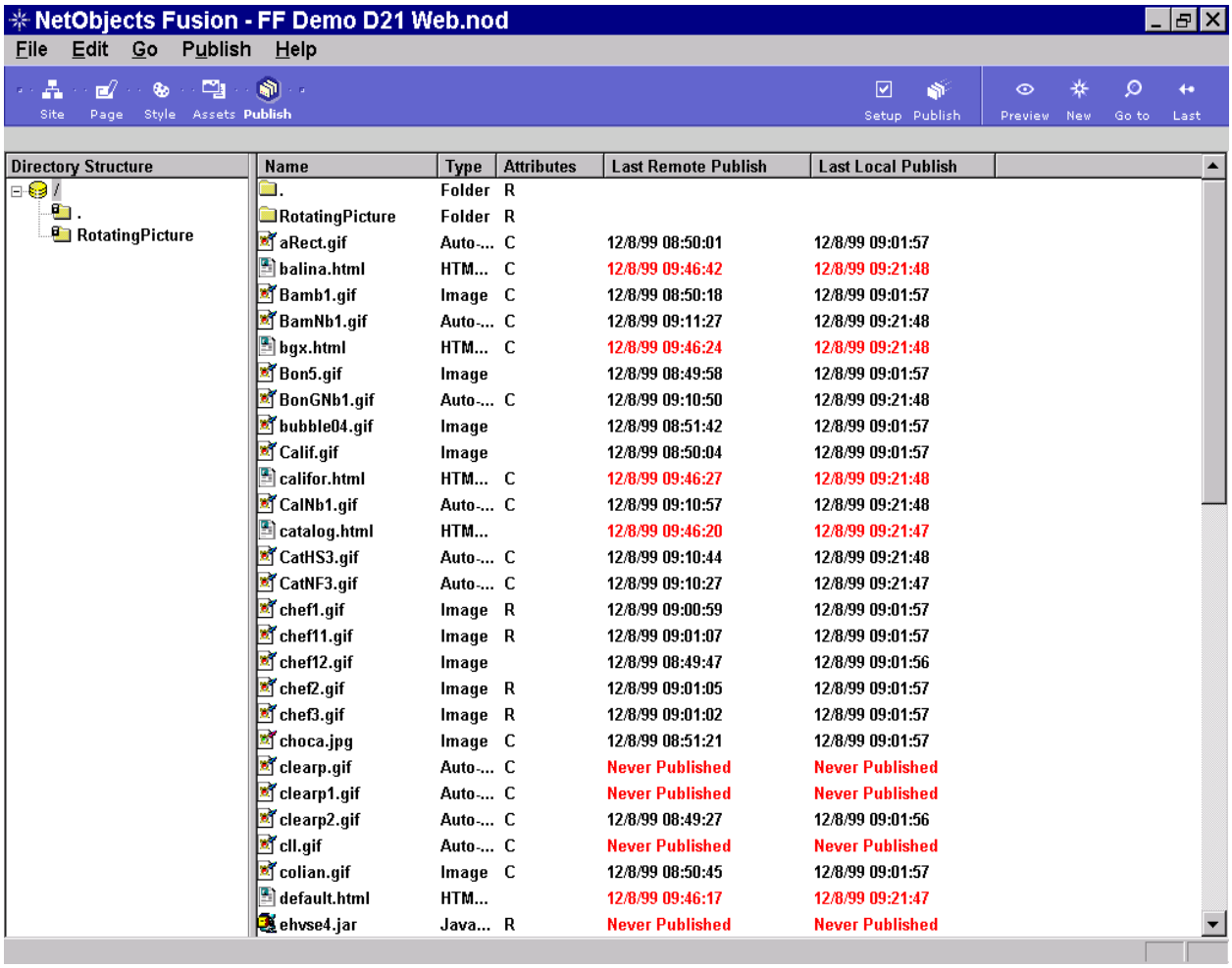


Figure 130. Publishing view ready to be published for IpServer

We start this process by clicking the **Publish** action button on the right side. We have to select the server we want to publish to. If we are publishing the site for the first time, we have to specify the address of the server, as well as the name of the base directory along with user ID and password for remote publishing via FTP; see Figure 131 on page 152.

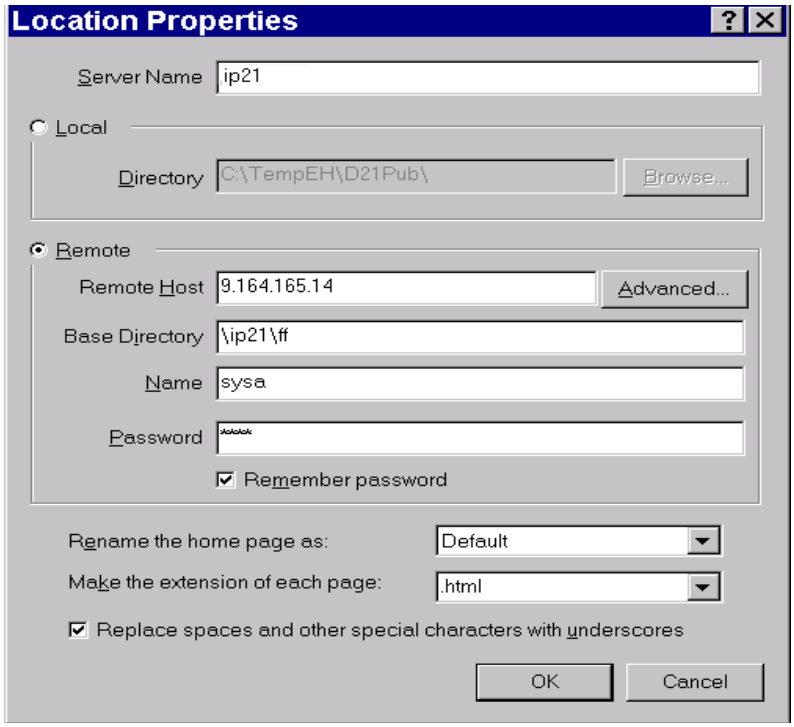


Figure 131. Setup location properties for publishing site for IpServer

Figure 132 shows the result of our CGI call.

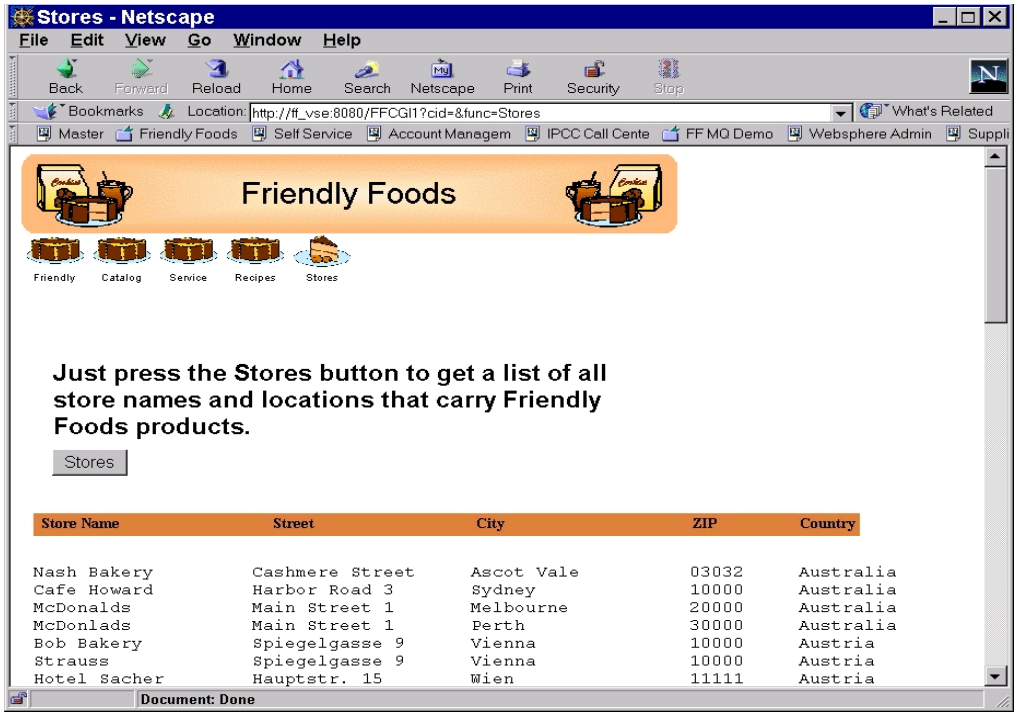


Figure 132. Result of our HTML page calling our sample CGI on IpServer

Chapter 8. Summary and outlook

Although Friendly Foods enjoys immediate business benefits from their e-business strategy, they are already planning the next steps that will accelerate their transformation into a world-class e-business.This chapter discusses the considerations needed to make that happen.

8.1 Improve network security

Network security is an important consideration for Friendly Foods as they expand their Internet presence. Friendly Foods will install a firewall. They plan to select IBM's SecureWay Firewall product. SecureWay Firewall comes in Windows NT (Netfinity) and AIX (RS/6000) versions. They are pleased with their existing Netfinity Web server, so have already decided to use Netfinity as their firewall as well.

At this point there is excess capacity on the existing Netfinity server so they could actually install the firewall on the same box as their WebSphere Application Server. However, Friend Foods believes a simpler, more adaptable, and more scalable approach is to just install a second Netfinity server that will only perform the firewall function.

Friendly Foods acknowledges network security is not one of their existing core IT competencies. Nor do they want to build these highly specialized skills and keep up-to-date on developments in the art and science of network security. Therefore they have decided to hire IBM Global Services or an IBM Business Partner with specific security expertise to design their overall network security plan, install and implement the Netfinity firewall, and periodically consult on network security matters on an ongoing basis as appropriate.

Friendly Foods is reviewing contractor bids at this time and plans to make a final selection soon. The award will specify a target completion within three months. Friendly Foods is currently trying to decide whether they should buy a new Netfinity for the firewall project, or use their existing Netfinity as a firewall and buy a newer, larger Netfinity to become the WebSphere Application Server.

8.2 Upgrade to VSE/ESA V2.4 and CICS Transaction Server for VSE/ESA

Friendly Foods' current system is based on VSE/ESA V2.3. They want to migrate to the most up-to-date release of VSE. The current release of VSE at this time is VSE/ESA V2.4.

VSE/ESA V2.4 provides a more robust, more capable version of CICS. CICS Transaction Server for VSE/ESA includes these benefits for Friendly Foods:

- A more robust environment
  - Support for the hardware subsystem storage protect feature to reduce unplanned outages
  - Enhanced security options including CA-Top Secret
- More e-business capabilities
  - CICS Web Support and 3270 bridge (shipped later in 2000)

8.2.1 VSE/ESA V2.4 security

Friendly Foods has transformed itself into a significant e-business. Their system is already open to employees, customers and suppliers. In addition to network security, overall security issues need to be addressed if they are to continue to expand their exploitation of Internet technology. VSE/ESA V2.4 enhances VSE security by adding a full-function external security manager: CA-Top Secret. With CA-Top Secret, Friendly Foods can implement additional security controls on critical VSE data and resources.

8.2.2 Exploiting CICS Web Support and 3270 bridge

CICS Web Support (CWS) provides a connection between TCP/IP and CICS. That is, CICS is no longer limited to SNA and 3270-compatible devices. CWS offers two ways to exploit the Internet. First, by using the 3270 bridge (which is also a component of CICS TS VSE/ESA), customers can access existing CICS applications with a text-based interface using standard Web browsers without changing the existing applications. Second, customers who separate presentation from logic can access CICS applications with an improved graphical interface using standard Web browsers.

One potential exploitation of CWS is to improve existing call center applications by replacing Host On-Demand (which they only recently installed as part of this project) implementation with CWS. The contractor Friendly Foods hired to handle the call center has significant employee turnover. That increases their charges to Friendly Foods because of substantial training and quality assurance costs. Using CWS, Friendly Foods plans to replace the existing text-based screen that is based on 3270 technology with newer, more intuitive graphical screens based on HTML technology. The new interface will be easier to learn and less error-prone.

8.2.3 Additional considerations

As the business expands and creative use of information technology grows, the workload on the Friendly Foods VSE system continues to grow. Therefore they are evaluating some options.

- They are considering replacing the existing S/390 with a Multiprise 3000 for additional capacity.
- Since they have rapidly growing storage needs for both S/390 and Netfinity, they are evaluating the IBM Enterprise Storage Server (known as Shark) as a way to provide reliable, high performance storage for both systems at a very attractive cost.

Friendly Foods also knows that at some point they may choose to migrate from VSE to OS/390. However, for the foreseeable future, they are comfortable with VSE working together with the complementary Windows NT-based Web application server. In the meantime they are taking care to keep their VSE system current and to ensure that all new work on VSE takes advantage of VSE-OS/390 affinity (CICS TS VSE/ESA, COBOL for VSE/ESA, DB2 Server, and so forth). Similarly, Friendly Foods ensures that all work on the Netfinity-based Web server exploits the standards and tools that are common between the WebSphere Advanced Edition they will soon install on Netfinity and the WebSphere Enterprise Edition that is standard on OS/390.

8.3 New business opportunities with e-commerce

Friendly Foods is realistic. They do not aspire to become a global “dot.com” powerhouse. Their focus is quality baked goods, not impressing mutual fund managers. However, they appreciate the fact that the Internet opens up a variety of new business opportunities that is limited only by their own commitment and imagination.

8.3.1 Business concept

In the past, Friendly Foods often thought about expanding their product line with high value, seasonal baked goods. For example, they believe an opportunity exists for new products such as fruitcake, Christmas cookies, gingerbread houses, and Lebkuchen.

These products are priced significantly higher than the normal product mix. They are often purchased to be given as gifts during the holiday season. That means there is the potential for generating substantial new revenues while still remaining within the Friendly Foods area of “core competency” and without requiring a major investment in new production facilities.

However, in the past Friendly Foods decided against adding such products to their stores. The cost of production is quite high and the sales turnover low. The inventory costs tend to be unacceptable. In addition, because demand is seasonal, sales are difficult to forecast. That means Friendly Foods would face a trade-off of missed sales opportunities or excessive cost of scrapped goods at the end of the season.

The Internet changes all that. From their Web site, Friendly Foods can market specialty baked goods worldwide. They can be produced in Friendly Foods’ existing production facilities and shipped by airborne package delivery directly to the customer. There is no need to stock the items in their stores. That eliminates costly inventory and scrap. And because they do not schedule any production until they have actually accepted electronic payment for the product and shipping costs, the problem of forecasting demand is greatly reduced.

8.3.2 Implementation

Many of the tools and much of the infrastructure is already in place. To support the specialty baked goods initiative, Friendly Foods must upgrade their Windows NT systems with the following steps:

- Upgrade from WebSphere Standard Edition to Advanced Edition.
- Add DB2 Universal Database.
- Install the net.Commerce product suite.
- Write a few Java programs to notify the VSE-based production system and accounting systems when a valid order is received. Communications with VSE would be through the existing MQSeries connection.
- Write a couple of new VSE applications to handle shipping labels, maintain customer information, and interface with the package delivery service.
- Consider increasing the capacity to their IBM Netfinity server to handle the increased load, especially the peak load that occurs during the holiday season.

8.3.3 Additional opportunities

Friendly Foods observed that other products have characteristics similar to specialty baked goods. For example, coffee and chocolates are high value items that are often purchased during the holiday season as gifts. Can Friendly Foods take advantage of the potential marketing synergy without making a huge investment to support these new, unfamiliar product lines in addition to the bakery business they know so well?

Internet technology lets them pursue these additional opportunities in ways that would not have been possible in the past. With e-business they can develop strategic partnerships with other companies. In this case Friendly Foods is negotiating with a Belgian manufacturer of high quality chocolates and with an American maker of premium coffee.

The plan is that Friendly Foods will add chocolates and coffee to their Web site. Customers can select whatever they want including fruitcakes, chocolates, and coffee. Orders for fruitcakes are sent to the Friendly Foods bakery and shipped to the customer. Orders for chocolate or coffee are relayed to the Belgian or American partners who then prepare the product, put a “Friendly Chocolates” or “Friendly Coffee” label on the package, and ship directly to the customer.

Notice that no new technology is needed. Friendly Foods has the skills and systems in place already. All that is required are some simple Java programs that communicate with the partners using MQSeries and a few new programs, or extensions to existing programs, on the VSE system to add information to the customer information file, accounting systems, and to the accounts payable system to pay business partners the wholesale price for the product they have shipped.

8.4 Improved customer service with pervasive computing

Friendly Foods is closely following developments in pervasive computing. In particular they are interested in new opportunities made possible by combining the Internet and wireless technology. They believe the technology and standards are beginning to solidify quickly and want to be in the early wave of companies that will be able to gain competitive advantage exploiting cellphones and PDAs.

8.4.1 Business concept

One application of wireless technology will be for store managers. Using a cellphone, they will be able to update tomorrow’s order from home, or inquire about the status of their shipment (for example: Has the truck that is carrying their order left the bakery yet?).

The second application will be for consumers. Especially valuable customers will be able to contact Friendly Foods and request that a particular product be “reserved”. That means a customer can use his cellphone before he leaves the office to send a message to the nearest Friendly Food shop and ask that they set aside a loaf of his favorite pumpernickel bread.

8.4.2 Implementation

Special purpose plug-ins will soon be available for WebSphere. These additions will support new wireless standards and recognize the limitations of special purpose browsers designed for handheld devices (compared with full function PC browsers such as NetScape). Friendly Foods will simply add these plug-ins to their WebSphere Application Server. They will use existing tools such as VisualAge for Java to develop the required applications.

It is not expected that the first round of pervasive applications will involve the VSE system at all. However, as Friendly Foods gains experience and imagines new application possibilities, it may be that interactions with new or enhanced VSE applications will be needed.

8.5 Conclusion

In summary, the e-business projects described in this redbook are just the beginning for Friendly Foods. They are receptive to the technology and eager to deploy it in ways that make them more competitive. In many ways they are not that different from most VSE customers. What sets them apart is simply a strong commitment to exploit Internet technology to benefit themselves and their customers.

Appendix A. e-business technologies

This appendix provides a high-level explanation of the major e-business technologies and new terms. It also gives a short description of the evolution of e-business. However, it does not explain the TCP/IP protocol and basic HTTP protocol. Instead, it focuses on the technologies available to develop Web applications.

A.1 User Interface technologies

e-business applications normally are accessible through thin clients consisting of a Java-enabled Web browser and a Java Virtual Machine (JVM), a TCP/IP stack, and (potentially) an encryption library. There are several different client technologies available to the Web developer. The design of the client side of a Web application varies considerably depending upon which technology is employed. The choice of client-side technology must be made in concert with the server-side technology choice. The following paragraphs describe the major User Interface (UI) technologies.

A.1.1 HTML

The Hypertext Markup Language (HTML) specification defines UI elements for text with various fonts and colors, lists, tables, images, and forms (text fields, buttons, check boxes, radio buttons). It is very similar to other markup languages like IBM BookMaster.

The provided elements are adequate to display the user interface for most applications. The disadvantage, however, is that these elements have a generic look and feel, and they lack customization. As a result, many e-business application developers augment HTML with other UI technologies to enhance the visual experience.

HTML has the advantage that the client-side Web application can be a simple HTML browser, enabling a less capable client to execute an e-business application. A good introduction to HTML can be found at:
`http://www.w3.org/MarkUp/`

A.1.2 JavaScript

JavaScript is a scripting language defined by Netscape. With this language the static HTML parts of a Web page can be extended by objects which enable the manipulation of HTML documents (e.g. checking form fields, submitting forms, and creating dynamic pages). With this, the page author can give the user a feeling of interaction with both the site and its contents.

A.1.3 Java applet

A Java applet is a program written in Java that is downloaded from the Web server and run on the Web browser. The applet to be run is specified in the HTML page using an APPLET tag.

It is the most flexible option of the UI technologies that can be run in a Web browser. Java provides a rich set of UI elements that include an equivalent for

each of the HTML UI elements. In addition, because Java is a programming language, an infinite set of UI elements can be built and used.

A disadvantage of using Java applets for UI generation is that any class that is not included as part of Java must be loaded from the Web server as it is needed. If the additional classes are large, the initialization of the applet may take from seconds to minutes depending upon the speed of the connection to the Internet.

A.2 Server-side logic

Server-side logic is used to bring business logic and dynamic content to the Web interface. This was first attempted by having Web servers that support the CGI-BIN interface to invoke application fragments. CGI-BIN is a good model in that it is standard across different vendors' Web servers. One of the disadvantages, however, is that CGI-BIN processing is quite expensive in terms of consuming server resources (especially execution cycles) because a new process must be launched for every HTTP request requiring application function.

The shortcomings of the CGI-BIN model caused each server vendor to develop a plug-in architecture that allows application elements to be linked directly into the Web server. This eliminates the overhead of CGI-BIN, but introduces new problems which are:

- Server plug-in APIs are vendor-specific, which therefore makes it very difficult to port from one vendor's Web server to another's.
- Server plug-ins run as part of the Web server and therefore a program error can compromise the entire Web server, resulting in a more complex and less reliable development model.

The IBM Application Framework for e-business defines that application elements can be implemented as specialized Java classes called Java servlets which exhibit the best features of CGI-BIN and server plug-ins, without their drawbacks.

A.2.1 Java servlets

Servlets are the units of Java programs that handle Web requests. They take as input the HTTP request from the Web client and output dynamically generated HTML. *A servlet can almost be thought of as an applet that runs on the server side--without a face*. The servlets are part of the Java standard as defined by Sun Microsystems, Inc. For more information on the Java Servlet API, refer to the home page at:

`http://java.sun.com/products/servlet/index.html`

The typical Web application involves generating large amounts of user interface content such as HTML. Some of this content is dynamic, depending on the results of computations, but much is static. The basic servlet mechanism specifies that servlets not only implement business logic but are also responsible for generating the user interface content that is sent to the client. *However, specifying the static part of a response page within a servlet is not the model promoted by the Framework*. This implementation would not allow a separation of skills into the user interface design and the program logic design. To solve these problems, the Framework's programming model includes Java Server Pages (JSPs).

A.2.2 Java Server Pages (JSPs)

Java Server Pages are HTML pages extended with a number of mechanisms to allow content to be added to the page as it is sent to the client. All processing of JSP HTML extensions is done on the server and each of the extensions is either removed or replaced before the page is sent to the client. Therefore, no additional support is required on the client.

A.3 e-business terms and Java-related definitions

The following is a collection of new e-business terms and definitions which are used throughout this redbook.

A.3.1 Java

Java normally is the term for two different things which are related, but not the same:

1. An object-oriented programming language for portable interpretive code that supports interaction among remote objects. Java was developed and specified by Sun Microsystems, Inc.
2. A runtime environment for the programs written in the Java language. It is called the Java Virtual Machine (JVM) and is the base for the portability of the Java code.

A.3.2 JavaBeans

The official definition for JavaBeans (as by Sun Microsystems) is:

"A JavaBean is a reusable software component that can be manipulated visually in a builder tool."

Sun Microsystems defined the JavaBeans API. The goal of the JavaBean APIs is to define a software component model for Java, so that third party ISVs can create and ship Java components that can be composed together into applications by end users. JavaBeans are Java components designed to be used on *client* systems. For more information on JavaBeans, refer to the home page at:

`http://java.sun.com/beans/`

A.3.3 Enterprise JavaBeans

The Enterprise JavaBeans architecture extends the JavaBeans architecture to provide a framework for developing and deploying component-based, multi-tier applications. Enterprise JavaBeans are server-side Java components that are designed for distributed environments.

Enterprise JavaBeans run in a specific run-time environment which is called the Enterprise JavaBean container. It is used to execute the components that provide services to an application (comparable to a Transaction Monitor). It does load balancing, manages state on the client, and automatically handles multi-threading, transaction management, security, and connection management so that programmers can focus on developing business logic.

The Enterprise JavaBean container is implemented by IBM WebSphere Application Server, Advanced Edition. More information on Enterprise JavaBeans can be found at:

http://java.sun.com/products/ejb/

A.3.4 Java Database Connectivity (JDBC)

JDBC is a Java API that allows Java programs to communicate with different database management systems in a platform-independent manner. Database vendors provide JDBC drivers for their platforms that implement the API for their database, allowing the Java developer to write applications to a consistent API no matter which database is used. For more information on JDBC, refer to:

http://java.sun.com/products/jdbc/

Appendix B. Sample programs

Following is the source of the COBOL program FFCGI1 that we used in 7.3.3, “Using dynamic pages with IpServer” on page 141. This program was derived from one of the sample CGI programs provided by Data 21, Inc.

```
 IDENTIFICATION DIVISION.
      PROGRAM-ID.    FFCGI1.
      AUTHOR.        IBM.
    *-----------------------------------------------------------------*
     * FFCGI1  - IBM APPLICATION TEST CGI PROGRAM.  WHEN 'Stores'    *
     *          IS SENT FROM A BROWSER, DB2 DATA                     *
     *          CONTAINING STORE NAMES WILL BE PRESENTED BACK TO     *
     *         THE BROWSER USING THE DATA21 IPSERVER WEB INTERFACE. *
     *                                                               *
     * PREREQUISITE:                                                 *
     *          1. DATA21 IPSERVER STARTED IN A CICS PARTITION.      *
     *                                                               *
     * FUNCTIONS:  1. ACTIVATED VIA WEB BROWSER REQUEST.             *
     *                                                               *
     * CALLED BY:  -- NONE --                                        *
     *                                                               *
     * CHANGE SUMMARY:                                               *
     *                                                               *
    *-----------------------------------------------------------------*
      ENVIRONMENT DIVISION.
      CONFIGURATION SECTION.
      DATA DIVISION.
    *-----------------------------------------------------------------*

      WORKING-STORAGE SECTION.
     * DECLARE DB2 VARIABLES
     *
          EXEC SQL BEGIN DECLARE SECTION END-EXEC.
       01  DB2RECORD.
          05  STORENAME  PIC X(20).                                  00000790
          05  LOCSTREET  PIC X(20).                                  00000800
          05  LOCCITY    PIC X(20).                                  00000810
          05  LOCZIP     PIC X(10).                                  00000820
           05  LOCCOUNTRY PIC X(18).
00000830
      *
       01  USERID    PIC X(8) VALUE 'SQLDBA'.
       01  PASSW     PIC X(8) VALUE 'SQLDBAPW'.
       01  UTIME     PIC S9(15) COMP-3.
      *
          EXEC SQL END DECLARE SECTION END-EXEC.
          EXEC SQL INCLUDE SQLCA END-EXEC.
      *     COPY COPYRWS.
     *-----------------------------------------------------------------*
      * COPYRIGHT WORKING STORAGE  FOR COBOL MODULES             *
     *-----------------------------------------------------------------*
       01   FILLER.
            05 FILLER                PIC X(72) VALUE
            'Licensed Materials - Property of IBM'.
            05 FILLER                PIC X(72) VALUE SPACES.
            05 FILLER                PIC X(72) VALUE
```

Figure 133. Friendly Foods CGI program for use with IpServer (part 1 of 6)

```
   '5686-A06  '.
            05 FILLER                PIC X(72) VALUE SPACES.
            05 FILLER                PIC X(72) VALUE
            '(C) Copyright IBM Corp. 1998           All Rights Reserve
       -     'd'.
            05 FILLER                PIC X(72) VALUE SPACES.
            05 FILLER                PIC X(72) VALUE
           'US Government Users Restricted Rights - Use, duplication or
       -     '  '.
            05 FILLER                PIC X(72) VALUE
           'disclosure restricted by GSA ADP Schedule Contract with IBM
       -     ' Corp.'.
      *-------------------------------------------------------------*
      *Debugging eyecatcher information for start of WORKING-STORAGE.
      *-------------------------------------------------------------*
        01  WS-RWS-PROGRAM-NAME1       PIC X(8).
        01  FILLER                     PIC X(16) VALUE
             ' Version V2.1.0'.
        01  WS-RWS-WHEN-COMPILED       PIC X(21).
        01  FILLER                    PIC X(7)  VALUE '======='.
      *-------------------------------------------------------------*
      *-------------------------------------------------------------------*

        01  WS-WORK-FIELDS.
            05  WS-MORE-FLAG             PIC  XX    VALUE SPACES.
              88 WS-MORE-DATA                       VALUE SPACES.
              88 WS-NOMORE-DATA                     VALUE 'Y'.

           05  WS-DATA-LENGTH           PIC S9(4)  COMP   VALUE ZERO.
           05  WS-APPL-MSG-LENGTH       PIC S9(8)  COMP   VALUE ZERO.
            05  WS-ABSTIME             PIC S9(15) COMP-3.
            05  WS-DATE.
               10  WS-DATE-CC          PIC  99  VALUE ZERO.
               10  WS-DATE-YYMMDD.
                 12  WS-DATE-YY        PIC  99  VALUE ZERO.
                 12  WS-DATE-MM        PIC  99  VALUE ZERO.
                 12  WS-DATE-DD        PIC  99  VALUE ZERO.
                 12  FILLER            PIC  XX  VALUE ZERO.

           05  WS-UNPACK-TIME-9        PIC 9(07) VALUE ZEROES.
           05  WS-UNPACK-TIME-X REDEFINES WS-UNPACK-TIME-9.
               10  FILLER              PIC  X(01).
               10  WS-TIME-HHMMSS.
                  12  WS-TIME-HH       PIC  X(02).
                  12  WS-TIME-MM       PIC  X(02).
                  12  WS-TIME-SS       PIC  X(02).
            05  WS-FORMATTED-TIME.
               10  WS-FORMAT-TIME-HH    PIC  X(02) VALUE SPACES.
                10  FILLER              PIC  X(01) VALUE ':'.
               10  WS-FORMAT-TIME-MM    PIC  X(02) VALUE SPACES.
```

Figure 134. Friendly Foods CGI program for use with IpServer (part 2 of 6)

```
10  FILLER               PIC  X(01) VALUE ':'.
           10  WS-FORMAT-TIME-SS    PIC  X(02) VALUE SPACES.
         05  WS-FORMATTED-DATE.
           10  WS-FORMAT-DATE-MM    PIC  X(02) VALUE SPACES.
            10  FILLER              PIC  X(01) VALUE '/'.
           10  WS-FORMAT-DATE-DD    PIC  X(02) VALUE SPACES.
            10  FILLER              PIC  X(01) VALUE '/'.
           10  WS-FORMAT-DATE-YY    PIC  X(02) VALUE SPACES.
      01  WORKAREA.
          05  BLANKS              PIC X      VALUE ' '.
          05  CID.
              10  CID-POS1        PIC X.
              10  CID-POS2-7      PIC X(7).
          05  TSITEM              PIC S9(4)   VALUE +0  COMP.
          05  TSLEN               PIC S9(4)   VALUE +88 COMP.
          05  DELITEM             PIC S9(4)   VALUE +0  COMP.
     * RETURN-PAGE MUST BE IN UPPER CASE
         05  RETURN-PAGE  PIC X(44) VALUE '/FF/STORES.HTML'.
          05  PARSE2-LENGTH       PIC S9(4)   VALUE +0  COMP.
    * PARSE2-TEMP-LENGTH VALUE MUST BE => PARSE2(SEE INSIDE D21IPARC)
          05  PARSE2-TEMP-LENGTH   PIC S9(8)   VALUE +512 COMP.

      01  TSDATA.
          05  ITEM-DESC           PIC X(88)   VALUE ' '.
          05  CRLF-0D25-1         PIC X(2)    VALUE X'0D25'.

    *-------------------------------------------------------------*
     *Debugging eyecatcher information for end  of WORKING-STORAGE.
    *-------------------------------------------------------------*
      01  FILLER                   PIC X(16)  VALUE
           '================'.
      01  WS-RWS-PROGRAM-NAME2       PIC X(8).
      01  FILLER                   PIC X(16) VALUE
           ' Version V2.1.0'.
      01  M0000                    PIC X(6) VALUE 'M0000 '.
      01  M1000                    PIC X(6) VALUE 'M1000 '.
      01  M2000                    PIC X(6) VALUE 'M2000 '.
      01  M3000                    PIC X(6) VALUE 'M3000 '.
      01  M3100                    PIC X(6) VALUE 'M3100 '.
    *-------------------------------------------------------------*
      01  M8000                    PIC X(6) VALUE 'M8000 '.
      01  M9000                    PIC X(6) VALUE 'M9000 '.
      01  STEP6                    PIC X(6) VALUE 'STEP6 '.
      01  OUR-LENGTH               PIC S9(4)  COMP   VALUE 300.
      01  DB21                     PIC X(6) VALUE 'DB21'.
    *-------------------------------------------------------------*
      COPY D21IPARC.
      LINKAGE SECTION.

      01  DFHCOMMAREA.
```

Figure 135. Friendly Foods CGI program for use with IpServer (part 3 of 6)

```
        COPY D21ICGIC.

     * LENGTH MUST BE = PARSE2-TEMP-LENGTH
      01  BUFFER-DATA-RETURN      PIC X(512).

      PROCEDURE DIVISION.

      0000-MAIN-LINE.
         MOVE 'FFCGI1'  TO WS-RWS-PROGRAM-NAME1
                           WS-RWS-PROGRAM-NAME2.
         MOVE WHEN-COMPILED TO WS-RWS-WHEN-COMPILED.
     *--SET UP CHECK FOR FIELDS PASSED
          IF VALID-COMMAREA
             PERFORM 1000-INITIALIZE
               THRU 1000-EXIT
             PERFORM 2000-INIT-ROUTINE
             EVALUATE PARSE-FUNCTION
                WHEN 'Stores' PERFORM 3000-GET THRU 3100-EXIT
             END-EVALUATE
             PERFORM 8000-PARSE2-IN-BUFFER2
            EXEC CICS DELETEQ TS QUEUE(CGI-RETURN-TSQID2) END-EXEC
             EXEC CICS SYNCPOINT END-EXEC
             MOVE '9' TO CID-POS1
             MOVE 88  TO TSLEN
            PERFORM 9000-COPY-TO-TSQ2 WITH TEST AFTER
                    VARYING TSITEM FROM 1 BY 1
                   UNTIL EIBRCODE NOT EQUAL LOW-VALUES
          END-IF.
      0000-EXIT.
         EXEC CICS RETURN END-EXEC.
    *---------------------------------------------------------------*
      1000-INITIALIZE.
    *---------------------------------------------------------------*
     * PURPOSE: SETUP DATA AREAS
    *---------------------------------------------------------------*

     *--SET UP ERROR AREA
          PERFORM   1050-SET-ERROR-INFO.
          PERFORM   1100-RECEIVE-INPUT.
     *

    *---------------------------------------------------------------*
      1000-EXIT.
          EXIT.

    *---------------------------------------------------------------*
      1050-SET-ERROR-INFO.
    *---------------------------------------------------------------*
     * PURPOSE: SET DEFAULT ERROR INFO
```

Figure 136. Friendly Foods CGI program for use with IpServer (part 4 of 6)

```
 *--SET CSMT DATE AND TIME
      EXEC CICS ASKTIME
                ABSTIME(WS-ABSTIME)
      END-EXEC.

      MOVE EIBTIME          TO WS-UNPACK-TIME-9.
      MOVE WS-TIME-HH       TO WS-FORMAT-TIME-HH
      MOVE WS-TIME-MM       TO WS-FORMAT-TIME-MM.
      MOVE WS-TIME-SS       TO WS-FORMAT-TIME-SS.

      EXEC CICS FORMATTIME
                ABSTIME (WS-ABSTIME)
                MMDDYY  (WS-FORMATTED-DATE)
                DATESEP ('/')
      END-EXEC.
 *
      EXEC  CICS FORMATTIME
                 ABSTIME(WS-ABSTIME)
                 YYMMDD (WS-DATE-YYMMDD)
      END-EXEC.

 *-- --SET CENTURY
      IF WS-DATE-YY >  50
          MOVE  19                TO  WS-DATE-CC
      ELSE
          MOVE  20                TO  WS-DATE-CC
      END-IF.


*-----------------------------------------------------------------*

*-----------------------------------------------------------------*
   1100-RECEIVE-INPUT.
 * TRANSLATE INPUT FIELDS/PARAMETERS TO UPPERCASE: YES/NO
 *    SET PARSE-TR-UPPERCASE-YES TO TRUE.
      SET PARSE-TR-UPPERCASE-NO  TO TRUE.
     CALL 'D21IPARS' USING DFHCOMMAREA, PARSE1, PARSE2, PARSE3.
*-----------------------------------------------------------------*
   2000-INIT-ROUTINE.

      EXEC CICS IGNORE CONDITION ERROR END-EXEC.

 * PASS UNIQUE CONVERSATION ID FOR DURATION OF TRANSACTION
 * CAN BE USED TO STORE TEMPORARY DATA
 * USE '8' OR '9' IN CID-POS1 FOR USER DATA
 * POS 2-7 IS THE CICS TASK NUMBER THAT STARTED CONVERSATION
      IF PARSE-CID = SPACES
         MOVE CGI-RETURN-TSQID TO CID
         MOVE '9'              TO CID-POS1
```

Figure 137. Friendly Foods CGI program for use with IpServer (part 5 of 6)

```
ELSE
           MOVE PARSE-CID       TO CID
         END-IF.
        MOVE CGI-RETURN-TSQID TO CGI-RETURN-TSQID2.
        MOVE '2'             TO CGI-RETURN-TSQID2-POS1.
    * DISPLAY SAME PAGE WITH DATA
        MOVE RETURN-PAGE        TO CGI-RETURN-PAGE-NAME.
  *-----------------------------------------------------------------*
     3000-GET.
         IF PARSE-SELNAME = SPACES
           MOVE 'Description is Blank' to PARSE-SELNAME
         END-IF.
    *--CONNECT TO DB2
         EXEC SQL CONNECT :USERID IDENTIFIED BY :PASSW
         END-EXEC.
         IF SQLCODE NOT = 0 THEN
           GO TO 0000-EXIT.
         EXEC SQL DECLARE C1 CURSOR FOR
            SELECT STORENAME, LOCSTREET, LOCCITY, LOCZIP,
            LOCCOUNTRY FROM SQLDBA.FFSTORES
            ORDER BY LOCCOUNTRY, LOCZIP
         END-EXEC.
         IF SQLCODE NOT = 0 THEN
           GO TO 0000-EXIT.
         EXEC SQL OPEN C1 END-EXEC.
         IF SQLCODE NOT = 0 THEN
           GO TO 0000-EXIT.
     3050-FETCH-NEXT.
         EXEC SQL FETCH C1
            INTO :STORENAME, :LOCSTREET, :LOCCITY, :LOCZIP,
             :LOCCOUNTRY
         END-EXEC.
         IF SQLCODE NOT = 0 THEN
           GO TO 3100-EXIT.
         EXEC CICS WRITEQ TS QUEUE(CID)
                  FROM(DB2RECORD) LENGTH(88) END-EXEC.
           GO TO 3050-FETCH-NEXT.
     3100-EXIT.
         EXIT.
     8000-PARSE2-IN-BUFFER2.
        EXEC CICS GETMAIN SET(CGI-RETURN-BUFFER2)
            FLENGTH(PARSE2-TEMP-LENGTH) INITIMG(BLANKS)
             END-EXEC.
       SET  ADDRESS OF BUFFER-DATA-RETURN TO CGI-RETURN-BUFFER2.
        MOVE LENGTH OF PARSE2            TO CGI-RETURN-LENGTH2.
        MOVE PARSE2                      TO BUFFER-DATA-RETURN.

     9000-COPY-TO-TSQ2.
        EXEC CICS READQ TS QUEUE(CID) ITEM(TSITEM)
                 INTO(ITEM-DESC) LENGTH(TSLEN) END-EXEC.
         IF EIBRCODE = LOW-VALUES
           EXEC CICS WRITEQ TS QUEUE(CGI-RETURN-TSQID2)
                    FROM(TSDATA) LENGTH(90) END-EXEC
         END-IF.
```

Figure 138. Friendly Foods CGI program for use with IpServer (part 6 of 6)

Appendix C. Special notices

This publication is intended to help customers, business partners, and IBMers to plan, install and configure e-business solutions for the VSE/ESA environment. The information in this publication is not intended as the specification of any programming interfaces that are provided by VSE/ESA V2R3 and the products described in this publication. See the PUBLICATIONS section of the IBM Programming Announcement for VSE/ESA V2R3 for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM ("vendor") products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

IBM ®
AIX
AS/400
CICS
CT
DB2
DB2 Universal Database
Enterprise Storage Server
MQSeries
Netfinity
OS/2
OS/400
RS/6000
SecureWay
TCS
VM/ESA
VTAM
Wizard
400
ACF/VTAM
APPN
AT
CICS/VSE
CUA
DB2 Connect
DRDA
FFST
Multiprise
Operating System/2
OS/390
RISC System/6000
S/390
SP
VisualAge
VSE/ESA
WebSphere
XT

The following terms are trademarks of other companies:

Tivoli, Manage. Anything. Anywhere.,The Power To Manage., Anything. Anywhere.,TME, NetView, Cross-Site, Tivoli Ready, Tivoli Certified, Planet Tivoli, and Tivoli Enterprise are trademarks or registered trademarks of Tivoli Systems Inc., an IBM company, in the United States, other countries, or both. In Denmark, Tivoli is a trademark licensed from Kjøbenhavns Sommer - Tivoli A/S.

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

PC Direct is a trademark of Ziff Communications Company in the United States and/or other countries and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through The Open Group.

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product, and service names may be trademarks or service marks of others.

Appendix D. Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

D.1 IBM Redbooks

For information on ordering these publications see “How to get IBM Redbooks” on page 177.

- *MQSeries for VSE/ESA*, SG24-5647
- *WebSphere Application Server: Standard and Advanced Editions*, SG24-5460
- *IBM SecureWay Host On-Demand: Enterprise Communications in the Era of Network Computing*, SG24-2149
- *Programming with VisualAge for Java Version 2*, SG24-5264
- *Getting Started with TCP/IP for VSE/ESA 1.4*, SG24-5626
- *CICS/VSE in a Networking World: Connecting to CICS Servers and Clients*, SG24-2047
- *Personal Communications Version 4.3 for Windows 95, 98 and NT,* SG24-4689
- *DB2 Solutions with VSE & VM Implementation and Usage*, SG24-2036
- *Multi-Language Solutions for Client/Server Database Environment*, SG24-4846
- *From Multiplatform Operational Data to Data Warehousing and Business Intelligence*, SG 24-5174
- *Getting Started with Data Warehouse and Business Intelligence*, SG24-5415

D.2 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at http://www.redbooks.ibm.com/ for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	Collection Kit Number
System/390 Redbooks Collection	SK2T-2177
Networking and Systems Management Redbooks Collection	SK2T-6022
Transaction Processing and Data Management Redbooks Collection	SK2T-8038
Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
AS/400 Redbooks Collection	SK2T-2849
Netfinity Hardware and Software Redbooks Collection	SK2T-8046
RS/6000 Redbooks Collection (BkMgr Format)	SK2T-8040
RS/6000 Redbooks Collection (PDF Format)	SK2T-8043
Application Development Redbooks Collection	SK2T-8037
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

D.3 Other resources

These publications are also relevant as further information sources:

- *VSE Applications: How e-business fits*, GF22-5137

D.4 Referenced Web sites

These Web sites are also relevant as further information sources:

http://www.data21.com/

http://www.intelliware.com/

http://www.ibm.com/software/ebusiness/library.html/

http://www.w3.org/MarkUp/

http://java.sun.com/products/servlet/index.html/

http://java.sun.com/beans/

http://java.sun.com/products/ejb/

http://java.sun.com/products/jdbc/

How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** http://www.redbooks.ibm.com/

  Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

  Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders by e-mail including information from the IBM Redbooks fax order form to:

	e-mail address
In United States	usib6fpl@ibmmail.com
Outside North America	Contact information is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and Redbooks by accessing the IBM Intranet Web site at http://w3.itso.ibm.com/ and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access MyNews at http://w3.ibm.com/ for redbook, residency, and workshop announcements.

IBM Redbooks fax order form

Please send me the following:

Title	Order Number	Quantity

First name	Last name	
Company		
Address		
City	Postal code	Country
Telephone number	Telefax number	VAT number

☐ Invoice to customer number

☐ Credit card number

Credit card expiration date	Card issued to	Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Abbreviations and acronyms

ACF/VTAM	Advanced Communications Facility/Virtual Telecommunications Access Method
APAR	authorized program analysis report
API	application programming interface
APPC	Advanced Peer-to-Peer Communication
APPN	Advanced Peer-to-Peer Networking
ASCII	American Standard Code for Information Interchange
ASI	automated system initialization
ATM	asynchronous transfer mode
AIX	Advanced Interactive Executive
BMS	basic mapping support
CAE	Client Application Enabler
CD	compact disc
CDRM	cross-domain resource manager
CGI	Common Gateway Interface
CLI	call level interface
CICS	Customer Information Control System
COBOL	Common Business Oriented Language
CP	control point
CPU	central processing unit
CSD	CICS System Definition
CSI	Connectivity Systems, Inc.
CTG	CICS Transaction Gateway
CTCA	channel-to-channel adapter
CWS	CICS Web Support
DASD	direct access storage device
DB2	Database 2
DLBL	Disk Label
DL/I	Data Language 1
DNS	domain name system
DOS	Disk Operating System
DRDA	distributed relational database architecture
DVMRP	Distance Vector Multicast Routing Protocol
EBCDIC	Extended Binary Communication Data Interchange Code
ECI	external call interface
EJB	Enterprise JavaBean
EPI	external presentation interface
ESDS	Entry Sequenced Data Set
ESI	external security interface
FSU	Fast Service Upgrade
FTP	File Transfer Protocol
GIF	graphical user interface
GMT	Greenwich Mean Time
GPS	General Print Server
GUI	graphical user interface
HLASM	high-level assembler
HP-UX	Hewlett-Packard UNIX
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
I/O	input/output
IBM	International Business Machines Corporation
ICCF	Interactive Computing and Control Facility
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IPL	initial program load
IPSec	IP Security Architecture
ISO	International Organization for Standardization
ITSO	International Technical Support Organization
IUI	interactive user interface
JCL	job control language
JDBC	Java Database Connectivity
JSP	Java Server Page
JVM	Java Virtual Machine

KSDS	Key Sequenced Data Set
LAN	local area network
LLC	logical link control
LPD	Line Printer Daemon
LPR	Line Printer Requester
LU	logical unit
L2F	Layer 2 Forwarding
L2TP	Layer 2 Tunneling Protocol
MAC	media access control
MB	Megabyte
MPTS	Multiple Protocol Transport Services
MQI	message queue interface
MVS	Multiple Virtual Storage Operating System
NFS	Network File System
NNTP	Network News Transfer Protocol
NSF	National Science Foundation
ODBC	open database connectivity
OEM	original equipment manufacturer
OS/2	Operating System/2
OSA	Open Systems Adapter
OSI	Open Systems Interconnect
OS/390	Operating System for the System/390 platform
OS/400	Operating System for the AS/400 platform
PC	personal computer
PDA	personal digital assistant
PL/I	Programming Language 1
PPTP	Point-to-Point Tunneling Protocol
PTF	program temporary fix
PU	physical unit
RDO	Resource Definition Online
REXX	Restructured Extended Executor
RS/6000	IBM RISC System/6000
SAM	Sequential Access Method
SAP	service access point
SCM	supply chain management
SD	Solution Developer (formerly Independent Software Vendor (ISV))
SNA	Systems Network Architecture
SQL	Structured Query Language
SSCP	system services control program
SSI	Server Side Include
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol /Internet Protocol
TLBL	tape label
TLS	transport layer security
TSQ	temporary storage queue
UDB	Universal Database
UI	user interface
URL	Uniform Resource Locator
USSTAB	Unformatted System Services Table
VM	virtual machine
VM/ESA	Virtual Machine/Enterprise Systems Architecture
VPN	virtual private network
VSAM	Virtual Storage Access Method
VSE/ESA	Virtual Storage Extended/Enterprise Systems Architecture
VSE/POWER	Virtual Storage Extended/Priority Output Writers, Execution processor, and input Readers
VSE/VSAM	Virtual Storage Extended/Virtual Storage Access Method
VTAM	Virtual Telecommunications Access Method
WAN	wide area network
WARP	Workstation Asset Reduction Program
WML	Wireless Markup Language
WWW	World Wide Web
XML	Extensible Markup Language

Index

Symbols

+ symbol 42

Numerics

2-tier model 1, 3
2-tier solution 125
3270 bridge 153, 154
3270 terminal control application 4
3-tier model 1, 5

A

access to 3270 application 125
account self-service 9
account tracking 9
add generated applet 32
alias name 21
APPC connection 14, 68, 100, 115
APPLET tag 159
application server 2
APPNCOS parameter 109
authorization code 39

B

BMS 4
business integration with suppliers 9

C

CA-Top Secret 153
cellphone 156
CGI-BIN interface 160
CICS BMS panel 125
CICS connection definition 110
CICS preprocessor 79
CICS session definition 111
CICS Transaction Gateway 68, 70, 98, 114
CICS Transaction Gateway flow 99
CICS Transaction Server for VSE/ESA 153
CICS Universal Client 68
  customization 112
CICS Web Support 4, 153, 154
class file 85
class name 89
CLI (ODBC) 17
client technology 159
client tier 2
codepage 114
Common Gateway Interface (CGI) 3
connector 5, 6
connector architecture 5
consumer marketing 8, 15
conversion table DFHCNV 115
customization 114

D

database wizard 26
DB2 18
  database connection 20
  establish connection 39
  RMTUSERS parameter 18
  startup job 18
  system configuration 15
  table access 27
  table sort 28
  TCPPORT parameter 18
DB2 Client Application Enabler 19
DB2 Connect 19
DB2 DRDA server 15
DB2 preprocessor 79
DB2 Universal Database 16, 155
DB2JSTRT command 22
DFHCNV 115
DYNLU parameter 109
DYNPU parameter 110

E

e-business connector 2
e-commerce 155
encrypted data transmission 134
Enterprise JavaBean 11, 12, 161
Enterprise JavaBean container 161
Enterprise Storage Server 154
error page 40
export package 85, 89
extended data stream 126
external call interface (ECI 98
external presentation interface (EPI 98
external security interface (ESI) 98
external security manager 154

F

FFCGI1 program 142, 163
firewall 153
framework 1
FTP daemon 140

H

Host On-Demand 154
  client 63
  configuration options 59
  configuration parameter 65
  server 63
  session2.html applet file 63
HTML 159
HTML template 128
HTTP request 160
HTTP server 11, 37
Hypertext link 130

I

IBM Application Framework for e-business 1, 35
industry standard 3270 59
initiation queue 76
integration with service providers 9
IPSec protocol 134
IpServer 125, 134
  calling CGI program 141
  configuration file 135
  copy book D21IPARC 142
  D21IPARS program 143
  installation 135
  license key 135
  sample CGI program 135
ITSO system setup 13

J

Java 161
Java applet 159
Java Database Connectivity (JDBC) 17, 162
Java Server Page (JSP) 11, 37, 49, 160
Java servlet 160
Java Virtual Machine 35, 159, 161
JavaBean 12, 161
JavaScript 159
JDBC
  applet driver 17, 22
  applet implementation 17
  client 17
  server 17

L

L2TP protocol 134
LAN connection 104
local queue 76

M

MQPECHO CICS definition 79
MQPECHO sample program 79, 116
MQSeries 68, 69, 156
  environment 71, 81
  naming convention 72
  object definition file 73
  queue manager creation 72
  RUNMQSC utility 77
  server to server connection 72
  start listener program 78
  start sender channel 77
MQSeries Client for Java 68
MQSeries for Windows NT 68
MQSeries Integrator 68, 69, 90
  control flow 92
  Formats Dialog 91
  queue handling 90
  rules database 91
  Rules Dialog 91
  setup 92
  startup file 94
Multiprise 3000 154

N

net.Commerce 155
NETID parameter 109
NetObjects BeanBuilder 24
NetObjects Fusion 23, 50, 85, 140
network security 153
no data page 40
NODETYPE parameter 109

O

object-oriented programming language 161
OS/390 154

P

pervasive computing 156
plug-in 157
port number 20, 26, 78, 126, 135
PPTP protocol 134
proxy server 134
public key 134
publish on VSE/ESA 149
publish process 33
Publish Wizard panel 30

R

receiver channel 73
reference to external HTML source file 56
remote queue 73
reverse proxy server 134
RUNMQSC utility 77

S

screen display suppression 130
screen unique color mapping 130
screen unique virtual keyboard 130
Secured Socket Layer 134
SecureWay Firewall 153
SecureWay Host On-Demand 59
SecureWay Personal Communications 68, 100
  configuration 101
  device definition 103
  LAN connection 104
  local LU6.2 definition 107
  LU6.2 sessions 116
  node definition 101
  partner LU6.2 definition 107
security in 2-tier solution 134
security in 3-tier solution 134
sender channel 73
sendMessage Java code 87
server integration 5
server plugin 160
Server Side Include (SSI) 3
Server Side Logic 160
server side technology 159

servlet configuration 89
servlet control file 41
SNA connection 68
SNA LU2 File Transfer 126
SQL Query Bean 40
SQL wizard 53
subsystem storage protect feature 153
symbolic name of Web server 26

T

TCP/IP connection 14, 15, 20, 67
TCP/IP for VSE/ESA 125, 134
TCP/IP socket 17
TCP/IP startup 140
Template facility 132
temporary storage queue 141
TestECI program 121
thin client 159
TN3270 protocol 125
TN3270 server 125
Token-Ring adapter 101
transaction monitor 134
translation table 114
transmission queue 73, 76
Transport Layer Security 134
trigger information 79
trigger statement 128

U

UNIX mode 140
User Interface (UI) 16, 159

V

Virtual Private Network 134
Visual Composition tool 120
visual layout tool 12
visual programming environment 12
VisualAge for Java 12
VTAM APPN support 101
VTAM configuration book 106
VTAM startup book 108
VTAM startup parameter 106, 108
VTAM XCA major node definition 110

W

Web application server 2
Web server solution 3
Web/VSE-Host 125
  configuration file 126
  configuration file options 128
  customization 126
  installation 126
  license code 126
  sample template file 129
  sample WebScreen definition 131
WebScreen facility 125, 130
WebSphere Application Server 11, 37
WebSphere Enterprise Edition 154
WebSphere Studio 12, 35
  animated GIF designer 13
  applet designer 13
  create database query bean 38
  Database wizard 40
  NetObjects Fusion 13
  NetObjects ScriptBuilder 13
  Page Designer 46
  project creation 37
  project view 48
  publishing options 44
  servlet control file 43
  SQL wizard 39
  visual page designer 13
  Web art designer 13
  Wizard 13
  workbench environment 13
Web-to-Host e-business solution 125
wireless technology 156

IBM Redbooks review

Your feedback is valued by the Redbook authors. In particular we are interested in situations where a Redbook "made the difference" in a task or problem you encountered. Using one of the following methods, **please review the Redbook, addressing value, subject matter, structure, depth and quality as appropriate.**

- Use the online **Contact us** review redbook form found at http://www.redbooks.ibm.com/
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Document Number **Redbook Title**	SG24-5662-00 e-business Solutions for VSE/ESA
Review	
What other subjects would you like to see IBM Redbooks address?	
Please rate your overall satisfaction:	O Very Good O Good O Average O Poor
Please identify yourself as belonging to one of the following groups:	O Customer O Business Partner O Solution Developer O IBM, Lotus or Tivoli Employee O None of the above
Your email address: The data you provide here may be used to provide you with information from IBM or our business partners about our products, services or activities.	O Please do not use the information collected here for future marketing or promotional contacts or other communications beyond the scope of this transaction.
Questions about IBM's privacy policy?	The following link explains how we protect your personal information. http://www.ibm.com/privacy/yourprivacy/

SG24-5662-00

Printed in the U.S.A.

e-business Solutions for VSE/ESA

SG24-5662-00