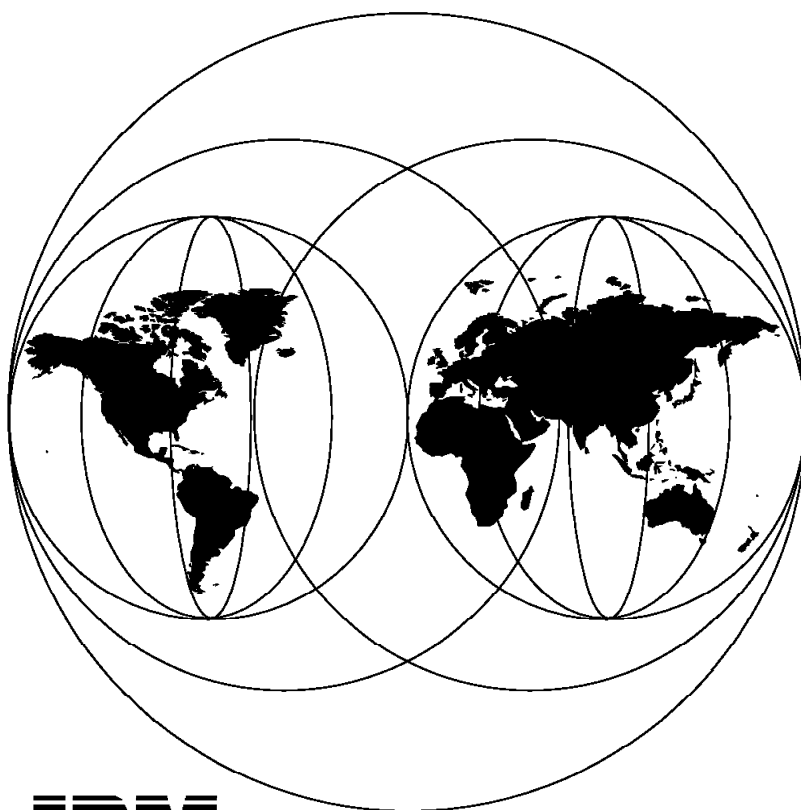


**Secure Electronic Transactions:
Credit Card Payment on the Web in Theory and Practice**

June 1997



**International Technical Support Organization
Raleigh Center**



International Technical Support Organization

SG24-4978-00

**Secure Electronic Transactions:
Credit Card Payment on the Web in Theory and Practice**

June 1997

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix D, "Special Notices" on page 311.

First Edition (June 1997)

This edition applies to Version 1 of the CommercePOINT products for SET (exact program names and numbers have not been finalized at the time of writing).

Warning

This book is based on a pre-GA version of a product and may not apply when the product becomes generally available. It is recommended that, when the product becomes generally available, you destroy all copies of this version of the book that you have in your possession.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HZ8 Building 678
P.O. Box 12195
Research Triangle Park, NC 27709-2195

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.

© **Copyright International Business Machines Corporation 1997. All rights reserved.**

Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

Figures	vii
Tables	xi
Preface	xiii
The Team That Wrote This Redbook	xiii
Comments Welcome	xiv

Part 1. The Theory	1
Chapter 1. Payment System Evolution and the World Wide Web	3
1.1 Anatomy of a Payment System	3
1.2 Commerce on the Web	4
1.2.1 Why the Web Is a Good Place to Sell Things	5
1.3 Security Threats and Responses	6
1.4 Web Payment Systems	7
1.4.1 The Origins of SET	7
1.4.2 Micropayments and Digital Cash	8
1.4.3 Smart Cards	9
1.4.4 JEPI	9
1.4.5 IBM Directions for Web Payments, SuperSET	9
Chapter 2. SET Roles and Responsibilities	11
2.1 SET Roles	11
2.2 SET Relationships	12
2.3 The Limits of SET	15
Chapter 3. SET Payment Protocols	17
3.1 Business Requirements	17
3.1.1 Confidentiality	18
3.1.2 Authentication	18
3.1.3 Integrity	18
3.1.4 Interoperability	18
3.2 SET Message Formats in General	19
3.2.1 SET Encrypted Message Example	19
3.2.2 Dual Signature	22
3.2.3 Dual Signature Example	22
3.3 SET Payment Transaction Processes	25
3.3.1 Purchase Process	25
3.3.2 Payment Authorization Process	34
3.4 Payment Capture Process	43
Chapter 4. SET Certification	51
4.1 Concept	51
4.2 Certificates	51
4.2.1 Cardholder Certificates	52
4.2.2 Merchant Certificates	52
4.2.3 Payment Gateway Certificates	52
4.3 Hierarchy of Trust	52
4.3.1 Root Key	53
4.4 SET Protocols For Issuing Certificates	55

4.4.1 Cardholder Certification Process	56
4.4.2 Merchant and Acquirer Certification Process	69

Part 2. The Practice	81
---------------------------------------	-----------

Chapter 5. Introduction	83
5.1 Lab Environment	83
Chapter 6. The Cardholder's View	85
6.1 Installing CommercePOINT Wallet	85
6.1.1 Preparing Netscape Navigator	87
6.2 Defining Credit and Debit Cards	88
6.2.1 Stand-Alone Card Definition	90
6.2.2 Online Card Definition and Certification	91
6.3 Shop Until You Drop	95
Chapter 7. The Merchant	101
7.1 Net.Commerce Components: Overview, Interfaces and Interactions	101
7.1.1 Components Overview	101
7.2 Installing the SET Extensions	107
7.2.1 Net.Commerce SET Extension Components Interactions	108
7.2.2 Installing the SET Extensions on AIX	110
7.2.3 Post-Installation DB2 Checking	113
7.2.4 Sample Acquirer	118
7.3 Configuring the Payment System	119
7.3.1 Key Generation and Certificate Request	119
7.3.2 SET Tables Configuration	120
7.3.3 Macro Customization	123
Chapter 8. The Certificate Server	139
8.1 IBM Registry for SET Components: Overview, Interfaces and Interactions	139
8.1.1 The Administrator	140
8.1.2 The Approver	141
8.1.3 The IBM Registry for SET Server	141
8.2 Cardholder Registration Scenario	141
8.3 Preparing the System	143
8.3.1 Checking System Memory	143
8.3.2 Checking the AIX System Level	143
8.3.3 Checking IBM DATABASE 2 Installation	144
8.3.4 Checking the Free Disk Space	144
8.4 Installing the IBM Registry for SET Product	144
8.4.1 Post Checking	146
8.5 Configuring the CA Server	147
8.5.1 Planning for Configuration	148
8.5.2 Modifying the Configuration File	149
8.6 Installing the CA Certificate	153
8.6.1 Key and Certificate Hierarchy Required for SET	153
8.6.2 Setting Up a CA Hierarchy for Test Purposes	155
8.7 Starting the CA Server	156
8.8 Creating Administrator and Approver Accounts	159
8.8.1 Creating a New Administrator Account	160
8.8.2 Creating an Approver Account	160
8.9 Creating a CCA WWW server	161
8.9.1 Creating an HTML Page that References the Wakeup Server	161

8.9.2 Replacing the Wakeup Process with a CGI Script	163
8.10 Signing Certificates with IBM Registry for SET	164
8.10.1 Producing Test Certificates for Merchant and Payment Gateway	165
8.10.2 Generating Cardholder Certificates	166
8.10.3 Implementing a User Exit	166
Chapter 9. The Payment Gateway	171
9.1 CommercePOINT Gateway: Overview, Interfaces and Interactions	171
9.1.1 The Payment Gateway Transaction Monitor	172
9.1.2 Payment Gateway Application	173
9.1.3 The Payment Gateway Listener	174
9.1.4 The Payment Gateway Legacy Application	174
9.1.5 Payment Authorization Scenario	175
9.2 Preparing the System	177
9.2.1 Verifying Prerequisites	177
9.2.2 Planning the Installation	177
9.3 Installing the Payment Gateway	178
9.4 Configuring the Payment Gateway	179
9.4.1 Create Server DB2 Instance	179
9.4.2 Create DB2 Database and Database Objects	181
9.4.3 PGTM Environmental Configuration	182
9.4.4 Payment Gateway Application Registration	186
9.4.5 Configure the Payment Gateway	192
9.4.6 PGTM Initialization and Termination	196
9.5 Customizing the Translation Exit	198
9.5.1 Sample Translation Exit Code	199

Part 3. Installing and Configuring Net.Commerce 213

Chapter 10. Installing Net.Commerce	215
10.1 Preparing the System	215
10.1.1 AIX System	215
10.1.2 Windows NT	222
10.2 Installing Net.Commerce	223
10.2.1 AIX System	223
10.2.2 Windows NT	223
10.3 Post-Installation Steps	226
10.3.1 Building Security Key Rings	226
10.3.2 Verifying That the Installation and Configuration Is Successful	236
10.3.3 Controlling the Net.Commerce Components	236
Chapter 11. Creating an Online Store with Net.Commerce	241
11.1 Configuring the Basics of Net.Commerce	241
11.1.1 Changing the Database	248
11.1.2 Creating and Configuring a Store	264
11.1.3 Customizing Your Macros	277
Appendix A. Certification Examples	283
A.1 Using setca_certmgr to Create a Certificate Hierarchy	283
A.2 Using setbrandca to Create a Certificate Hierarchy	289
A.3 Displaying the Contents of a CA Key Database	291
Appendix B. Cryptographic Processes	299
B.1 Symmetric Key Encryption	299

B.1.1 Characteristics of Symmetric Key Algorithms	300
B.2 Public Key Encryption	301
B.2.1 Characteristics of Public Key Encryption	302
B.3 Secure Hash Functions	302
B.4 Combinations of Cryptographic Techniques	303
B.5 Public Key Certificates	303
Appendix C. Secure Sockets Layer	305
Appendix D. Special Notices	311
Appendix E. Related Publications	313
E.1 International Technical Support Organization Publications	313
E.2 Redbooks on CD-ROMs	313
E.3 Other Publications	313
How to Get ITSO Redbooks	315
How IBM Employees Can Get ITSO Redbooks	315
How Customers Can Get ITSO Redbooks	316
IBM Redbook Order Form	317
Index	319
ITSO Redbook Evaluation	321

Figures

1.	Internet Host Growth Trends	4
2.	Narrowcasting	6
3.	Contractual Relationships	13
4.	Administrative Relationships	14
5.	Operational Relationships	15
6.	Encryption Example	20
7.	Dual Signature Example	23
8.	Summary of the Purchase Process	26
9.	Cardholder Initiates Purchase	27
10.	Merchant Sends Initiate Response	28
11.	Cardholder Receives Initiate Response	29
12.	Cardholder Sends Purchase Request	30
13.	Merchant Verifies Purchase Request Message	32
14.	Merchant Sends Purchase Response	33
15.	Cardholder Receives Purchase Response	34
16.	Payment Authorization Diagram	35
17.	Merchant Requests Authorization	37
18.	Payment Gateway Receives Authorization Request	39
19.	Payment Gateway Sends Authorization Response	41
20.	Merchant Receives Authorization Response	43
21.	Payment Capture Diagram	44
22.	Merchant Payment Capture Request	45
23.	Payment Gateway Receives Capture Message	47
24.	Merchant Sends Capture Response	48
25.	Merchant Receives Capture Response Message	49
26.	Certification Hierarchy (Original Design)	53
27.	Root Certificate	54
28.	Replacing the Root Certificate	55
29.	Cardholder Registration	56
30.	Cardholder Initiates Registration	57
31.	CA Sends Initiate Response	58
32.	Cardholder Receives Initiate Response	59
33.	Cardholder Requests Registration Form	60
34.	CA Receives Registration Form Request Message	61
35.	CA Sends Registration Form	62
36.	Cardholder Receives Registration Form	63
37.	Cardholder Requests Certificate	64
38.	CA Receives Request	66
39.	CA Sends Certificate	67
40.	Cardholder Receives Certificate	68
41.	Merchant Registration Diagram	69
42.	Merchant Initiates Request	70
43.	CA Sends Registration Form	71
44.	Merchant Receives Registration Form	72
45.	Merchant Requests Certificate	73
46.	CA Receives Certificate Request	74
47.	CA Sends Certificates	76
48.	Merchant Receives Certificates	78
49.	Lab Environment	84
50.	Initial Installation Logo Screen	86
51.	CommercePOINT Wallet Folder in Windows 95	86

52.	Defining Helper Applications in Netscape Navigator	87
53.	User Data Question Pop-Up	88
54.	Prompt For a New User ID	89
55.	Password For New User ID	89
56.	Initial Cardholder Screen	90
57.	Add Card Dialog with Default Brand List	91
58.	Web Page with Link to SET Certificate Server	91
59.	Add Card Dialog with Pass Brand Added	92
60.	One Card in the Wallet	93
61.	Brand Policy Screen	94
62.	Certificate Registration Form	95
63.	Completed Certification	95
64.	Mall Home Page	96
65.	Shopper Registration Form	97
66.	Excellent Taste in Redbooks	97
67.	Shopping Cart Contents	98
68.	Order Details Screen	98
69.	Electronic Wallet Request to Confirm Payment	99
70.	Check Merchant Distinguished Name Information	100
71.	Payment Successful Message	100
72.	Net.Commerce Components	102
73.	IBM Internet Connection Secure Server Interfaces and Interactions	103
74.	Net.Commerce Server Interfaces and Interactions	105
75.	Net.Commerce SET Extension and Net.Commerce	108
76.	Net.Commerce SET Extension Daemon StartUp Sample Script	112
77.	SET Order Failure Task	115
78.	Order Information Page	116
79.	Order List	117
80.	Marking for Capture	118
81.	Acquirer StartUp Sample Script	119
82.	Order Details Store Page	124
83.	ITSO Corner Order Details Page	125
84.	itso_orderdsp.d2w Macro Source File Extract	126
85.	Task Management Form	127
86.	Macro Assignment Form	128
87.	Order Details: ITSO Store	129
88.	itso_orderdsp.d2w Macro File - Extract #1	130
89.	itso_orderdsp.d2w Macro File - Extract #2	132
90.	ITSO Order Failure Page	133
91.	itso_ordfailed Macro File	135
92.	IBM Registry for SET Interfaces and Interactions	140
93.	Cardholder Registration Flow	142
94.	Install and Update Software from LATEST Available (1 of 2)	145
95.	Install and Update from LATEST Available Software (2 of 2)	146
96.	Global Values in IBM Registry for SET Configuration File	149
97.	Cardholder CA Definition in Configuration File	150
98.	Merchant and Payment Gateway Definitions in setca.conf	152
99.	Hierarchy of Key Databases, Keys, and Certificates for SET	154
100.	Initial Prompt and Menu for setca_certmgr	155
101.	setadmin Logon Panel	157
102.	setadmin Main Menu	158
103.	Administration Activation - Password Panel	158
104.	Changing ibmset User Password	159
105.	Creating an Additional Administrator Account	160
106.	Creating an approver account	161

107. Registration Initiation Message Example	162
108. MIME-Encapsulated Registration Initiation Message Example	162
109. CCA Home Page	163
110. CCA Home Page Source Code	163
111. CGI Script to Wake Up the Cardholder Plug In Module	164
112. AuthChk Exit Sample Source Code	168
113. CommercePOINT Gateway Interfaces and Interactions	172
114. CommercePOINT Gateway Interfaces and Interactions	175
115. Sample rc.db2 Startup Script	185
116. Register TSM	185
117. Register TSM (Continued)	186
118. Register TSM (Continued)	186
119. Register Payment Gateway Interface	188
120. Register Payment Gateway Interface	188
121. andRegister Main Menu, Select Option 3	189
122. Two TMSs Registered, Select The Payment Gateway, pgw_tsm	190
123. One Application Registered to the TSM, Select It	190
124. Initial Application Details Screen, Select Interfaces	191
125. One Local Interface Defined	191
126. Interface Details, Select Communications Protocol	192
127. UNIX Socket Definition for Local Application	192
128. pgconfig Main Menu	193
129. Add a Payment Gateway	193
130. Select 1 for Basic Configuration	194
131. Select 2 to Update the TCP/IP Configuration	194
132. Select 1 to Change the TCP/IP Port	194
133. Enter Your Chosen Port Number	195
134. Changing a Payment Gateway	196
135. Deleting a Payment Gateway	196
136. Example of PGTM System Log	197
137. Checking that LAD is Active	198
138. Example of PGTM System Log at Termination	198
139. Payment Gateway Legacy Application Sample Source Code	200
140. Checking the Size of the /home Directory	215
141. Increasing the Size of the /home Directory	216
142. Checking the Paging Space Size	217
143. Checking the Paging Space Size	218
144. Increasing the Size of an Existing Paging Space	219
145. Creating a New Paging Space	219
146. Creating a DB2 Group	220
147. Creating a DB2 User ID	221
148. Checking If httpd Is Running	226
149. Creating Keys	227
150. Creating a Key	228
151. Creating a Key	228
152. Creating a Key	229
153. Creating a Key	230
154. Changing the Current Ring	231
155. Receiving the Certificate	232
156. Receiving the Certificate	233
157. Receiving the Certificate	234
158. Net.Commerce Server Manager Main Page	242
159. Specifying the Database Name	243
160. Administrator User ID and Password	244
161. Confirmation Page	245

162. Database Management	246
163. Confirmation Page	247
164. Server Control Form	248
165. Shutting Down the Server	249
166. Shutting Down the Server	250
167. Checking to See If Net.Commerce Is Working with the Correct Database	251
168. Net.Commerce Administrator Main Page	252
169. Site Manager Main Page	253
170. Mall Front Home Page	254
171. Mall Header	256
172. Mall Footer	256
173. Mall Front Configuration	257
174. Defining Categories	258
175. Creating a Site Administrator User	259
176. Creating a Store Administrator User	260
177. Creating a Store Administrator User	261
178. Creating Store Records	262
179. Creating a Shipping Provider	263
180. Store Manager Main Page	265
181. Store Front Home Page	266
182. Store Header	268
183. Store Footer	268
184. Store Information	270
185. Product Categories	271
186. Shipping Service	274
187. Calculation Method	275
188. Shipping Details	276
189. db2www.ini Sample File	278
190. Change Registration Page	279
191. Selecting the Task for an Assignment	280
192. Assigning a Macro	281
193. setbrandca Dialog Example	290
194. cmsutil Command Output Example	292
195. Symmetric Key Encryption	299
196. Public Key Encryption	301
197. SSL Handshake Process	306
198. SSL Session Indicator in Netscape	307
199. Certifying Authorities in Netscape Navigator	308
200. Server Certificate Signed by Unknown CA	309
201. Option to Add Trusted Server	310

Tables

1. Boundaries of SET	16
2. Net.Commerce Default Order Macros	123
3. SETSSTATCODE Values	131
4. Net.Commerce SET Extension Order Macro	133
5. IBM Registry for SET Files Location	147
6. Key Databases Created by setbrandca	156

Preface

Secure Electronic Transactions (SET) is a group of protocols designed for safe credit card payments over the World Wide Web. This redbook explains the operation of the standard and illustrates it using IBM SET-compliant products, including the Net.Commerce server suite.

This redbook will help readers gain a theoretical understanding of what SET is and a practical knowledge of how it is implemented.

There are three parts to the book:

- Part 1, "The Theory"

This section introduces the Secure Electronic Transactions (SET) environment and defines the roles of the users and providers that it serves. It also discusses other approaches to electronic payments including some product directions. The transaction flows in SET are then discussed in detail.

- Part 2, "The Practice"

This section describes how the redbook project team implemented early versions of the IBM products that deliver the SET function. The text includes installation, customization and sample coding for programming exits.

- Part 3, "Installing and Configuring Net.Commerce"

This section describes the installation and customization of the Net.Commerce server, which is a prerequisite for the SET payment processes described in Part 2.

Some knowledge of Internet security and Web server operations is assumed.

The Team That Wrote This Redbook

This redbook was produced by a team of specialists from around the world working at the Systems Management and Networking ITSO Center, Raleigh.

Rob Macgregor is a technical specialist at the Systems Management and Networking ITSO Center, Raleigh. He writes extensively and teaches IBM classes worldwide on aspects of Internet security and distributed systems management. Before joining the ITSO three years ago, Rob worked in the UK as a specialist in systems and network management.

Catherine Ezvan is a Systems Engineer in the AIX National Support and Services Center in France. She graduated as an Electronics Engineer from the Ecole Supérieure d'Electronique de l'Ouest in Angers, France. She has five years of experience in the fields of TCP/IP networking, AIX administration and Web site administration. She has worked at IBM for four years.

Luis Fernando Liguori is a Security Specialist in Brazil. He has two years of experience in the security field. He has worked at IBM for four years. His areas of expertise include LAN products, TCP/IP, firewalls and cryptography. His work involves advising customers on the design and implementation of network security solutions.

JaeSool Han is an Advisory Specialist in IBM Korea. He has three years of experience in the Network computing field. He has worked at IBM for seven years. His areas of expertise include Web-technology, e-commerce, multimedia and development of communication emulator, spread sheet and word processor software.

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

David Boone and Linda Robinson
Systems Management and Networking ITSO Center, Raleigh.

John Wilson and his team at the SET Integration Test lab in
IBM Research Triangle Park.

Gerry Mcfadden and Juha Nevalainen
of the IBM Installation Support Centre, Hursley, UK.

Niklas Montonen of IBM Finland and
Peter Weinert of IBM Germany for their assistance with the IBM Registry for SET
chapter.

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 "ITSO Redbook Evaluation" on page 321 to the fax number shown on the form.
- Use the electronic evaluation form found on the Redbooks Home Pages at the following URLs:

For Internet users <http://www.redbooks.ibm.com>

For IBM Intranet users <http://w3.itso.ibm.com>

- Send us a note at the following address:
 redbook@vnet.ibm.com

Part 1. The Theory

Chapter 1. Payment System Evolution and the World Wide Web

The explosion of public interest in the Internet in the last few years has been truly phenomenal. Net awareness has reached into every corner of society; people from pre-school children to their grey-haired grandparents have embraced the Internet and have become fluent in the language of URLs and e-mail IDs. This coming together of people from around the world has created a new community that crosses social and national boundaries. Where there is a community there is a potential for trade, and there is no shortage of traders setting up their stalls in this promising new marketplace.

In this book we look at one of the technologies that will make the Internet marketplace a reality, *Secure Electronic Transactions* (SET). First, however, we look at the factors that make credit card transactions on the Web an attractive proposition, despite the natural concerns about security.

1.1 Anatomy of a Payment System

We can reasonably assume that the trading of goods has gone on since the dawn of mankind. We can imagine primitive hunters/gatherers wishing to trade their surplus catch for animal skins, or stone implements or other goods. However, the concept of a *payment system*, rather than simple bartering of goods, came much later. The first recorded use of a form of money was in Mesopotamia, about 3000 years ago.

Many forms of money followed. In the more primitive cases, the money itself had intrinsic value, being made of a precious metal for example. As time went by the payment systems became more sophisticated and the money that changed hands became something that *promised* payment, instead having real value itself. A credit card payment takes this one stage further. When you pay with plastic, you are making not one promise, but two: one that tells the seller that the credit card company will pay for the purchase and one to the credit card company that says you will eventually pay the bill.

Regardless of the type of payment system, it relies on some fundamental principles:

Trust At the basic level, the buyer has to be able to trust that the seller will actually deliver the goods. Then the seller has to trust that the promise represented by the payment will be realized. When the payment was in the form of gold or silver, trust could be established by rubbing it on a touchstone to check for authenticity. In the case of paper money, watermarks and special printing perform the same function. For credit cards the original mechanism for establishing trust was to compare the signature of the buyer with that on the back of the card. However, with the increase in credit card fraud it has become increasingly common to do an online authorization check, even for small-value payments. Electronic credit card payments in SET use a variation of both techniques, employing digital signatures as well as online authorization checking.

Security The transaction needs to take place in a safe way, so that the payment is not stolen in transit. For cash payments, this is just a matter of physical security. For credit card payments, it is more

complex. As the buyer, you want to be sure that your credit card details are not stolen. As the seller, you want to be sure that the payment is transferred without interference. In SET, cryptographic techniques are used to protect against these risks.

In the best of all possible worlds, all buyers and sellers would be completely trustworthy and all transactions would be totally secure. In reality, any transaction has some degree of risk attached and the payment system attempts to minimize that risk with a response that is appropriate for the value of the payment and the dangers of the environment in which it takes place. For cash payments in an all-night convenience store the appropriate response may be a drop safe and a big, ugly dog. For a credit card payment on the World Wide Web the appropriate response is provided by the cryptographic processes of SET.

1.2 Commerce on the Web

The growth of the Internet and World Wide Web usage has been fueled by increasing commercialization of the network. Figure 1 shows the increasing numbers of hosts under particular high-level domain names over the past few years. You can clearly see the shift from academic sandbox to worldwide marketplace, reflected in the rate at which .com hosts have increased compared with .edu.

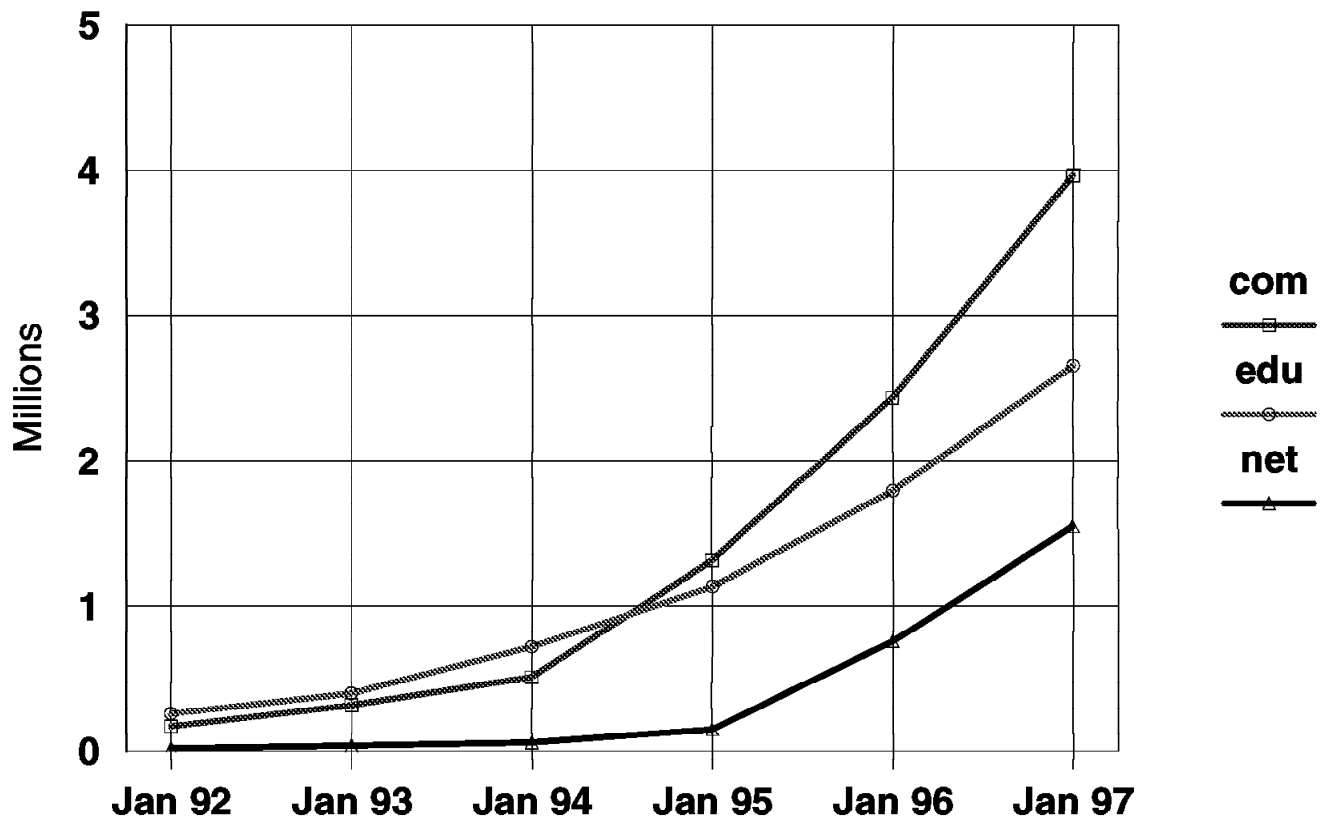


Figure 1. Internet Host Growth Trends. This diagram is based on the biannual domain survey published by Network Wizards at <http://www.nw.com>.

1.2.1 Why the Web Is a Good Place to Sell Things

Many companies are using their World Wide Web servers as a method to improve their corporate image and disseminate information. In fact, any organization that does *not* have a Web site is viewed as suspiciously old-fashioned.

Most commercial uses of the Web are safe and effective ways to improve customer service. For example, product information and customer support applications can improve responsiveness and ease the load on the public relations department (and improve customer perception too, assuming that the customer prefers watching a Web page loading to listening to a voice on the telephone saying that “All service representatives are currently busy.”).

When you start to think about applications that involve money, questions of security come up and you may wonder if the risk is worth taking. The answer to this question is partly to be found in Figure 1 on page 4. The growth of Internet access providers (IAPs), represented by the .net domain qualifier, has been even more meteoric than that of .com. These hosts support a whole group of network users that did not previously exist. This has caused a demographic shift of the user population, towards a group with serious purchasing power. In fact, if we paint a picture of the “average” Web user, we find the kind of person that consumer marketing specialists dream about:

- 33 years old
- Lives in the U.S.A or northern Europe
- Household income between 50,000 and 75,000 US dollars per year
- College educated

NUA Internet Marketing at <http://www.nua.ie> has a good summary of several Internet user surveys if you are looking for more information about this population.

Not only does the Web deliver you an attractive user population to sell goods and services to, it also provides ways to target specific groups of consumers within that population. Imagine you are passionately interested in cheese. To keep up to date with happenings in the world of cheese without using the Web, you would have to invest a lot of effort reading periodicals and visiting specialist stores. Alternatively, a simple search on one of the Web index sites will yield a plethora of cheese-related information, without having to leave your seat (see Figure 2 on page 6). For a vendor, this promises low-cost access to the one person in 10000 who is likely to buy your product. This kind of *narrowcasting* is what makes the Web, potentially, a very efficient place to promote and sell products.

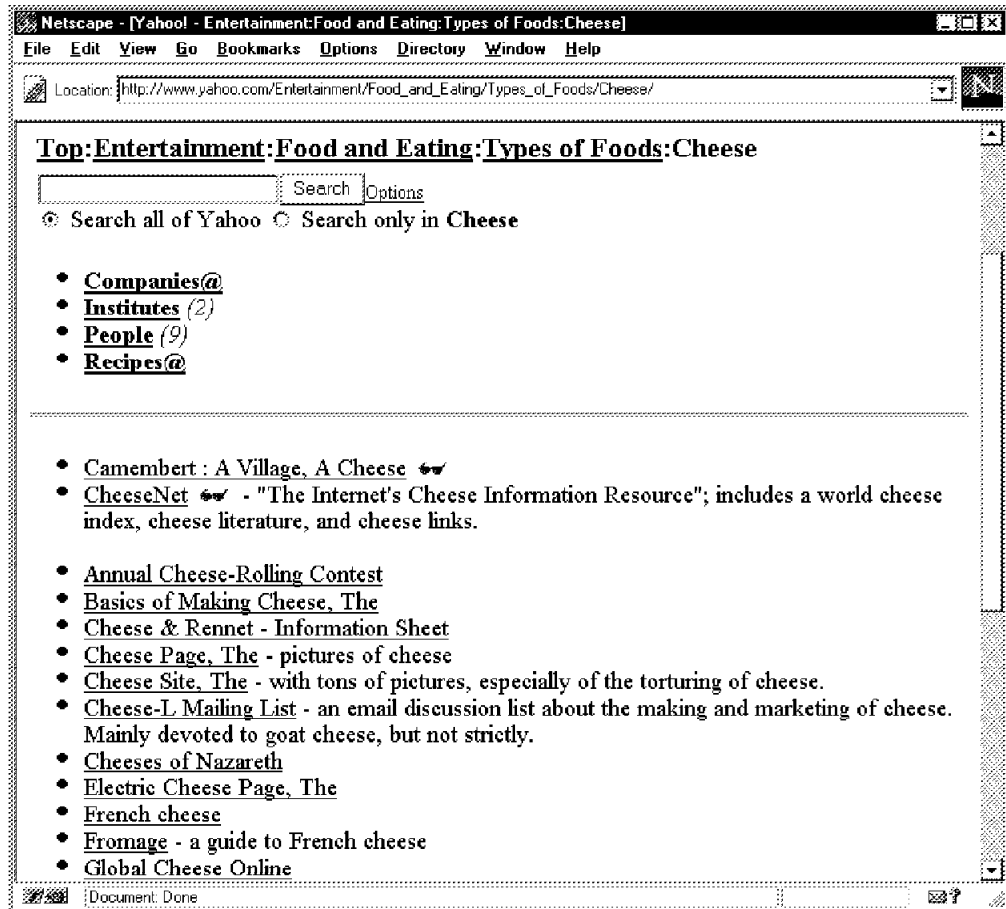


Figure 2. Narrowcasting. This shows the result of a simple search on the Yahoo Web index site (www.yahoo.com).

1.3 Security Threats and Responses

A number of surveys and reports have identified the lack of a secure payment scheme as the main inhibitor of Web commerce. The problem lies in the nature of the Internet itself. It was conceived as a way to link different networks together, providing global public access with minimal central control and bureaucracy. It has achieved these objectives very successfully; however, security suffers as a result.

We are all familiar with the exploits of hackers who try to break into Web sites and other Internet hosts. Often these attacks are more a form of network joyriding than serious attempts to damage a system or organization. However, if we plan to send credit card information across the Internet, we are providing a very visible and potentially lucrative target for them to go after. There are a number of modes of attack to consider:

Eavesdropping Owing to the nature of the Internet you have to assume that every message can be intercepted. This means that any message that contains sensitive information must be encrypted and the encryption mechanism must be strong enough to withstand a realistic attack. SET uses strong encryption technology to combat this threat.

Impersonation If you are purchasing goods or services online, you want to be assured that the organization you are sending your credit card information to is really who they claim to be. Conversely, if someone steals your credit card details, you don't want them to be able to run up charges on your account. SET uses digital signatures to combat this threat.

Trojan Horse While it is important that your credit card information is safe in transit and is going to the right server, it is equally important that it is not stolen when it reaches the server. Web servers are exposed to attack and it is possible for an attacker to place a program on the system to monitor traffic. As a cardholder, you have no idea whether the server has been breached in this way. SET overcomes this problem by using an innovative cryptographic approach (discussed in Chapter 3, "SET Payment Protocols" on page 17).

As we go deeper into the SET processes, it will become apparent how it counters different types of threat.

1.4 Web Payment Systems

SET is not the only electronic payment system designed for the World Wide Web. It is, however, emerging as the only significant standard for credit card transactions. In this section we give a brief history of the origins of SET, and also discuss other payment approaches.

1.4.1 The Origins of SET

Banks and financial institutions have had networks for electronic payment processing for many years. These networks connect highly secure, trusted computer systems, using dedicated links and powerful cryptographic hardware. A number of international standards exist to define the protocol for messages exchanged over the network.

The challenge for Internet credit card processing lies in producing a scheme that can provide adequate protection at a reasonable cost without compromising trust in any of the existing systems.

During 1995, various financial organizations and technology companies formed a number of alliances aimed at producing standards for credit card payment. This was a confusing time, with a number of competing standards and consortia. The technical community would probably still be arguing the merits of one solution or another, but the two largest credit card companies, Visa and Mastercard, realized that nothing would happen without a globally accepted standard. They joined forces with the key software companies to produce a single proposal, SET.

SET is based on ideas from previous proposed standards and is also heavily influenced by *Internet Keyed Payment Protocols* (iKP), which is the result of research carried out at the IBM Zurich Laboratory.

Other credit card payment systems do exist, but they are generally not aimed at the broad market, as SET is. For example, First Virtual Internet Payments System (FVIPS), operated by First Virtual Holdings Inc. is a scheme by which the prospective buyer registers credit card details with First Virtual and receives a personal identification number (PIN). The buyer can then use the PIN in place of

a card number at any merchant that has an account with First Virtual. Payment details must be confirmed by e-mail before any purchase is completed. Although this scheme has been successful it is limited due to the requirement for both buyer and seller to be affiliated with the same service. SET more closely follows the model of normal credit card payments, in which the only relationship between the organization that issues the card and the one that processes the purchase is that they subscribe to the same clearing network.

1.4.2 Micropayments and Digital Cash

Credit and debit cards are not the answer to all payment requirements. For small payments, the process of authorization and funds capture is too much of an overhead. Exactly what constitutes a small payment is an open question, but the consensus seems to be that anything under \$10 U.S. falls into that category.

Apart from inefficiency, there is one other aspect of credit card payments that sometimes causes concern. Any payment made by credit card can be linked directly back to you, the cardholder. Even if you are not concerned about the way this may infringe on your privacy in general, there are probably some payments you would rather the rest of the world did *not* know about.

Out in the real world beyond the edge of your browser, the answer to these dilemmas is to pay with cash. On the Web there are a number of initiatives that provide *electronic cash*: payment methods that mimic coins and notes. The methods differ in a number of ways. Some of them use digital coins: tokens that represent some small monetary amount that are authenticated by the digital signature of a bank. Others are closer to the credit card model, with a third party that manages the cash on behalf of a customer.

Here is a sample of some electronic cash schemes:

DigiCash This is the largest electronic cash scheme, based on electronic coins. It has a large number of subscribers, both buyers and merchants, and is supported by a number of banks. It uses an innovative blind signature scheme to protect the anonymity of the buyer. DigiCash is located at <http://www.digicash.com>.

NetBill This is a scheme developed at Carnegie Mellon University. In this case the cash is not held directly by the buyer, but by a NetBill server. It is primarily designed for delivering for-fee data content. When the buyer elects to buy the data or service, the seller sends the data in an encrypted form. It also sends a billing request to the NetBill server. If there are sufficient funds in the buyer's account, the server sends the buyer the key to unlock the data. If the buyer accepts, the cost is deducted from his or her account. Details of NetBill can be found at <http://www.netbill.com>.

Minipay This is a scheme proposed by IBM research. It is similar in some ways to NetBill, but its unique feature is that for small payments there is no need for the seller to request funds from the server that holds the account. Each buyer has a daily spending limit and, as long as it is not exceeded, the seller can be relatively sure that the bill will be paid. The advantage of this scheme is faster, lighter transactions, at the cost of a small additional risk. Minipay is described at http://www.ibm.net.il/ibm_il/int-lab/mpay.

1.4.3 Smart Cards

In all of the electronic cash examples, a cash balance is maintained on a computer, in some kind of electronic wallet or vault. However, this ties the transaction to a specific machine. Smart cards offer an alternative, by allowing electronic money to be stored in a portable medium.

In most of the payment protocols (including SET) the buyer has a public key pair, which is usually held in a file on his or her machine. Again, smart cards offer a solution whereby the keys can be held in a very secure, but portable form.

The number of applications of smart cards is increasing at a high rate, not only for payment schemes but also as security keys and as repositories of personal data. So far, these have mostly been based on private networks, because there are relatively few Internet devices with card readers. However, as the cost of card readers decreases, we can expect many of these schemes to start exploiting the public global access offered by the Internet.

1.4.4 JEPI

The emergence of a single standard for credit card payments, SET, is a very positive development for Web payments. However, as the previous sections have shown, there are many situations in which SET is not appropriate, and many other payment systems that browser and server software needs to accommodate.

In fact this diversity implies two requirements:

1. Electronic wallet and till technology that can handle multiple payment types
2. A negotiation protocol for client and server to determine what payment options they share

In real life, we take this latter protocol for granted. It goes something like this:

Buyer: Do you accept American Express?

Seller: No, we only take Mastercard and Visa.

Buyer: How about a personal check?

Seller: (laughs) That's very funny.

Buyer: I'll have to pay in cash then.

Seller: No problem, so long as it's in small-denomination used bills.
(etc.)

In cyberspace, the same exchange has not yet been finalized, but a project called JEPI (Joint Electronic Payments Initiative) is working hard to define the protocol. This is a combined effort of CommerceNet and the World Wide Web Consortium (W3C). You can find out more about JEPI at <http://www.w3.org/pub/WWW/Payments/jepi.html>.

1.4.5 IBM Directions for Web Payments, SuperSET

Having delivered products and services that cover all of the roles and functions in the SET framework, IBM development is working to expand the product set to embrace any other payment method. This development effort, known internally as SuperSET, will deliver electronic wallet and electronic till software that provides a number of interfaces to allow other payment modules to be easily integrated. It will also provide protocol negotiation capability, including JEPI, as soon as it is finalized.

Chapter 2. SET Roles and Responsibilities

In any credit card transactions there are a number of parties involved, all of whom have some relationship with each other. For Internet payments, SET defines the way that these relationships are carried on.

2.1 SET Roles

SET defines a number of different roles, some that are intuitively obvious, others that are more confusing:

Cardholder	This is you or me, the average person in the street with a wallet full of credit cards.
Merchant	A person or organization that has goods or services to sell to the cardholder.
Issuer	This is the company that provides the cardholder with the credit card in the first place. Ultimately it is the issuer that is responsible for the payment of the debt of the cardholder. In other words the issuer balances the risk of a cardholder defaulting against the income from interest payments. Issuers are financial institutions, such as banks. There is no reason why the cardholder should have any relationship with the issuer except for the credit card account, but in practice most cardholders have at least one card from the bank that holds their checking account.
Brand	Brand recognition and loyalty are key to the marketing of credit cards. Some brands are owned by a single financial organization which is also the card issuer. Other brands are owned by <i>bankcard associations</i> , consortiums of financial institutions that promote and advertise the brand, establish operating rules and provide a network for payment authorizations and fund transfer. SET provides controlled access to these networks from the Internet.
Acquirer	This is an organization that provides card authorization and payment capture services for merchants. A merchant will normally want to accept more than one credit card brand, but does not want to have to deal with multiple bankcard associations. They achieve this by using the services of an acquirer. These services include such things as verbal or electronic telephone authorization support and electronic transfer of payments to the merchant's account. Now the services can include SET protocol support as well. Acquirer services are paid for by the merchant in the form of a small percentage charge on each transaction.

Payment Gateway

This is a function provided by or on behalf of an acquirer. (It is more accurately called the *acquirer payment gateway*.) The payment gateway interfaces between SET and the existing bankcard association networks for authorization and capture functions. To put it another way; the payment gateway acts as a *proxy* for the bankcard network functions.

Certification Authority (CA)

This is a function that provides public key certification. There are separate certifiers for each of the different key-using roles in SET (cardholders, merchants and payment gateways). However, they are all bound together in a hierarchy so that any SET party can use public key certificates to establish trust in any other. At the time of writing there is a separate trust hierarchy for each credit card brand, because no ultimate CA has been identified to be the top of the hierarchy. However, Visa and Mastercard have proposed that CertCo LLC should assume this role. CertCo is a company that was founded to enable banks and other financial institutions to provide a trustworthy infrastructure capable of supporting large-scale, international, secure electronic commerce. In December 1996, CertCo was spun out of Bankers Trust, the seventh largest U.S. bank and the world's fourth largest clearer of money.

2.2 SET Relationships

Many of the inter-role relationships are implicit in the definitions of the roles themselves (above). We can divide the relationships into three types:

1. *Contractual* relationships. These represent agreements in law between the different parties to provide services and accept responsibilities. They have nothing to do with SET directly, except that SET assumes the relationships already exist.
2. *Administrative* relationships. These are relationships required to set up the SET environment before any payments can flow. They also maintain the environment and keep it secure. Some of these are actual SET protocol flows.
3. *Operational* relationships. These are the short-term relationships that take place when a payment happens. These are all defined by SET protocol flows.

Figure 3 on page 13 to Figure 5 on page 15 illustrate the three kinds of relationships.

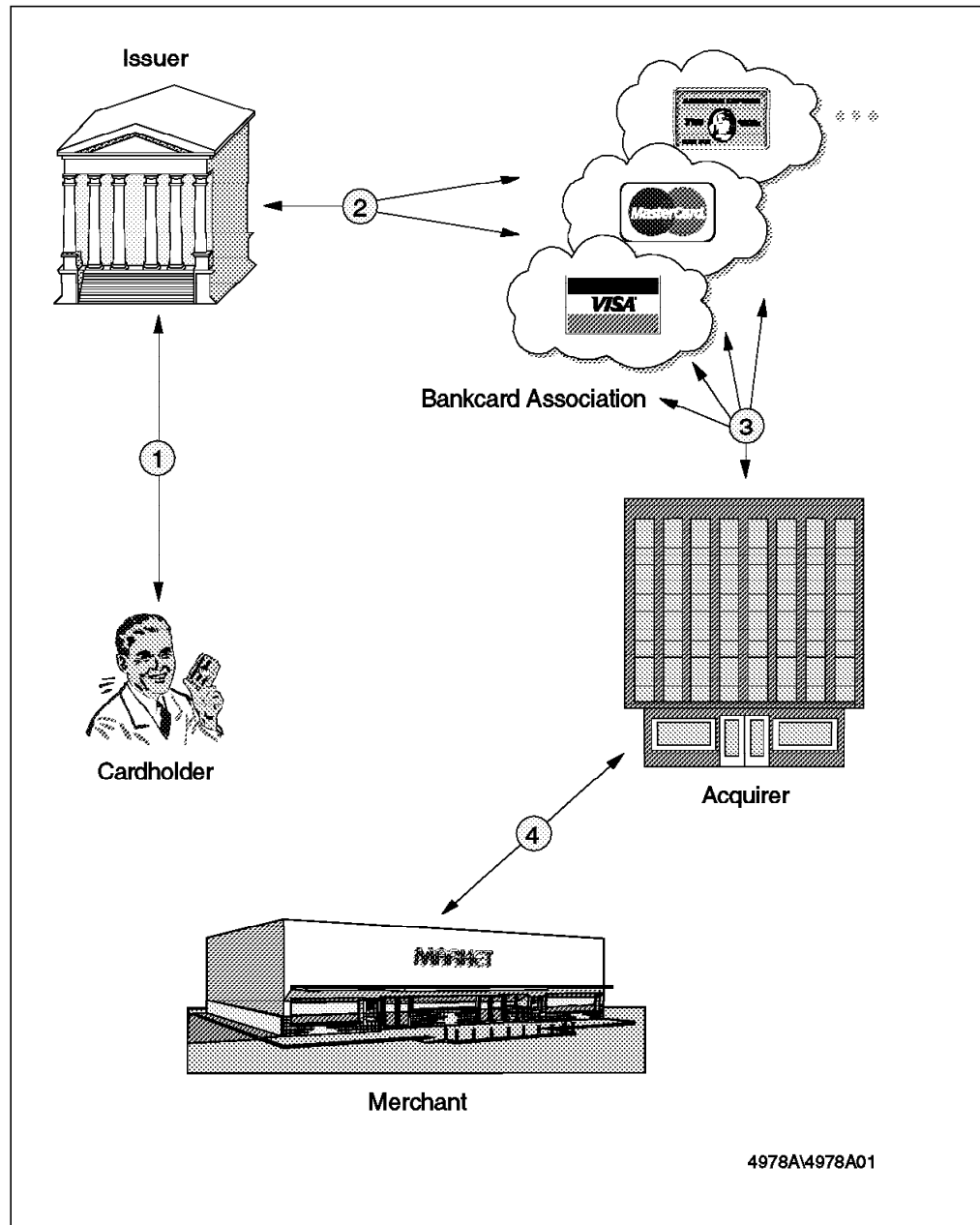


Figure 3. Contractual Relationships

The contractual relationships shown are as follows:

1. The cardholder has a contract with the issuer in which he or she promises to pay the minimum monthly payment and stay within a credit limit.
2. The issuer has a contract (or other legal instrument) with the bankcard association in which it agrees to accept the risk of card debt and to support the association.
3. The acquirer has an agreement with the bankcard associations in which it agrees to operate securely and within their rules. In practice the acquirer is often part of a financial institution itself, so different parts of a company may play different roles.
4. The merchant has a contract with the acquirer in which it receives payment processing services in return for a percentage fee.

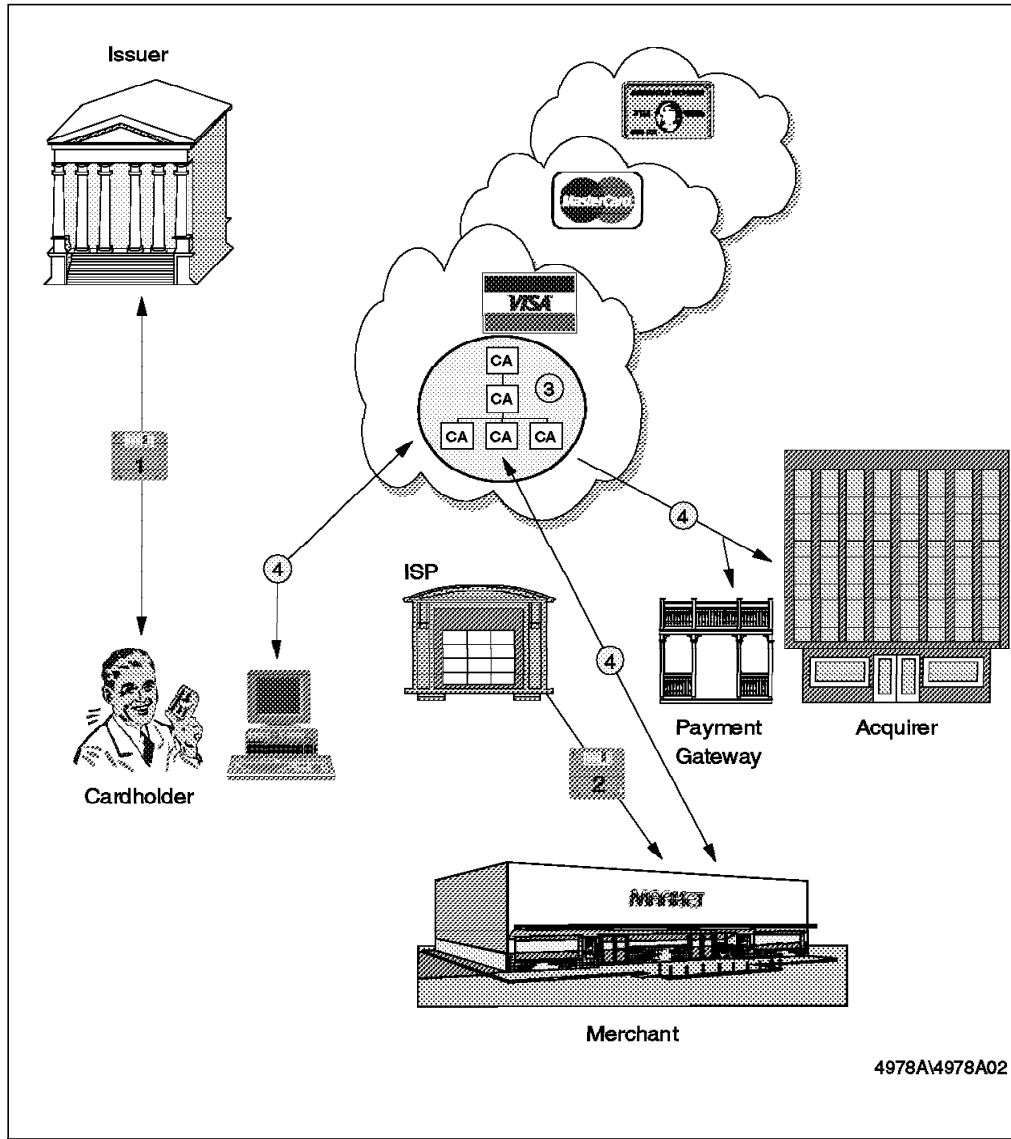


Figure 4. Administrative Relationships

The administrative relationships shown are as follows:

1. The cardholder has a relationship of trust with a software provider. This will probably be the bank that provided his or her credit card. Based on this trust, the cardholder will install the SET code and upgrades on his or her personal computer. Trust is necessary, because the cardholder needs to be sure that the code is genuine.
2. The merchant has a contract with a software and services provider. This may be an Internet Services Provider (ISP) or other organization, probably related to the acquirer. Whatever the organization is, it will supply and (possibly) install the merchant SET code, so the merchant needs to trust it not to install a Trojan horse or otherwise damage the integrity of the system.
3. The different certification authorities have hierarchical relationships between them based on public key certificates.
4. The cardholder, merchant and payment gateway have relationships with the appropriate certification authorities. They must each prove their legitimacy,

for which they receive a signed public key certificate. The details of these relationships are part of the SET specification.

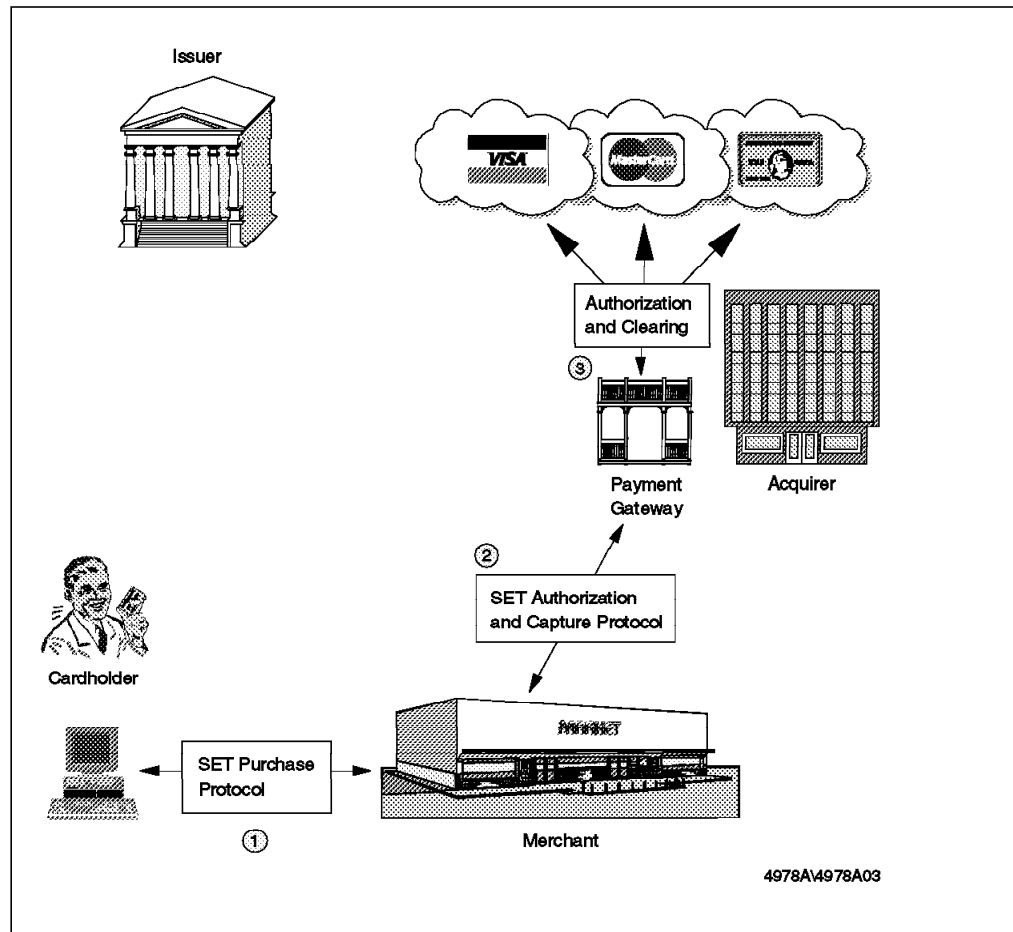


Figure 5. Operational Relationships

Finally, we get to do some shopping. The operational relationships are the heart of SET:

1. The cardholder connects to the merchant system to perform payment, inquiry and refund transactions.
2. The merchant system connects to the payment gateway to perform authorization and capture transactions.
3. The payment gateway connects to the bankcard association networks to relay the requests from the merchant.

2.3 The Limits of SET

SET is specifically a *payment protocol*. It defines the communication between cardholder, merchant and payment gateway for card purchases and refunds. It defines the communication between the different parties and certification authorities for public key signature. It does *not* define anything beyond that.

To see what this means in practice, consider Table 1 on page 16. The table goes through the steps involved when a cardholder decides to install SET support and then purchase something. The right-hand column indicates which steps are defined by SET. We explore the SET functions in detail in the next two

chapters and show how the complete payment cycle is implemented in Part 2, “The Practice” on page 81.

<i>Table 1. Boundaries of SET</i>		
	Description	Defined in SET?
1	Cardholder receives and installs software.	
2	Cardholder connects to CA and requests a certificate.	
3	Cardholder SET software invoked by CA.	
4	Cardholder provides public key and details.	yes
5	Certificate signed by CA and returned to cardholder.	yes
6	Cardholder connects to merchant Web site and selects goods to purchase.	
7	Cardholder selects payment option.	
8	Cardholder SET software invoked by merchant.	
9	Payment request sent to merchant.	yes
10	Authorization request sent to payment gateway.	yes
11	Gateway forwards authorization to bankcard network and receives response.	
12	Authorization response returned to merchant.	yes
13	Payment response sent to cardholder.	yes

Chapter 3. SET Payment Protocols

We have described in the preceding chapters the job that SET performs and the environment in which it operates. Let us now look in detail at the protocol itself. If you want some further insight into these processes, refer to the *Secure Electronic Transactions Specification*, which is in three parts:

- Book 1, Business Description
- Book 2, Programmer's Guide
- Book 3, Formal Protocol Definition

The documents are available in several different formats from <http://www.mastercard.com/set>.

3.1 Business Requirements

SET exists to allow businesses to receive payment for goods and services in a safe, reliable and consistent manner. When you consider how the protocols work, you should keep in mind the requirements that such businesses have for secure transactions. These requirements are addressed in SET by using cryptographic and others techniques.

The major business requirements are:

- | | |
|-------------------------|---|
| Confidentiality | We do not want any details about the payment and the order to which it applies to be visible to someone listening in on the session, whether they are monitoring the network path or lurking on one of the systems involved in the transfer. |
| Authentication | We want to be sure of the identity of all of the parties involved in the transaction. |
| Integrity | We want to be sure that the data received in a SET transaction is identical to what was sent. Otherwise it may be vulnerable to a man-in-the-middle attack. |
| Interoperability | We do not want to have multiple versions of the SET software installed, particularly on the customer's browser. Therefore, it is important that the protocol will allow implementations from different manufacturers to interoperate with each other. |

Note that, although we focus on the use of SET in an Internet environment, using the World Wide Web as an application vehicle, there is nothing in the specification that mandates this. You should also keep in mind that SET is solely concerned with *payments*. There are many other pieces of sensitive information that may flow in a Web session, such as names and addresses, personal details, and bank account information. The Web server must use other techniques to protect this kind of data in addition to implementing SET for credit card transactions. We show how the IBM implementation handles these problems in Part 2, "The Practice" on page 81.

Let us look at each of the requirements in more detail.

3.1.1 Confidentiality

To facilitate and encourage electronic commerce using payment card products, it will be necessary to assure cardholders that their payment and order instruction are safe and accessible only by the intended recipient. The payment instruction, cardholder account and order instruction must be secured as they travel across the network, preventing interception of account numbers and expiration dates by unauthorized individuals.

Note

Confidentiality is ensured by the use of message encryption.

3.1.2 Authentication

- Cardholder Account Authentication

Merchants need a way to verify that a cardholder is a legitimate user of a valid branded payment card account number. A mechanism that uses technology to link cardholder to a specific payment card account number will reduce the incidence of fraud and therefore the overall cost of payment processing.

- Merchant Authentication

The SET specifications must provide a way for cardholders to confirm that a merchant has a relationship with a financial institution allowing it to accept payment cards. Cardholders also need to be able to identify merchants with whom they can securely conduct electronic commerce.

Note

Authentication is ensured by the use of digital signatures and certificates.

3.1.3 Integrity

The SET specification must guarantee that message content is not altered during the transmission between originator and recipient.

Payment information sent from cardholders to merchants includes order information, personal data and payment instruction. If one of these components is altered, the transaction will not be processed correctly. The SET specification must provide a way to ensure that the information received matches the information sent.

Note

Integrity is ensured by the use of digital signatures.

3.1.4 Interoperability

The SET specification must ensure that it works in different software and hardware platforms. Any cardholder using any compliant software and hardware will be able to communicate with a merchant or a Certification Authority that use another hardware and software.

Note

Interoperability is ensured by the use of specific protocols and message formats.

3.2 SET Message Formats in General

SET uses combinations of cryptographic techniques to address the business specification of a secure transaction. In this section we assume that you are familiar with basic cryptographic concepts, specifically:

- Symmetric key (or bulk) encryption
- Public key encryption
- Secure hash (or digest) functions
- Digital signatures
- Public key certification mechanisms

If you want an introduction to these concepts, you should first read Appendix B, “Cryptographic Processes” on page 299 before continuing here.

SET defines a series of message exchanges for the payment processes and for certificate management. In this chapter we explore the payment processes in some detail and in Chapter 4, “SET Certification” we look at certification. However, all of the processes use similar message packaging techniques, so let us look at an example of how SET creates a secure message.

3.2.1 SET Encrypted Message Example

In our example, Alice wants to send a message to Bob. She does not want anyone else to be able to read this message (confidentiality) and she wants to be sure that Bob reads the same message that she sent without modifications (data integrity). Bob wishes to know if the message that he received is really from Alice (authentication). Also, he may be using different hardware and software from Alice (interoperability).

Figure 6 on page 20 shows the complete mechanism of encryption and decryption.

Encryption Example

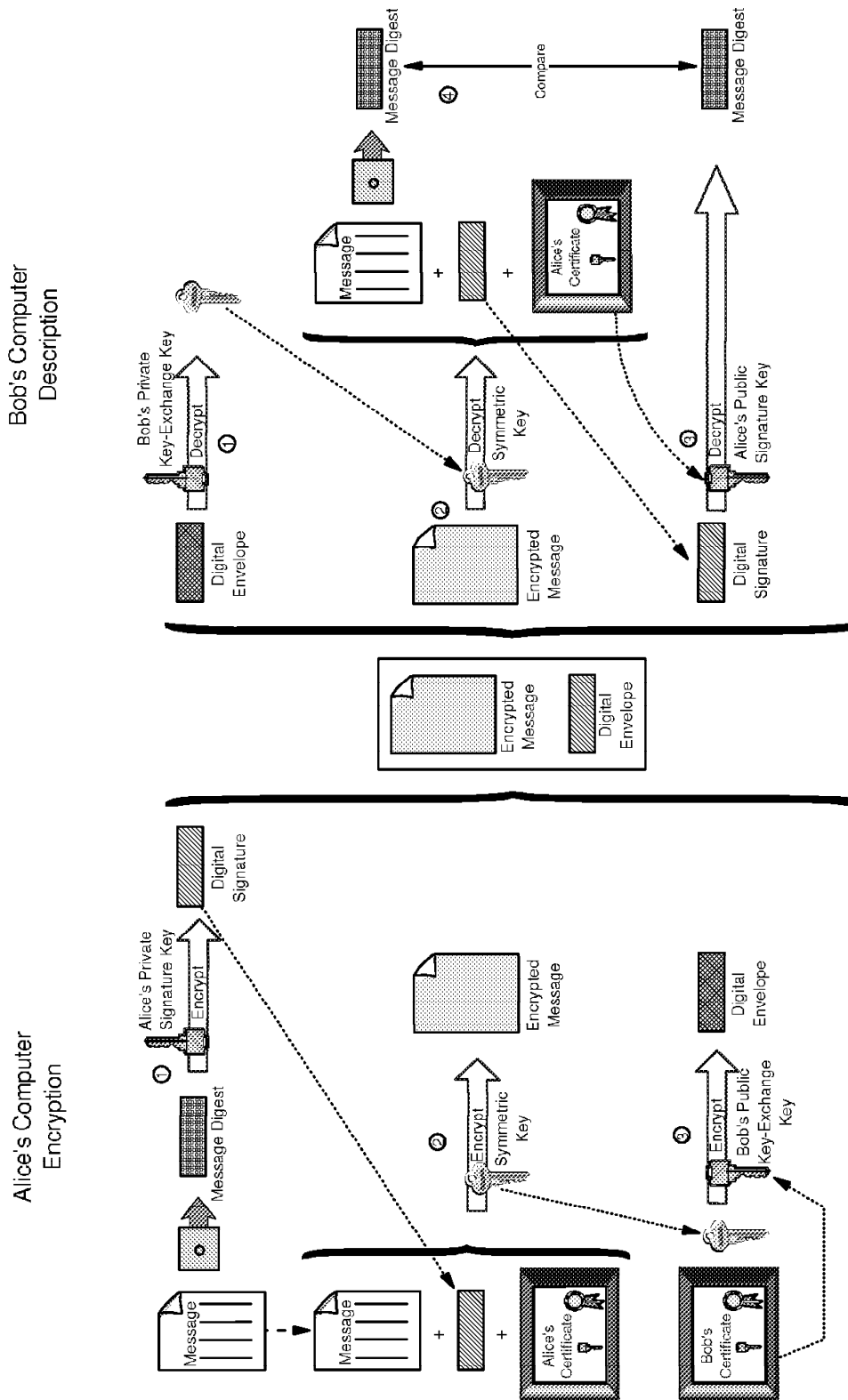


Figure 6. Encryption Example

In the following sections, we examine the processes of encryption and decryption, step by step.

3.2.1.1 Encryption

Alice has a message that she wants to send to Bob as part of a SET transaction.

1. First, Alice wants to create a digital signature to prove to Bob that the message really came from her. She needs to perform the following actions:
 - a. Run her message through a secure hash function (SET uses the SHA-1 secure hash algorithm). This will create a *message digest* that is a unique fingerprint for her message.
 - b. Using RSA, encrypt the message digest with her private signature key. We discuss how Alice came to be the owner of a private key in Chapter 4, "SET Certification" on page 51; for now assume that she has a key pair to use for digital signatures and a certificate that goes with it. The result of encrypting the digest in this way is a *digital signature*.
2. Next, Alice wants to encrypt the content of her message to ensure that nobody can read it when it travels across the network:
 - a. Alice generates some random data that will be used as a key for a bulk encryption algorithm. (By default SET uses the DES algorithm.)
 - b. Alice uses this symmetric key to encrypt her message, digital signature and certificate. The result of this is the encrypted message. She needs to send her certificate because it contains the public half of her signature key that will be used by Bob to decrypt her digital signature.
3. Now Alice has a package that she can safely send to Bob, which also authenticates her. However, in order for Bob to be able to read the message, she needs to also send him the symmetric key that she used to encrypt the message. To send this key with privacy, she needs to use Bob's public key. Alice encrypts the symmetric key using Bob's public key-exchange key. The result of this is called a digital envelope.
4. Finally she sends both the digital envelope and the encrypted message to Bob.

3.2.1.2 Decryption

1. Bob decrypts the digital envelope to obtain the symmetric key using his private key-exchange key.
2. Bob decrypts the encrypted message to obtain the message, digital signature and Alice's public key certificate using the symmetric key. Bob can use the certificate to verify that Alice is trustworthy by checking the signature on it.
3. Bob decrypts the digital signature to obtain the message digest using Alice's public signature key (extracted from her certificate).
4. Bob does not know if the message that he received is really from Alice. To ensure that she sent this message, he must follow these steps:
 - a. Run the received message through the secure hash function to create a message digest.
 - b. Compare this message digest with the message digest that he obtained from step 3. Alice used her private signature key to encrypt the message digest and Bob decrypts it using her public signature key. If they are equal, the message came from the owner of the private half of the signature key, that is to say, Alice.

3.2.2 Dual Signature

Within the SET protocols there is a situation where the cardholder communicates with both the merchant and payment gateway in a single message. The message contains an order section, with details of the products/services to be purchased, plus a payment section. The payment instruction will be used by the acquirer and the order by the merchant, but the messages are both sent together. This means that the message packaging must:

1. Prevent the merchant from seeing the payment instruction
2. Prevent the acquirer from seeing the order instruction
3. Link the two parts of the message, so that they can only be used as a pair

In this case, SET uses a technique called *dual signature*. When the order and payment instruction is sent by the cardholder, the merchant will be able to see only the order instruction, and the acquirer only the payment instruction. The merchant will not see the cardholder's account information. In a SET transaction, the transfer of money and offer are linked allowing the money to be transferred to the merchant only if the cardholder accepts the offer.

3.2.3 Dual Signature Example

This diagram provides an overview of the encryption process for dual signature. It is only an example to understand dual signatures. The entire process of purchasing is explained in 3.3.1, "Purchase Process" on page 25.

Consider the following example: Alice is a buyer and Bob is a merchant. She wants to purchase some products from Bob's store. After she accepts what Bob is offering, she will send an acceptance message to the bank. Using dual signature, the bank will be able to verify the authenticity of Bob's transfer authorization and ensure the acceptance is for the same offer. (In fact, in the real SET protocol flow the message is first sent to the merchant, which then forwards it to the bank, but here we will keep it simple.) See Figure 17 on page 37 for the real SET dual signature flow.

Dual Signature

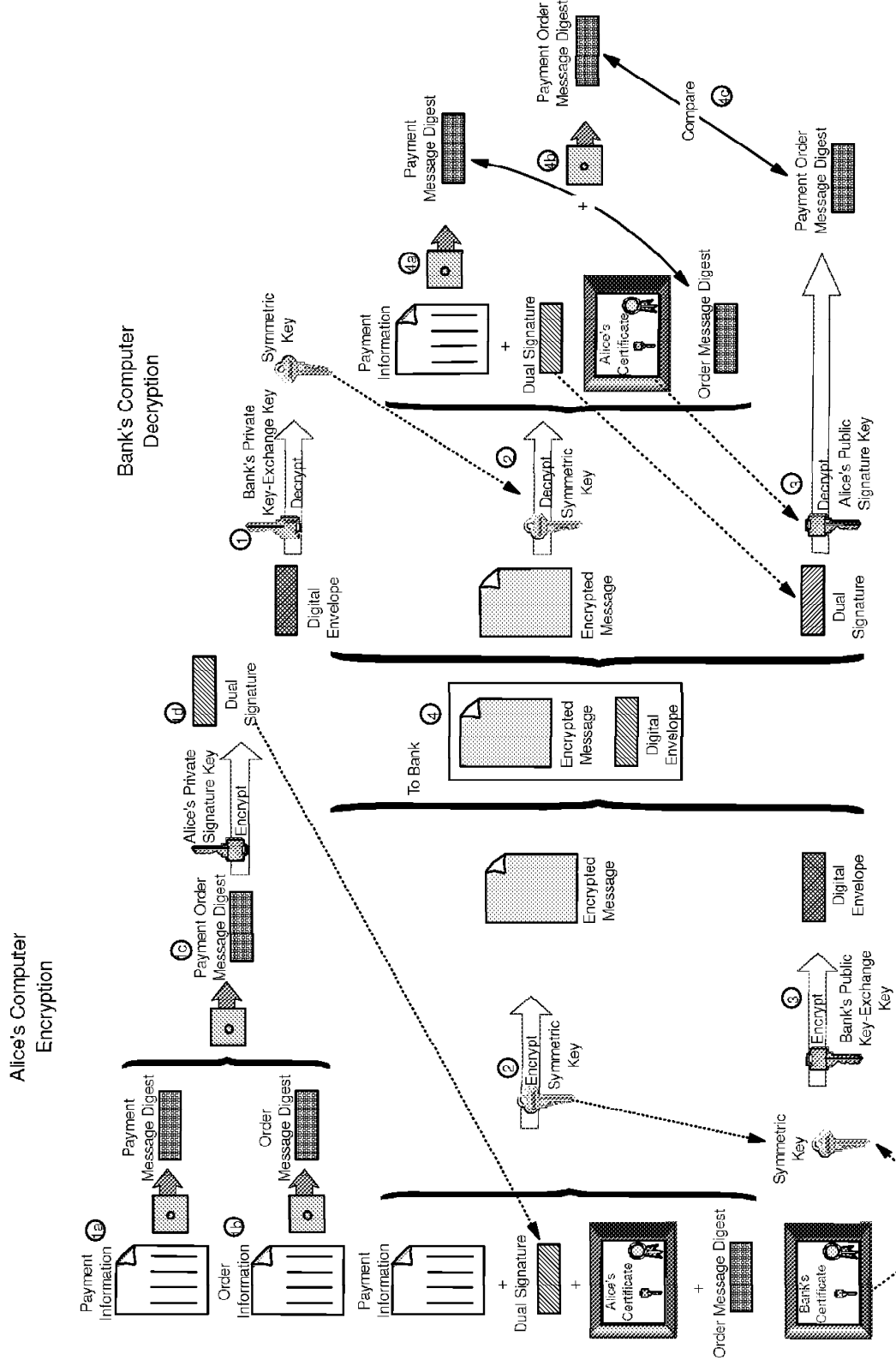


Figure 7. Dual Signature Example

There are two parts to this diagram, the encryption and decryption processes.

3.2.3.1 Encryption

1. Alice wants to create a digital signature to prove to Bob and the bank that the message came from her. She needs to do the following:
 - a. Run her payment instruction through a hash function. This will create a message digest that is unique to her message.
 - b. Run her order instruction through a hash function. Now she has a second message digest that is unique to her order instruction message.
 - c. Concatenate the two message digests together and run the result through a hash function.
 - d. Encrypt the combined message digest with her private signature key. This is called the *dual signature*.
2. Alice wants to encrypt the content of her message to ensure that nobody can read it when it travels across the network.
 - a. She generates some random data that will be used as a key for a bulk encryption algorithm (DES).
 - b. She uses the bulk key to encrypt her payment instruction message, the dual signature, her certificate and the digest of the order message. The result is an encrypted message. She needs to send her certificate because it contains her public signature key that will be used by the bank to decrypt the dual signature. The order message digest will be used by the bank to match up against the same message digest provided by the merchant, to prove that the payment information and the order instruction were generated together by Alice.
3. Alice needs to send the symmetric key that she used to create the encrypted message. To send this key with privacy, she uses the bank's public key.
 - Alice encrypts the symmetric key using the bank's public key-exchange key. The result is called a digital envelope.
4. Alice sends the digital envelope and the encrypted message to the bank.

3.2.3.2 Decryption

1. The bank decrypts the digital envelope to obtain the symmetric key using its private key-exchange key.
2. The bank decrypts the encrypted message to obtain the payment instruction, dual signature, Alice's certificate and the order message digest using the symmetric key.
3. The bank extracts Alice's public signature key from her certificate and uses it to decrypt the dual signature to obtain the payment order message digest.
4. The bank does not know if the message that it received is really from Alice. To ensure that she sent this message, it must follow these steps:
 - a. Run the payment instruction through a hash function to create a payment message digest.
 - b. Concatenate the payment message digest with the order message digest and run the result through a hash function.
 - c. Compare this message digest with the message digest obtained from step 3. If they are equal, the message came from Alice.

The bank can then go on to process the order instruction from Bob's store, which uses a similar construction and which contains the same dual signature.

If the signatures match, the bank can be sure that the two messages are a pair and have not been altered. But, you say surely an attacker could have modified *both* messages? This is only possible if the attacker has access to both Bob and Alice's private keys, because each half of the message was individually signed by its sender.

3.3 SET Payment Transaction Processes

The entire lifecycle of an electronic transaction includes the browsing, purchase, payment authorization and payment capture processes. SET protocols define the last three of these.

The cardholder accesses an electronic store, browses the products and services and prepares an order. The cardholder software receives the information about the order, as the description of the products or services and the total price including the shipping and handling. The user must confirm this order and supply the payment instruction. The merchant accepts the order and then uses the payment instruction to create authorization and capture requests to send to the payment gateway. The payment gateway translates the request and forwards it to the appropriate bankcard network. If the authorization is given, the payment gateway will send a response to the merchant, who will ship the goods or perform the services indicated in the order.

The merchant uses the capture request to initiate the payment from the payment gateway. As in the case of authorization, the payment gateway translates the capture request into a funds transfer to the merchant's account.

The whole process will only occur if all the parties involved in the SET transactions have valid public key certificates.

We describe this process in detail, dissecting it into three parts:

- Purchase process
- Payment authorization process
- Payment capture process

3.3.1 Purchase Process

Figure 8 on page 26 shows at a high level how a cardholder requests a purchase.

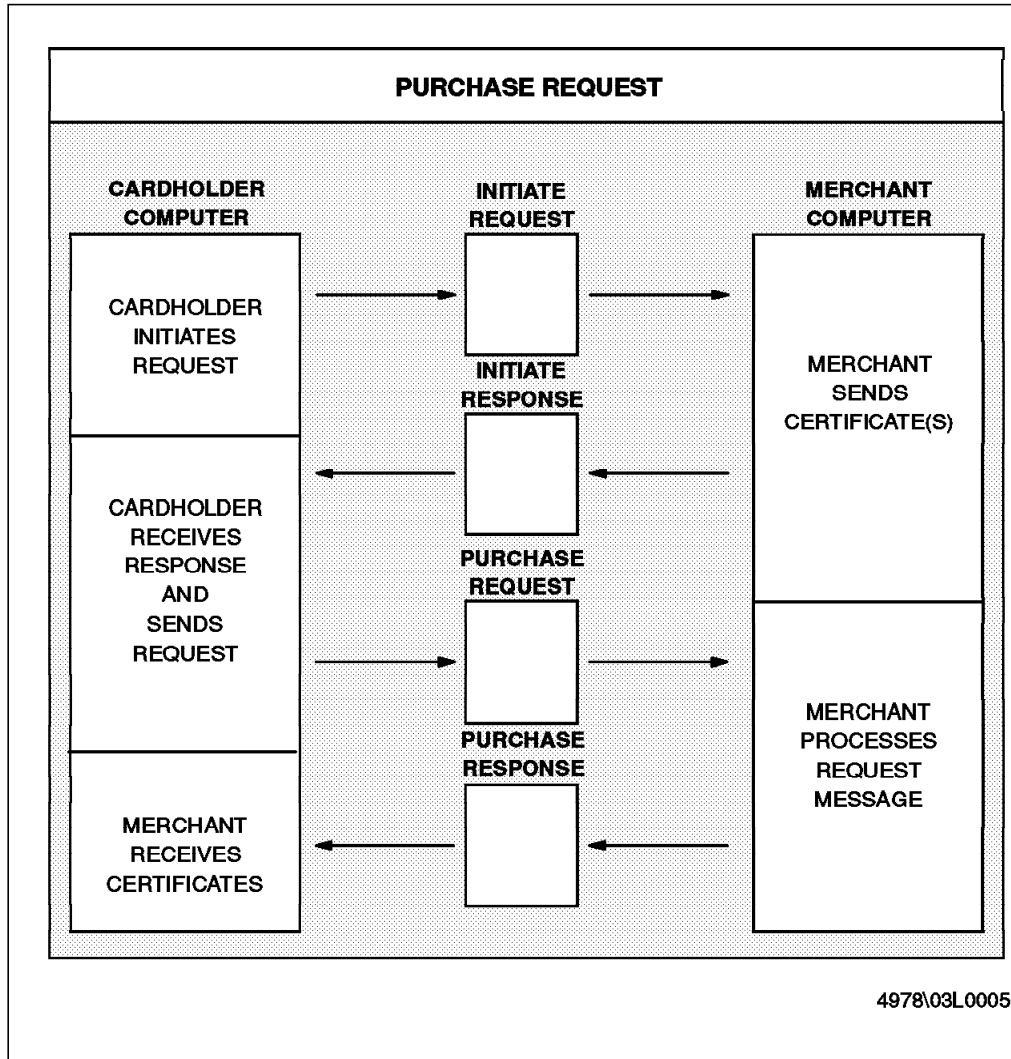


Figure 8. Summary of the Purchase Process

The payment process begins when the cardholder sends an initiate request (P INIT) to the merchant. The merchant receives the initiate request and sends a initiate response. This response includes the merchant’s certificates that will be used to encrypt messages later in the transaction. The cardholder receives the certificates and generates a purchase request to send to the merchant. The purchase request is a two-part message, including a dual signature (see 3.2.2, “Dual Signature” on page 22). The two parts of the message are the *order instruction* (OI), which is required by the merchant, and the *payment instruction* (PI), which is required by the payment gateway. The payment information is encrypted with the payment gateway public key, making it unintelligible to the merchant. Only the payment gateway can read it.

The merchant receives the purchase request and generates a payment authorization request to send to the payment gateway. We describe this request in the next section (3.3.2, “Payment Authorization Process” on page 34). At this point (without waiting for a response to the authorization request) the merchant completes the purchase process by responding to the cardholder’s purchase request.

The cardholder can inquire about the status of the order to know if the payment authorization was approved and when the goods will be delivered. We now look at each of these messages in more detail.

3.3.1.1 Cardholder Initiates Request

Cardholder software creates a *purchase initiate* request, containing the name of the credit card brand that the cardholder has elected to use. This request is effectively asking for a copy of the merchant certificate containing its public signature key and payment gateway certificate containing its public key-exchange key. The cardholder sends the initiate request to the merchant.

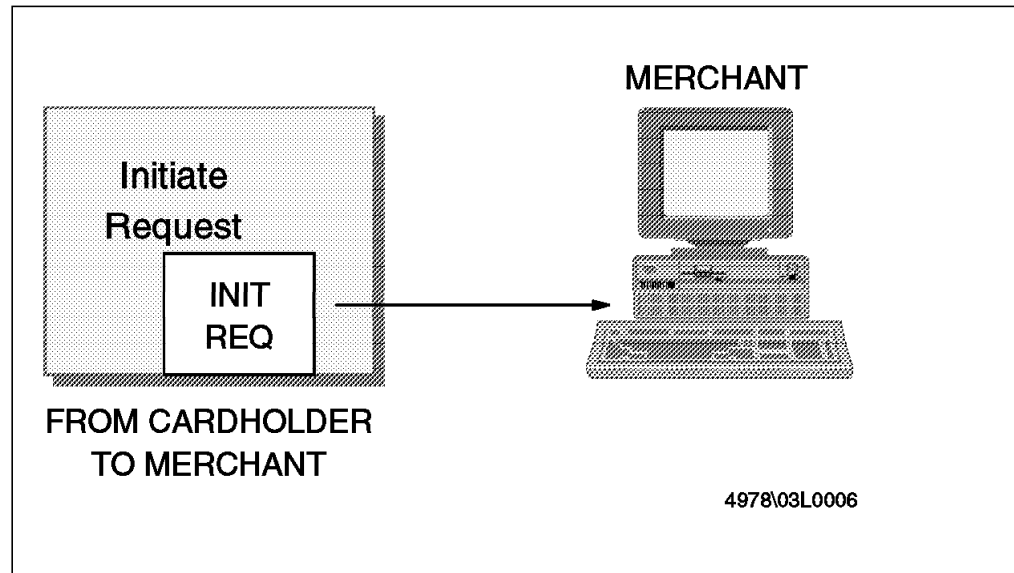


Figure 9. Cardholder Initiates Purchase

3.3.1.2 Merchant Sends Response

The merchant receives the cardholder initiate request and transmits a response containing the certificates. The payment gateway certificate will be used to protect the payment information that is sent to the payment gateway. The merchant is affiliated with an acquirer and part of the setup of the merchant system is to provide it with a copy of the acquirer payment gateway certificates. The merchant certificate will be used to protect the order instruction sent to the merchant. Figure 10 on page 28 shows how the merchant constructs the payment initiate response.

1. The merchant receives the initiate request.
2. The merchant generates the response message and digitally signs it by passing the response through a hash function. The message digest created by this is encrypted with the merchant private signature key, resulting in a digital signature.
3. The merchant sends the initiate response, the digital signature, the merchant certificate containing the public signature key and the payment gateway certificate containing the public key-exchange key to the cardholder.

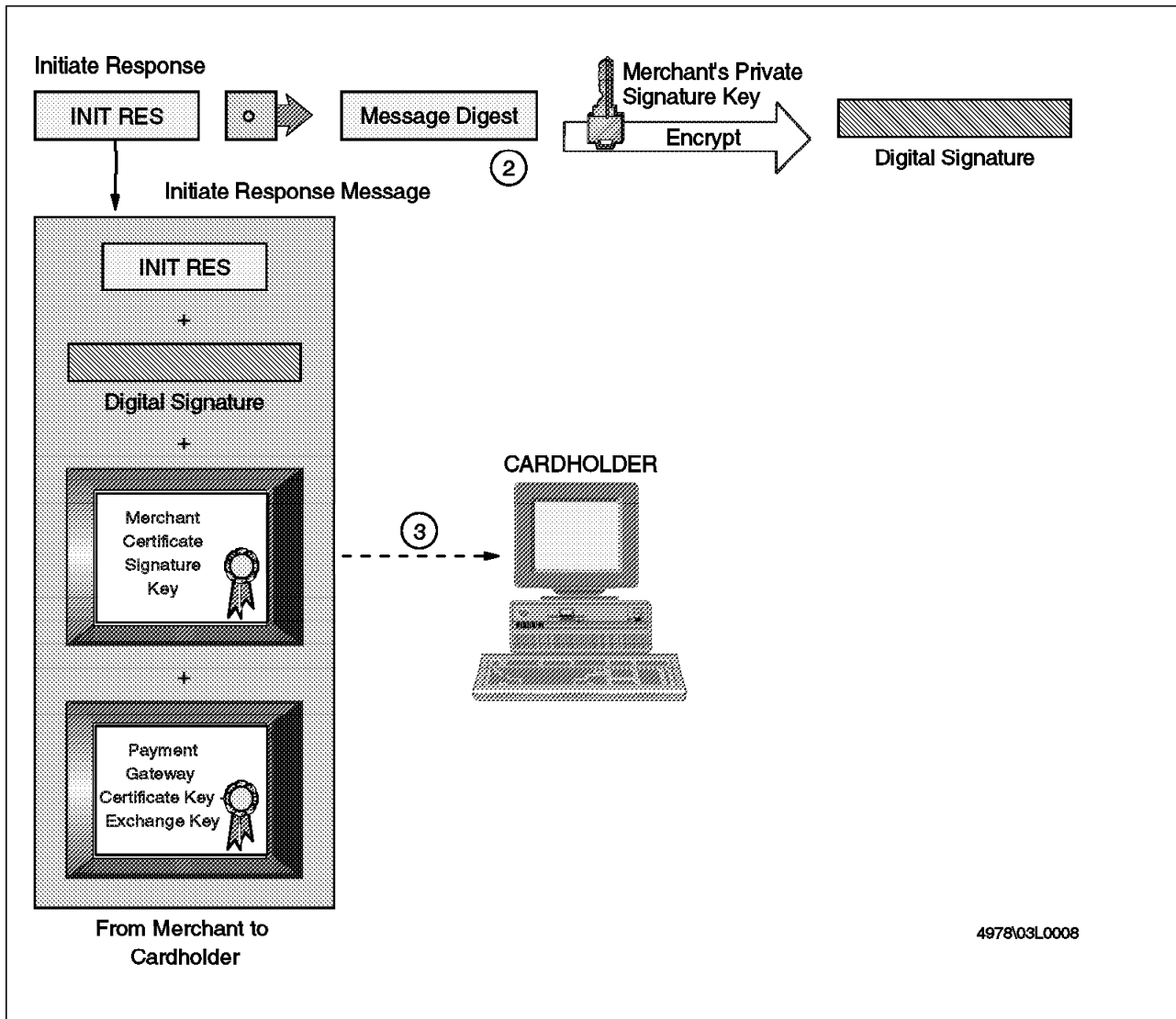


Figure 10. Merchant Sends Initiate Response

3.3.1.3 The Cardholder Sends Request

Next, the cardholder will send the payment request to the merchant. This is a message in two parts, the *Order Instruction*, which is for the merchant to process, and the *Payment Instruction*, which is for the payment gateway. The two parts are bound together using a dual signature, as described in 3.2.2, "Dual Signature" on page 22.

1. The cardholder receives the initiate response message from the merchant and verifies the merchant and payment gateway certificates by traversing the trust chain to the root. These certificates will be used later during the ordering process.
2. The merchant signature is verified by running the initiate response through a hash function and creating a message digest. The digital signature is decrypted using the merchant public signature key and the result is compared with the message digest obtained from the received message. If they are equal, the integrity of the message is assured (see Figure 11 on page 29).

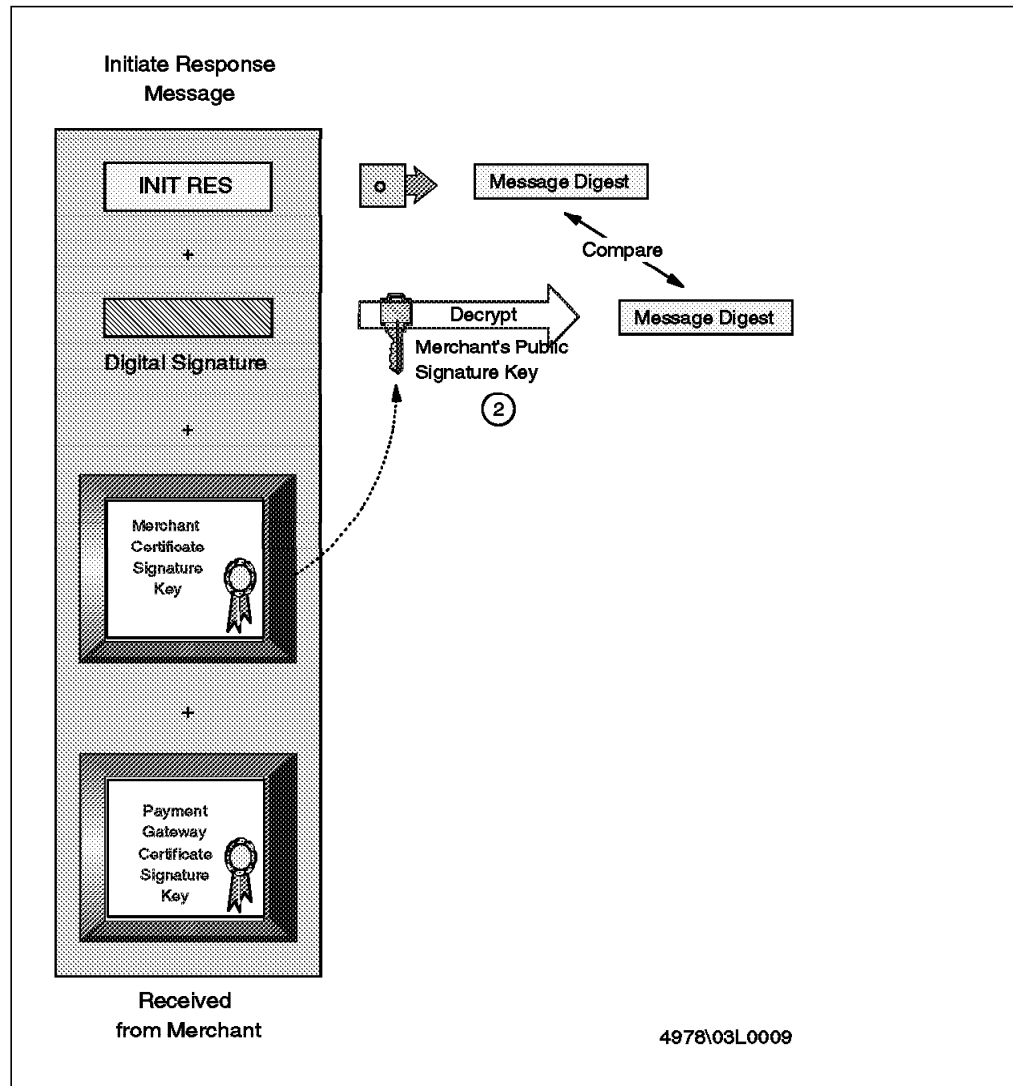


Figure 11. Cardholder Receives Initiate Response

3. The cardholder software creates the order instruction (OI) portion of the purchase request message using information from the shopping phase. The OI does not contain the description of the goods purchased. This information was exchanged between the cardholder and the merchant during the shopping process and before the first SET message.
4. The cardholder creates the second portion of the purchase request, the payment instruction (PI). This contains details of the credit card that the cardholder has chosen to use.
5. A transaction identifier, received from the merchant in the initiate response, is placed in the OI and PI. This identifier will be used by the payment gateway to link the OI and PI when the merchant requests payment authorization.
6. The cardholder generates a dual signature by passing the order instruction and payment instruction through a hash function. The two message digests created (OI message digest and PI message digest) are concatenated. The resulting message is run through a hash function and is encrypted with the cardholder private signature key. This is the dual signature.

7. The PI, dual signature and OI message digest are encrypted using a randomly generated symmetric key. This is the encrypted payment message, which will be passed on to the payment gateway.
8. The symmetric key used to construct the payment message and the cardholder's account number are encrypted with the payment gateway public key-exchange key, generating the payment digital envelope.
9. The encrypted payment message, PI message digest, order instruction (OI) message, payment digital envelope, dual signature and the cardholder certificate containing its public signature key are sent to the merchant.

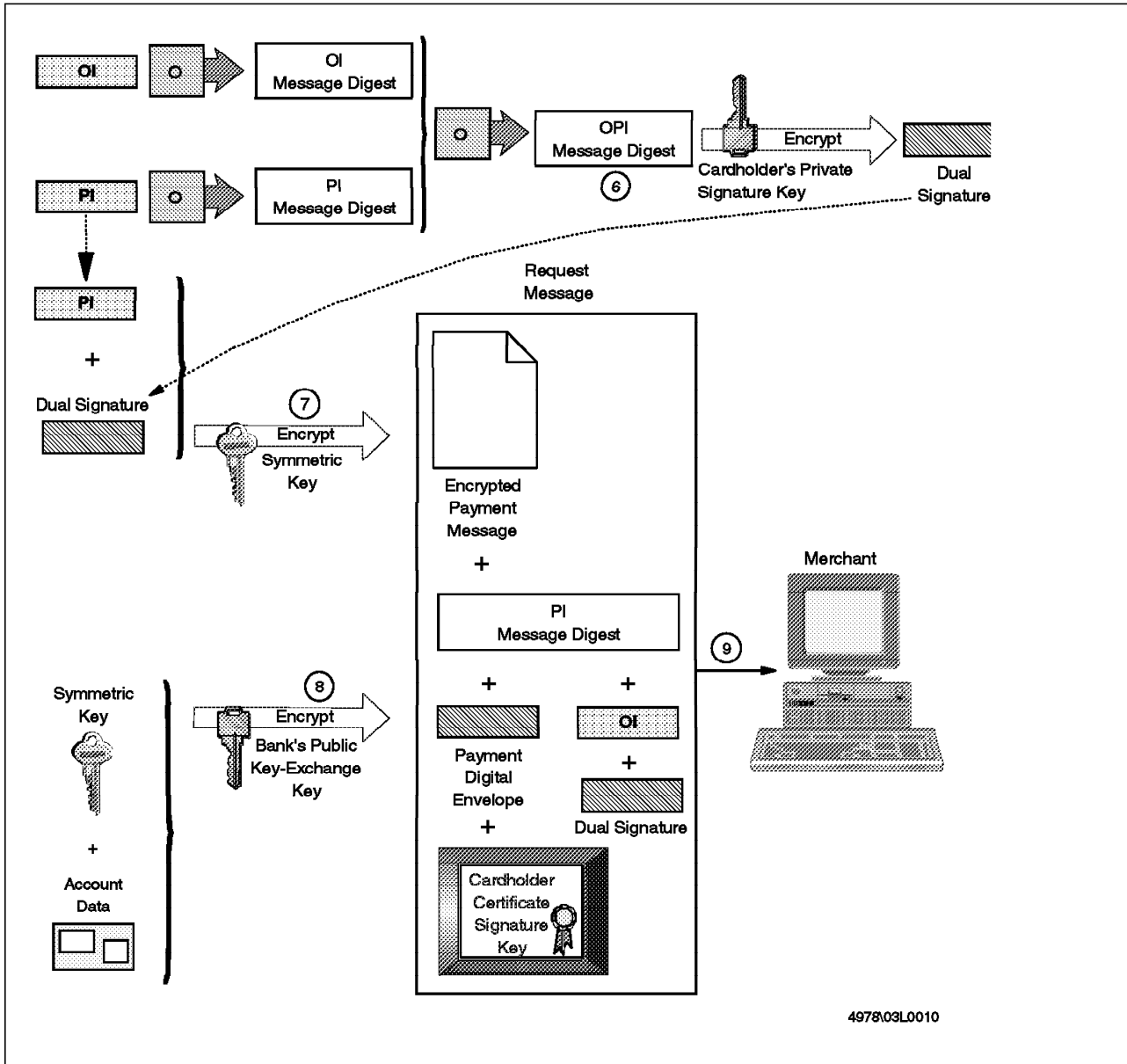


Figure 12. Cardholder Sends Purchase Request

3.3.1.4 Merchant Processes Purchase Request Message

This process is shown in Figure 13 on page 32. It involves the following steps:

1. The merchant receives the request message and verifies the cardholder certificate by traversing the trust chain to the root. The public key in this certificate will be needed to check the dual signature.
2. The dual signature is verified by running the order instruction (OI) through a hash function and creating the OI message digest. This message digest is concatenated with the PI message digest that was received within the request message. The dual signature is decrypted using the cardholder public signature key and the result is compared with the OPI message digest obtained locally. If they are equal, the merchant can be assured of the integrity of the request.
3. The merchant processes the order request and forwards the encrypted payment message and payment digital envelope to the payment gateway for payment authorization, as we describe in 3.3.2.1, "Merchant Requests Authorization" on page 35.

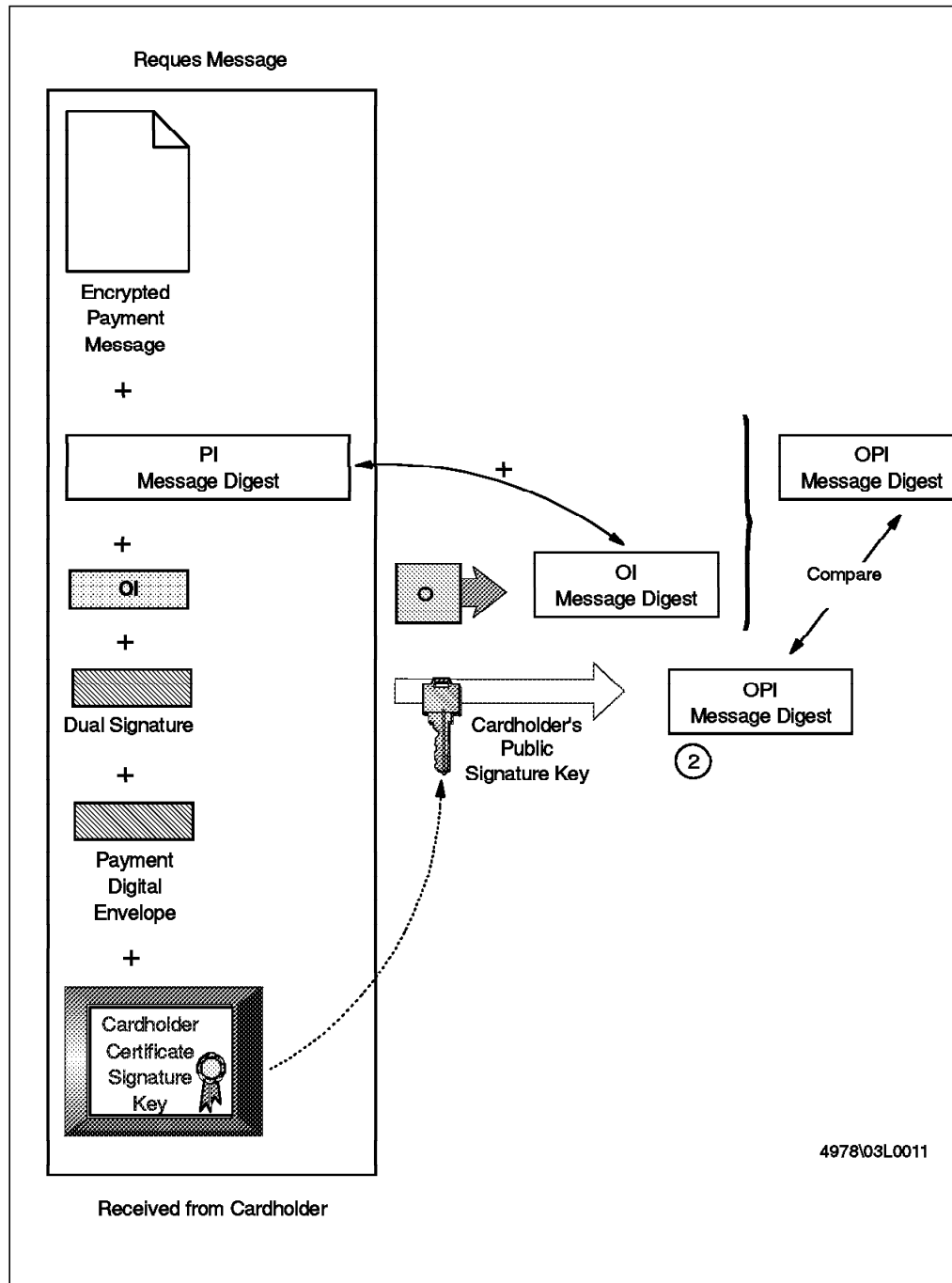


Figure 13. Merchant Verifies Purchase Request Message

The merchant does not need to wait for a response to its authorization request before it sends a response to the cardholder's purchase request.

4. The merchant creates the response message and digitally signs it by passing it through a hash function. The message digest so created is encrypted with the merchant private signature key, resulting in a digital signature.
5. The CA sends the purchase response, the digital signature and the merchant certificate containing the public signature key to the cardholder (see Figure 14 on page 33). This message only indicates that the merchant received the order. The services or goods purchased by the cardholder will

only be executed or shipped when the merchant receives a payment authorization response from the payment gateway. After the cardholder receives the confirmation that the merchant received the order information, he or she can send inquiries to the merchant to know if the authorization has been performed.

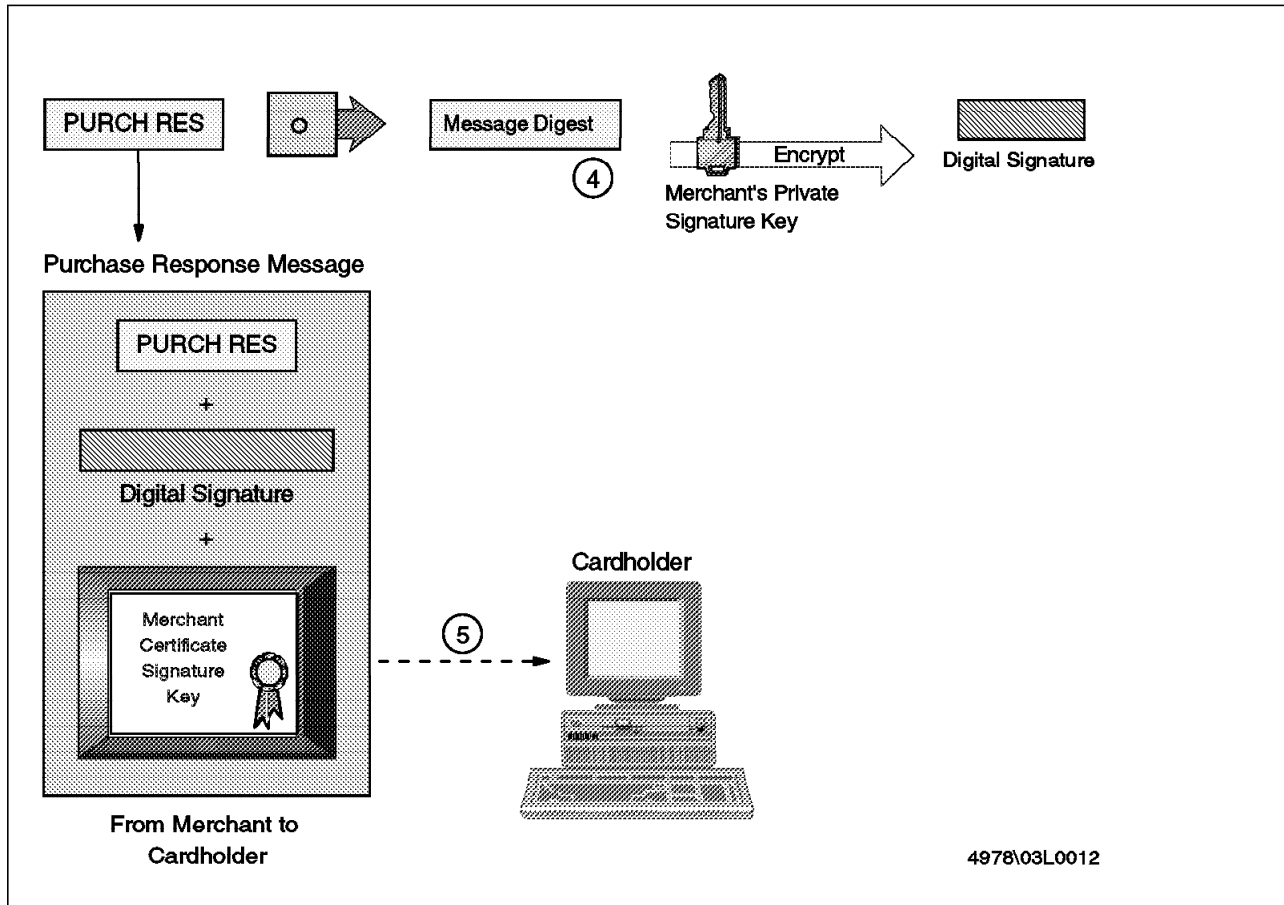


Figure 14. Merchant Sends Purchase Response

3.3.1.5 Cardholder Receives Purchase Response

The cardholder receives the response from the merchant. This tells him or her that the purchase request has been accepted and that he or she can expect to receive the goods, as long as the card has enough credit remaining.

1. The cardholder receives the purchase response message from the merchant and verifies the certificates by traversing the trust chain to the root.
2. The merchant signature received is verified by running the purchase response through a hash function and creating a message digest. The digital signature from the response is decrypted using the merchant public signature key and the result is compared with the message digest obtained locally. If they are equal, the cardholder is assured of the integrity of the message.
3. The cardholder stores the purchase response. The cardholder can determine the status of the order (if the payment gateway approved the transaction) by sending an order inquiry message to the merchant. Order inquiry is also defined by the SET specification, although we do not go into details of it here.

If it was approved, the goods purchased will be shipped or the services will be performed.

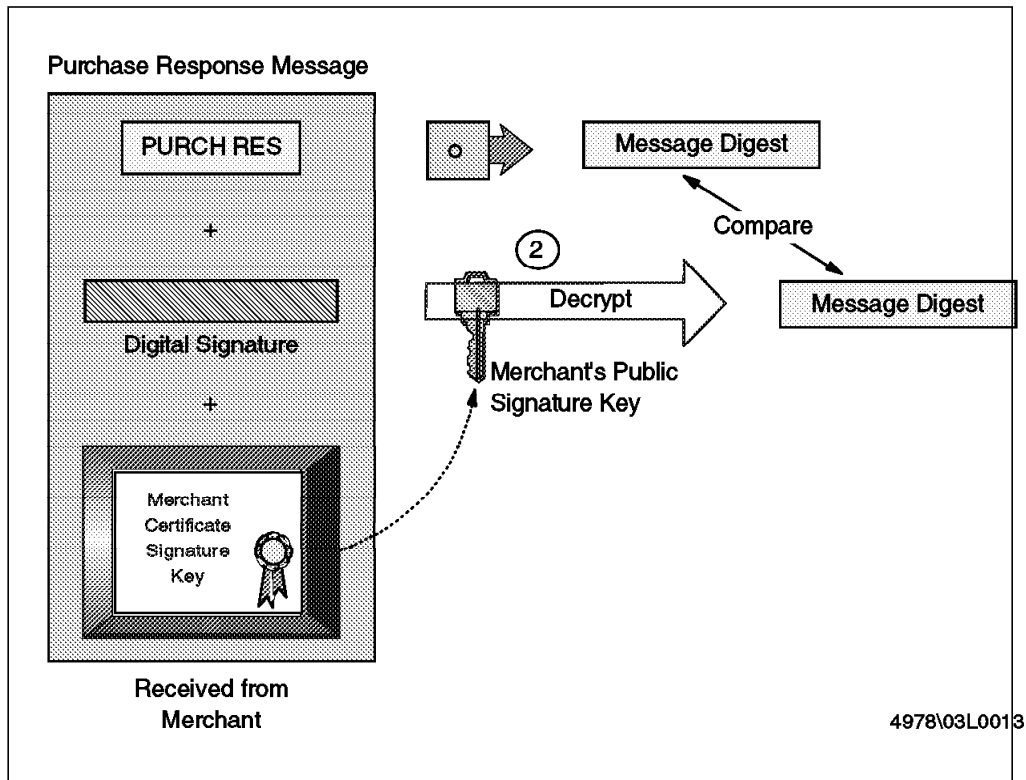
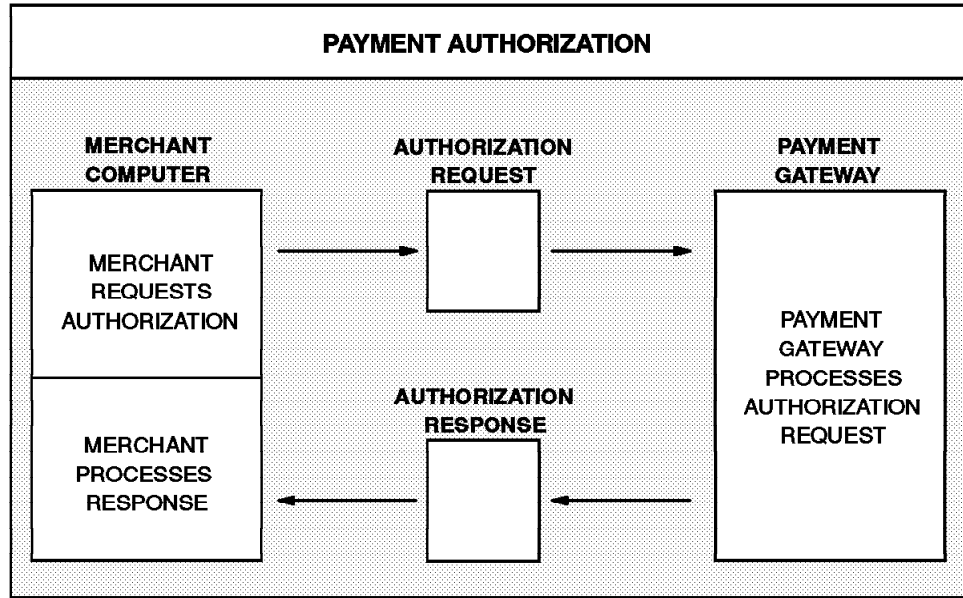


Figure 15. Cardholder Receives Purchase Response

3.3.2 Payment Authorization Process

Figure 16 on page 35 shows at a high level how a merchant requests authorization for a payment.



4978\03L0014

Figure 16. Payment Authorization Diagram

We show the details of the messages when a merchant asks for a payment authorization with the payment gateway. This authorization ensures that the transaction was approved by the issuer (cardholder's financial institution). This approval guarantees to the merchant that it will receive payment and the merchant can therefore go ahead and deliver the goods or perform the services requested by the cardholder.

The process to obtain a payment authorization begins when the merchant sends an authorization request. The authorization request is in two parts: the payment message originally sent by the cardholder inside the purchase request and an authorization message created by the merchant. When the payment gateway receives the authorization request, it verifies the consistency between the two parts of the request using the dual signature and the transaction identifier.

The payment gateway selects the appropriate private bank card network for the particular credit card brand and sends an authorization request using the proper protocol of that network. If the request is approved, the payment sends an authorization response to the merchant containing the capture token that will be used when the merchant requests that the payment be made (or *captured* in the jargon of SET). This capture token is stored by the merchant until the goods purchased by the cardholder are shipped or the services are performed.

We now look at this message flow in detail.

3.3.2.1 Merchant Requests Authorization

This request is shown in Figure 17 on page 37.

1. The merchant software creates an authorization request including the amount to be authorized, the transaction identifier from the order instruction (OI), a locally generated digest of the OI and other information related to the transaction.

2. The merchant digitally signs the authorization request by passing it through a hash function, creating a message digest. The message digest is encrypted with the merchant's private signature key resulting in a digital signature.
3. The authorization request and the digital signature are encrypted using a randomly generated symmetric key(1). We refer to this as key(1) because a second key is also used in this process.
4. Symmetric key(1) is then encrypted with the payment gateway public key-exchange key, generating the digital envelope.
5. The merchant sends the authorization request message, the payment message created during the cardholder's purchase process (see 3.3.1.4, "Merchant Processes Purchase Request Message" on page 31 step 3), the cardholder certificate containing the public signature key, the merchant certificate containing the public signature key, and the other merchant certificate containing the public key-exchange key to the payment gateway.

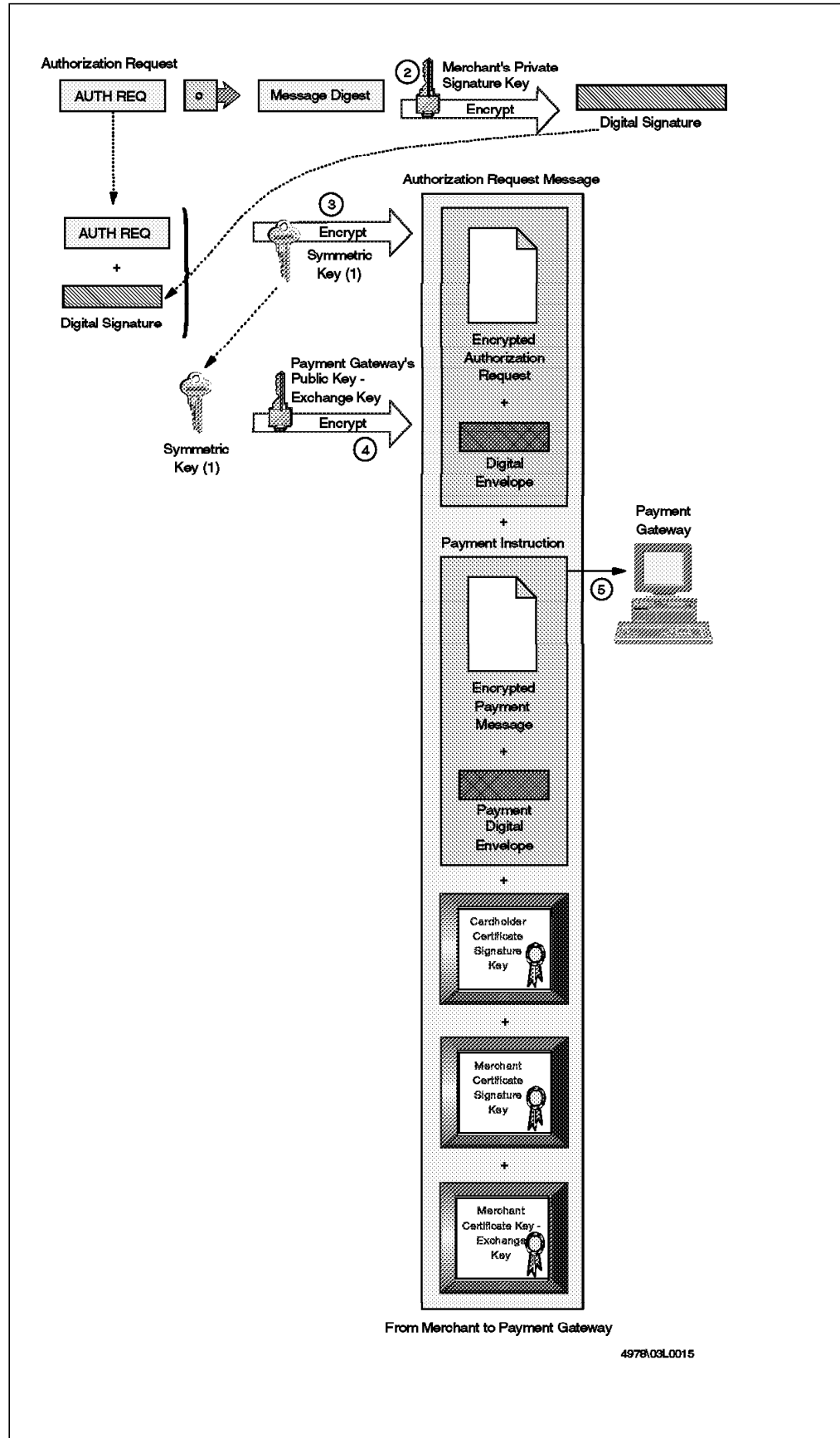


Figure 17. Merchant Requests Authorization

3.3.2.2 Payment Gateway Processes the Authorization Request

When the payment gateway receives the authorization request, it first has to disassemble it and validate it. The process is shown in Figure 18 on page 39.

1. The payment gateway receives the authorization message and verifies the merchant certificates by traversing the trust chain to the root. It also verifies that the certificates have not expired.
2. The payment gateway decrypts the digital envelope contained in the authorization request message using the payment gateway private key-exchange key, thereby obtaining symmetric key(1).
3. Symmetric key(1) is used to decrypt the encrypted authorization request to obtain the authorization request message and the digital signature.
4. The digital signature is verified by running the authorization request through a hash function to create a message digest. The digital signature from the request is decrypted using the merchant public signature key and the result is compared with the message digest obtained locally. If they are equal, the integrity of the request is assured.
5. The payment gateway verifies the cardholder certificate by traversing the trust chain to the root. It also verifies that the certificate has not expired.
6. The payment gateway decrypts the payment digital envelope contained in the payment message using the payment gateway private key-exchange key to obtain the symmetric key(2) and the cardholder account information.
7. This symmetric key is used to decrypt the encrypted payment message to obtain the payment instruction (PI) and dual signature.
8. The dual signature is verified by running the PI through a hash function to create the PI message digest. The PI message digest is concatenated with the OI message digest received from the merchant as part of the authorization request. The two digests are then run through a hash function to generate the OPI message digest. The dual signature is decrypted using the cardholder public signature key and the result (which is the OPI digest originally calculated by the cardholder) is compared with the OPI message digest generated locally. If they are equal, the payment gateway can be assured that the two halves of the message match each other and they have not been altered in any way.

The payment gateway also verifies the integrity of the transaction by checking that the transaction identifier received from the merchant matches the identifier sent with the cardholder payment instruction.

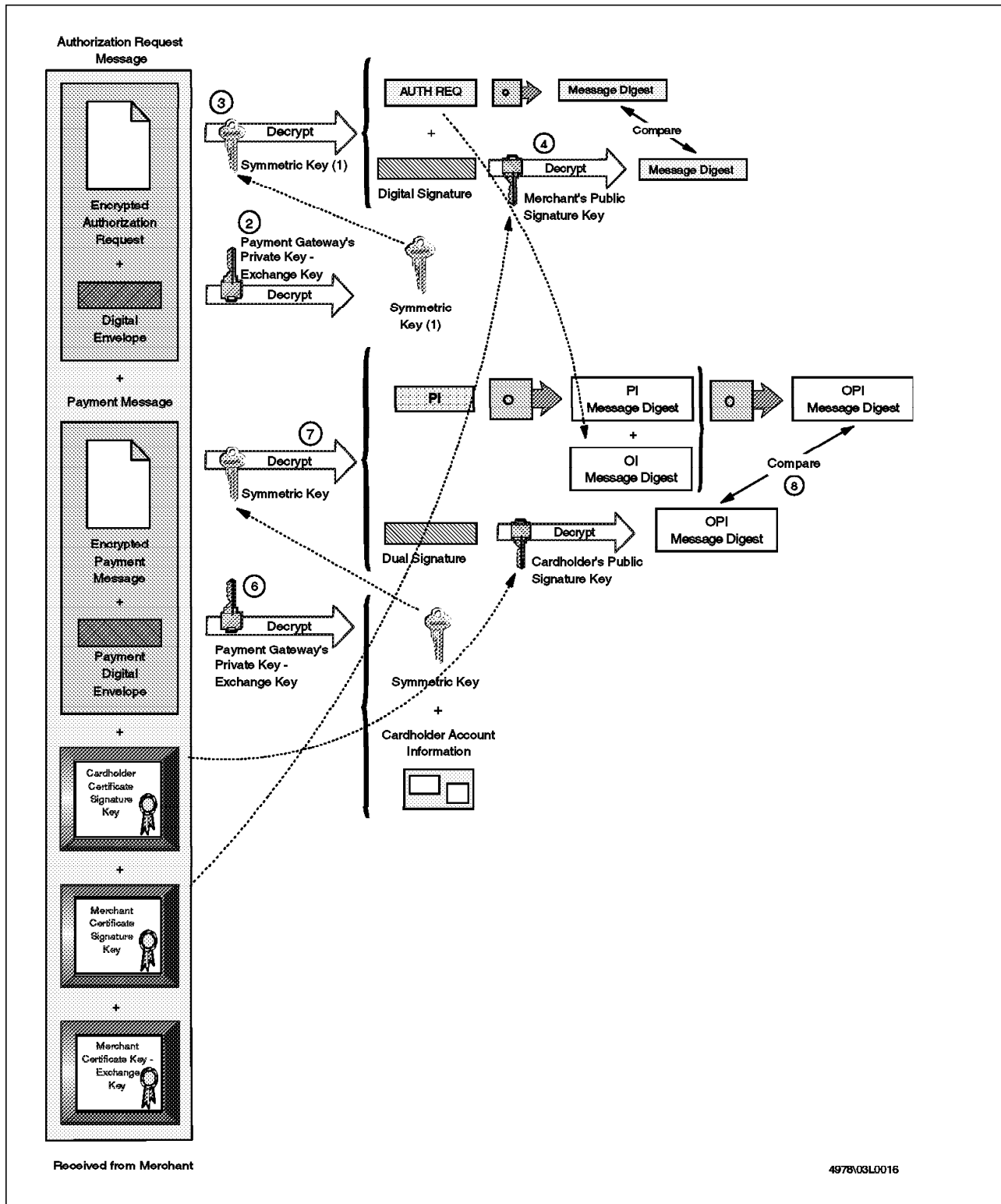


Figure 18. Payment Gateway Receives Authorization Request

The payment gateway formats and sends an authorization request to the appropriate bankcard association network. After receiving authorization from the card issuer, the payment gateway creates an authorization response. This response is in two sections, an authorization message containing the issuer's response and a capture token. (The payment gateway will need this information

to process the capture request.) Figure 19 on page 41 shows the components of the response.

1. The payment gateway creates the authorization response and passes it through a hash function to produce a message digest. This is then encrypted with the payment gateway's private signature key, creating a digital signature.
2. The authorization response and the digital signature are encrypted using a randomly generated symmetric key(2). The result is the encrypted authorization message.
3. Symmetric key(2) is encrypted with the merchant's public key-exchange key, generating the authorization digital envelope.
4. The payment gateway creates a capture token and encrypts it using a randomly generated symmetric key(3). The result of this is the encrypted capture token message.
5. Symmetric key(3) is encrypted with the payment gateway public key-exchange key, generating the capture token digital envelope.
6. The encrypted authorization message, authorization digital envelope, encrypted capture token message, capture token digital envelope and the payment gateway certificate including the signature key are sent to the merchant.

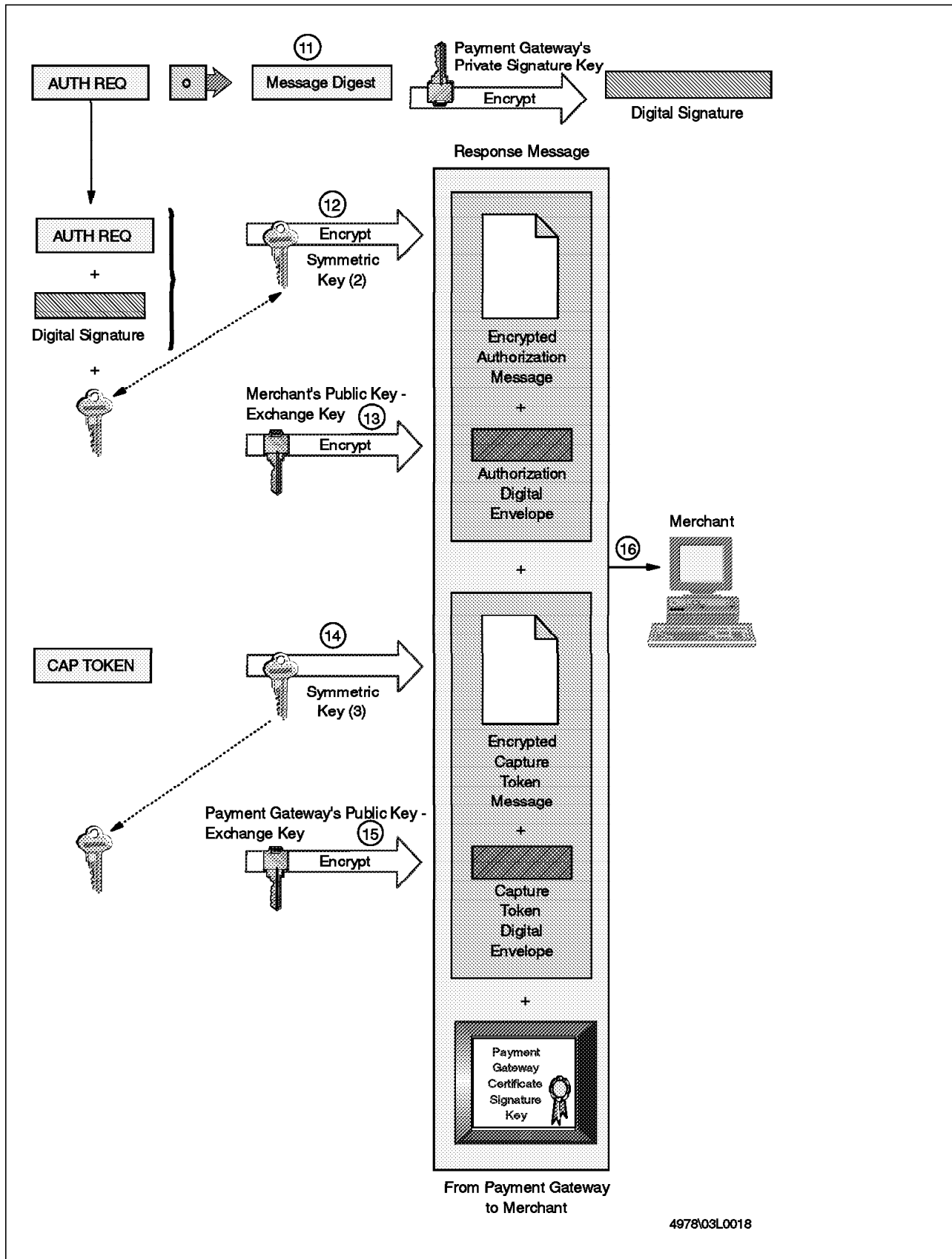


Figure 19. Payment Gateway Sends Authorization Response

3.3.2.3 Merchant Processes Response

Figure 20 on page 43 shows how the authorization response is processed by the merchant.

1. The merchant receives the response message and verifies the payment gateway certificate by traversing the trust chain to the root. The public key in this certificate will be used to check the digital signature included in the encrypted authorization message.
2. The merchant decrypts the authorization digital envelope using the merchant's private key-exchange key to obtain the symmetric key(2).
3. Symmetric key(2) is used to decrypt the encrypted authorization message to obtain the authorization response and digital signature.
4. The digital signature is verified by running the authorization response through a hash function and creating a message digest. The digital signature is decrypted using the payment gateway public signature key and the result is compared with the message digest obtained locally. If they are equal, the integrity of the message has been proved.
5. The merchant stores the encrypted capture message and capture digital envelope for later capture processing.

This now completes the SET processing for a purchase request and payment authorization. Next, the merchant must complete the process by shipping the goods or performing the services indicated in the order.

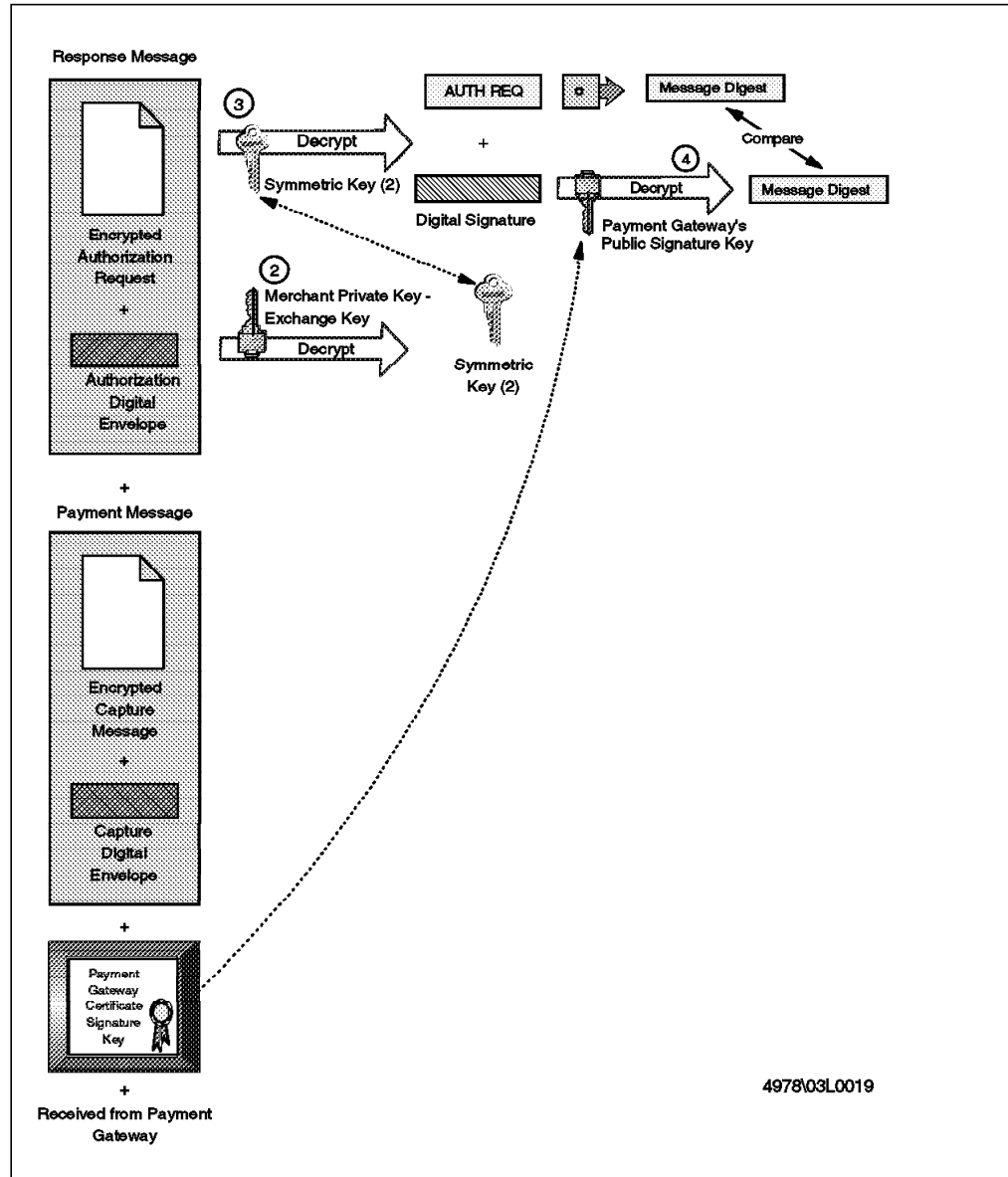
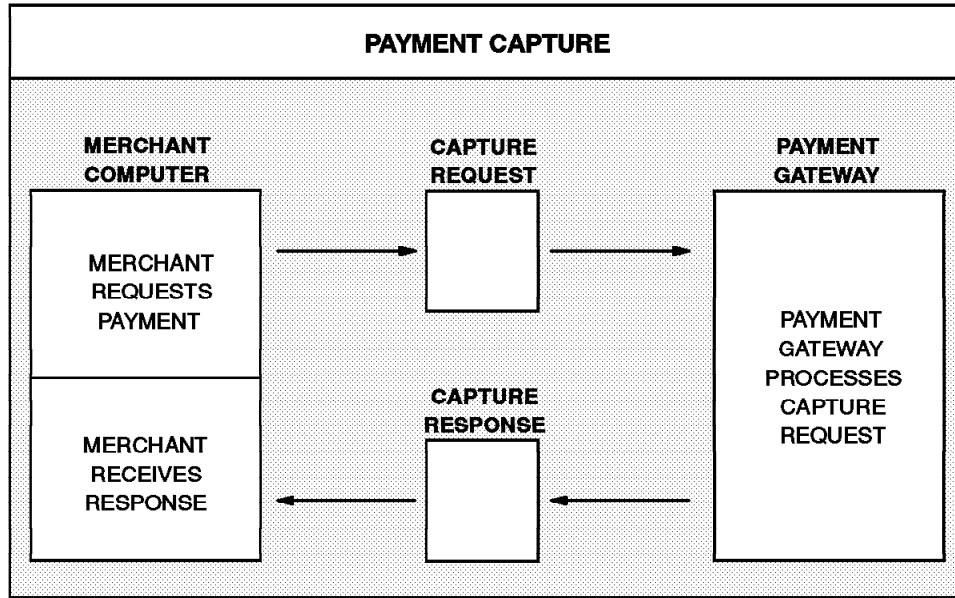


Figure 20. Merchant Receives Authorization Response

3.4 Payment Capture Process

At some point in time after the purchase transaction, the merchant will want to get paid. Figure 21 on page 44 shows at a high level how a merchant requests payment for the goods or services delivered to the cardholder.



4978\03L0020

Figure 21. Payment Capture Diagram

The process to request the payment begins when the merchant sends a capture request to the payment gateway including the amount of the transaction authorized. The payment gateway receives the request and sends it to the issuer (cardholder's financial institution) via a financial network. The payment gateway sends a capture response to the merchant that will store it to be used for reconciliation with payment received from the acquirer. We now describe this process in more detail.

3.4.1.1 Merchant Requests Payment

Figure 22 on page 45 shows the construction of the capture request.

1. The merchant software creates a capture request including the final amount to be authorized, the transaction identifier from the order instruction (OI) and other information related to the transaction.
2. The merchant digitally signs the capture request by passing it through a hash function. The message digest created is encrypted with the merchant's private signature key to create a digital signature.
3. The capture request and the digital signature are encrypted using a randomly generated symmetric key(4), resulting in the encrypted capture request message.
4. Symmetric key(4) is encrypted with the payment gateway public key-exchange key, generating the capture request digital envelope.
5. The capture token message, which was received from the payment gateway in the authorization response will be sent to the payment gateway together with the encrypted capture request message, capture request digital envelope, the two merchant certificates containing the signature key and the key-exchange key.

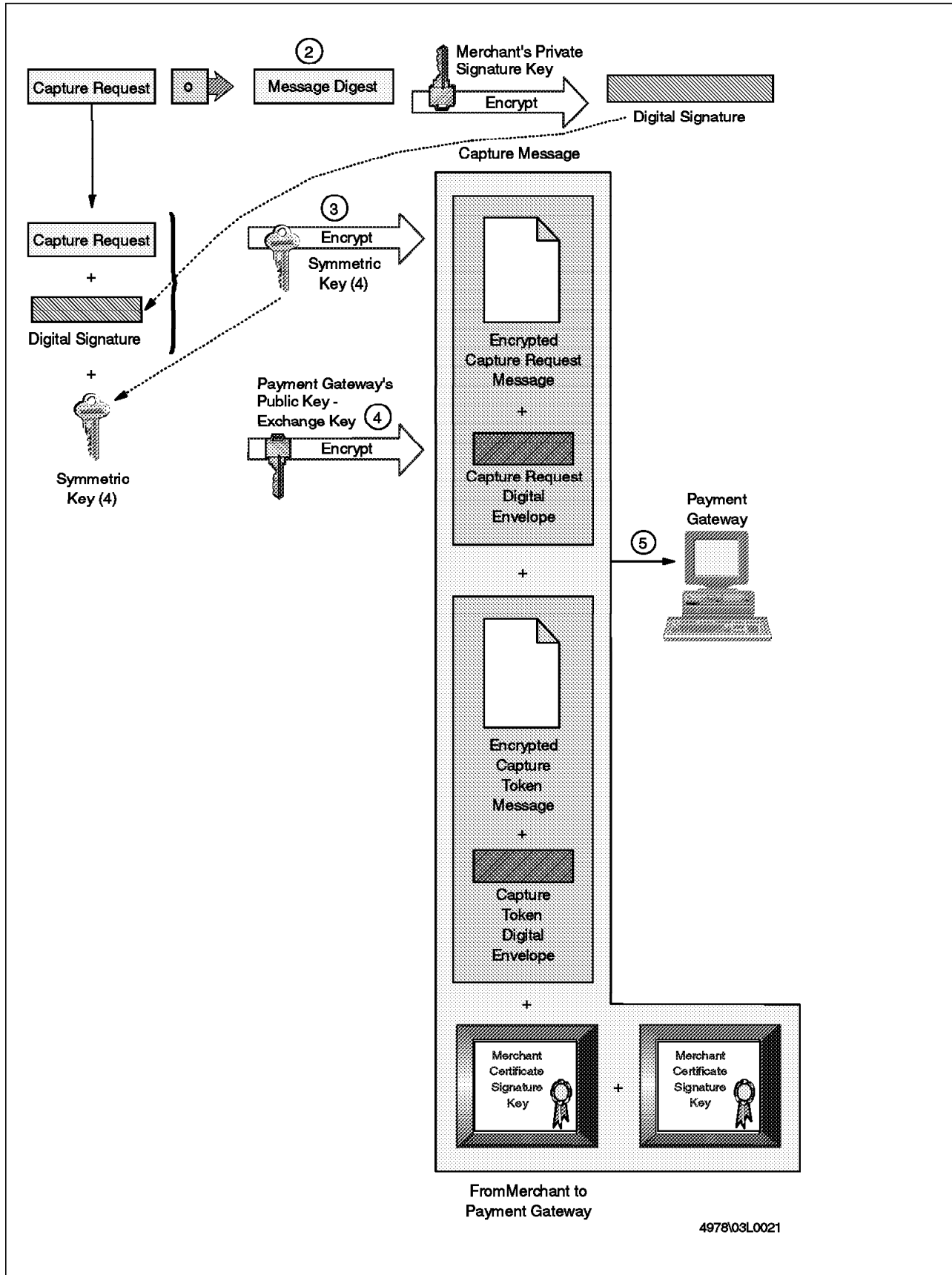


Figure 22. Merchant Payment Capture Request

3.4.1.2 Payment Gateway Processes Capture Request

When the payment gateway receives the capture request from the merchant, it must first unscramble the message, then initiate the funds transfer and inform the merchant of the result.

1. The payment gateway receives the capture message and verifies the merchant certificates by traversing the trust chain to the root.
2. The payment gateway decrypts the capture request digital envelope using the payment gateway private key-exchange key to obtain the symmetric key(4).
3. This symmetric key(4) is used to decrypt the encrypted capture request message to obtain the capture request and digital signature.
4. The digital signature is verified by running the capture request through a hash function and creating a message digest. The digital signature is decrypted using the merchant public signature key and the result is compared with the message digest obtained. If they are equal, the payment gateway is assured of the integrity of the message.
5. The payment gateway decrypts the capture token digital envelope using the payment gateway private key-exchange key to obtain the symmetric key(3).
6. This symmetric key(3) is used to decrypt the encrypted capture token message to obtain the capture token.

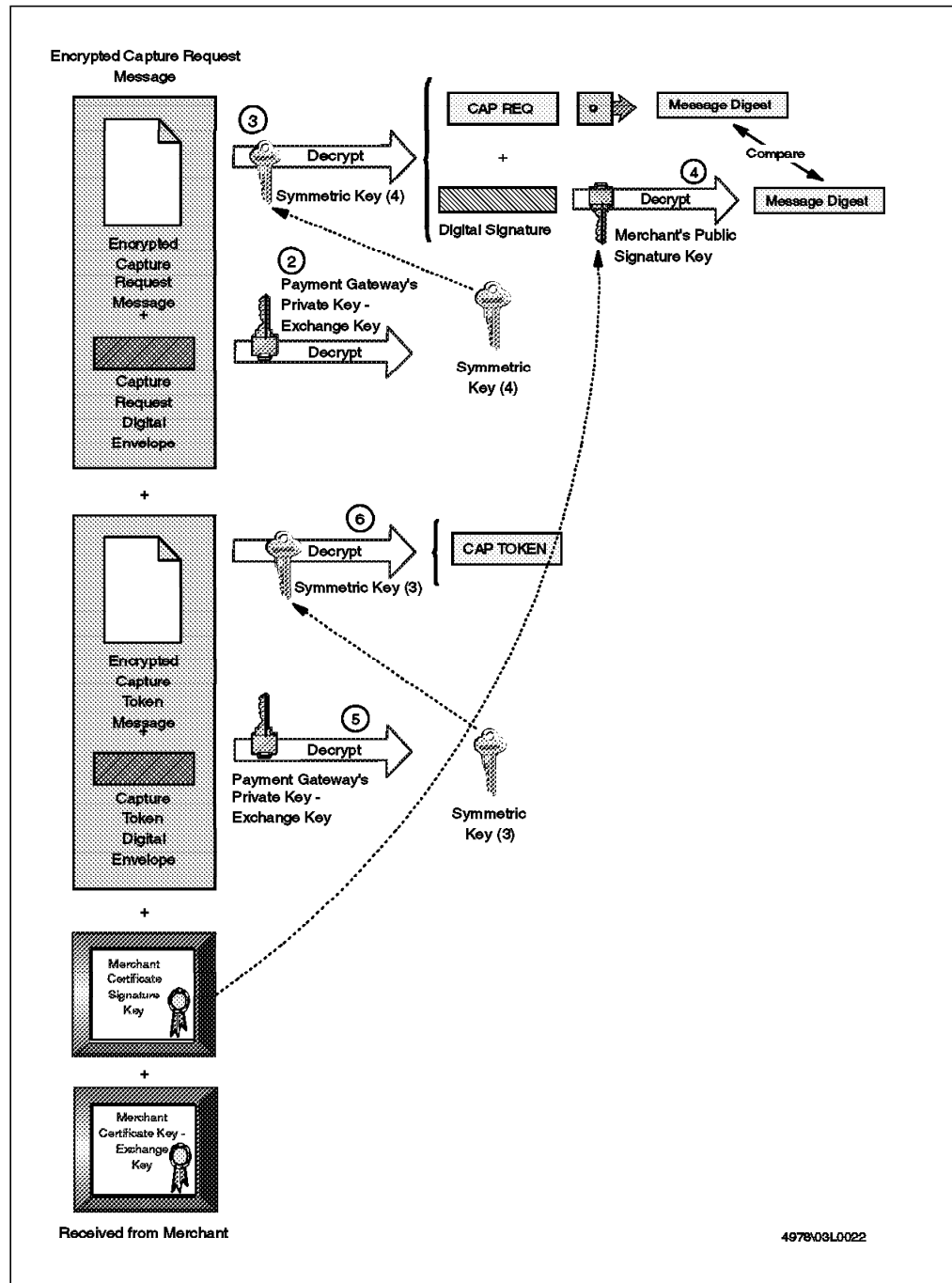


Figure 23. Payment Gateway Receives Capture Message

The payment gateway checks for consistency between the capture request and capture token. It then uses the information from the capture request and capture token to create a clearing request that is sent to the issuer using a private bank card network. This request causes funds to be transferred to the merchant's account.

Finally, the payment gateway has to send notification of the result of the capture operation back to the merchant.

1. The payment gateway creates a capture response and passes it through a hash function, creating a message digest. This message digest is encrypted with the payment gateway private signature key, creating a digital signature.

2. The capture response and the digital signature are encrypted using a randomly generated symmetric key(5). The result is the encrypted capture response message.
3. The symmetric key(5) is encrypted with the merchant public key-exchange key, generating the capture response digital envelope.
4. The capture response message including the encrypted capture response message, capture response digital envelope and payment gateway certificate including the signature key are sent to the merchant.

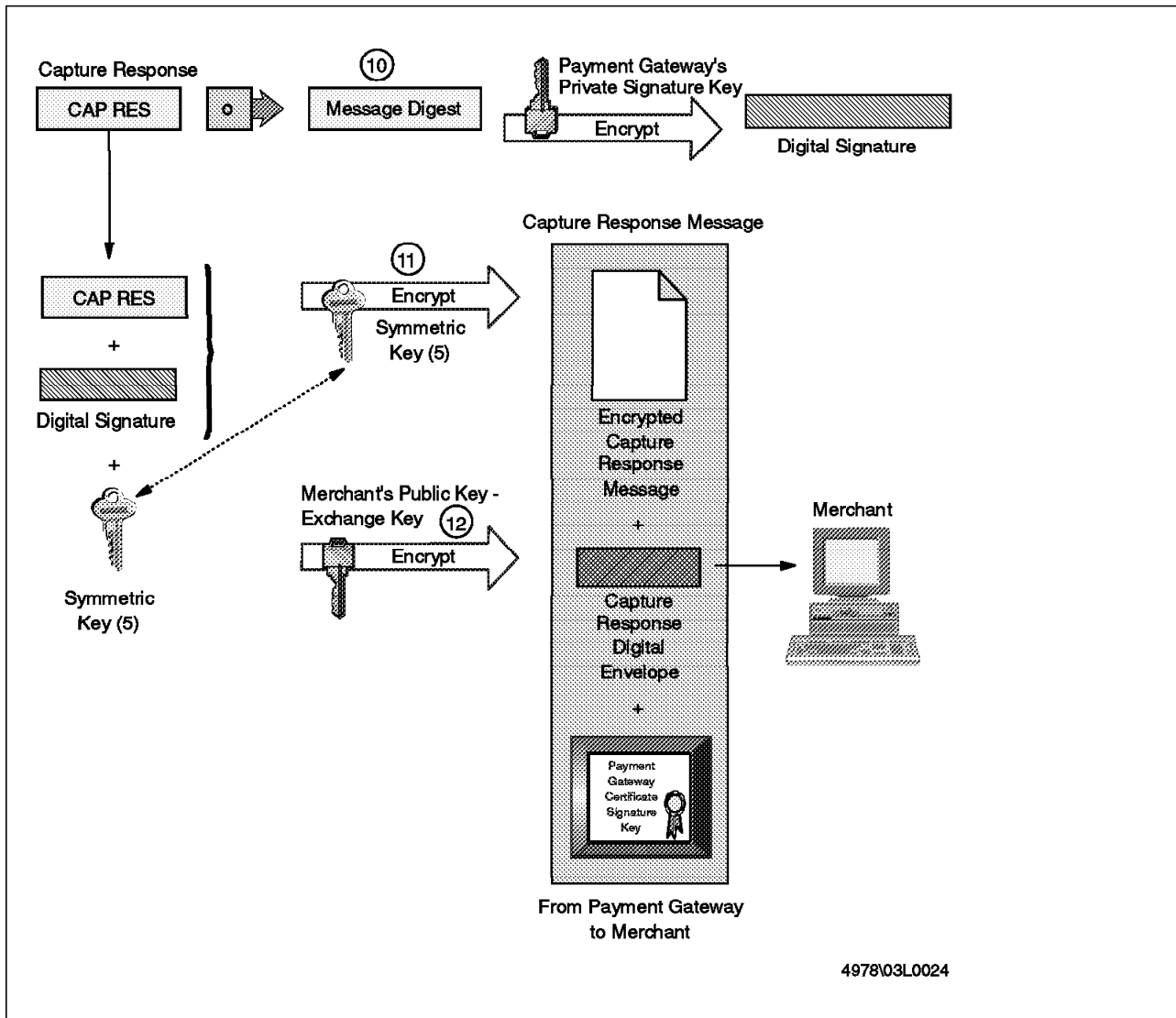


Figure 24. Merchant Sends Capture Response

3.4.1.3 Merchant Processes Response

The capture response is the merchant's proof that payment has been made. It needs to log the information contained in it in case of discrepancy.

1. The merchant receives the capture response message and verifies the payment gateway certificate by traversing the trust chain to the root. This certificate will be used to check the digital signature included in the encrypted capture response message.

2. The merchant decrypts the capture request digital envelope using the merchant private key-exchange key to obtain the symmetric key(5).
3. This symmetric key(5) is used to decrypt the encrypted capture response message to obtain the capture response and digital signature.
4. The digital signature is verified by running the capture response through a hash function to create a message digest. The digital signature is decrypted using the payment gateway public signature key and the result is compared with the message digest obtained locally. If they are equal, the message integrity has been assured.
5. The merchant stores the capture response to be used for reconciliation with payment received from the acquirer.

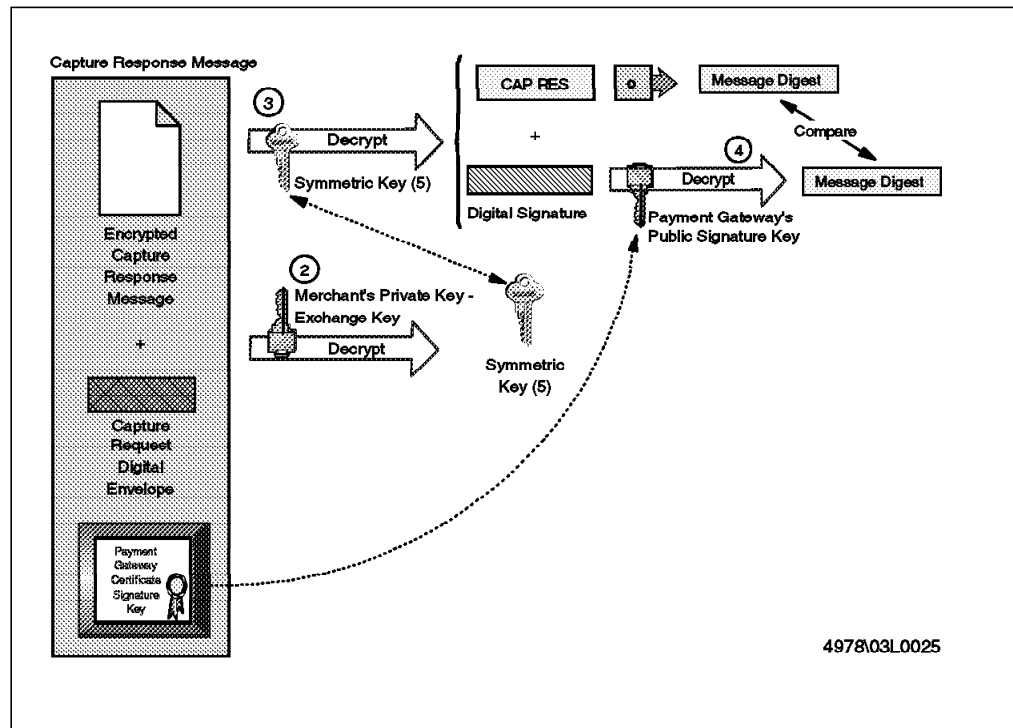


Figure 25. Merchant Receives Capture Response Message

Chapter 4. SET Certification

This chapter describes the SET certification process. It explains how a certificate works and shows in detail the registration processes for cardholders and a merchants to obtain a certificate.

4.1 Concept

When someone goes to a real store to buy a product, all the order and payment information (credit card, signatures, etc.) is presented to the merchant personally. The merchant knows that the signature belongs to the person that presented the card. That is not to say that the card necessarily belongs to the person who presented it, of course. But at least the merchant can check the signature against the one on the back of the card.

When a cardholder goes to a virtual store, the cardholder presents the merchant with credit card details and a digital signature as authentication. However, in this case the merchant cannot see the cardholder write the signature, and therefore cannot know that the signature is valid. The technique used to overcome this problem is the *public key certificate*. In the case of SET, the cardholder's certificate is effectively a statement from the credit card brand that says: "This public key belongs to this person who is a bona fide cardholder and you should trust any signature made using it." Similarly, the other SET roles (for example, merchant or payment gateway) also have certificates to authenticate their public keys.

We have said that the credit card brand is ultimately responsible for issuing certificates. They may run the certification operation directly, or contract it to some other company. The organization that performs the function is called a *certificate authority*.

Although the different SET roles all need public key certificates, they each require different levels of authentication. For example, it may be permissible to create a certificate for the cardholder online, with minimal checking, because the risk of an error is, financially, small. On the other hand, the impact of issuing a certificate for a fraudulent merchant could be much higher, so you would expect some stringent offline checks to be made. For this reason, SET defines separate certification authorities for each role.

4.2 Certificates

In a transaction using SET, all the members need a certificate in order to be authenticated. A certificate must be issued for the cardholder, merchant and payment gateway (in fact, the protocol allows for a cardholder to operate without a certificate, but initial implementations are overriding this option).

4.2.1 Cardholder Certificates

The Cardholder Certificate links the holder of the payment card to the account details (account number and expiration date). The account information is a secret, and SET goes to great lengths to reveal it only to the payment gateway and not the merchant (see the distinction between PI and OI in 3.3.1.3, “The Cardholder Sends Request” on page 28). With this in mind, it is clear that the account information cannot be placed in the cardholder certificate in clear. In fact, the account information and a secret value only known by the cardholder are run through a hash function to generate a digest that is included in the certificate. When the cardholder presents this certificate or while it is travelling through the network, nobody can see the account information and the secret value, because they were encoded using a one-way hashing algorithm. In this way the certificate can be used to verify that the cardholder is valid, but not to learn any account information.

However, when payment authorization is requested the acquirer must be able to check that the credit card account really does belong to the cardholder. The payment instruction (PI) SET protocol message includes the cardholder account information and the secret value. This message is encrypted so that it can only be decrypted by the acquirer payment gateway. This means that the payment gateway can verify the digest in the signature and establish the link between the cardholder and his or her account details, while still hiding the information from the rest of the world.

4.2.2 Merchant Certificates

This certificate represents a relationship between the merchant and a financial institution, allowing it to accept a payment card brand. These certificates are approved by the acquiring financial institution and provide assurance that the merchant has a valid agreement with an acquirer.

The merchant needs two public key pairs for SET, one for digital signatures and one for encrypting key exchanges. It will therefore need two certificates for each payment card brand that it accepts.

4.2.3 Payment Gateway Certificates

The payment gateway receives its certificates from the acquirer. During the process of purchasing, the cardholder encrypts the payment information using the public key of the payment gateway. This key was extracted from the payment gateway certificate.

4.3 Hierarchy of Trust

We have seen that a certificate ties a public key to some information about the owner of that key. In order for the certificate to be acceptable it has to be signed by some trusted third party. Certificates are created by certificate authorities (CAs). In SET there are a number of different types of certificate authorities, for the various roles within the protocol. Each CA type may occur many times, based on credit card brand, location or usage.

It would be possible to require each SET participant (cardholder, merchant, acquirer, etc.) to know about all of the CAs, but this would be impractical and an administrative nightmare. In fact, there is a *hierarchy of trust* among the SET certificate authorities. Each certificate is linked to the CA that digitally signed it.

A merchant certificate is digitally signed by the Merchant Certification Authority (MCA). The MCA certificate is digitally signed by the Brand Certificate Authority (BCA) and the BCA certificate is digitally signed by the Root Certificate Authority (RCA). When a SET participant presents its certificate, it includes the complete certificate chain, up to the root. The recipient of the message can follow this chain and, as long as it has the root certificate, can establish trust in the certificate.

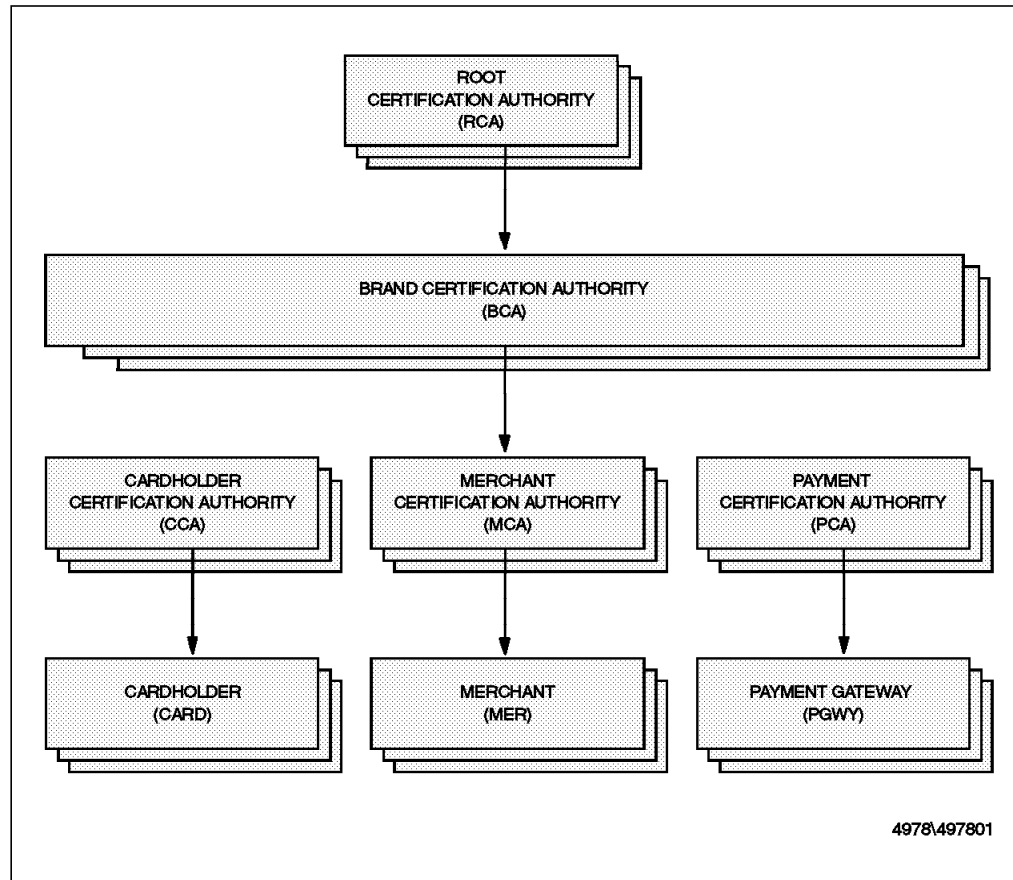


Figure 26. Certification Hierarchy (Original Design)

4.3.1 Root Key

Under the original SET design, there is one single root key. The private key would be kept offline and all brands would be below it in the hierarchy of trust. Visa and Mastercard have nominated CertCo to take the role of root key-holder, but current SET projects are each using separate root keys during the evaluation phase..

The root key is a self-signed certificate and if it is necessary to confirm this key, a system must send an inquiry to the certificate authority containing the hash (message digest) of the root certificate. The response will indicate whether the digest matches the current root, or whether it matches an old root key that has been replaced.

When a root key pair is created, a replacement key pair is also created and stored securely. The self-signed root certificate contains the root public key and a hash of the replacement public key (see Figure 27 on page 54).

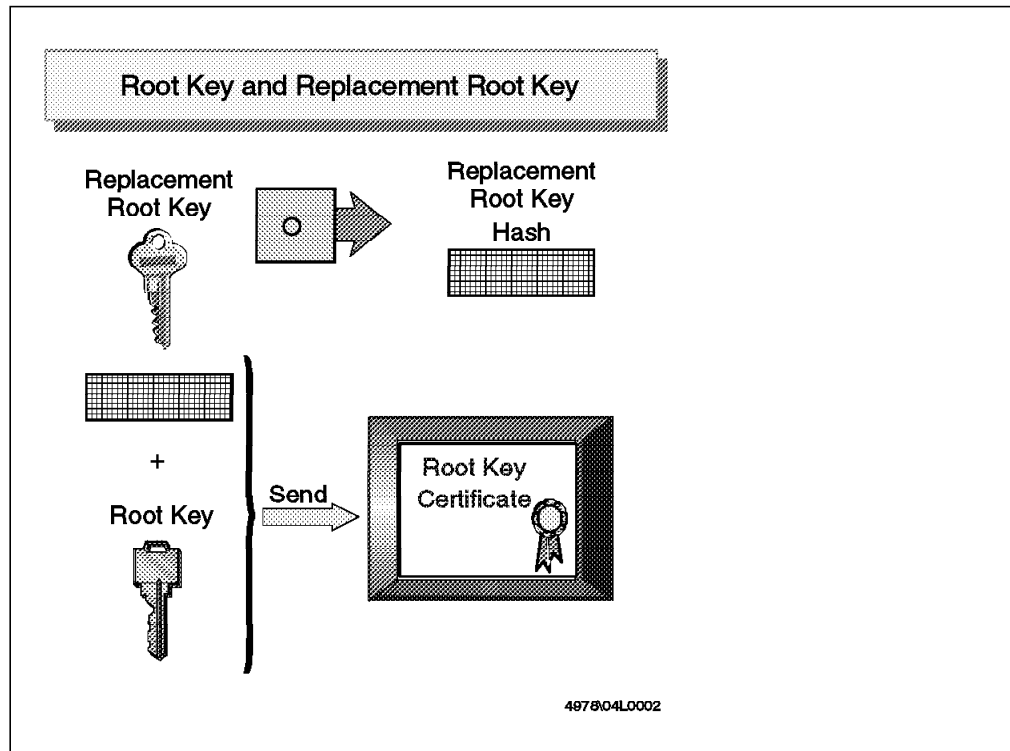


Figure 27. Root Certificate

If, for any reason, it is necessary to replace the root key, the root CA will generate a new replacement key pair (a replacement for the replacement). Then it will create a new root certificate in the same way as before, except that the old replacement key is now the current key and the newly generated key is now the replacement.

This new certificate then needs to be communicated to all SET participants. For payment gateways, which are in frequent communication with their CA (for certificate revocation lists for example), this is not a problem. The merchants and cardholders do not frequently have to refer to the CA, however. The root CA may send out a replacement for the root certificate in e-mail messages, but otherwise the first thing the cardholders and merchants will know about the problem will be when payment requests start to fail because the payment gateway cannot resolve the signatures on their certificates back to the root. Then they will have to access their own CA to receive the new root key, request a new certificate and get back in business. Clearly, root key replacement is not something that will happen very frequently.

When any of the SET players request the replacement root key they will receive a self-signed certificate of the replacement key and a hash of the next replacement key. They will confirm that the certificate of the replacement key is valid by passing it through the hash function and comparing the result with the hash received with the original root certificate (see Figure 28 on page 55).

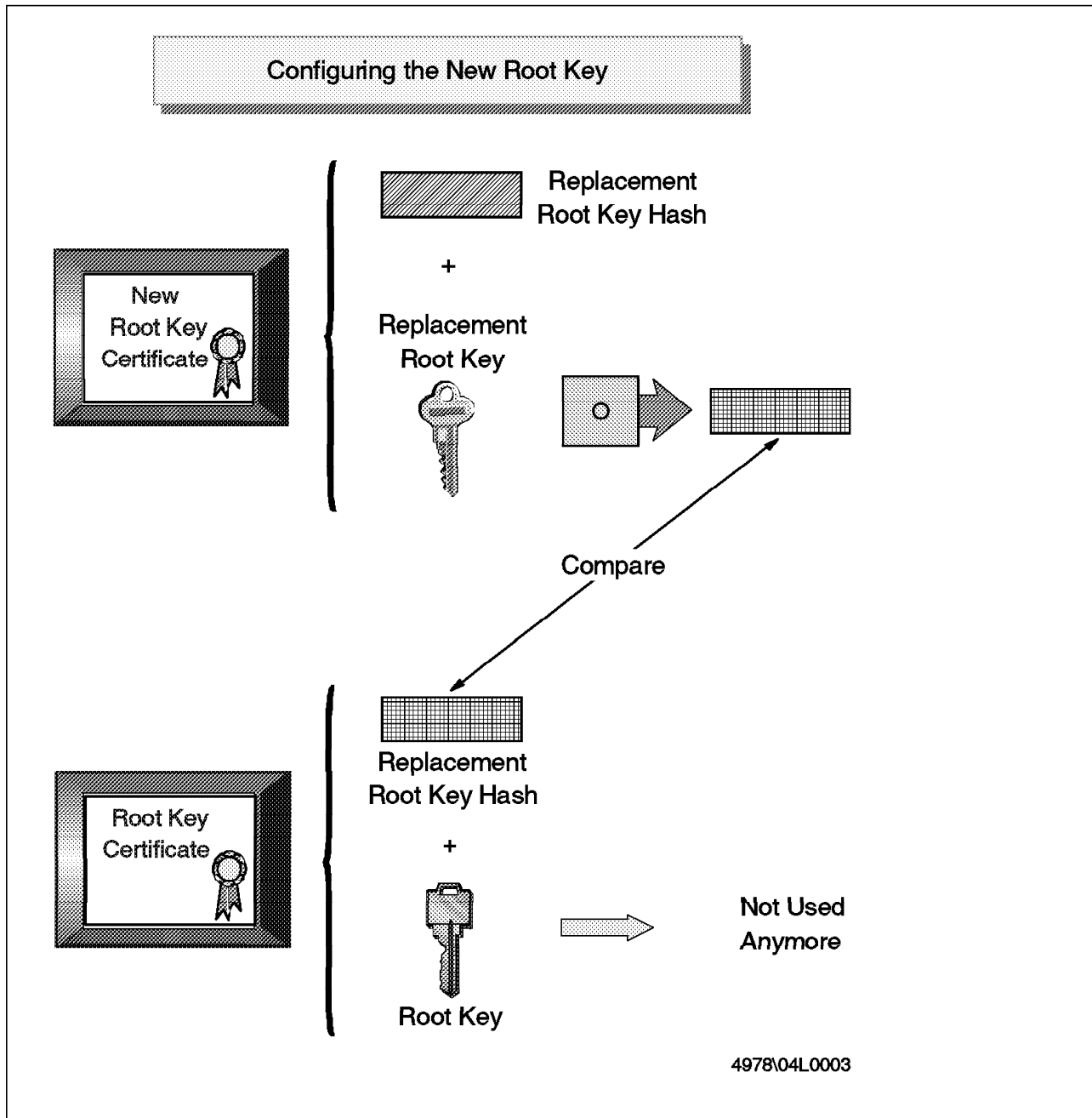


Figure 28. Replacing the Root Certificate

4.4 SET Protocols For Issuing Certificates

The main responsibilities of the certificate authority are:

1. Receive requests to issue a certificate
2. Approve or reject the requests
3. Send certificates

The following sections explain in detail the process of issuing a certificate to a cardholder and to a merchant.

4.4.1 Cardholder Certification Process

Figure 29 is a high-level look at how a cardholder certificate is issued.

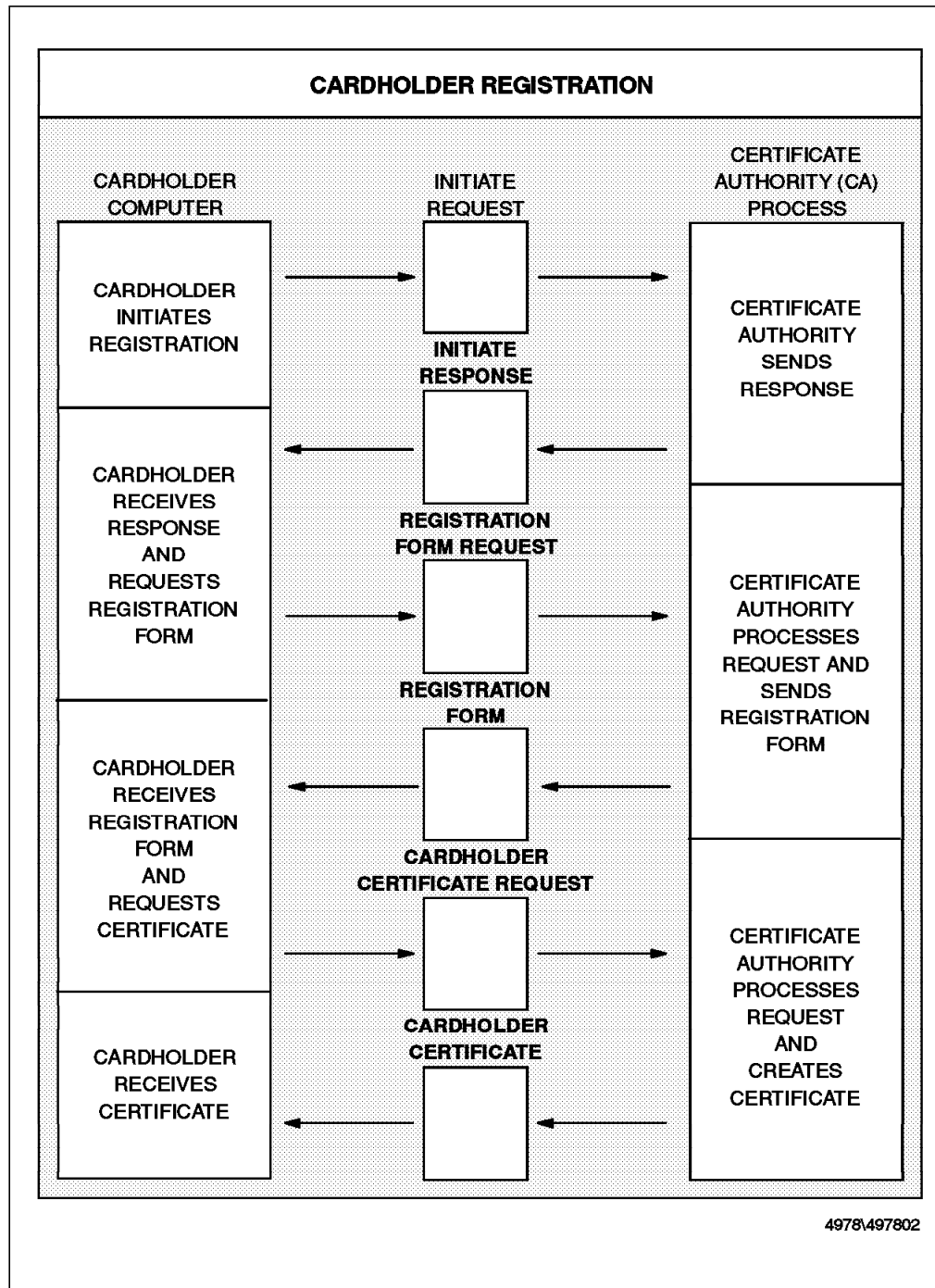


Figure 29. Cardholder Registration

The process to obtain a certificate begins when the cardholder sends an initiate request to the certificate authority. The CA receives the initiate request and sends an initiate response. This response includes the CA certificates. The cardholder receives the initiate response and requests the registration form. The CA receives the request and sends the registration form to the cardholder. The cardholder fills out the registration form and creates a pair of keys (public and private key). The cardholder sends the registration form, his or her public

key and the certificate request to the CA. The CA verifies the cardholder data filled in the registration form and if it is approved, the CA creates the certificate including the cardholder public key, and sends to the cardholder. The cardholder receives and stores the certificates for future electronic commerce.

We will now describe the certificate request flow in detail.

4.4.1.1 Cardholder Initiates Registration

The cardholder software creates a request to ask for a copy of the CA's certificates and sends it to the CA.

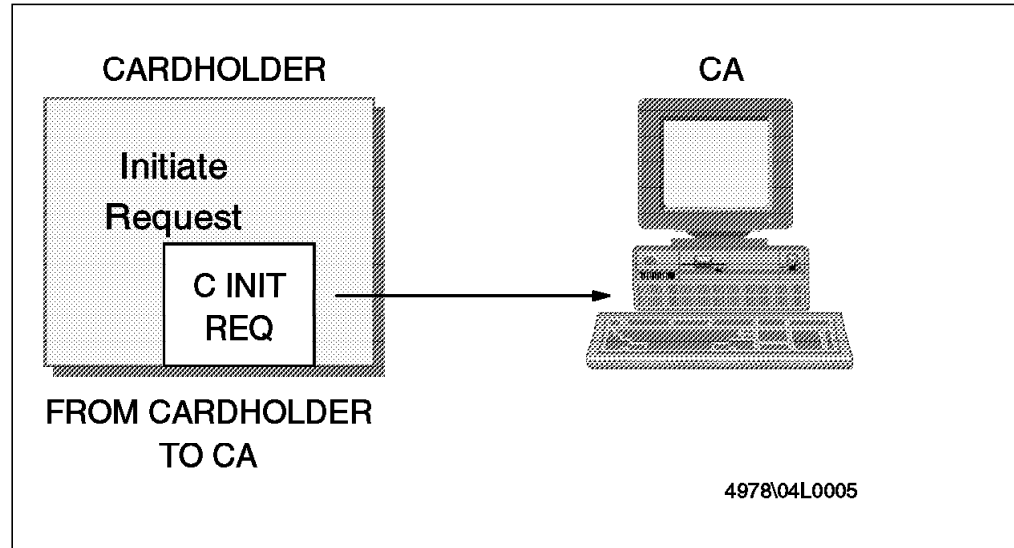


Figure 30. Cardholder Initiates Registration

4.4.1.2 CA Sends Response

The CA receives the cardholder initiate request and transmits its two certificates, for the public signature key and public key-exchange key. These certificates will be used to protect the payment card account number in the registration form request. Figure 31 on page 58 shows the process.

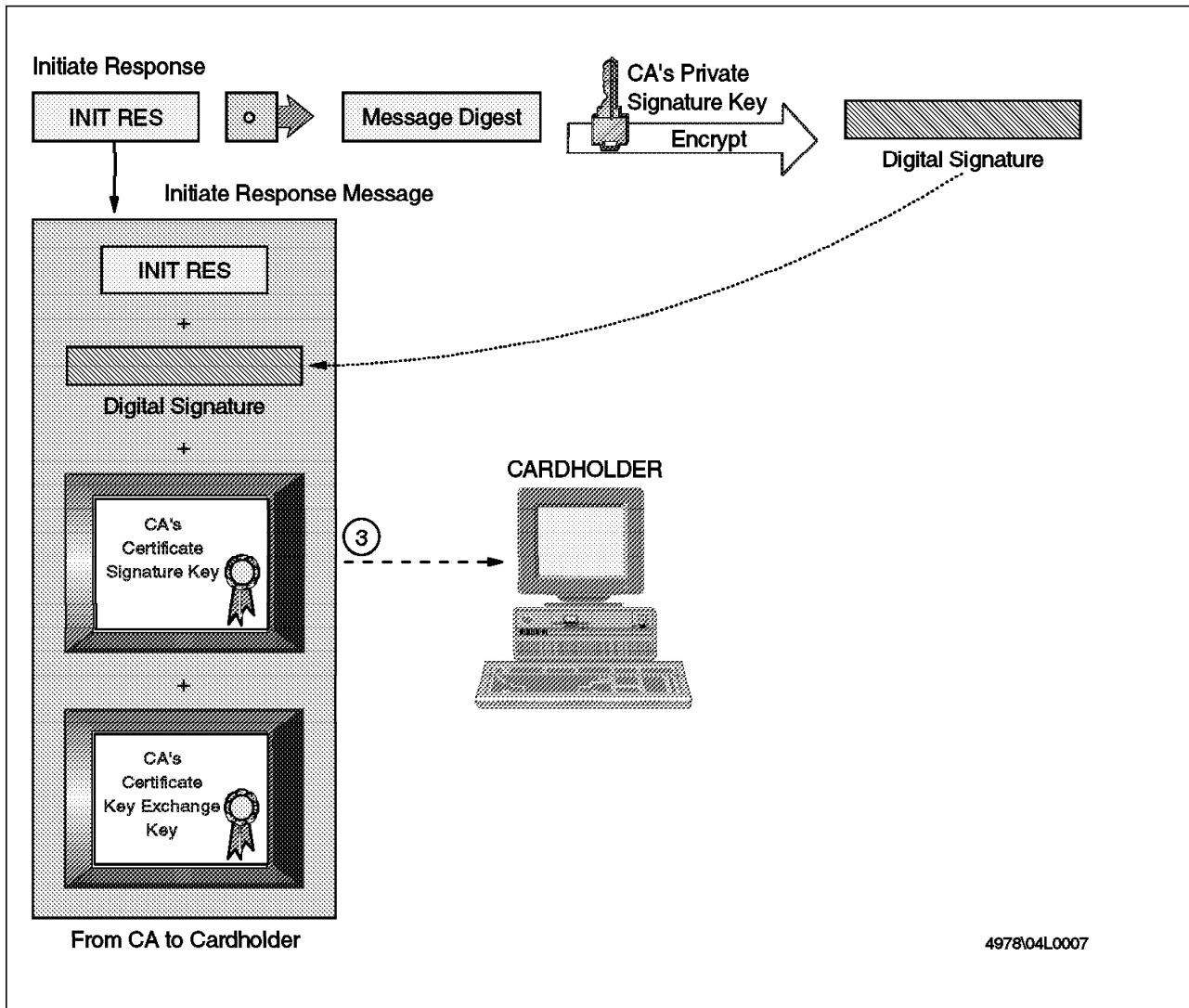


Figure 31. CA Sends Initiate Response

1. The CA receives the initiate request.
2. The CA generates a response and digitally signs it by passing the response through a hash function. The message digest so created is encrypted with the CA private signature key, resulting in a digital signature.
3. The CA sends the initiate response, the digital signature and the two CA certificates containing the public signature key and public key-exchange key to the cardholder.

4.4.1.3 Cardholder Requests Registration Form

The CA will require some details so that it can validate that the cardholder is who he or she claims to be. Instead of having a hard-coded registration form, the form is supplied by the CA on request.

1. The cardholder receives the initiate response message from the CA and verifies the certificates by traversing the trust chain to the root.
2. The CA signature is verified by running the initiate response through a hash function and creating a message digest. The digital signature is decrypted using the CA public signature key and the result is compared with the

message digest obtained locally. If they are equal, the integrity of the CA response has been assured.

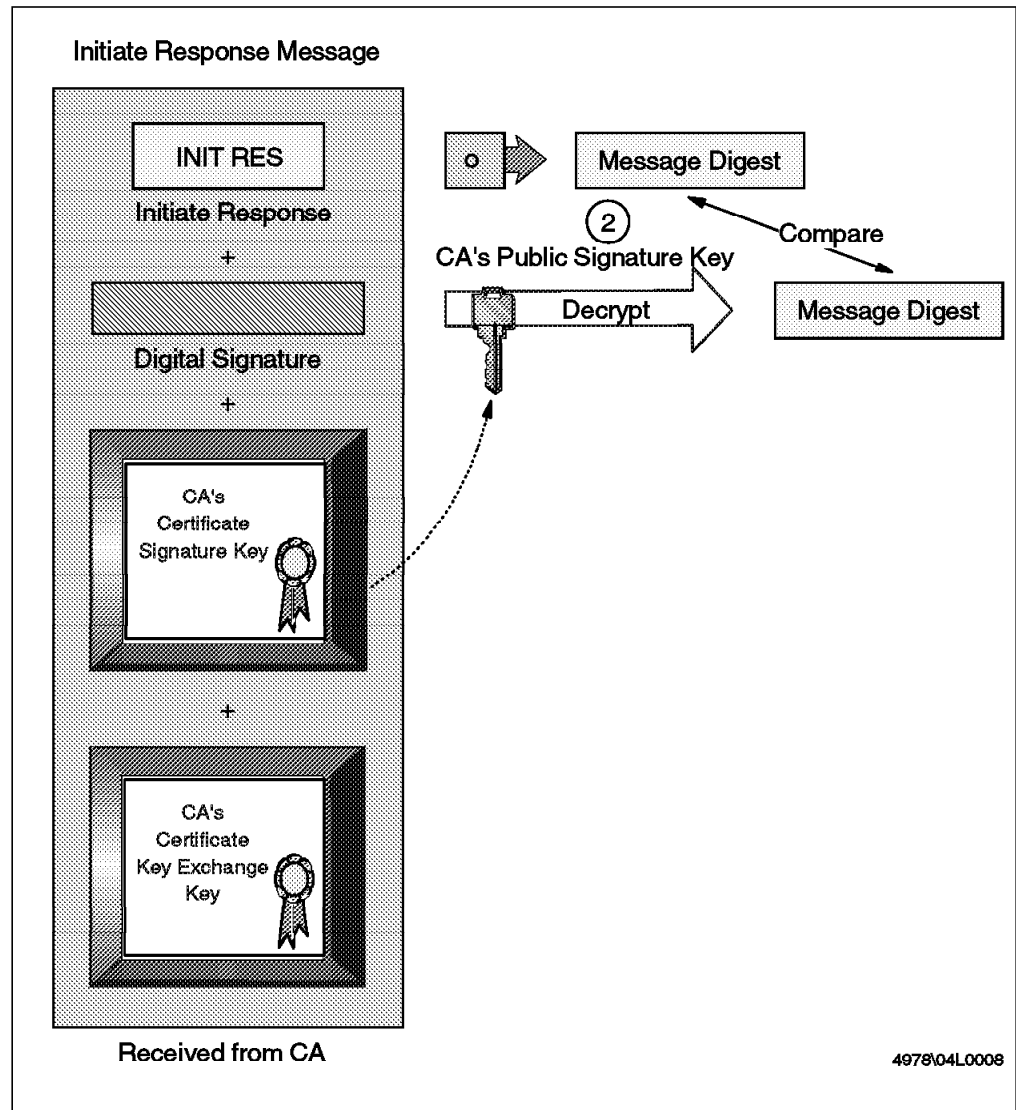


Figure 32. Cardholder Receives Initiate Response

3. The cardholder enters the credit card account number and the cardholder software then generates a registration form request.
4. The registration form request is encrypted with a randomly generated symmetric key. This symmetric key and the cardholder's account number are encrypted with the CA public key-exchange key, generating a digital envelope.
5. The digital envelope and the encrypted registration form request are sent to the CA.

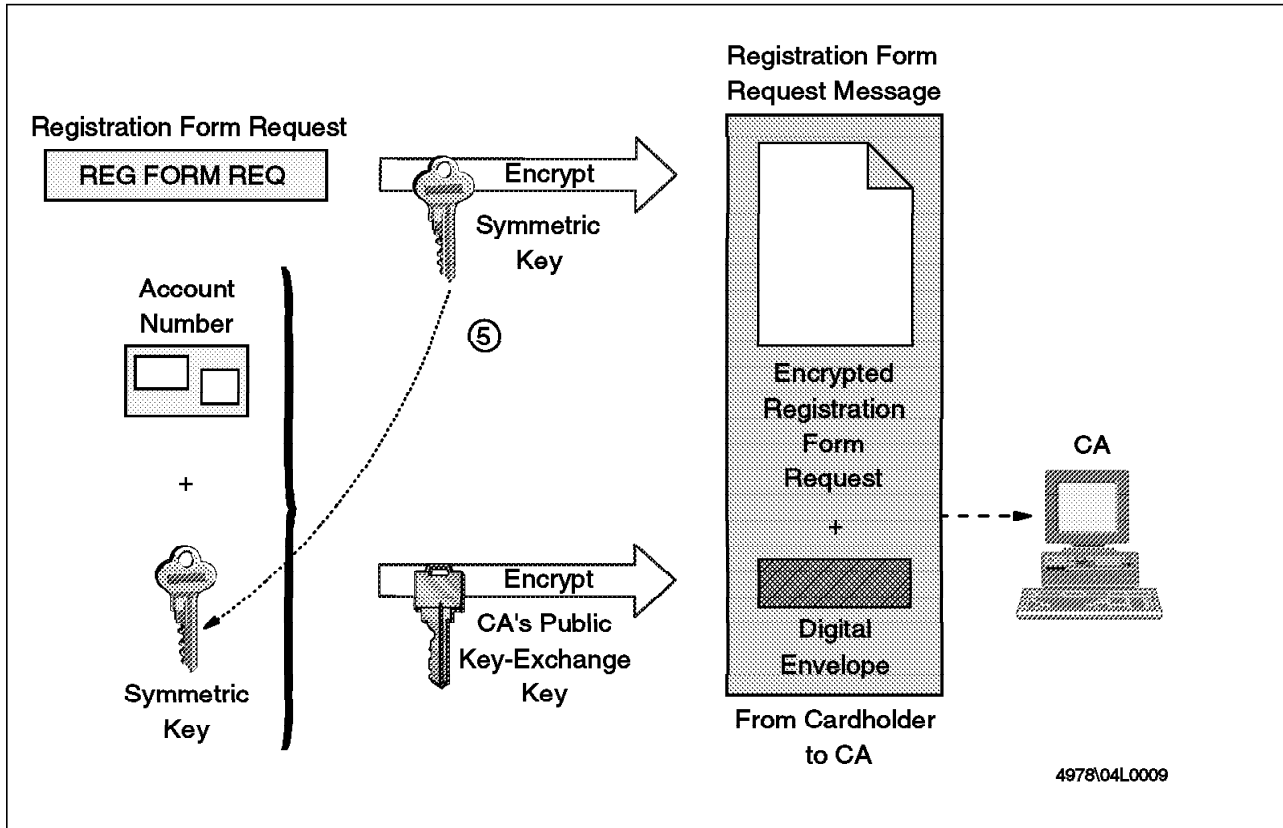


Figure 33. Cardholder Requests Registration Form

4.4.1.4 CA Sends Registration Form

The registration form is sent in the clear, but the CA signs the message so that the cardholder can be sure of its authenticity.

1. The CA receives the registration form request message and decrypts the digital envelope with the CA private key-exchange key to obtain the cardholder's account number and the symmetric key.
2. The symmetric key is used to obtain the registration form request by decrypting the encrypted registration form request.

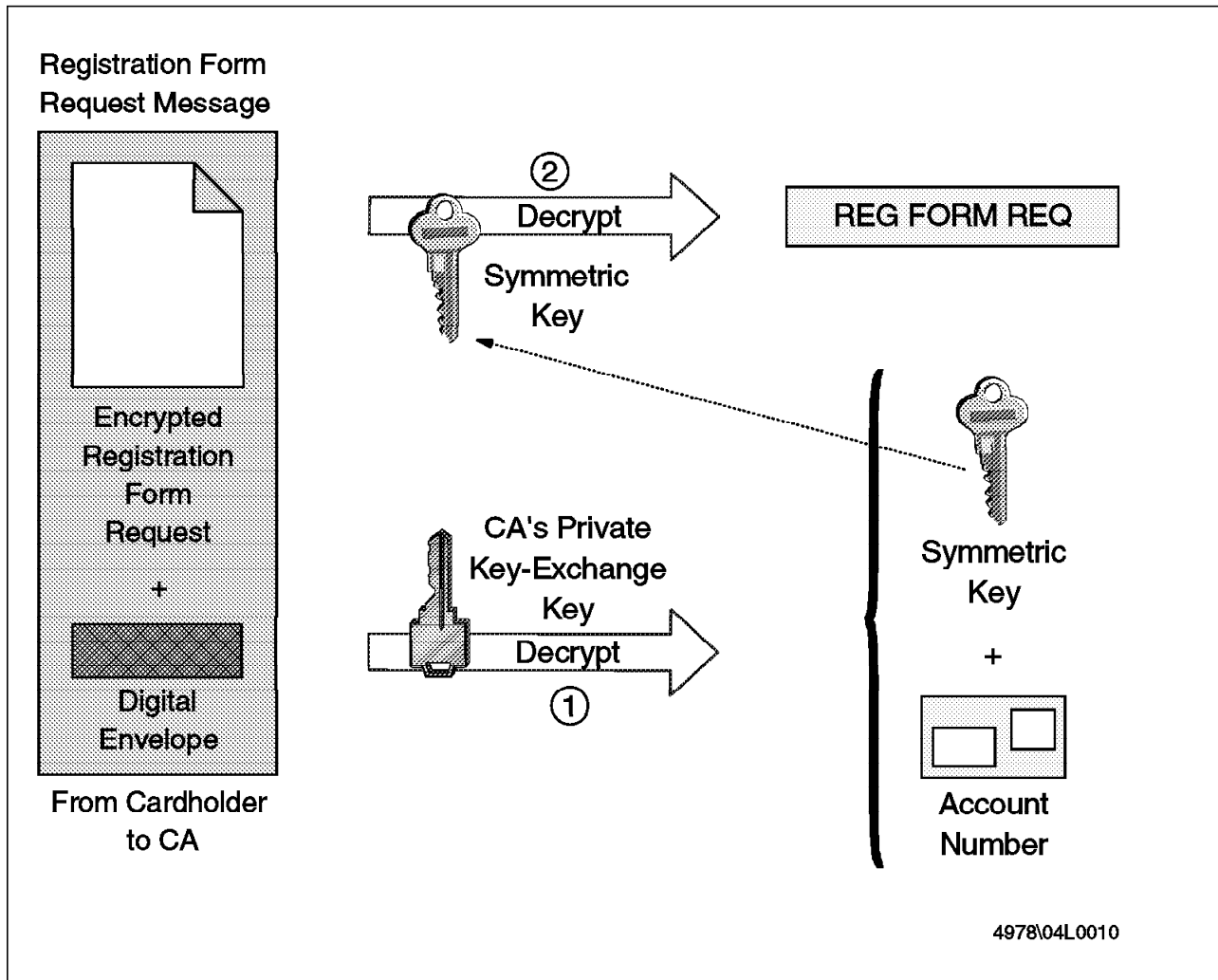


Figure 34. CA Receives Registration Form Request Message

3. The CA determines the registration form appropriate for this user and digitally signs it by passing it through a hash function. The message digest so created is encrypted with the CA private signature key, resulting in a digital signature.
4. The CA sends the registration form, the digital signature and the CA certificate containing the public signature key to the cardholder.

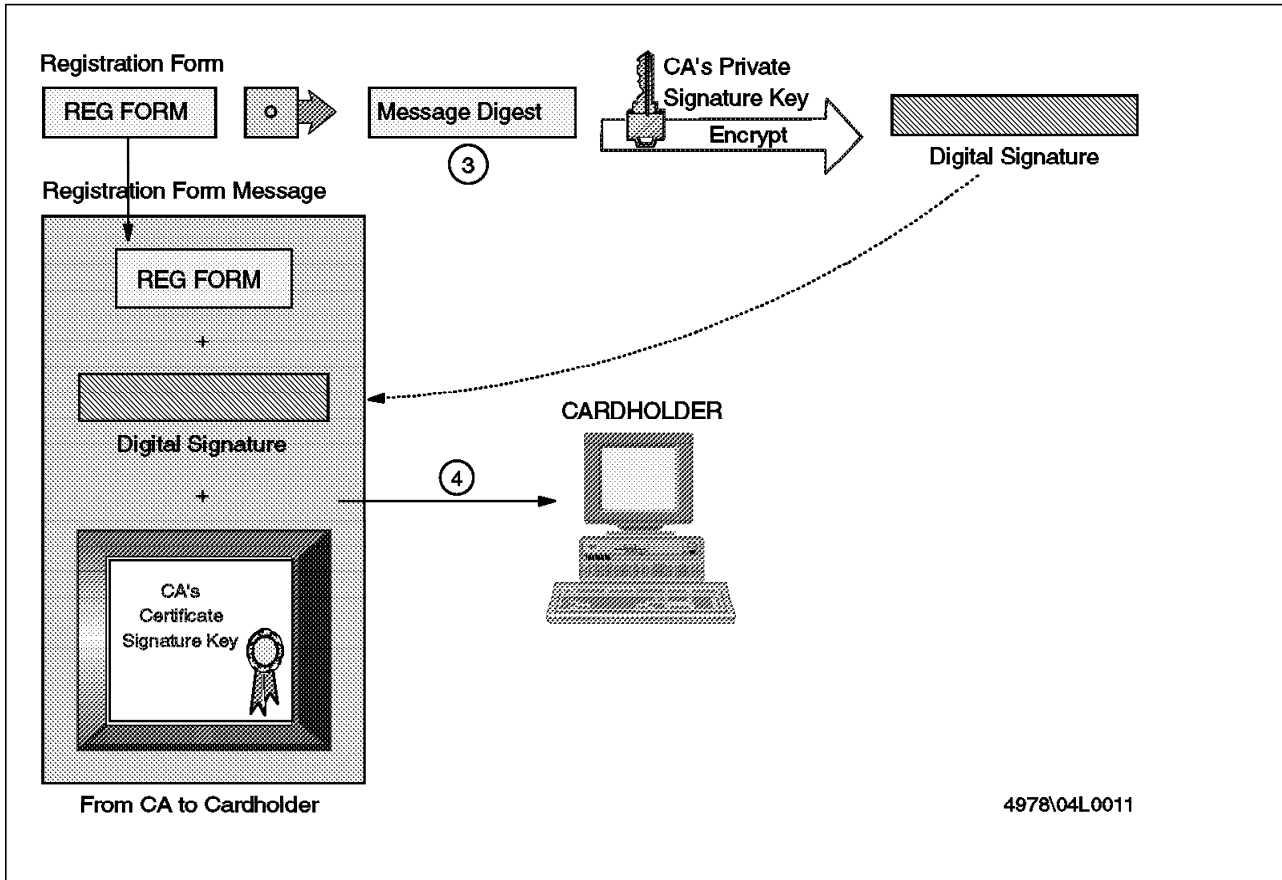


Figure 35. CA Sends Registration Form

4.4.1.5 Cardholder Requests Certificate

Now the cardholder software can present the request form and generate a request message. Figure 36 on page 63 and Figure 37 on page 64 show the process.

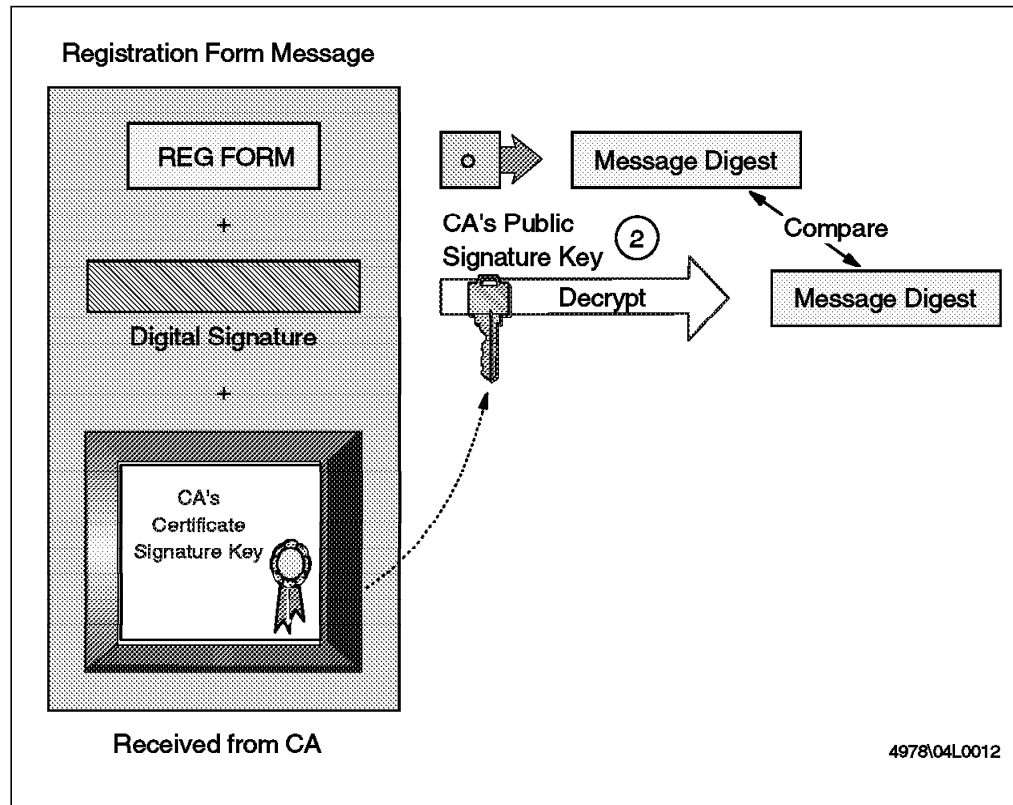
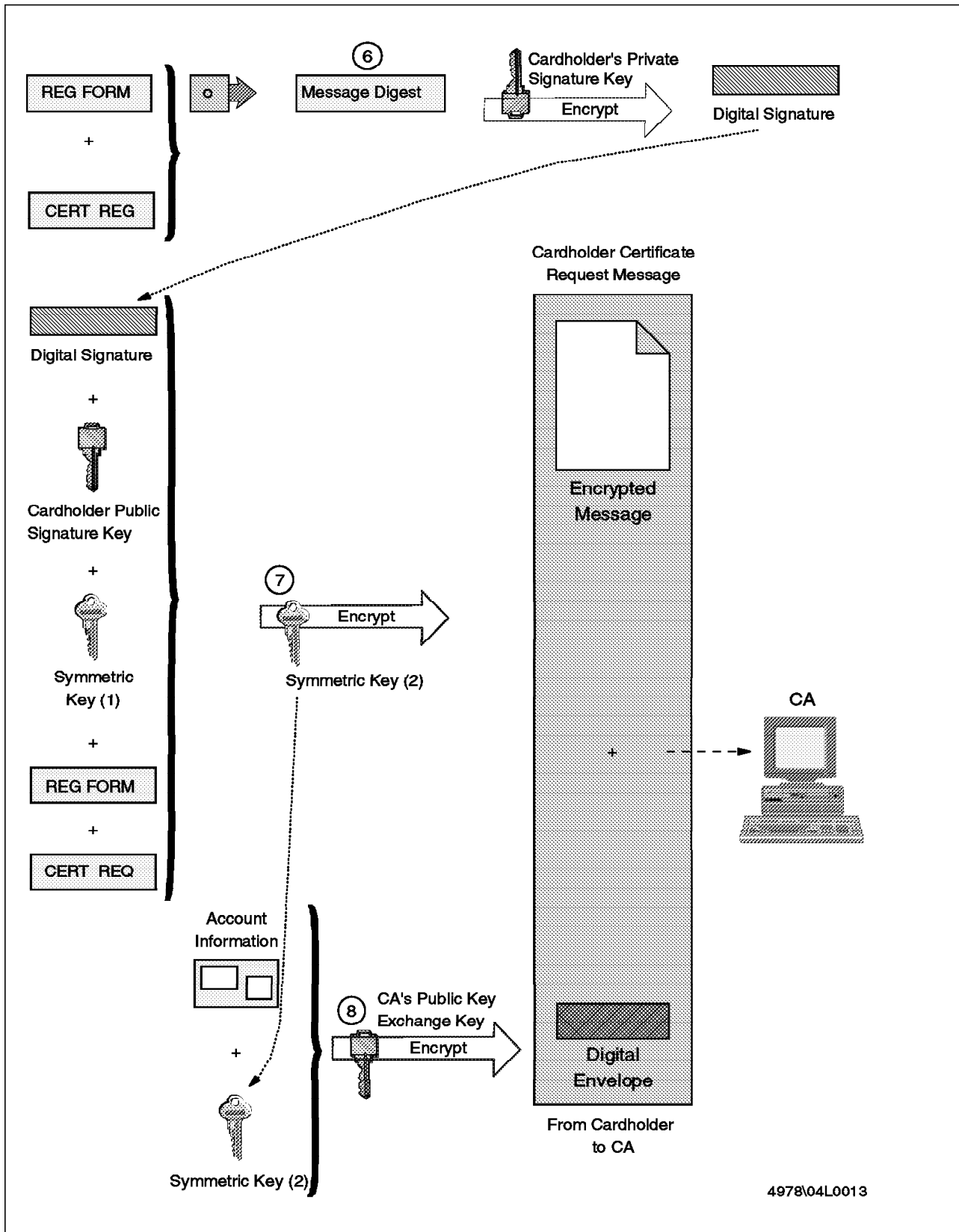


Figure 36. Cardholder Receives Registration Form

1. The cardholder receives the registration form message from the CA and verifies the certificates by traversing the trust chain to the root.
2. The CA signature received is verified by running the registration form through a hash function and creating a message digest. The digital signature is decrypted using the CA public signature key and the result is compared with the message digest obtained locally. If they are equal, the integrity of the message is assured.



497804L0013

Figure 37. Cardholder Requests Certificate

1. The cardholder software generates a key pair (public and private signature keys). It stores them securely on disk or other storage (such as a smart card, for example).
2. The cardholder completes the registration form and submits it. To this data, the cardholder software adds a random number that will be used by the CA to create the certificate.
3. The cardholder software generates a certificate request message using the information filled in on the registration form and the random number.
4. The certificate request and registration form are digitally signed by passing through a hash function. The message digest created is encrypted with the cardholder private signature key, creating a digital signature.
5. The cardholder creates two randomly generated keys. (We call them symmetric key (1) and symmetric key (2).)
6. The cardholder takes the registration form and certificate request, the digital signature, the cardholder public signature key, and symmetric key (1) and encrypts them using the other symmetric key (2).
7. The symmetric key (2), cardholder account information, expiration date and random number are encrypted with the CA public key-exchange key, generating a digital envelope.
8. The digital envelope and the encrypted message are sent to the CA.

4.4.1.6 CA Receives Request

Now the certificate authority must unscramble the message sent by the cardholder and process the registration request. Figure 38 on page 66 shows the process.

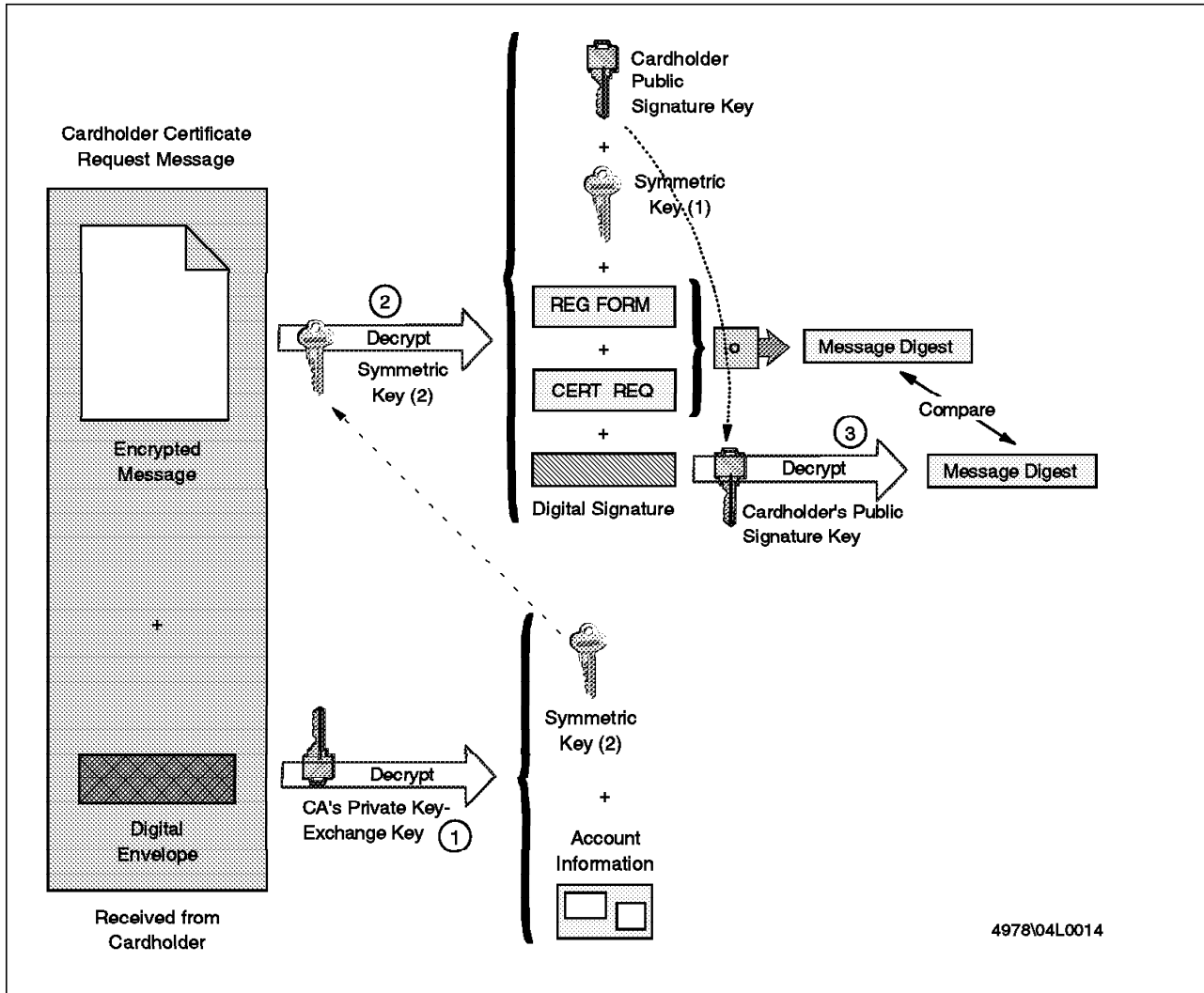


Figure 38. CA Receives Request

1. The digital envelope is decrypted using the CA private key-exchange key, revealing the symmetric key (2), account information and the random number.
2. The encrypted message is decrypted using symmetric key (2) to obtain the cardholder public signature key, symmetric key (1), the certificate request, registration form, and the digital signature.
3. The cardholder signature received is verified by running the certificate request and registration form through a hash function and creating a message digest. The digital signature is decrypted using the cardholder public signature key and the result is compared with the message digest obtained locally. If they are equal, the integrity of the message has been assured.

Now the CA verifies the certificate request by checking the cardholder account and registration form details against the brand's database of cardholder information. If the information is valid, the CA can create and send the certificate, as shown in Figure 39 on page 67.

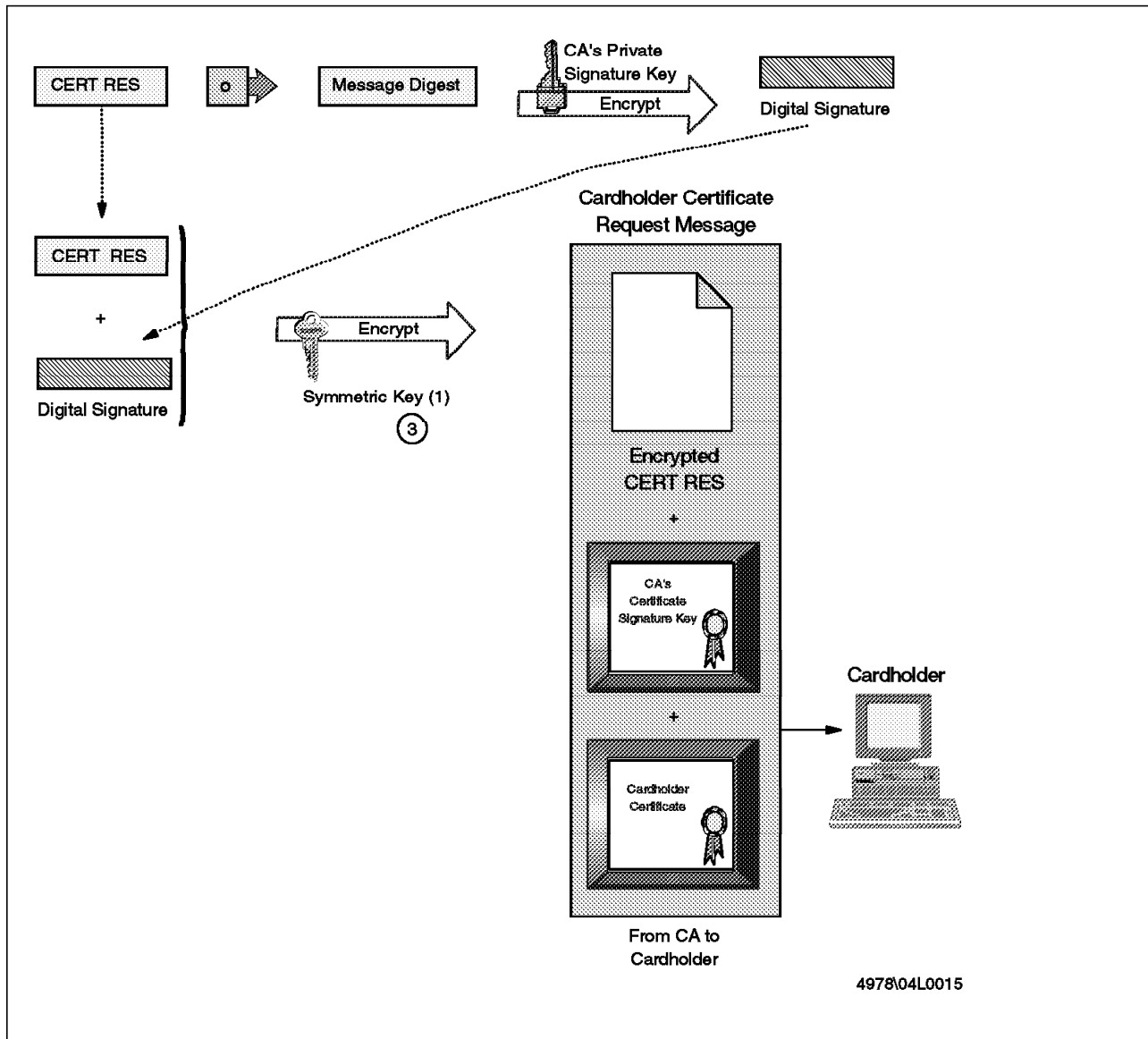


Figure 39. CA Sends Certificate

1. First the CA generates a random number that is combined with the random number created by the cardholder software to make a secret value. The account number, expiration date and secret value are encoded using a one-way hashing algorithm. The result is placed in the cardholder certificate. The certificate is then digitally signed by the CA using its private signature key.
2. The certificate is placed in a certificate response message that is digitally signed by passing through a hash function. The message digest so created is encrypted with the CA private signature key resulting in a digital signature.
3. The CA encrypts the digital signature and the certificate response containing the random number generated by the CA and other information (such as brand logo) with symmetric key (1) that it received from the cardholder. The resulting message, CA certificate and cardholder certificate are then sent to the cardholder.

4.4.1.7 Cardholder Receives Certificate

The cardholder checks the validity of the certificate it receives and saves it (see Figure 40).

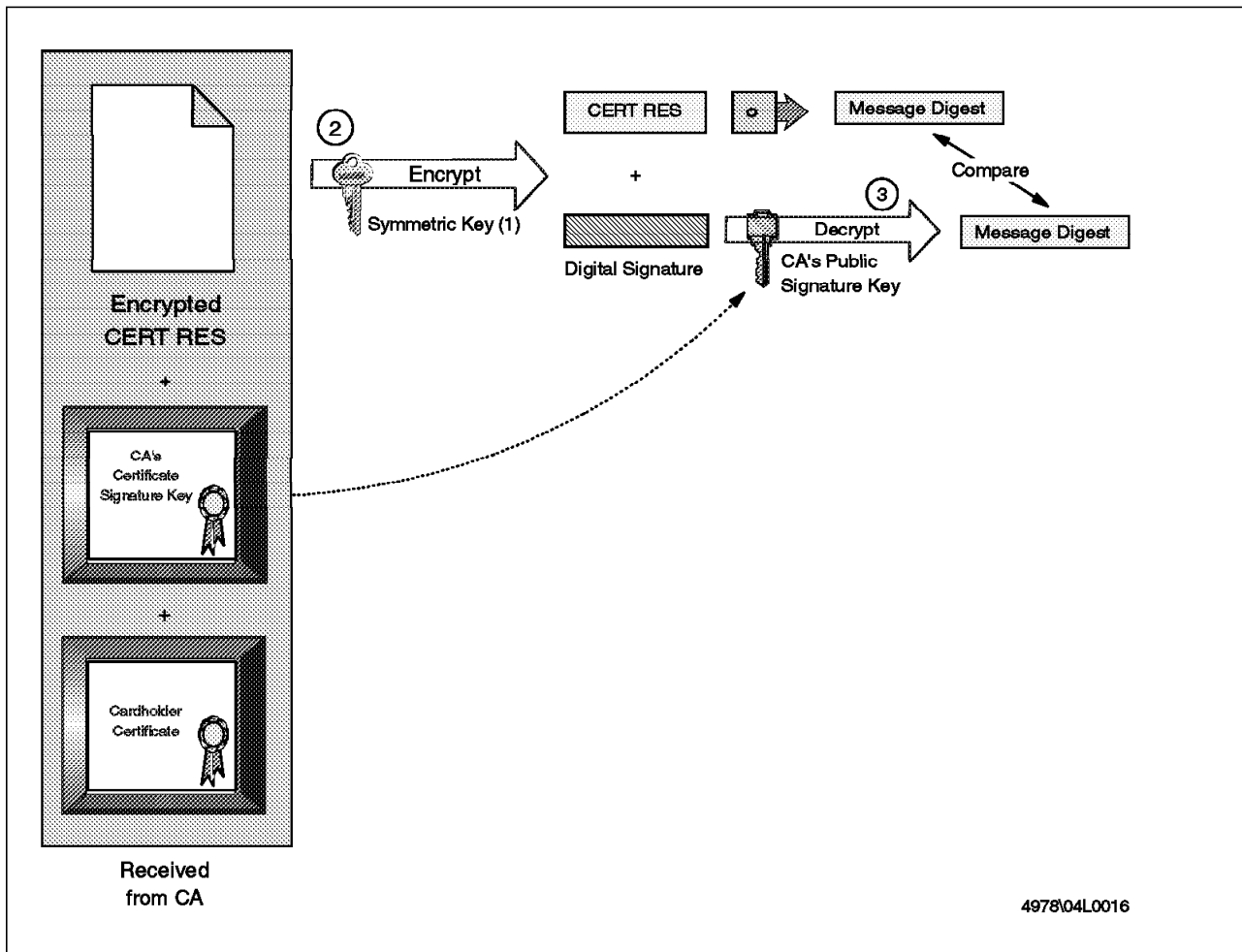


Figure 40. Cardholder Receives Certificate

1. The cardholder verifies the certificates by traversing the trust chain to the root.
2. The encrypted certificate response is decrypted using symmetric key (1), obtaining the certificate response and digital signature. The cardholder software already knows symmetric key (1) because it created it in the first place (see 4.4.1.5, "Cardholder Requests Certificate" on page 62). It combines the random number sent in the registration form with the random number in the certificate response to determine the secret value. This value is stored to use with the certificate.
3. The CA signature received is verified by running the certificate response through a hash function and creating a message digest. The digital signature is decrypted using the CA public signature key and the result is compared with the message digest obtained locally. If they are equal, the integrity of the message has been assured.
4. The cardholder software stores the certificate securely, on disk or other medium, for future use. The cardholder is now ready to go shopping.

4.4.2 Merchant and Acquirer Certification Process

The certification process for merchants and acquirers is very similar to that for cardholders, but there are some significant differences, due to the nature of the parties involved. Figure 41 shows the high-level process flow.

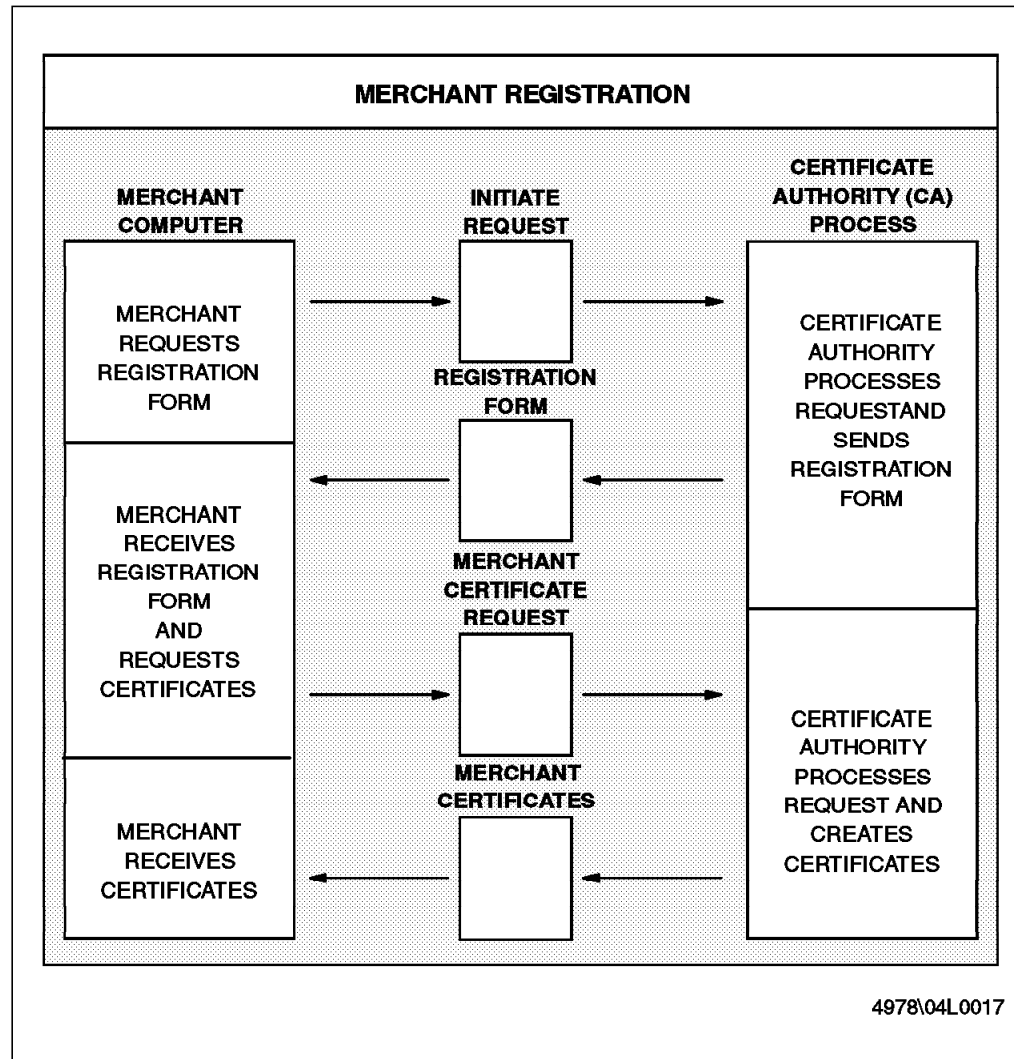


Figure 41. Merchant Registration Diagram

In our explanations we refer to the merchant, but the process for the acquirer is the same.

During an electronic transaction, the cardholder must verify if the merchant that is selling the products or services is trusted. To prove this, the merchant must have a certificate generated by a certificate authority for its valid public key. This certificate is sent to the cardholder during the purchase process. The cardholder will verify the certificate and check that the CA that signed it is a valid certificate authority. The merchant will also use its certificate to process transactions through a payment gateway.

The process to obtain a certificate begins when the merchant requests the registration form and sends an initiate request to the certificate authority. The CA receives the request and sends the registration form to the merchant. The merchant fills the registration form and creates two key pairs (public and private

keys). The merchant sends the registration form, the two public keys and a certificate request to the CA. The CA verifies the merchant data filled in the registration form and if it is approved, the CA creates the certificates and sends them to the merchant. The merchant receives and store the certificates for future electronic commerce.

The major differences from the cardholder case are:

- The merchant has two key pairs (a pair for signatures and another for key exchanges).
- The process for cardholder certification is likely to take place totally online, whereas in the merchant case there will always be offline verification.

We now step through the process in detail.

4.4.2.1 Merchant Initiates Registration

The merchant creates a request to ask for a copy of the CA's certificates and registration form. This request is sent to the CA.

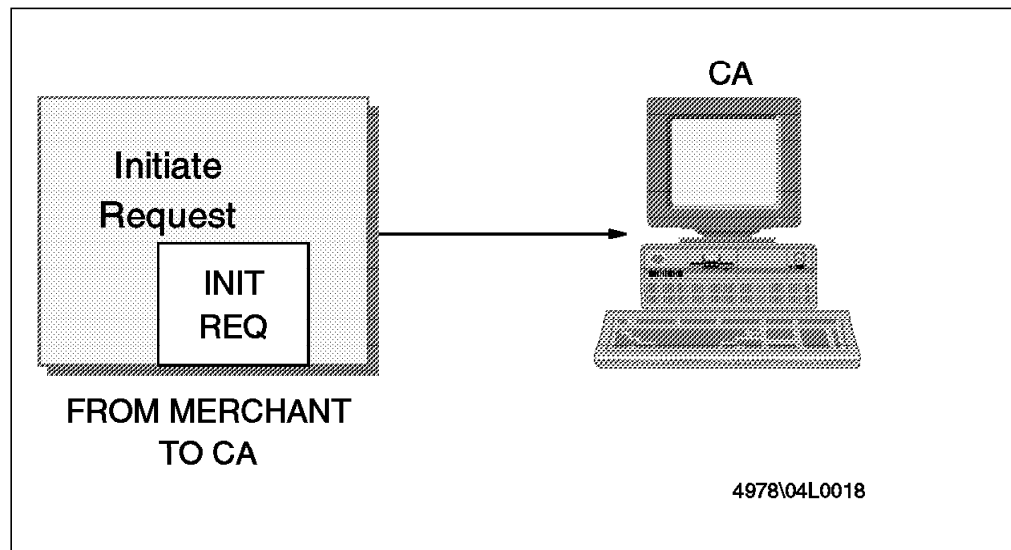


Figure 42. Merchant Initiates Request

4.4.2.2 CA Sends Registration Form

The CA receives the merchant initiate request and determines the appropriate registration form. It then sends it to the merchant in a signed message, as shown in Figure 43 on page 71.

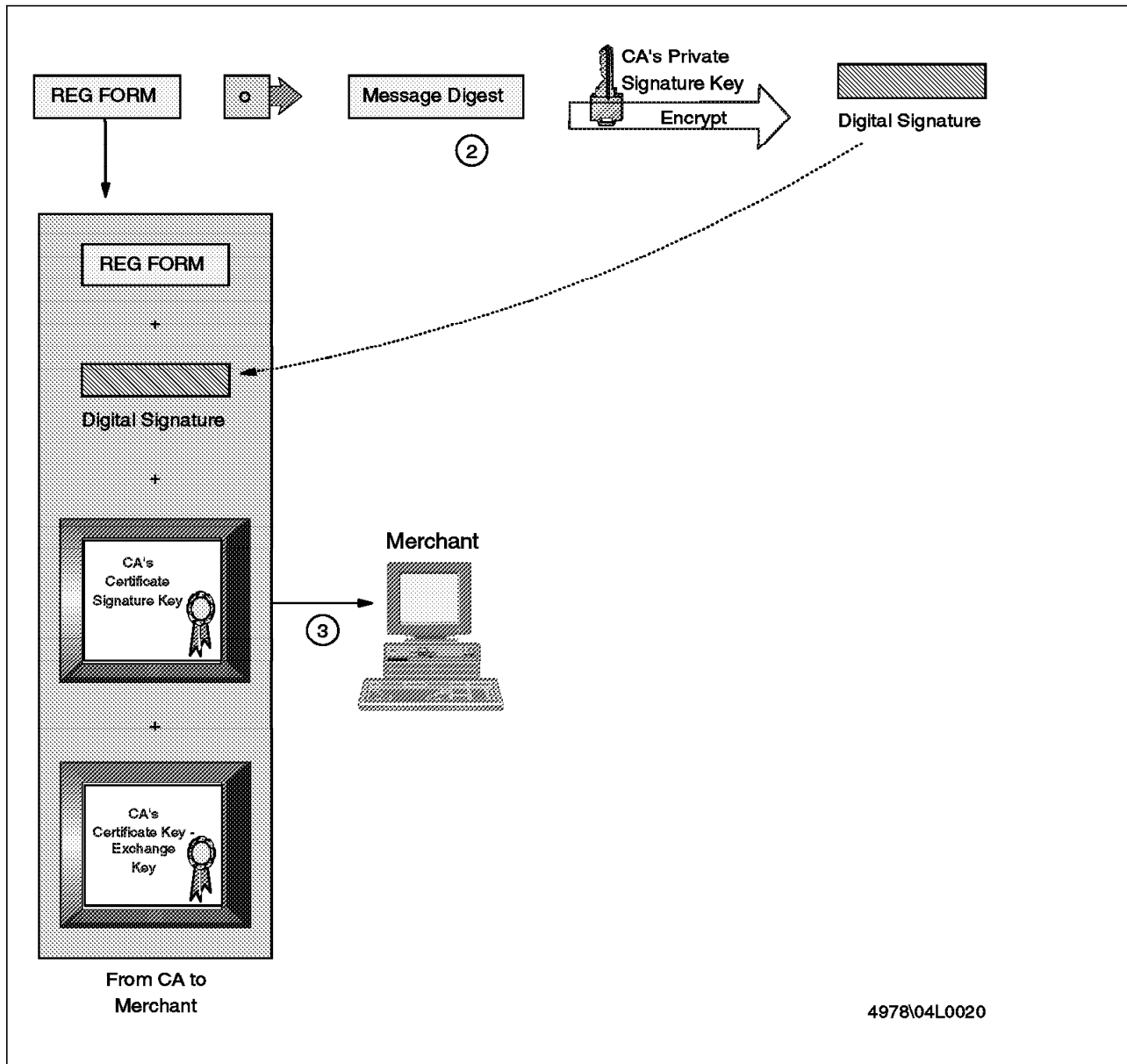


Figure 43. CA Sends Registration Form

1. The CA digitally signs the form by passing it through a hash function. The message digest so created is encrypted with the CA private signature key, creating a digital signature.
2. The CA sends the registration form, the digital signature and the certificates containing its public signature key and public key-exchange key to the merchant.

4.4.2.3 Merchant Requests Certificate

Now the merchant can receive the registration form and use it to request a certificate, as shown in Figure 44 on page 72 and Figure 45 on page 73.

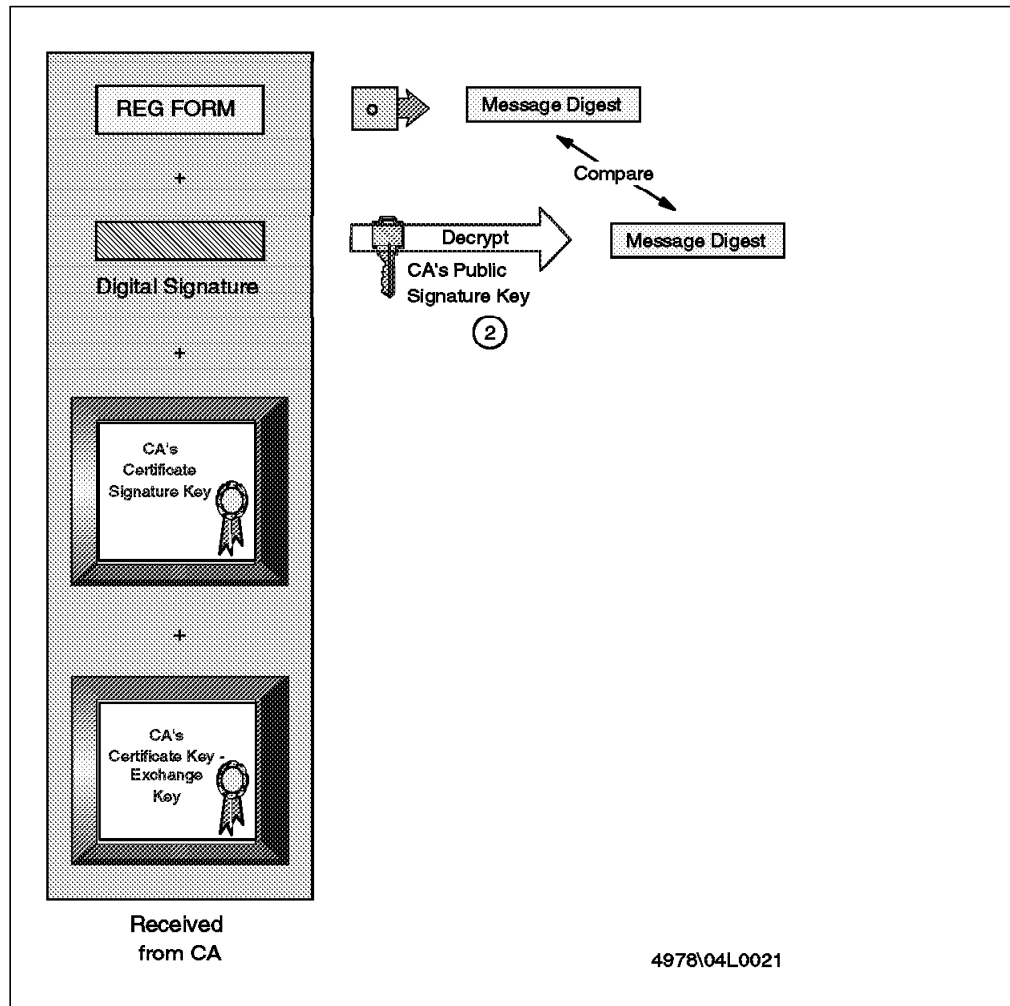


Figure 44. Merchant Receives Registration Form

1. The merchant receives the registration form from the CA and verifies the certificates by traversing the trust chain to the root.
2. The CA signature received is verified by running the registration form through a hash function and creating a message digest. The digital signature is decrypted using the CA public signature key and the result is compared with the message digest obtained locally. If they are equal, the integrity of the message has been assured.

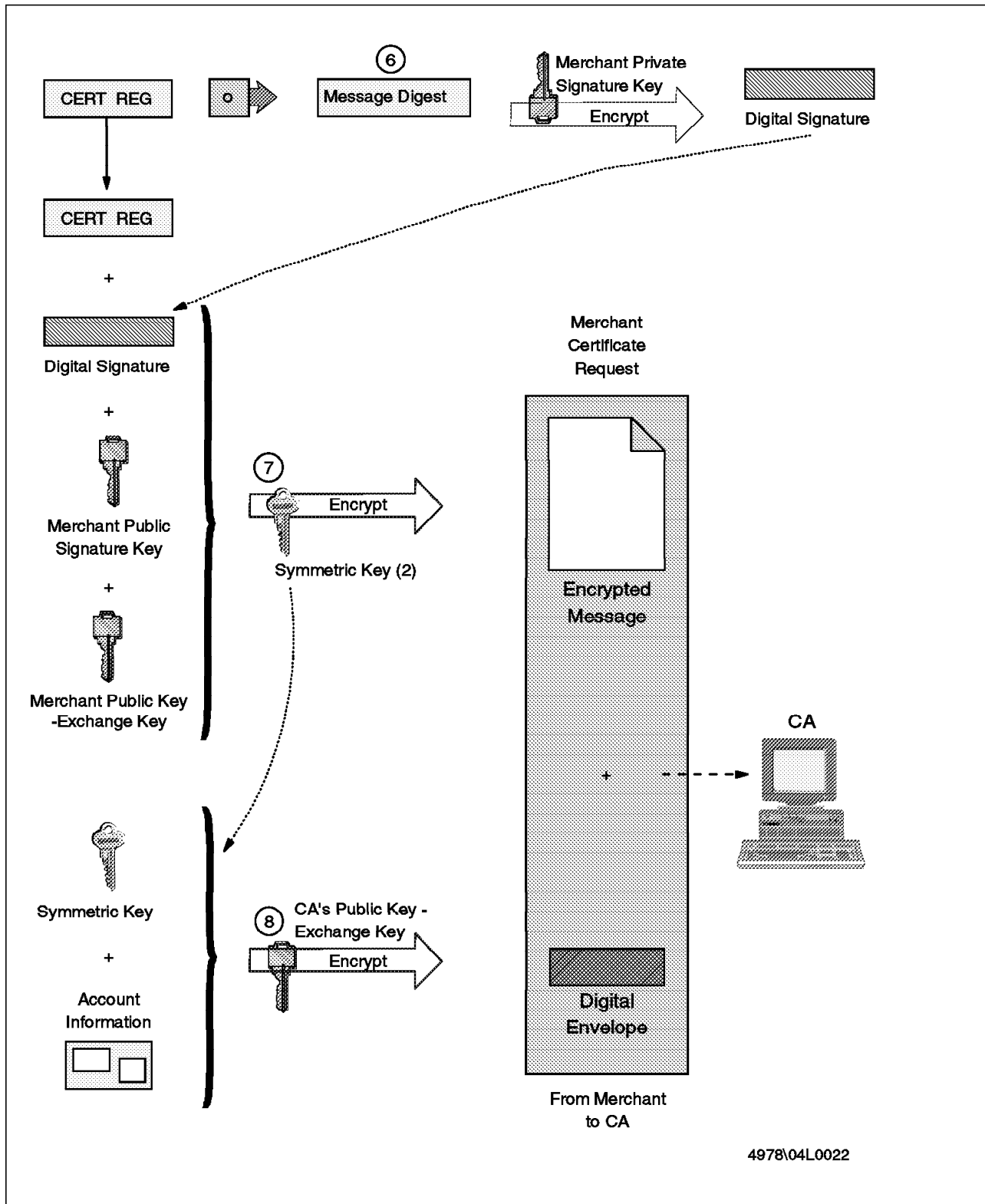


Figure 45. Merchant Requests Certificate

3. The merchant software creates two key pairs (public and private signature key and public and private key-exchange key).
4. The merchant completes the registration form.

5. The merchant software generates a certificate request including the information from the registration form.
6. The certificate request is digitally signed by passing it through a hash function. The message digest so created is encrypted with the merchant private signature key, resulting in a digital signature.
7. The merchant creates an encrypted message by encrypting the certificate request, the digital signature, the merchant public signature key, and merchant public key-exchange key using a randomly generated symmetric key.
8. The symmetric key and merchant account data are encrypted with the CA public key-exchange key generating a digital envelope.
9. The digital envelope and the encrypted message are sent to the CA.

4.4.2.4 CA Creates Certificates

Next, the CA receives the certificate request and processes it, including any supporting offline data. Then it generates a certificate and sends it to the merchant. Figure 46 and Figure 47 on page 76 show the process.

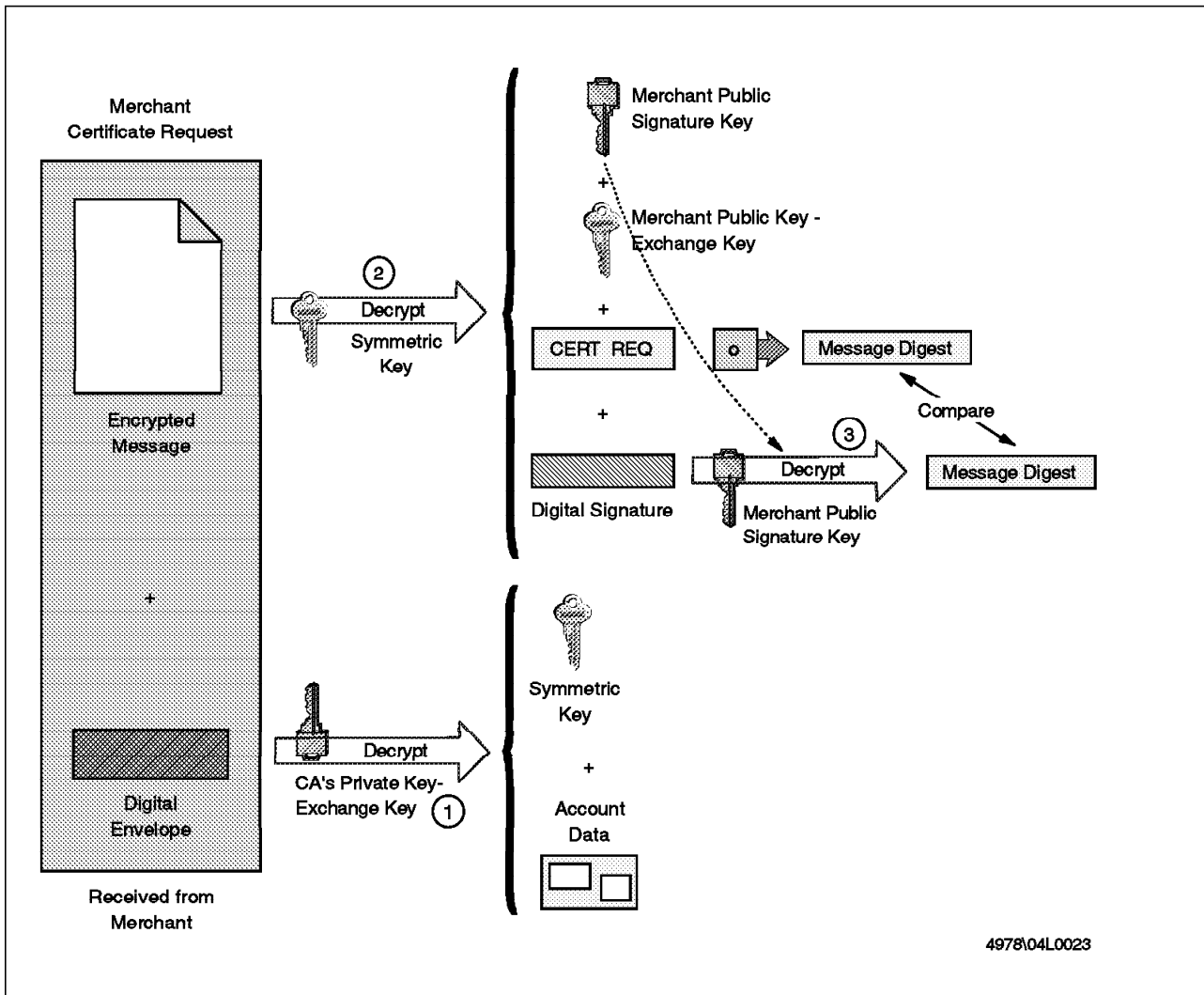


Figure 46. CA Receives Certificate Request

1. The CA receives the certificate request message from the merchant containing the digital envelope and encrypted message. The digital envelope is decrypted using the CA private key-exchange key, yielding the symmetric key and merchant account data.
2. The encrypted message is decrypted using the symmetric key. The CA now has the merchant public signature key, merchant public key-exchange key, the certificate request, and the digital signature.
3. The merchant signature is verified by running the certificate request through a hash function and creating a message digest. The digital signature is decrypted using the merchant public signature key and the result is compared with the message digest obtained locally. If they are equal, the integrity of the message has been assured.
4. The CA verifies the certificate request using the merchant account data, offline documentation and the registration form. If the information is valid, the CA can go ahead and return the certificates.

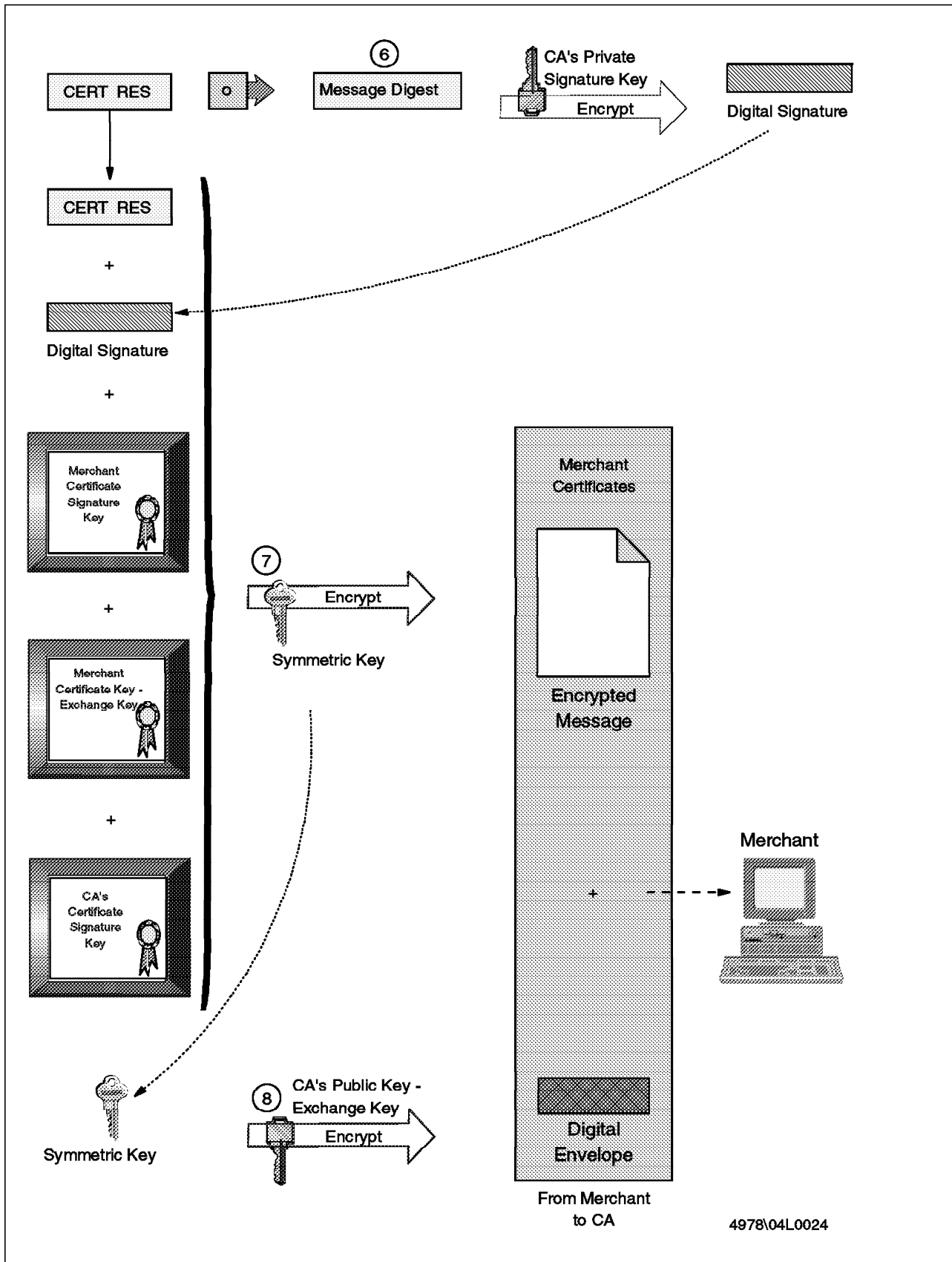


Figure 47. CA Sends Certificates

5. The CA generates the certificates by digitally signing the requests using the CA private signature key.
6. It creates a certificate response message and passes it through a hash function. The message digest so created is encrypted with the CA private signature key, resulting in a digital signature.
7. The CA encrypts the certificate response, digital signature, the merchant certificates containing the signature key and key-exchange key, and the CA certificate containing the signature key with a randomly generated symmetric key.
8. The symmetric key data is encrypted with the merchant public key-exchange key, generating a digital envelope.
9. The digital envelope and the encrypted message are sent to the CA.

4.4.2.5 Merchant Receives Certificates

Finally, the merchant receives the certificate and stores it.

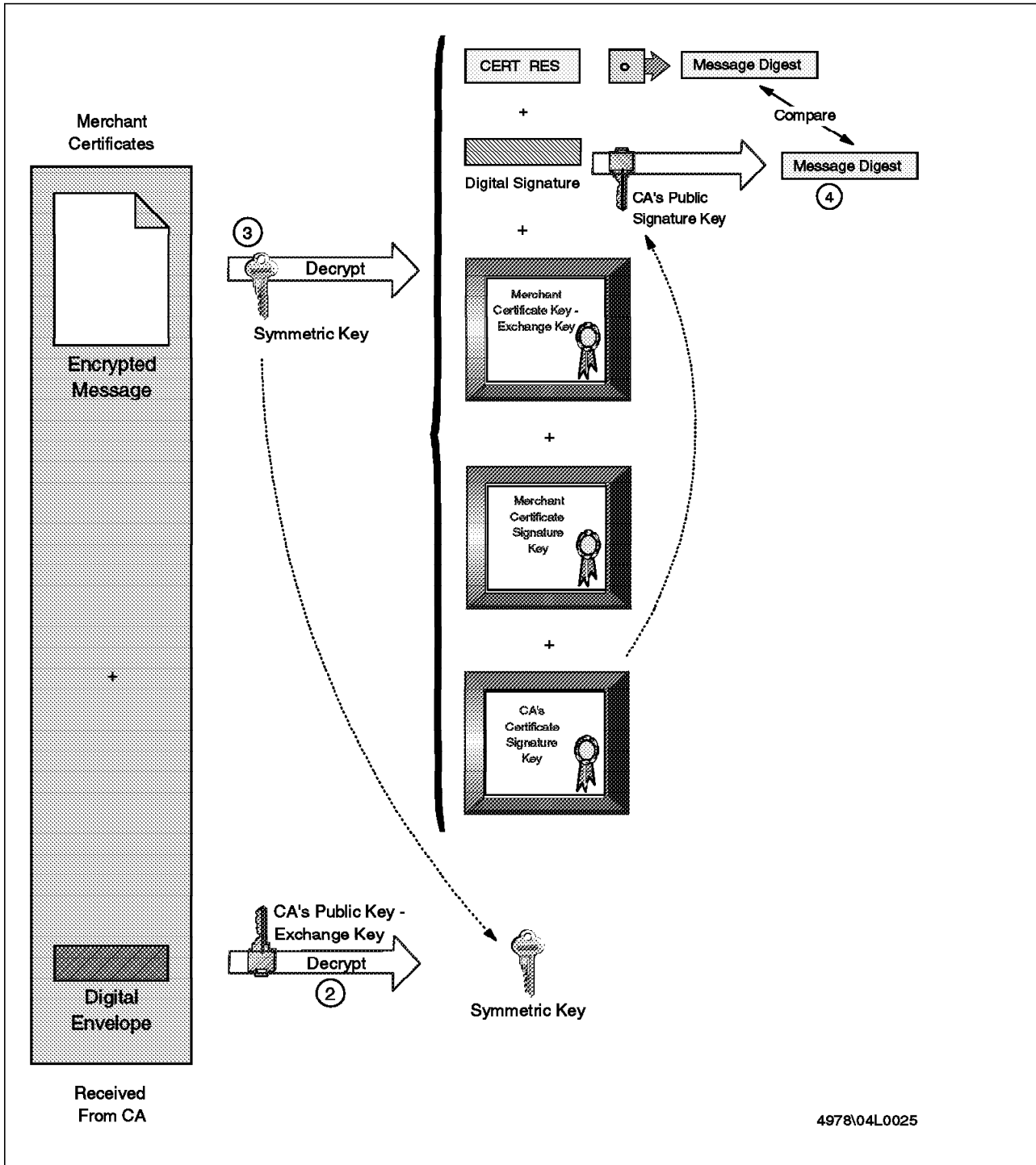


Figure 48. Merchant Receives Certificates

1. The merchant verifies the certificates by traversing the trust chain to the root.
2. The digital envelope is decrypted using the merchant private key-exchange key yielding the symmetric key.
3. The encrypted message is decrypted using the symmetric key obtaining the certificate response, digital signature, merchants certificates and CA certificate.

4. The CA signature is verified by running the certificate response through a hash function and creating a message digest. The digital signature is decrypted using the CA public signature key and the result is compared with the message digest obtained locally. If they are equal, the integrity of the message has been assured.
5. The merchant stores the certificate for future use. It is now ready to take credit card orders.

Part 2. The Practice

Chapter 5. Introduction

In this part of the book we show how all the different functions of SET are implemented by IBM products. Each role is taken by a different product, as follows:

Cardholder

This function is provided by CommercePOINT Wallet, which is a plug-in module for Web browsers.

Merchant

This function is provided by the Net.Commerce server, together with an additional function which implements the SET protocol as an extension to the standard payment processing options. The complete application (Net.Commerce plus SET extensions) will be known as CommercePOINT Till.

SET Certifying Authority

This function is provided by the IBM Registry for SET. IBM Registry is a general-purpose certificate management server. The SET edition adds SET protocol support to the base server.

Acquirer Payment Gateway

This function is provided by CommercePOINT Gateway, which implements the acquirer gateway portion of the SET protocol and provides a number of tools and interfaces to interface it to existing authorization systems.

We discuss the detailed installation and use of each of these components in turn in the following chapters. At the time of writing this book, none of the components were yet in their final form so the examples here are based on alpha and beta versions of the products.

5.1 Lab Environment

Our lab environment for this book included RS/6000s for each of the three server roles, plus a number of Windows 95 and Windows NT machines.

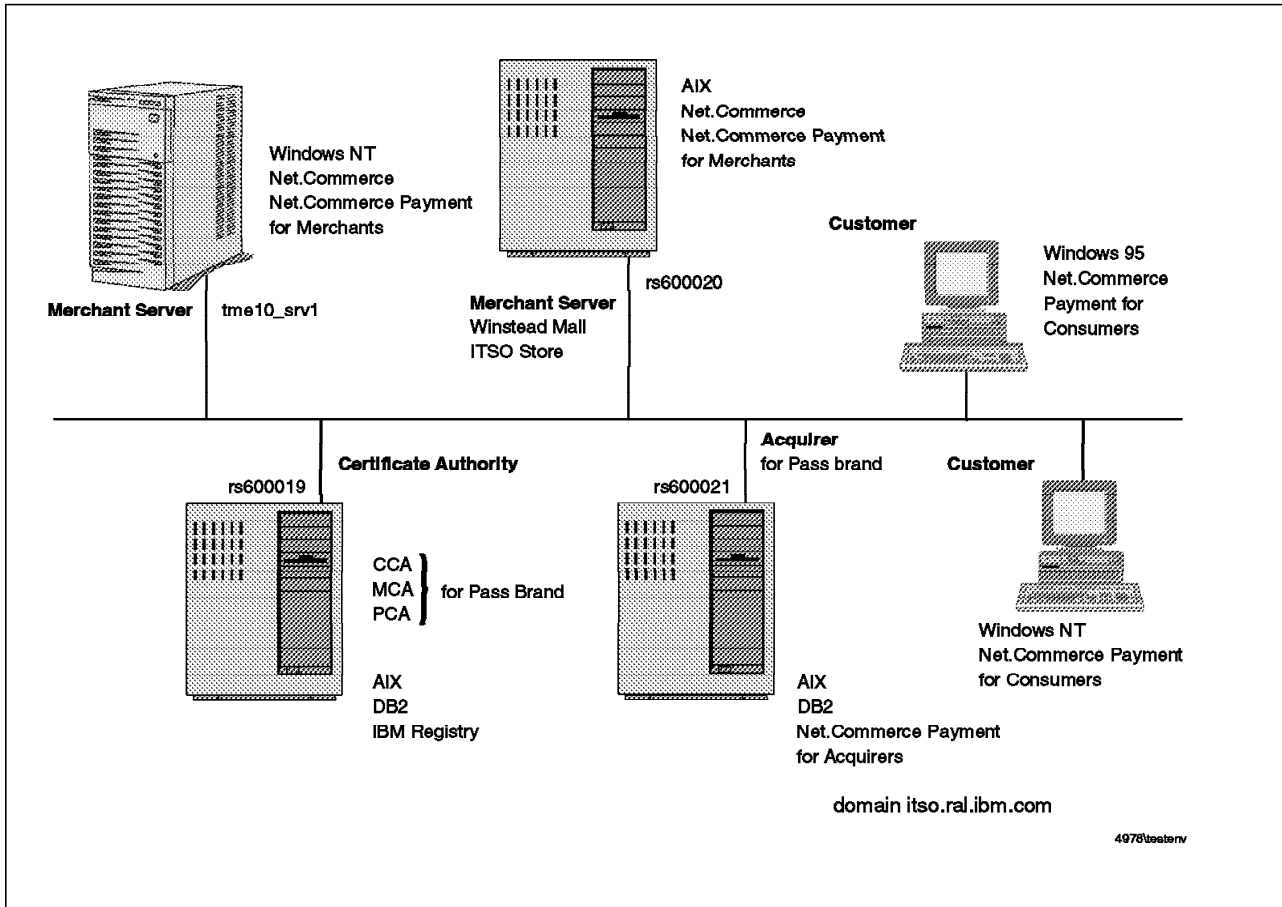


Figure 49. Lab Environment

Chapter 6. The Cardholder's View

In this chapter we show the end result of implementing SET using IBM products: the process by which a user can safely purchase goods or services over the World Wide Web using a credit card. There are three parts to the process:

1. Installing the electronic wallet (the cardholder plug-in module)
2. Defining credit cards and obtaining public key certificates for them
3. Making purchases

For SET to be successful it is imperative that the cardholder is protected from the underlying complexity of the protocol. In this chapter we aim to demonstrate this simple view by showing the sequence of screens that the user sees. Of course, this simplicity is supported by the basic design of SET and by the implementation of it in the three servers (Merchant, CA and Payment Gateway). At each step in the process we cross-reference the appropriate part of the SET protocols and the relevant server configuration. In this way you can get a good idea of what a fully implemented SET environment looks like, without having to go into detail unless you want to.

6.1 Installing CommercePOINT Wallet

The cardholder plug-in module is supplied as a self-extracting zip file. It installs on any Windows 95 or Windows NT system with Netscape Navigator. You will need about 12 MB of disk space available. There is no minimum processor requirement for the product, but you should keep in mind that encryption algorithms require a lot of CPU resource, so be prepared for CommercePOINT Wallet to run slowly if your machine does not have sufficient power. In our experience the module runs well on a 75 MHz 486 DX4 machine or faster, but we did not do any systematic testing of processor requirement.

Installation uses the standard Windows 95/NT InstallShield wizard. To perform the installation you simply:

1. Receive the file onto your hard disk.
2. Execute it (either by double-clicking it in Windows Explorer or by selecting **Run** from the Start menu).

The logo screen shown in Figure 50 on page 86 will then appear and you will be prompted to supply basic details:

- Your name.
- The directory in which to place the CommercePOINT Wallet code. The default is C:\Netcommerce.



Figure 50. Initial Installation Logo Screen

You will also be required to click on a button to accept the terms and conditions of the product. We envision there being some variation in the details of the install process and of the terms and conditions. In general, a cardholder is likely to receive the CommercePOINT Wallet installation code from his or her bank or credit card provider and therefore the information required from the user and the detailed operation of the electronic wallet will be customized to fit in with the policy of the card issuer.

On Windows 95, the installation program creates a new folder under the Programs folder, containing icons for CommercePOINT Wallet itself, a README file and help information. Under Windows NT this folder is not created.



Figure 51. CommercePOINT Wallet Folder in Windows 95

6.1.1 Preparing Netscape Navigator

It is possible to invoke the electronic wallet directly by selecting the icon or by launching NETCOMM.EXE from the CommercePOINT Wallet install directory. However, normally it is only invoked when the client machine receives SET messages with specific MIME header types. In order to make this happen, NETCOMM.EXE has to be defined as a *helper* application in Netscape Navigator.

In the alpha version of CommercePOINT Wallet that we were using, we had to manually define the helper entries. In production code this would probably be an automatic function of the installation. To define the helper entries, select **Options** from the menu bar followed by **General Preferences** and then click on the **Helpers** tag. Figure 52 shows the four entries that need to be added.

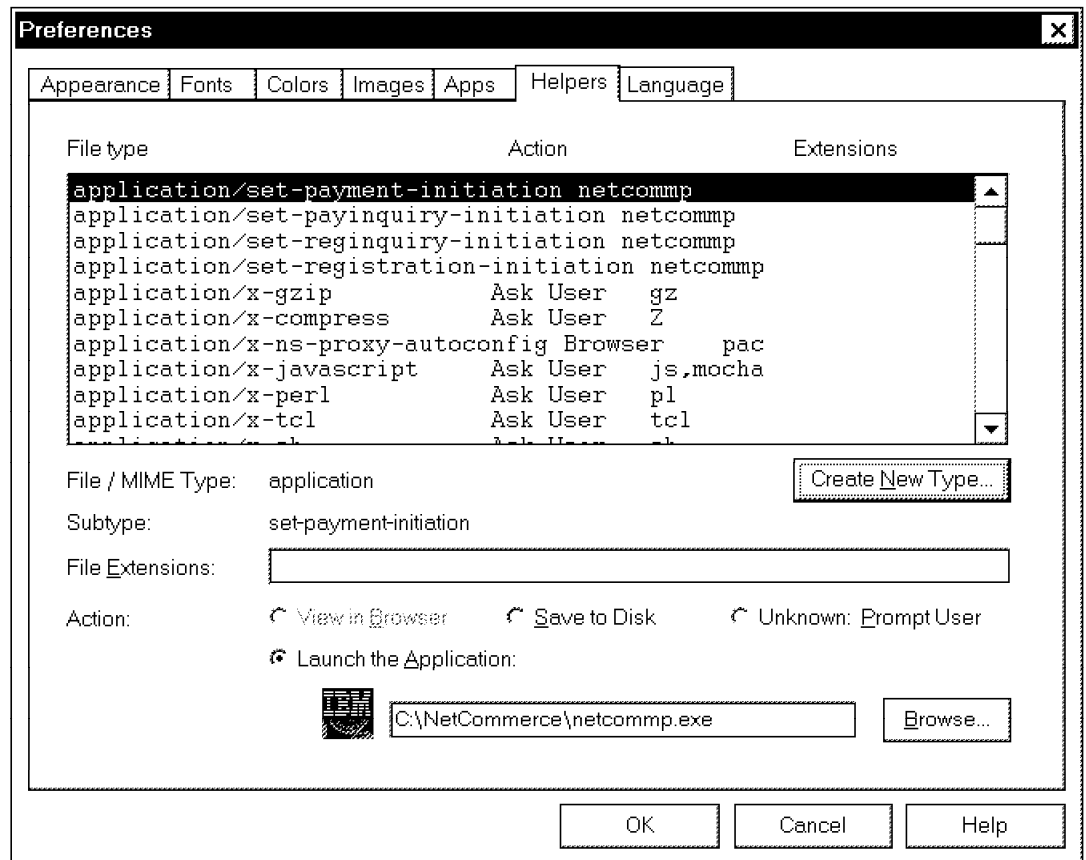


Figure 52. Defining Helper Applications in Netscape Navigator

Each entry has a MIME type of application together with a SET-defined sub-type:

- set-payment-initiation is invoked by the response to a payment request. Note that this is *not* a SET PINIT response (see Figure 11 on page 29), but part of a preamble to the SET payment process.
- set-payinquiry-initiation is the result of a request to inquire on the status of a payment.
- set-registration-initiation is invoked by the response to an initial connection to a certificate server. Note that this is *not* a SET CINIT response (see Figure 32 on page 59). As in the payment case, it is part of a preamble to SET certification.

- set-reginquiry-initiation is the result of a request to inquire on the status of a certificate request.

All four message types invoke the same helper application, namely NETCOMM.EXE in the CommercePOINT Wallet install directory.

6.2 Defining Credit and Debit Cards

As we have said, you can invoke CommercePOINT Wallet directly from the Windows desktop. In this mode you can define credit cards for any of the brands that are predefined within the product, but you cannot create cardholder keys to go with them. To do this and to define additional card brands, CommercePOINT Wallet has to be invoked online, from within a browser session.

No matter which way it is invoked, the first screen that you see when you enter the program will ask you where your user data resides, on a diskette or on the hard drive (see Figure 53).

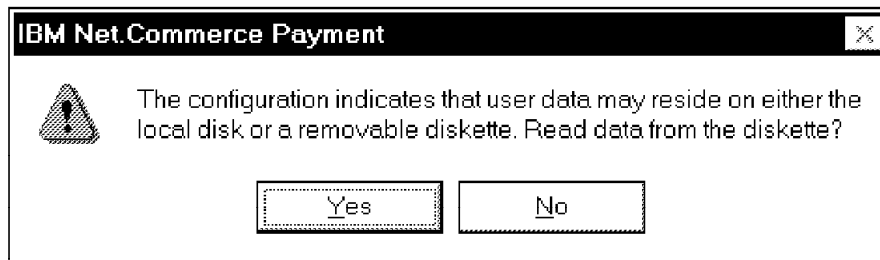


Figure 53. User Data Question Pop-Up

If this is the very first time that you have used CommercePOINT Wallet you will be asked to enter a user ID and password (see Figure 54 on page 89 and Figure 55 on page 89). Note the unusually large size required for the password. Details of this sort are likely to vary, depending on the policy of the bank or brand that issued the cardholder software.



Figure 54. Prompt For a New User ID

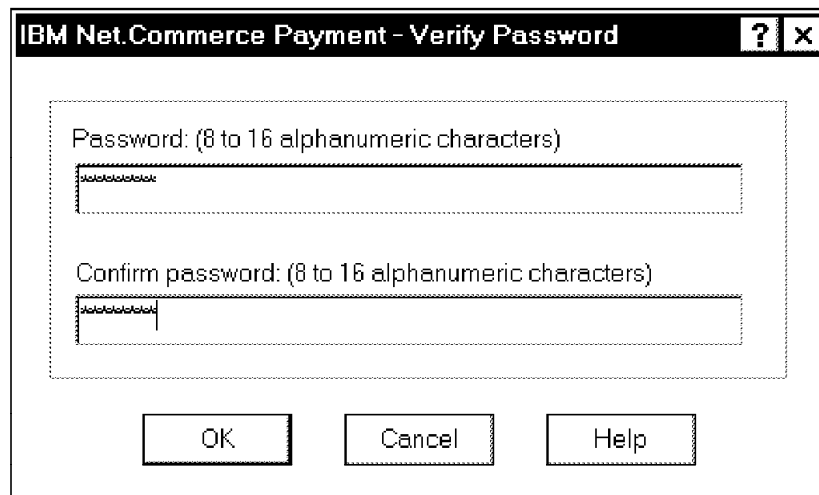


Figure 55. Password For New User ID

Note that this password is not used in any network communications. For all online transaction, SET uses digital signatures to authenticate the user. This password simply acts as a key to lock up the details and keys belonging to the user. This means that if the PC or diskette containing your SET configuration is stolen, the thief cannot use it without first guessing the password.

6.2.1 Stand-Alone Card Definition

After providing an ID and password you will be presented with the main CommercePOINT Wallet screen, as shown in Figure 56. The pop-up message is telling you that you do not yet have any cards in your wallet. We'll soon fix that.

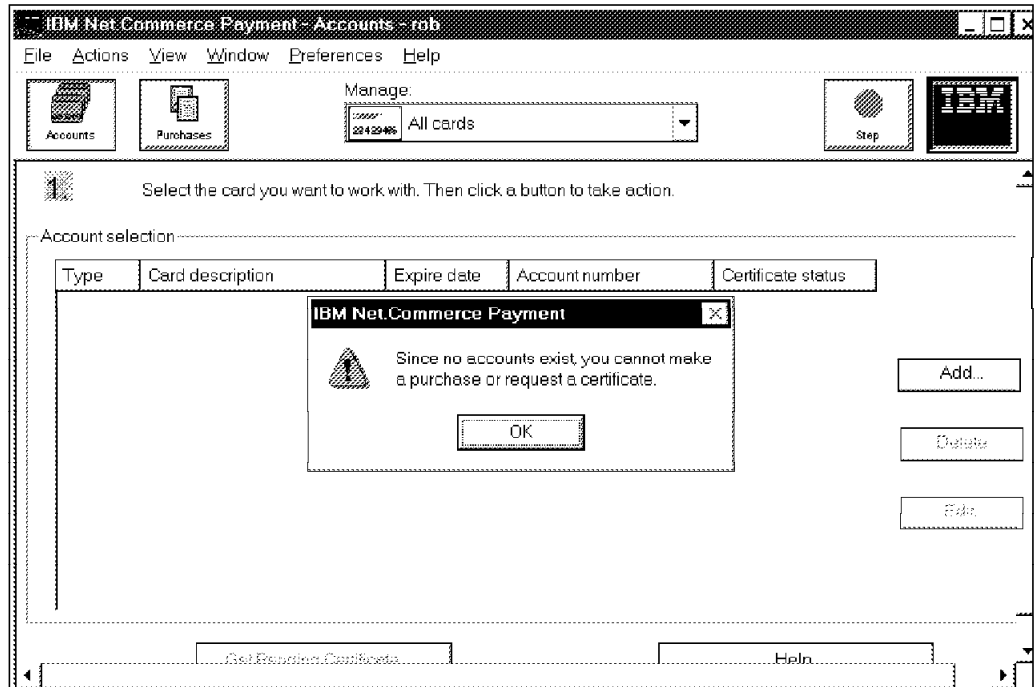


Figure 56. Initial Cardholder Screen

If you click on **Add** at this point you will get the chance to define a new card. Figure 57 on page 91 shows the range of brands that are available. In our case, however, the card we want to define is from the exclusive (and imaginary) *Pass* brand. To define a card we need to connect to a SET certificate server that handles the Pass brand.

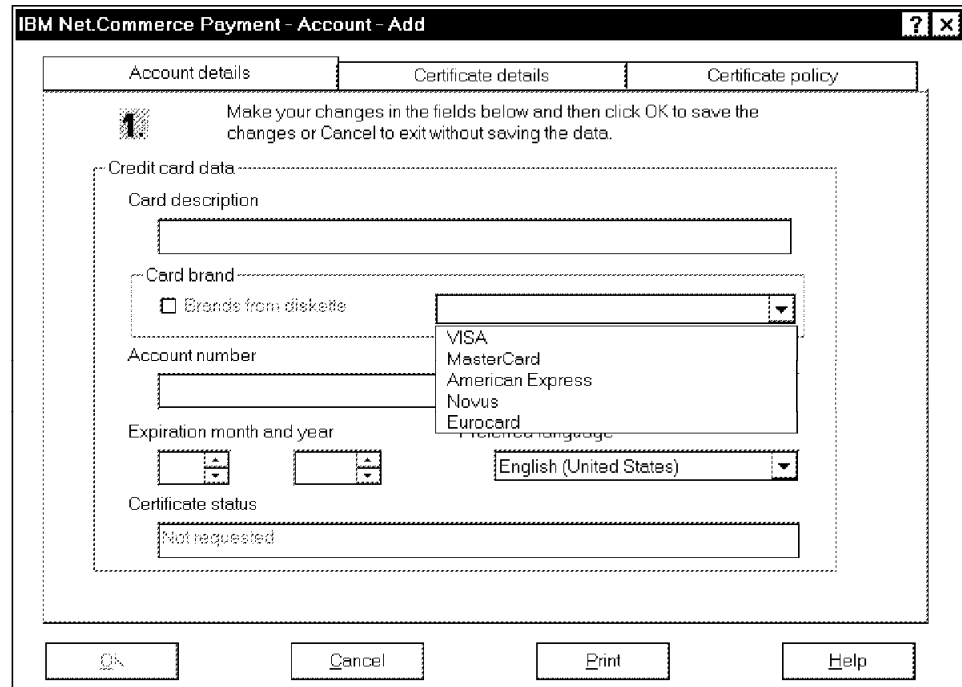


Figure 57. Add Card Dialog with Default Brand List

6.2.2 Online Card Definition and Certification

To invoke the electronic wallet online we need to select a URL directed to the CA server which will kick off the certification process. In real life, this would probably be a link on a Web page belonging to the brand. We show how to create this page in 8.9, "Creating a CCA WWW server" on page 161. In our test case the brand Web page is on the same machine as the certificate server itself, but normally it would be different. In fact, from a security standpoint you would not want a regular Web server operating on the CA system at all. Figure 58 shows the Web page.

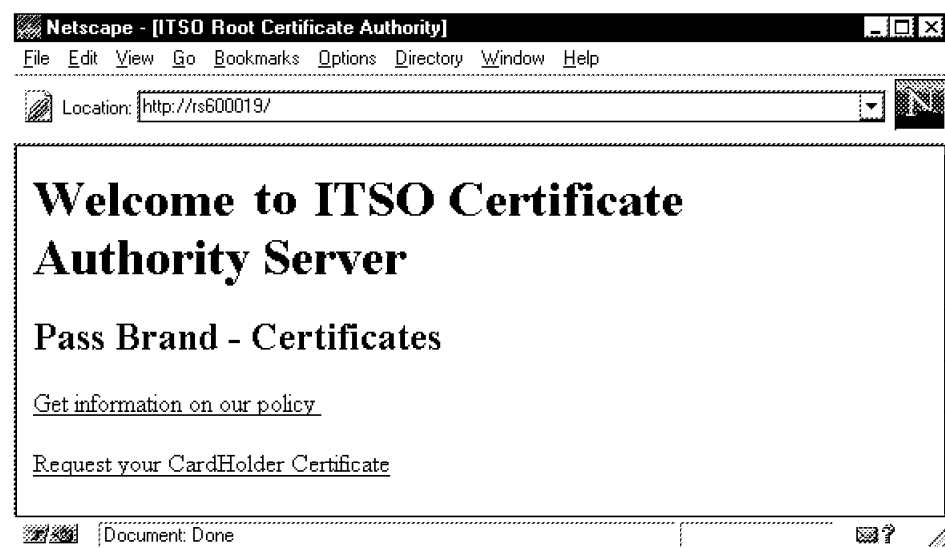


Figure 58. Web Page with Link to SET Certificate Server

When you click on the link to request a certificate you are passed to the certificate server, which returns a set-registration-initiation response causing the electronic wallet to start up (see 6.1.1, "Preparing Netscape Navigator" on page 87). However, this time the certificate server has also updated the brand information with the addition of our plastic friend, the Pass card. Figure 59 shows the new list of supported brands.

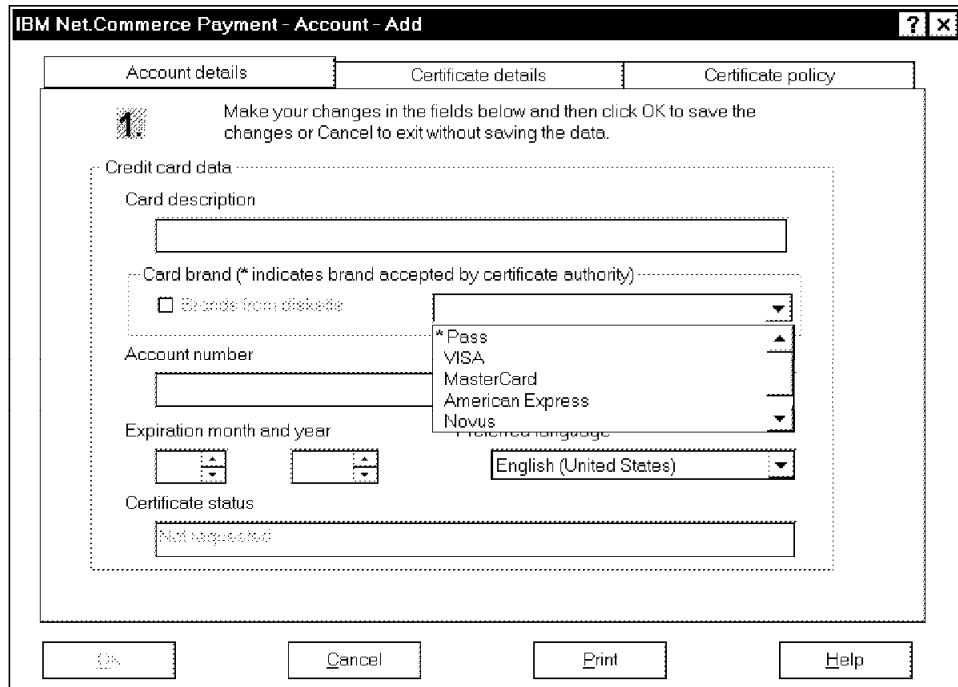


Figure 59. Add Card Dialog with Pass Brand Added

Note also that the Pass brand has an asterisk (*) beside it, indicating that this server can sign certificates for holders of the card.

Now you can fill in all the card details and save them. Figure 60 on page 93 shows the main CommercePOINT Wallet screen with our newly added card. Note that at this point the card only exists as an entry in the user database on your own PC. You have not yet requested a certificate to go with the card details.

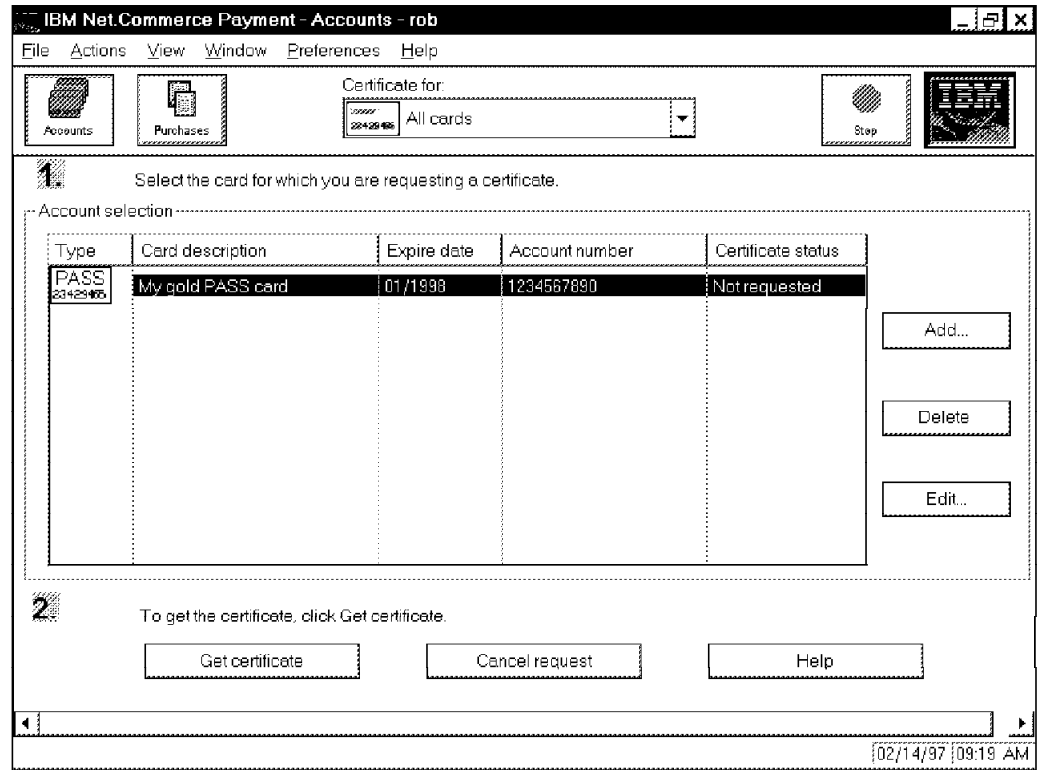


Figure 60. One Card in the Wallet

If you now click on **Get Certificate** you will generate a certification initiate request to the SET CA server. Note that this is the *first* time that we have invoked one of the defined SET protocol flows (see 4.4.1, “Cardholder Certification Process” on page 56).

Cardholder Keys

At this point we are assuming that the cardholder software has a key database in which the Root certificate has been pre-installed. For production cardholders this will always be the case. In the case of the early code that we used for our project, we had to generate a Root certificate for our own test CA hierarchy and then install the database onto the cardholder machine manually.

Refer to 8.6.1, “Key and Certificate Hierarchy Required for SET” on page 153 for an overview of the keys and certificates needed by SET and refer to 8.10.2, “Generating Cardholder Certificates” on page 166 for the details of the files we had to copy.

The Get Certificate request will first cause the Brand policy details to appear, as shown in Figure 61 on page 94 (seems like the Pass brand may need to check this policy with their lawyers).

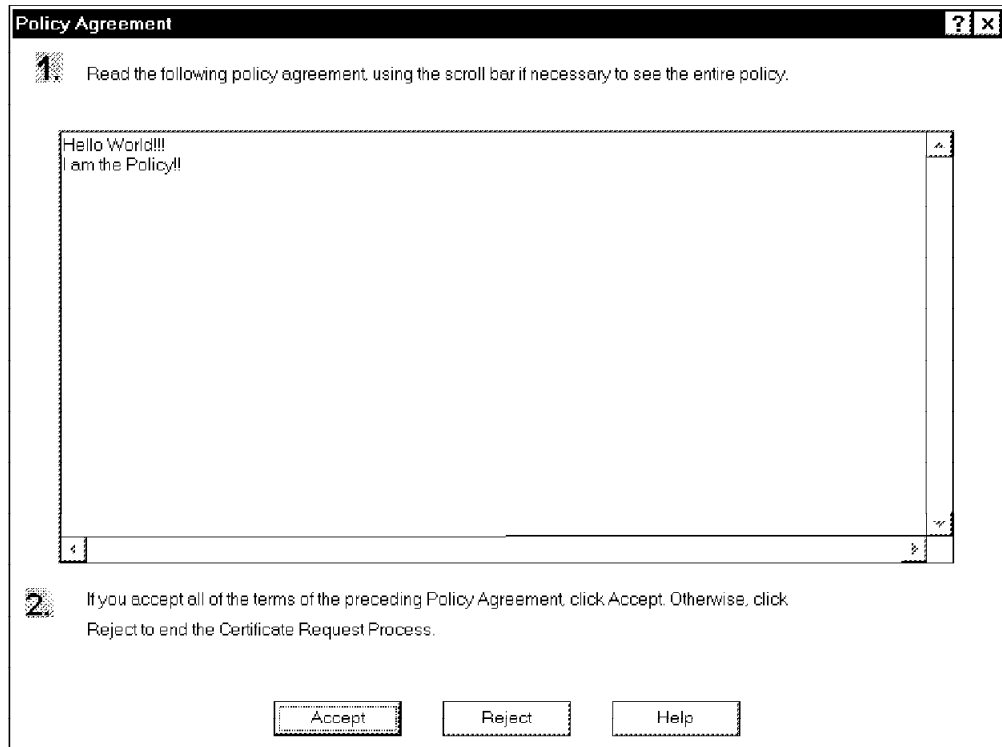
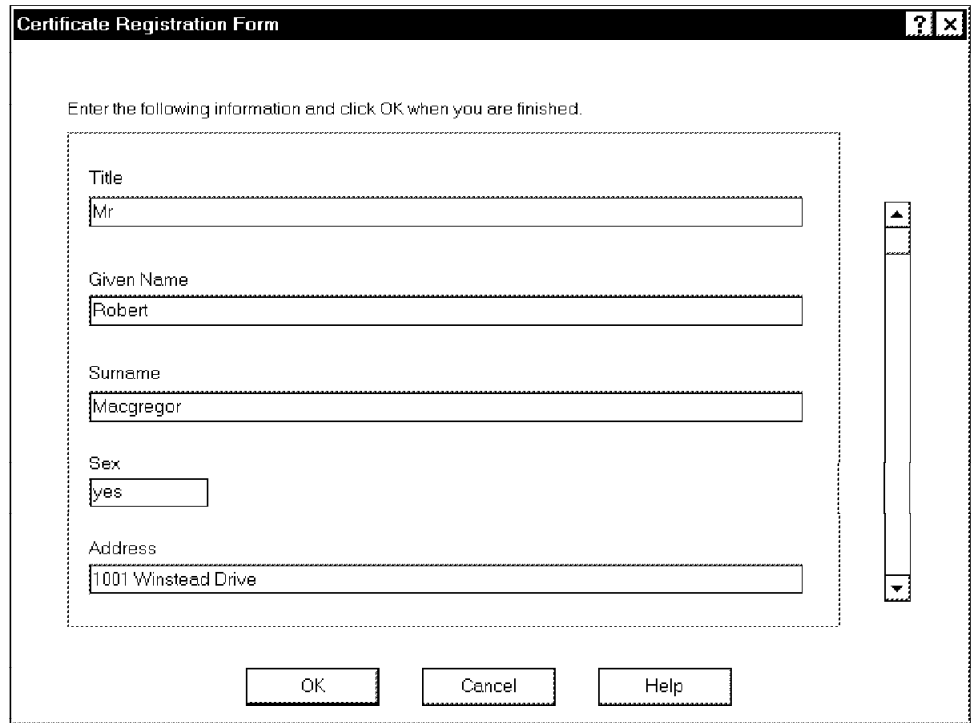


Figure 61. Brand Policy Screen

If you then click on **Accept**, you will be prompted for further details (see Figure 62 on page 95). This may appear to be the kind of standard form-filling that we all do in our daily lives. In fact, from a certification point of view, this information will be used to construct the *distinguished name* for the certificate. When the certificate has been built it will provide proof of a connection between this name information and a public key.



The image shows a dialog box titled "Certificate Registration Form". It contains a text area with the instruction "Enter the following information and click OK when you are finished." Below this are several input fields: "Title" with "Mr", "Given Name" with "Robert", "Surname" with "Macgregor", "Sex" with "yes", and "Address" with "1001 Winstead Drive". There are three buttons at the bottom: "OK", "Cancel", and "Help". A vertical scrollbar is visible on the right side of the form area.

Figure 62. Certificate Registration Form

Finally you can submit the certification request by clicking on **OK**. If all is well, your certificate will be returned directly and you will be returned to the browser window, as shown in Figure 63.

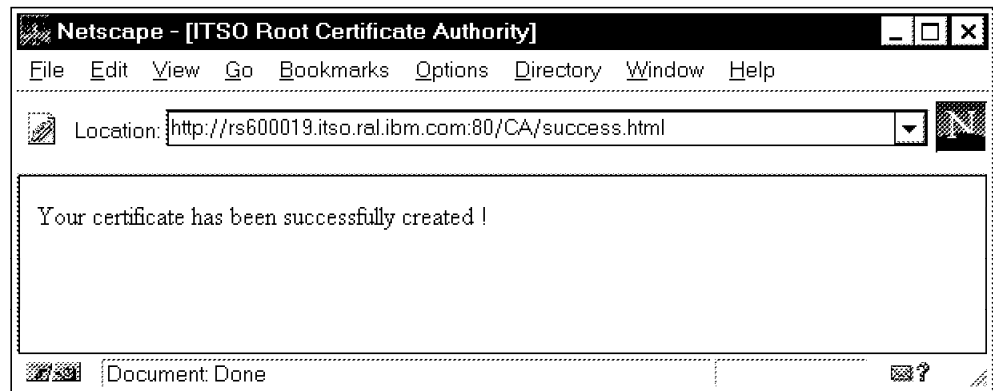


Figure 63. Completed Certification

6.3 Shop Until You Drop

Now your electronic wallet contains a credit card, so you can exercise your spending power. In our example we do this by shopping in an online store, but any situation with online payment for goods or services could use SET. In our case we assume that you are shopping for redbooks at the ITSO online bookstore. First you select the home page of the electronic mall where the bookstore is found, as shown in Figure 64 on page 96.

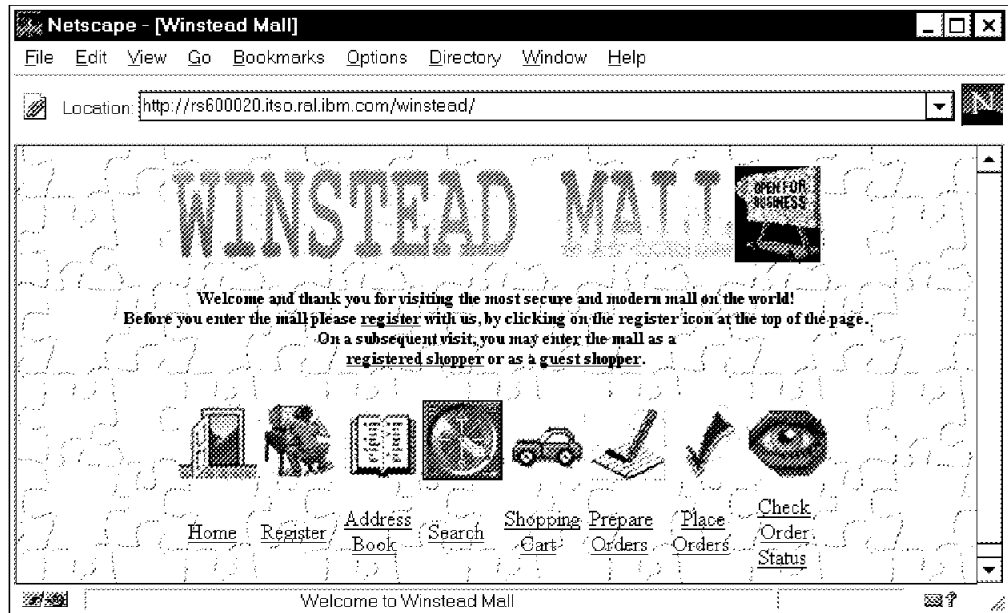


Figure 64. Mall Home Page

This is the first time you have shopped at this mall, so you have not registered yet. You could just browse, and put off registration until you find what you want to buy. However, in this case let's assume that you want to register straight away, by clicking on the **Register** hot link. You must now provide some personal and demographic information. Note that, so far, this has nothing to do with SET. This means that you should be concerned whether your personal information is being handled safely. Net.Commerce uses SSL to protect the data and to authenticate the server to you. You can tell that the form is being sent securely from the key symbol at the foot of the screen (see Figure 65 on page 97). The https: prefix to the URL indicates that SSL is in use.

Netscape - [Registration Information]

File Edit View Go Bookmarks Options Directory Window Help

Location: <https://rs600020.its0.ral.ibm.com/cgi-bin/nph-msvr/register/form>

New Registration

To register as a new shopper, fill in the form below. Items listed in **bold** must not be left blank. All information will be kept strictly confidential by the mall and store owners. We will not communicate this information to others as part of a mailing list.

Personal Information

Shopper's Login ID	<input type="text" value="robmacg"/>		
Password	<input type="password" value="*****"/>	Verify Password	<input type="password" value="*****"/>
Title	<input type="text" value="Mr"/>	Last Name	<input type="text" value="Macgregor"/>
First Name	<input type="text" value="Robert"/>	Middle Name	<input type="text" value="Stevens"/>
Company	<input type="text"/>		

Document Done

Figure 65. Shopper Registration Form

After registration you will be returned to the mall entrance, from where you can select the ITSO bookstore and find the redbook you want to buy (see Figure 66).

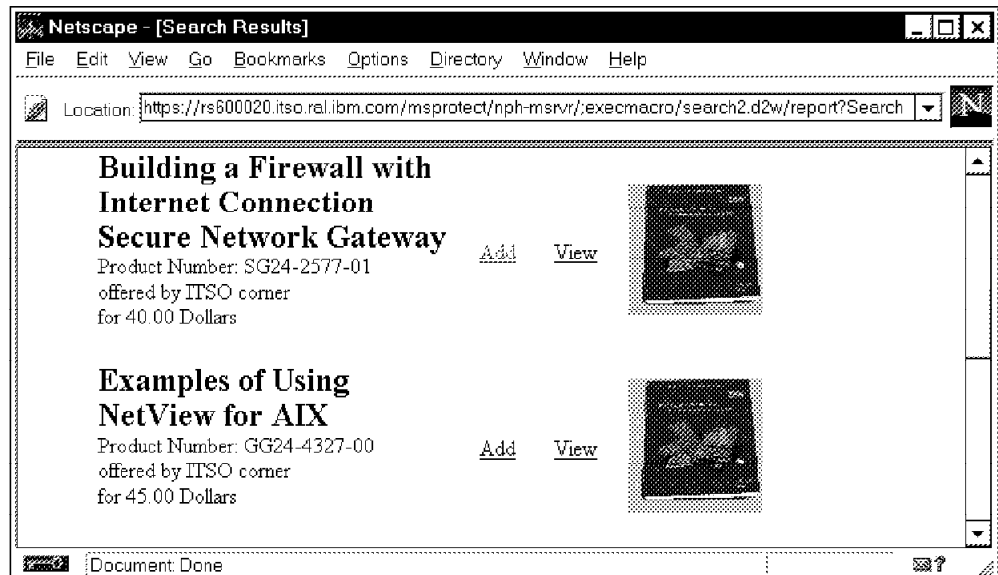


Figure 66. Excellent Taste in Redbooks

Now you can place the book of your choice in your shopping cart by selecting **Add**. Figure 67 on page 98 shows the resulting display of your cart contents.

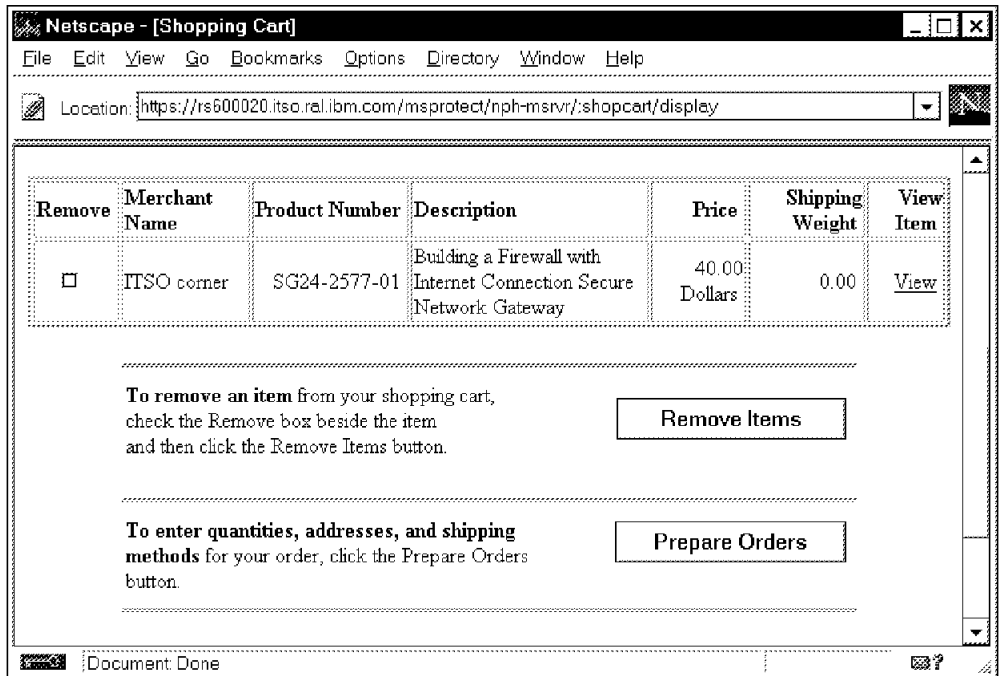


Figure 67. Shopping Cart Contents

At this point you can either continue shopping or go ahead with the purchase straight away by clicking on **Place Orders**. When you place the order you will be guided through a series of screens in which you can select the shipping details, quantities and so on. Finally you arrive at the order detail screen, shown in Figure 68.



Figure 68. Order Details Screen

Now SET takes over to process the payment. Click on **Purchase with Electronic Wallet** to initiate the SET payment process. The response to this request has a MIME type of application/set-payment-initiation, causing CommercePOINT Wallet to start up, as shown in Figure 69 on page 99. Note that the payment details appear here too.

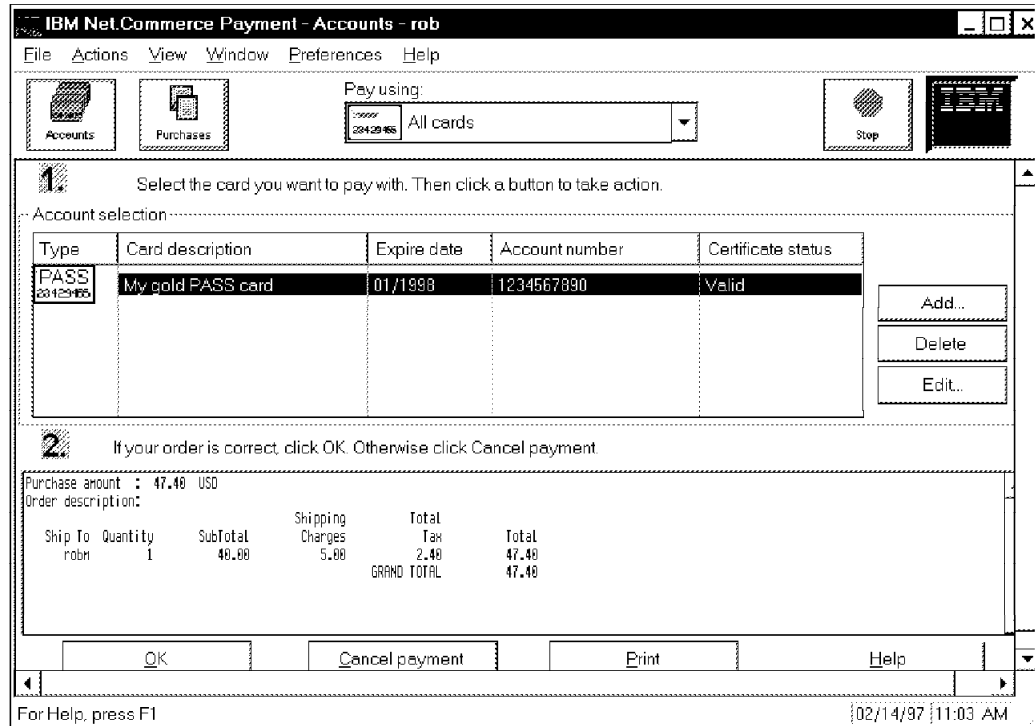
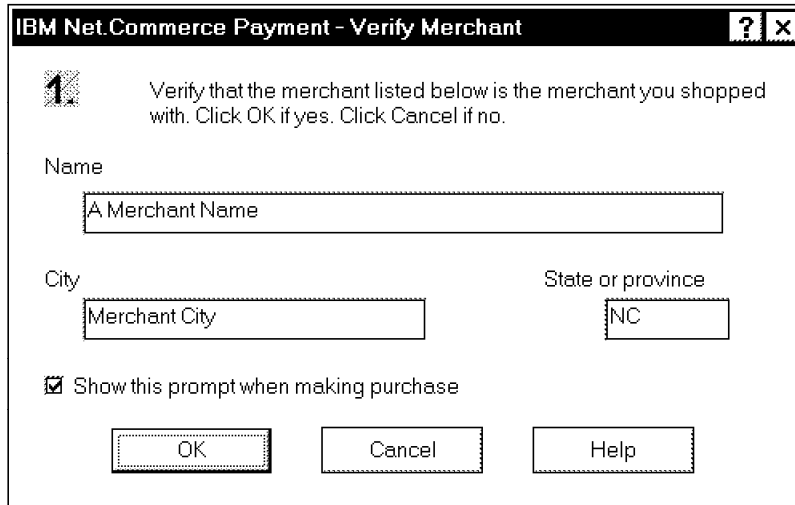


Figure 69. Electronic Wallet Request to Confirm Payment

Now click on **OK**. This invokes the SET payment protocol, as described in 3.3.1, “Purchase Process” on page 25. If you examine the first step of the payment process, you will find that the merchant presents its certificate as proof of identity. The wallet will automatically verify the trust hierarchy of the certificate back to the brand root. However, this only checks that the merchant holds a valid certificate, not that it is the same merchant that you went shopping with. You are asked to check the distinguished name in the certificate yourself, as shown in Figure 70 on page 100, to make sure that everything is as it should be.



The dialog box is titled "IBM Net.Commerce Payment - Verify Merchant" and contains the following elements:

- A numbered instruction: "1 Verify that the merchant listed below is the merchant you shopped with. Click OK if yes. Click Cancel if no."
- A "Name" label followed by a text input field containing "A Merchant Name".
- A "City" label followed by a text input field containing "Merchant City".
- A "State or province" label followed by a text input field containing "NC".
- A checked checkbox labeled "Show this prompt when making purchase".
- Three buttons at the bottom: "OK", "Cancel", and "Help".

Figure 70. Check Merchant Distinguished Name Information

If the merchant details are correct, click on **OK**. There will be a short delay, during which time the merchant server completes the SET payment protocol, obtaining authorization from the acquirer payment gateway. Finally you are returned to the browser screen with a confirmation message (see Figure 71).

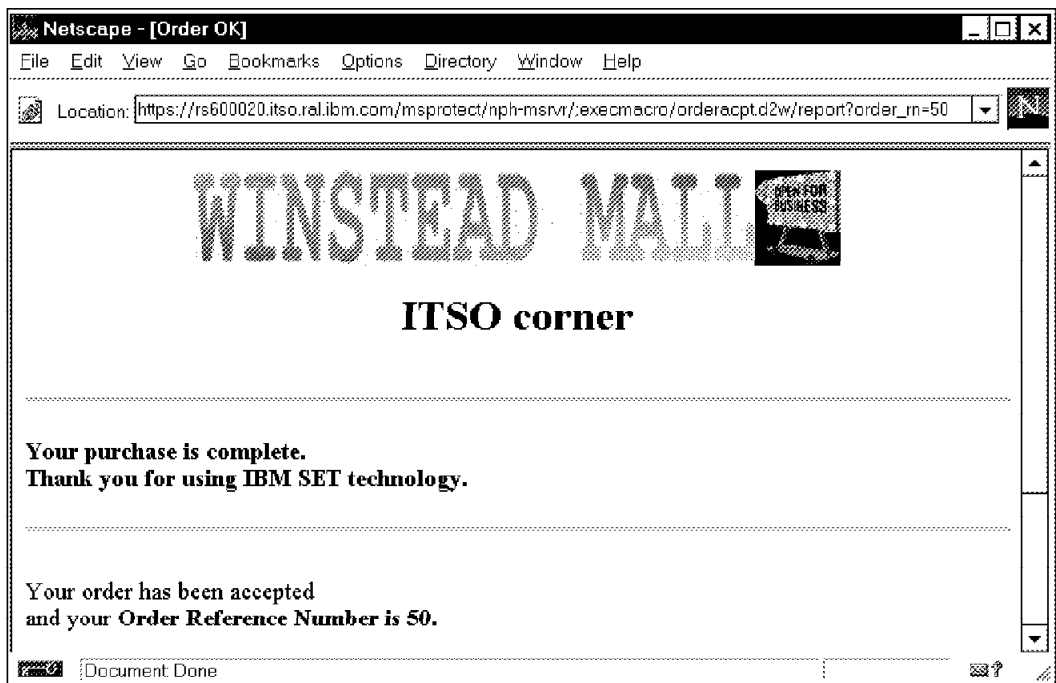


Figure 71. Payment Successful Message

Chapter 7. The Merchant

The merchant role is a pivotal point for many of the SET protocol flows. It is also a point at which the payment protocol has to be integrated into a commercial operation. SET does not place any restrictions on the nature of the commercial application, except that it requires credit card payments. However, the application that first comes to mind is online shopping. The IBM product that gives you the capability to create a Web site for online shopping is Net.Commerce. Version 2 of Net.Commerce has been extended to support the SET protocol.

This chapter describes how to install and configure the SET extensions for Net.Commerce. In this chapter we assume that you are familiar with the installation of Net.Commerce and the way that you create an online store. If you are not familiar with it, Chapter 10, "Installing Net.Commerce" on page 215 and Chapter 11, "Creating an Online Store with Net.Commerce" on page 241 describe how we created the "ITSO Corner" bookstore, which is used as an example throughout the book.

First, let us look at the structure of the Net.Commerce product and at how the SET extensions are integrated into it.

7.1 Net.Commerce Components: Overview, Interfaces and Interactions

Net.Commerce is not a simple Web server, but rather a combination of Web server, relational database, transaction interface, and administrative tools. This section provides an overview of the different components of the Net.Commerce system and describes the interfaces available to access these components and the interactions between them.

For a full description of the components, refer to Chapter 1, "Inside the Net.Commerce System" in the *Net.Commerce Installation and Operations Guide*.

7.1.1 Components Overview

The Net.Commerce system relies on four major components:

- IBM Internet Connection Secure Server (ICSS)
- IBM DATABASE 2 (DB2)
- Net.Commerce Server
- Net.Commerce Administrator

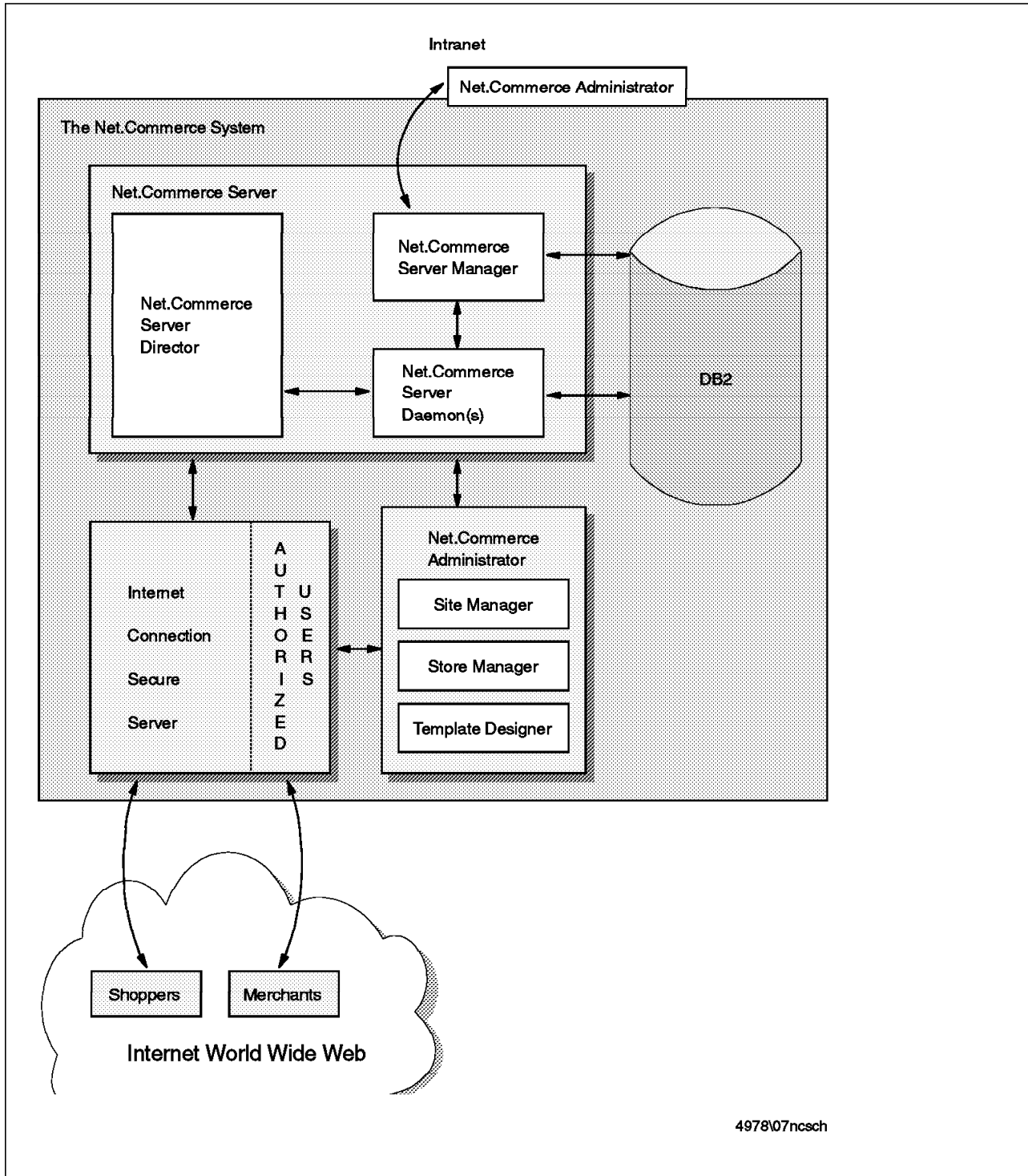


Figure 72. Net.Commerce Components

Figure 72 shows the four Net.Commerce components and the interactions between them when they are installed on a single processor.

7.1.1.1 The IBM Internet Connection Secure Server

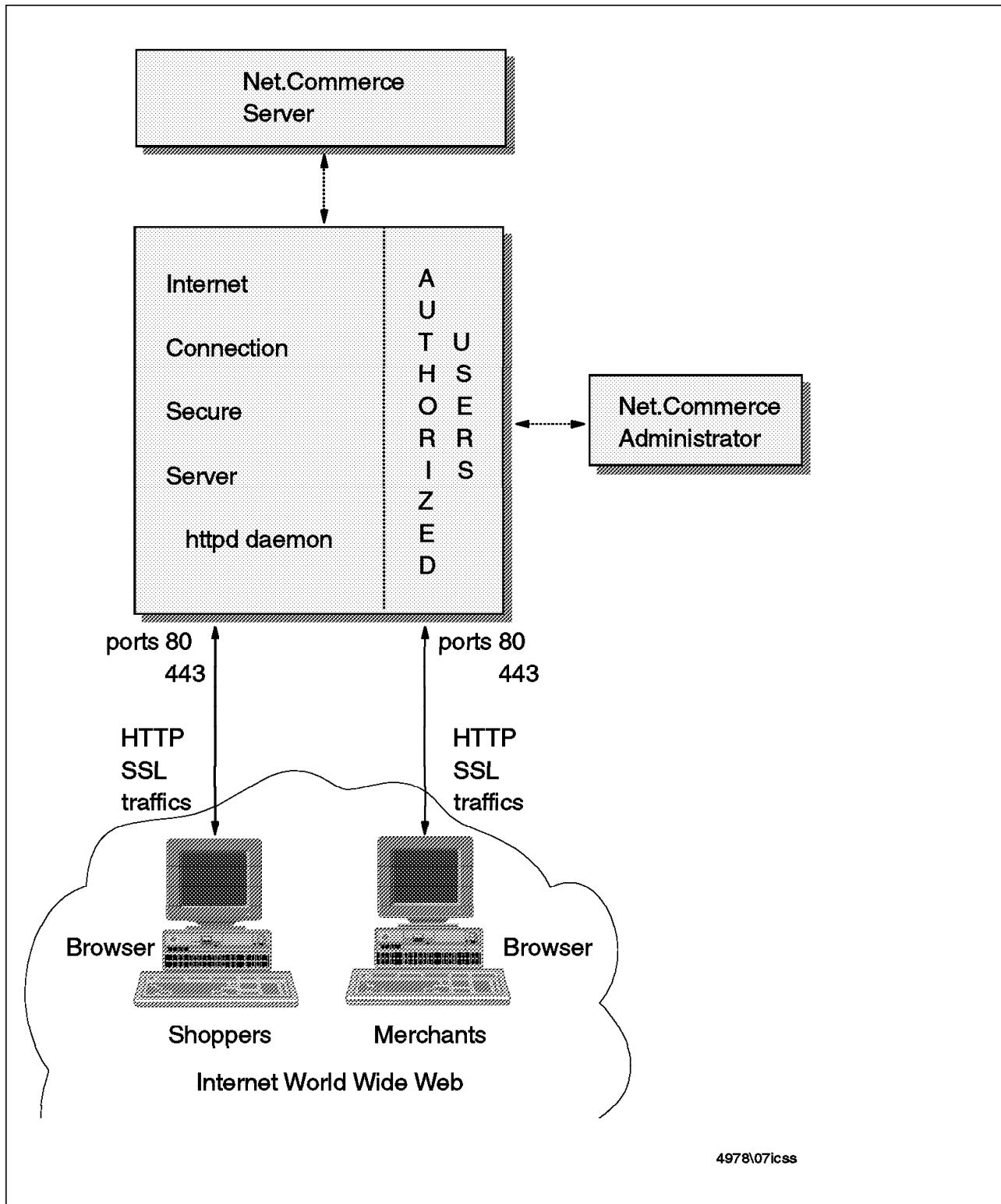


Figure 73. IBM Internet Connection Secure Server Interfaces and Interactions

The IBM Internet Connection Secure Server is a standard Web server that complies with the HTTP and SSL protocols. It listens on the IP ports dedicated to HTTP and SSL traffic (TCP ports 80 and 443, respectively).

The IBM Internet Connection Secure Server is the interface and gateway to the Net.Commerce Server for customers, and to the Net.Commerce Administrator for merchants. It relies on the SSL protocol to ensure confidentiality of customer transactions and administrators' maintenance operations. Refer to Appendix C, "Secure Sockets Layer" on page 305 for a brief overview of SSL. The IBM Internet Connection Secure Server is also in charge of the authentication of administrators. Customers can access the Internet Connection Secure Server with any SSL-compliant WWW browser. As you will see in 7.1.1.4, "Net.Commerce Administrator" on page 107, the administrators need a WWW browser that complies with a number of standards.

The IBM Internet Connection Secure Server runs as a background process. On AIX this is a daemon, httpd, which runs under the control of the subsystem resource controller. Under NT it runs as an NT service. In both cases, after you have installed the product it is automatically restarted after subsequent reboots.

You will see in the following sections how the IBM Internet Connection Secure Server interacts with Net.Commerce Server and Net.Commerce Administrator.

7.1.1.2 IBM DATABASE 2

IBM DATABASE 2 (DB2) is the relational database management system that is used by the Net.Commerce system. All the information about malls, stores, merchandise, customers and orders is stored in DB2 tables.

The implementation of IBM DATABASE 2 is highly dependent on the operating system. The scope of this document is to explain which daemons and services are utilized by IBM DATABASE 2. For more information about DB2, see *DATABASE 2 Information and Concepts Guide*.

The structure of the Net.Commerce database and the information it contains are described in *IBM Net.Commerce Customization Guide and Reference*. However, except for maintenance operations such as backup and table reorganization, you will probably never access the databases using DB2 or SQL commands. The customer accesses the database indirectly through the IBM Internet Connection Secure Server and Net.Commerce Server, and the store administrators access it through the IBM Internet Connection Secure Server and Net.Commerce Administrator. You will find further details on how this operates in Chapter 11, "Creating an Online Store with Net.Commerce" on page 241.

7.1.1.3 Net.Commerce Server

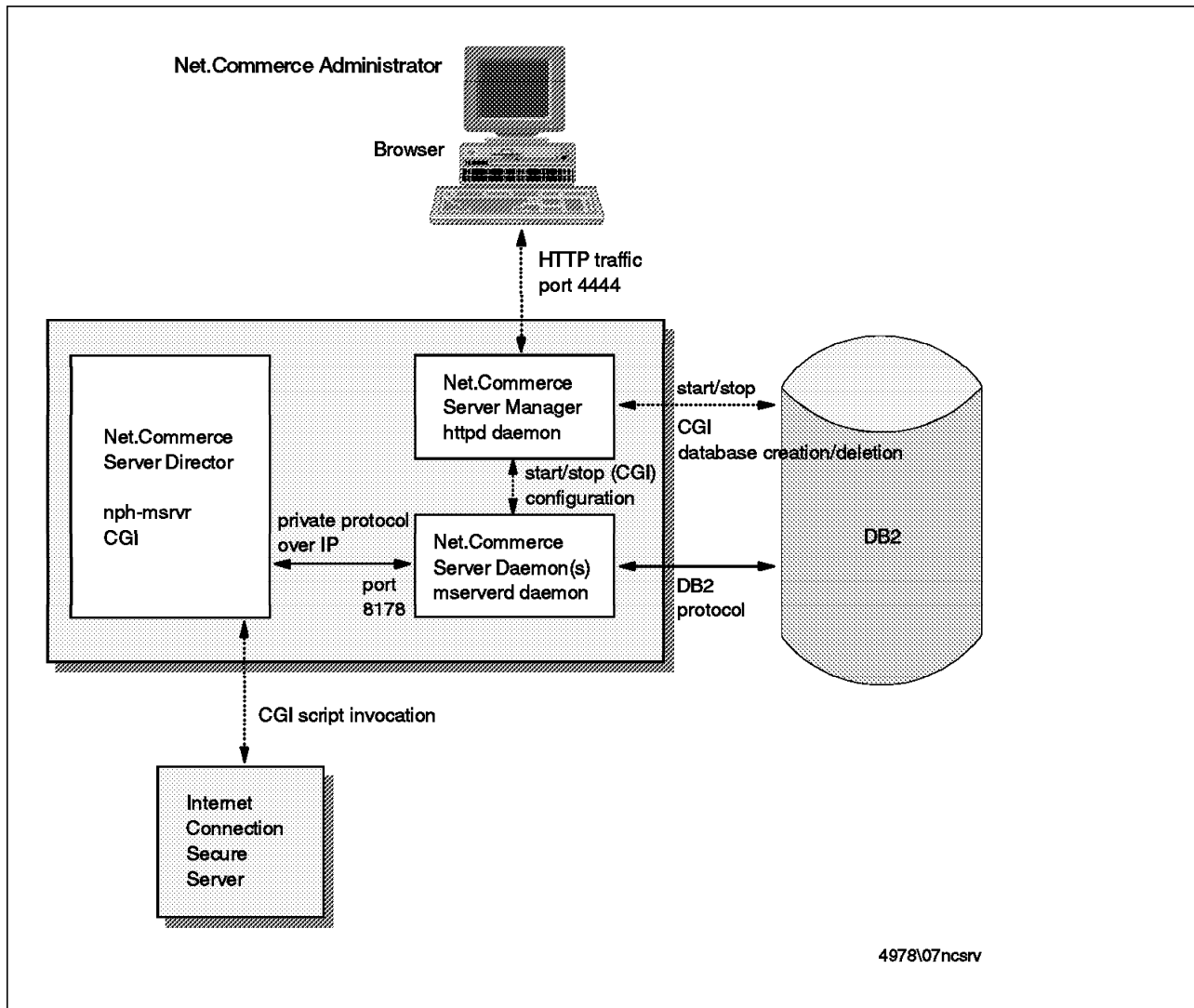


Figure 74. Net.Commerce Server Interfaces and Interactions

Net.Commerce Server is the heart of the Net.Commerce System. It acts as a gateway between the IBM Internet Connection Secure Server and IBM DATABASE 2 for operations such as customer transactions and merchant administrative tasks.

According to the *Net.Commerce Installation and Operations Guide*, the Net.Commerce Server contains two subcomponents:

- Net.Commerce Server Daemon
- Net.Commerce Server Director

However, we consider that Net.Commerce Server Manager is also a key subcomponent.

Net.Commerce Server Daemon: The Net.Commerce Server daemon retrieves data from the Net.Commerce database and builds HTML pages for either the customer or the merchant. The Net.Commerce Server daemon maintains a continuous connection to the database in order to operate efficiently. Depending

on your configuration, you can have several instances of the Net.Commerce Server daemon running, thus enabling Net.Commerce to handle more requests from customers and merchants.

None of the operators (customers, merchants or administrators) have a direct interaction with the Net.Commerce Server daemon. This is because it shares a private protocol with the Net.Commerce Server Director. The two-way communication between director and daemon operates over IP; the daemon listens on a dedicated IP port (8178 by default) and serves any incoming request from the director. So requests are all routed through the director before being executed on one of the Net.Commerce Server daemons.

The number of Net.Commerce Server daemons and the IP port used to communicate with Net.Commerce Server are configured at installation time or may be modified later via the System Configuration menu of Net.Commerce Server Manager.

Net.Commerce Server Director: The Net.Commerce Server Director is a Common Gateway Interface (CGI) program that works as an interface between the IBM Internet Connection Secure Server and the Net.Commerce Server daemon. This program, *nph-msrvr*, is invoked by the IBM Internet Connection Secure Server each time a customer accesses the database (to display products, process orders, etc.), or when a merchant maintains the data related to his store.

nph-msrvr is located in:

- /usr/lpp/NetCommerce/cgi-bin directory on AIX
- \IBM\NETCOMMERCE\CGI-BIN directory on Windows NT

CGI programs are not persistent; that is, a new instance of the program is invoked for each request. Due to this, there are no special considerations for the Net.Commerce Server Director during Net.Commerce startup or shutdown.

Net.Commerce Server Manager: The Net.Commerce Server Manager provides an HTML interface to both the Net.Commerce Server daemon and IBM DATABASE 2. It helps you configure the Net.Commerce Server daemon during the installation process or later on. It also offers you a user-friendly interface to start up and shut down the Net.Commerce Server daemon and IBM DATABASE 2. Net.Commerce Server Manager is another HTTP server (separate from the Net.Commerce ICSS server) listening on a specific port (4444 by default). This HTTP server is actually another instance of the IBM Internet Connection Secure Server with a specific configuration. Access to this Web server requires an administrator account, using HTTP basic authentication (see Appendix C, "Secure Sockets Layer" on page 305.). However, this is not considered a secure method of authentication. Because it is not mandatory to use Net.Commerce Server Manager to start and stop the Net.Commerce Server Daemon or IBM DATABASE 2 and, since the configuration of the Net.Commerce Server is an operation that you perform on rare occasions, we recommend that you do not keep the Net.Commerce Server Manager active.

You *must* change the default password of the administrator, since he or she has the authority to start or stop the Net.Commerce Server or IBM DATABASE 2, and can even drop (that is, destroy) databases!

Note that, as a security measure, Net.Commerce Server Manager is not started at boot time.

7.1.1.4 Net.Commerce Administrator

Net.Commerce Administrator provides access to the suite of tools used to maintain the sites and the stores.

This suite of tools consists of the Site Manager, the Store Manager and the Template Designer.

Net.Commerce Site Manager and Net.Commerce Store Manager are actually a collection of forms for managing malls and stores. In other words, Net.Commerce Site Manager and Net.Commerce Store Manager are not processes or daemons, but a set of HTML pages and CGI scripts that help you maintain the database information related to your malls and stores.

In fact, there is only one CGI program, shared with the Net.Commerce Server, which is the Net.Commerce Director. As we have seen above, this program interacts with IBM DATABASE 2 via the Net.Commerce Server Daemon.

Net.Commerce Template Designer is a Java applet that is downloaded into your browser from ICSS whenever you want to design and develop pages for your mall or store server. The pages that make up your mall and store are really a combination of static HTML and dynamic data fields, extracted on the fly from the DB2 database. These pages are specified as *macros*. The Template Designer is the easiest way to initially generate macros, although you may still have to tweak the result manually to get exactly the effect you want.

Because the Net.Commerce Template Designer runs on your client station it does not correspond to any process, daemon or service on the Net.Commerce host system. Net.Commerce Site Manager, Net.Commerce Store Manager and Net.Commerce Template Designer are all accessible using a Web browser, as long as this browser is:

- SSL-compliant
- Java- and JavaScript-enabled
- Capable of supporting tables and frames
- Capable of handling cookies

The browser gives you access to the IBM Internet Connection Secure Server. You are assigned an account (login and password) so that you are viewed by the IBM Internet Connection Secure Server as an authorized user for access to the Net.Commerce Administrator component.

7.2 Installing the SET Extensions

CommercePOINT Till provides secure Internet transactions for the merchant, performing the critical role in the handling of SET transactions from the Internet. It serves in the combined roles of message management, encryption, certification and records keeping.

CommercePOINT Till:

- Acquires customer orders over the Internet
- Forwards customer card information and certification to the bank

- Supports all SET messages and processes
- Manages SET messages at the merchant level
- Provides cryptographic functions
- Offers records keeping functions

7.2.1 Net.Commerce SET Extension Components Interactions

CommercePOINT Till comes as an extension to the Net.Commerce product available on the AIX and Windows NT platforms. The integration of the Net.Commerce SET Extension and its interaction with Net.Commerce is shown in Figure 75.

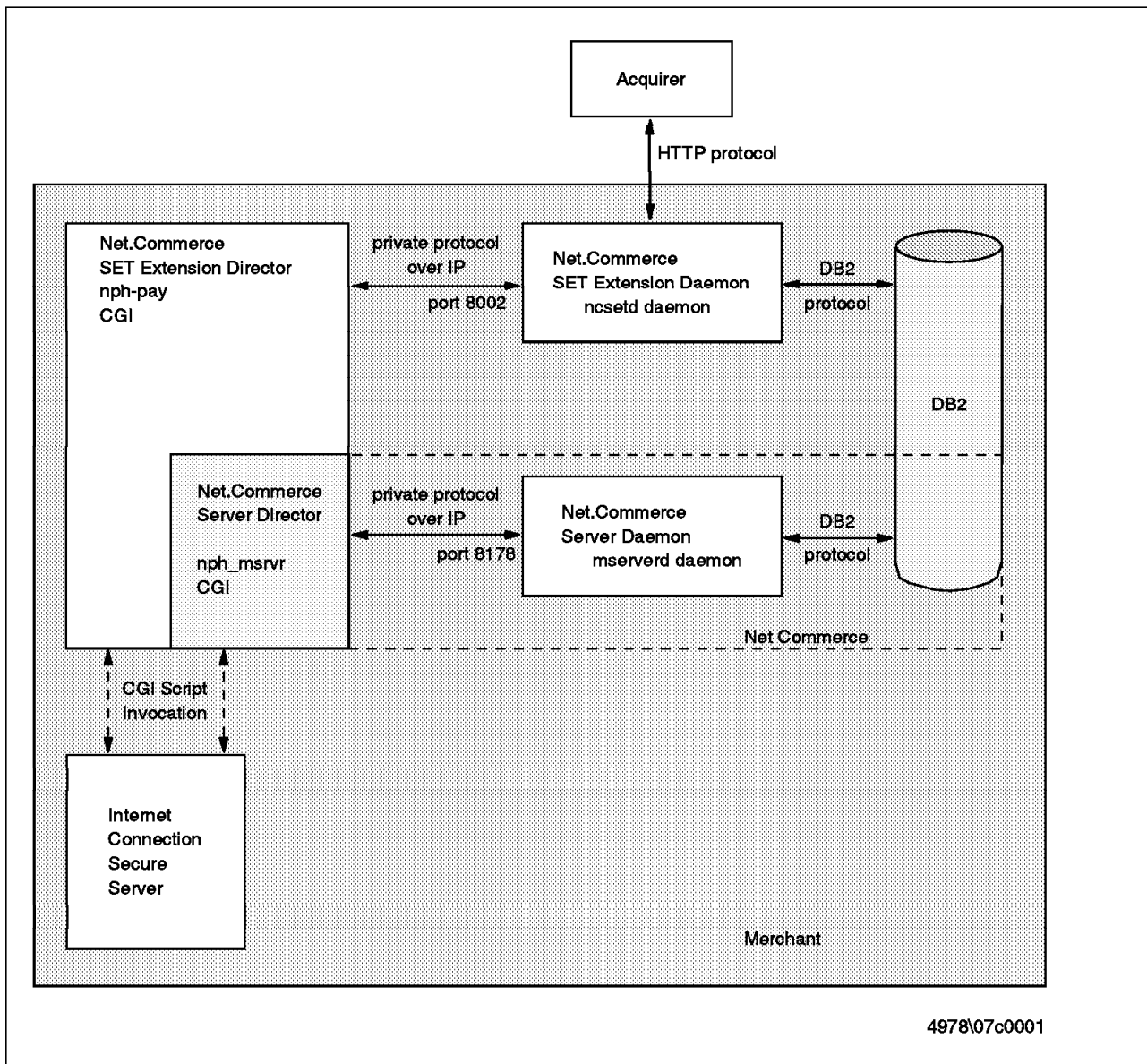


Figure 75. Net.Commerce SET Extension and Net.Commerce

The CommercePOINT Till includes two key components:

- The Net.Commerce SET Extension Director
- The Net.Commerce SET Extension Daemon

The Net.Commerce SET Extension Director is a CGI script that works as an interface between the IBM Internet Connection Secure Server and the Net.Commerce SET Extension Daemon. This CGI script, `nph-pay`, is invoked by the IBM Internet Connection Secure Server whenever a customer sends SET messages to the merchant.

The Net.Commerce SET Extension Daemon is the heart of the CommercePOINT Till:

- It processes cardholder transactions
- It communicates with the acquiring bank
- It stores information in the database

The Net.Commerce SET Extension Daemon shares a private protocol with the Net.Commerce SET Extension Director. The two-way communication between director and daemon operates over IP; the daemon listens on a dedicated IP port (8002 by default) and serves incoming requests from the director.

The Net.Commerce SET Extension Daemon communicates with the acquirer over IP. The flow of transactions between the Net.Commerce SET Extension Daemon and the acquirer is fully described in the Secure Electronic Transaction specification. The HTTP protocol is used by these two entities to communicate over the Internet; the acquirer listens on a dedicated IP port.

The Net.Commerce SET Extension Daemon maintains tables in the database. These specific tables contains information such as the status of authorization and capture requests, etc.

The CommercePOINT Till package includes:

- The Net.Commerce SET Extension Director CGI script
- The Net.Commerce SET Extension Daemon executable and the accompanying CMS library
- A set of sample Net.Commerce macros specific to the SET Extensions
- A GUI tool to administer credit card payments and orders
- A sample acquirer daemon, for test purposes

Let us look at the roles of the different components of the SET extensions in more detail.

7.2.1.1 Net.Commerce SET Extension Daemon

The `ncsetd` daemon runs permanently. It listens on a dedicated port and serves all incoming requests from the Net.Commerce SET Extension Director. It sends requests to the acquirer payment gateway when orders are processed.

At startup time, the `ncsetd` daemon checks that it holds the certificates of all the acquirers declared in the database. If the daemon detects that some certificates are missing, it requests them from the acquirers. To allow merchants to refresh acquirer certificate information without restarting `ncsetd`, `ncsetd` daemon wakes up regularly to check the "have certificate" field of the acquirer's records; it requests the certificate from the acquirer if required.

The `ncsetd` daemon also wakes up every 60 seconds to find if any orders are ready for payment capture. If it finds any in the queue, it sends the capture

request to the acquirer, waits for the response, and changes the state accordingly.

The ncsetd daemon records SET messages, authorization requests and capture requests in the database for its own processing and also for tracking purpose.

7.2.1.2 Net.Commerce SET Extension Director

The Net.Commerce SET Extension Director formats requests coming from the customer and passes them to the Net.Commerce SET Extension Daemon. According to the parameters passed to the Director, it asks the daemon to send to the cardholder a SET payment initiation message or a payment initiation message, or to process a SET purchase request or a SET inquiry request.

The Net.Commerce SET Extension Director is a CGI program that processes POST and GET requests with arguments corresponding to different call methods. Most of these methods are invoked automatically by the cardholder software. As an administrator you only need to know one call method, that is, the command to ask the server to send a SET wakeup initiation message to the cardholder.

The URL to access in order to initiate a purchase request is:

`http://<merchant_server_name>/cgi-bin/nph-pay/wakeup?SMSPSG=1&merchant_rn=1&order_rn=37`

where:

- 1 is the merchant reference number
- 37 is the order reference number

The URLs that are used to invoke this script to send the other SET messages are passed in the SET messages themselves; these URLs are built by the Net.Commerce SET Extension Daemon based on data from the incoming message and passed to the cardholder in the response message.

The format of this URL is:

`http://<merchant_server_name>/cgi-bin/nph-pay/ncset/1/37`

where:

- 1 is the merchant reference number
- 37 is the order reference number

7.2.2 Installing the SET Extensions on AIX

We ran our project using early (alpha level) code and the installation packaging of the SET extensions had not been completed. We therefore had to perform a number of manual steps that will not be required in the final product. Therefore we do not give detailed installation instructions here. You should follow the instructions in the documentation that comes with the system.

The steps performed by the installation procedure of Net.Commerce SET Extension are as follows:

- Create directories and install files on your AIX system
- Create new tables in the store database
- Modify Net.Commerce configuration files

7.2.2.1 Net.Commerce SET Extension Files and Directories on AIX

Assuming that /usr/lpp/NetCommerce is your Net.Commerce installation root, and that your language is en_US, the following files are installed during the installation process:

- /usr/lpp/NetCommerce/bin/ncsetd
- /usr/lpp/NetCommerce/bin/acqdaemon_mod.exe
- /usr/lpp/NetCommerce/cgi-bin/nph-pay
- /usr/lpp/NetCommerce/macro/en_US/ncsample/orderdssp.d2w
- /usr/lpp/NetCommerce/macro/en_US/ncsample/orderacpt.d2w
- /usr/lpp/NetCommerce/macro/en_US/order/s_orddef.inc
- /usr/lpp/NetCommerce/macro/en_US/order/s_ord.d2w
- /usr/lpp/NetCommerce/macro/en_US/order/s_orcdc.d2w
- /usr/lpp/NetCommerce/macro/en_US/order/s_ordn.d2w
- /usr/lpp/NetCommerce/macro/en_US/order/s_ordr.d2w
- /usr/lpp/NetCommerce/macro/en_US/order/s_orps.d2w
- /usr/lpp/NetCommerce/macro/en_US/order/s_orpu.d2w
- /usr/lpp/NetCommerce/html/en_US/ncadmin/sitemgr
- /usr/lpp/NetCommerce/lib/libcms.a

Note that the libcms.a library is linked to the /usr/lib directory or any other directory included in the LIBPATH environment variable.

A directory is required by the Net.Commerce SET Extension to store its key ring file. When the installation process is finalized this directory may be created automatically. In our case we created a directory called /usr/lpp/NetCommerce/set/data/merchdatabase.

In order to establish a complete list of the files installed by Net.Commerce SET Extension on your AIX system, you can use the following command:

```
ls_lpp -f <package_name>
```

where package_name is the name of the Net.Commerce SET Extension package.

7.2.2.2 Net.Commerce Configuration File

The port on which the Net.Commerce SET Extension Daemon will listen and wait for requests from the Net.Commerce SET Extension Director must be set up. This is done by adding the following directive in the Net.Commerce configuration file /etc/mserver.conf

```
MS_SET_PORT <port_number>
```

where port_number is the chosen port. The default value for the SET port is 8002.

Note that you can add an entry in /etc/services in order to declare this new service:

```
ncset      8002/tcp      # Net.Commerce SET Extension Daemon
```

7.2.2.3 Environment Setup

The Net.Commerce SET Extension Daemon and Director require that you set up your environment before operation.

In order to participate in the SET environment, the CommercePOINT Till must have at least two key pairs:

- One to use for digitally signing messages, to prove authenticity.
- One to use for encrypting keys that the merchant passes to other SET entities.

In fact, two key pairs are required for each credit card brand that the merchant accepts. Each key pair must have a public key certificate, signed by a merchant Certification Authority acting for the brand. The keys and certificates are kept in a key database, which is a password-protected file where the Net.Commerce SET Extension Daemon can retrieve them whenever it needs them. The file is called key.db and must be located in the merchant key ring directory. This directory is identified by the CmsPath environment variable.

For more information about the necessary hierarchy of key databases used by SET, refer to 8.6.1, “Key and Certificate Hierarchy Required for SET” on page 153.

As we have said, we created /usr/lpp/NetCommerce/set/data/merchdatabase as our key ring directory and therefore we set the CmsPath variable to /usr/lpp/NetCommerce/set/data/merchdatabase before starting the Net.Commerce SET Extension Daemon. The daemon will not start without a valid key database (key.db file). We discuss the process to generate keys and request certificates in 7.3.1, “Key Generation and Certificate Request” on page 119.

The Net.Commerce SET Extension Daemon and the sample acquirer both use the libcms.a library. The path to this library should be added to your LIBPATH environment variable. This can be done by modifying the profile of the administrator or by setting the LIBPATH variable before starting the daemon.

Considering all these requirements, the script to start the Net.Commerce SET Extension daemon on AIX platform will be like the one shown in Figure 76.

```
#!/bin/ksh
##
## start Net.Commerce SET Extension Daemon
##

export LIBPATH=$LIBPATH:/usr/lpp/NetCommerce/lib
export CmsPath=/usr/lpp/NetCommerce/set/data/merchdatabase

/usr/lpp/NetCommerce/bin/ncsetd
```

Figure 76. Net.Commerce SET Extension Daemon StartUp Sample Script

7.2.3 Post-Installation DB2 Checking

During the installation process, a set of new tables is added in the store database. To verify that the tables were created successfully, log in as the owner of the Net.Commerce database, connect to the database and list the tables.

```
db2 "connect to <database>"  
db2 "list tables" | more
```

The following tables should appear in the list:

1. SETACQAUTHRESP
2. SETACQOFFDAYS (*)
3. SETACQUIRER
4. SETAUTH (*)
5. SETBRAND
6. SETCAP (*)
7. SETMERCH
8. SETSTATUS (*)
9. SETTRX (*)

The tables marked with an asterisk are empty after the installation of Net.Commerce SET Extension; they will be updated by the Net.Commerce SET Extension Daemon during operation. The other tables should be populated before the initial startup of the Net.Commerce SET Extension. See 7.2.3, "Post-Installation DB2 Checking" for further details on customization of these tables. There is also a sample script, `settbldemo`, provided with the product that will populate the tables based on the DEMOMALL sample mall. This is useful if you are testing and do not want to create your own mall.

The following descriptions give some detail about the content of each table.

SETMERCH table: This table contains the declarations of the merchants in the mall operating with the SET protocol. There is one record per merchant. You should therefore have at least one entry in this table before starting up the Net.Commerce SET Extension Daemon.

SETACQUIRER table: The SETACQUIRER table contains the definitions of the acquirers. There is one record per acquirer operating for a unique brand. You should therefore have at least one entry in this table.

SETACQAUTHRESP table: The SETACQAUTHRESP table contains the specifications of the response codes returned by the acquirers. As each acquirer can specify the meaning of the authorization code, this table must be filled in for each acquirer, and each response code that you expect to receive from each acquirer. The response codes are mapped in the table to an action field, which indicates what action to take on the request. Valid values for the action field are:

- AUTH_SUCCESS - put the transaction into manual capture state
- AUTH_REJECTED - put the transaction into rejected state
- AUTH_CONFIRM - retry this authorization
- AUTH_REV_SUCCESS

- AUTH_REV_REJECTED

SETBRAND table: This table contains the declarations of the relationships between the merchants and the acquirers according to the brand. There should be one record per merchant and per brand.

7.2.3.1 Net.Commerce SET Extension Macros and Tasks

The order process of your mall must be changed in order to activate the SET mechanism. Several macros and tasks are affected in the implementation of the Net.Commerce SET Extension.

Purchase validation: Instead of activating the standard order process when the customer validates his purchase, the server must send a SET wakeup initiation message to the cardholder. The Net.Commerce command to ask the server to send such a message is:

```
/cgi-bin/nph-pay/wakeup?SPMSG=1&merchant_rn=<merchant_no>&order_rn=<order_no>
```

where:

- merchant_no is the merchant reference number
- order_rn is the order reference number

This command will therefore replace the command:

```
/cgi-bin/nph-msrvr/;order/process?merchant_rn=<merchant_no>&order_rn=<order_no>
```

in the orderdsp.d2w macro or whichever macro is affected by the ORD_DSP_PEN task in your mall/store.

A sample macro orderdsp.d2w is provided with the Net.Commerce SET Extension package. Make sure to copy it in the directory containing the macros for your mall or your store.

SET Order Failure Task: During the installation of the Net.Commerce SET Extension on the merchant server, a new task is added on the Net.Commerce system. This new task is known as ORD_SETFAIL and is associated with SET order failure as shown in Figure 77 on page 115.

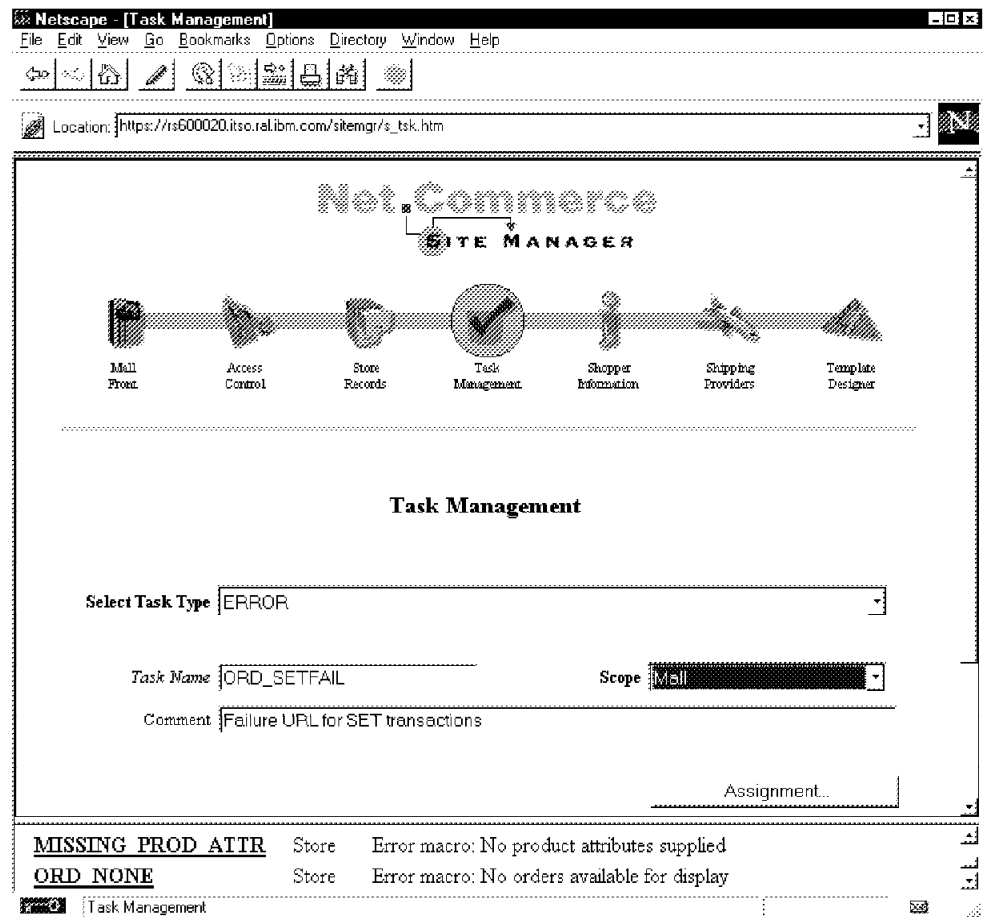


Figure 77. SET Order Failure Task

The macro that is assigned to execute the task is, by default, `ord_failed.d2w`. This macro should be created and installed in your store in the pertinent directory.

7.2.3.2 Net.Commerce SET Extension Director

If the Net.Commerce SET Extension Director is unable to communicate or receives no response from the `ncsetd` daemon, it will use an error HTML page to signify it to the requestor. This error page has the following URL:

`http://<merchant_server_name>/ncerror/seterror.html`

A sample `seterror.html` page for use with the DEMOMALL is provided with the Net.Commerce SET Extension. Like all the HTML pages provided with the Net.Commerce package, you may modify it to be coherent and uniform with the other HTML pages.

7.2.3.3 Order Administration

In order to provide manual capture kickoff and tracking capability, a simple administration tool is provided in Net.Commerce SET Extension. The URL to bring up the administration tool is:

`http://<host_name>/sitemgr/s_ord.htm`

After typing in the URL, you will be prompted to enter a user ID and password. You must log in as a site administrator to be authorized to use this tool for

capture. Please refer to the Net.Commerce documentation for more details about administration accounts.

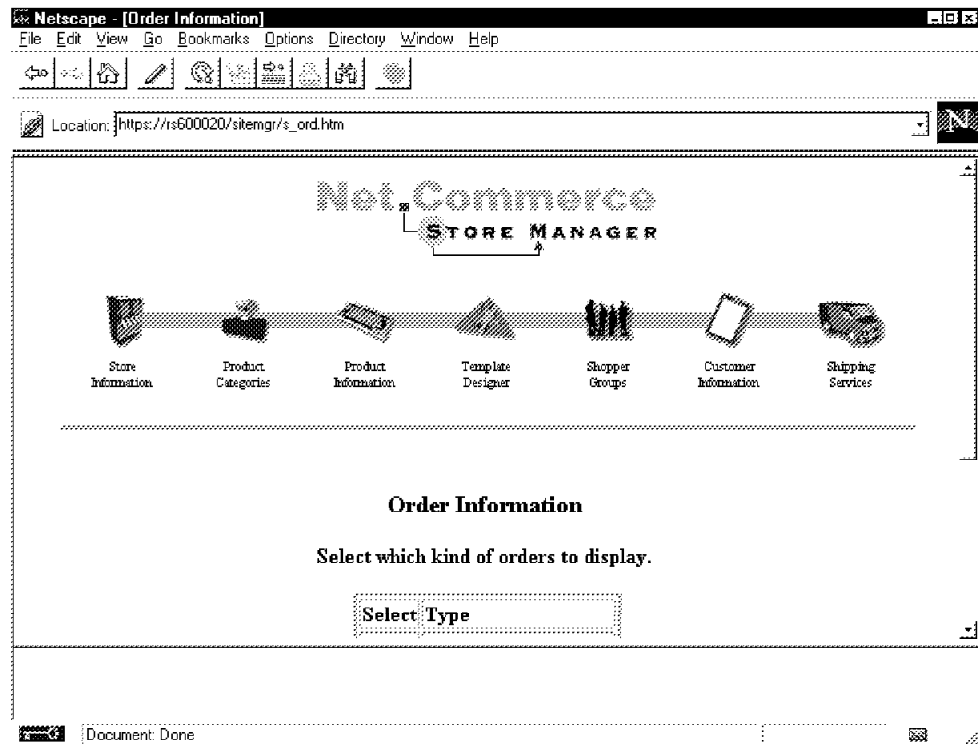


Figure 78. Order Information Page

The administration tool allows you to list the order information by category. The orders for which an authorization has been obtained from the acquirer but for which the capture request has not been sent yet are in the "awaiting capture" state.

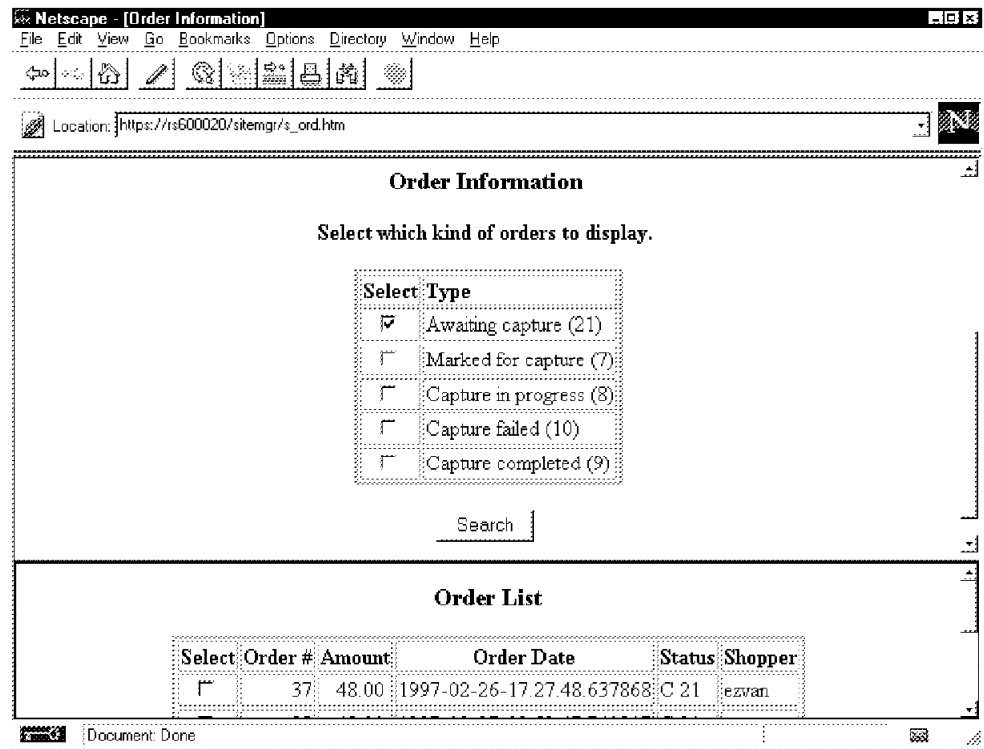


Figure 79. Order List

This GUI tool lets you mark the orders you wish to capture or revise. You can use the GUI to change their state to "marked for capture" so that ncsetd, when it wakes up to find out the next order which is ready to capture, will send the capture request to the acquirer. The ncsetd daemon will change the state to "capture success" if the response is successful.

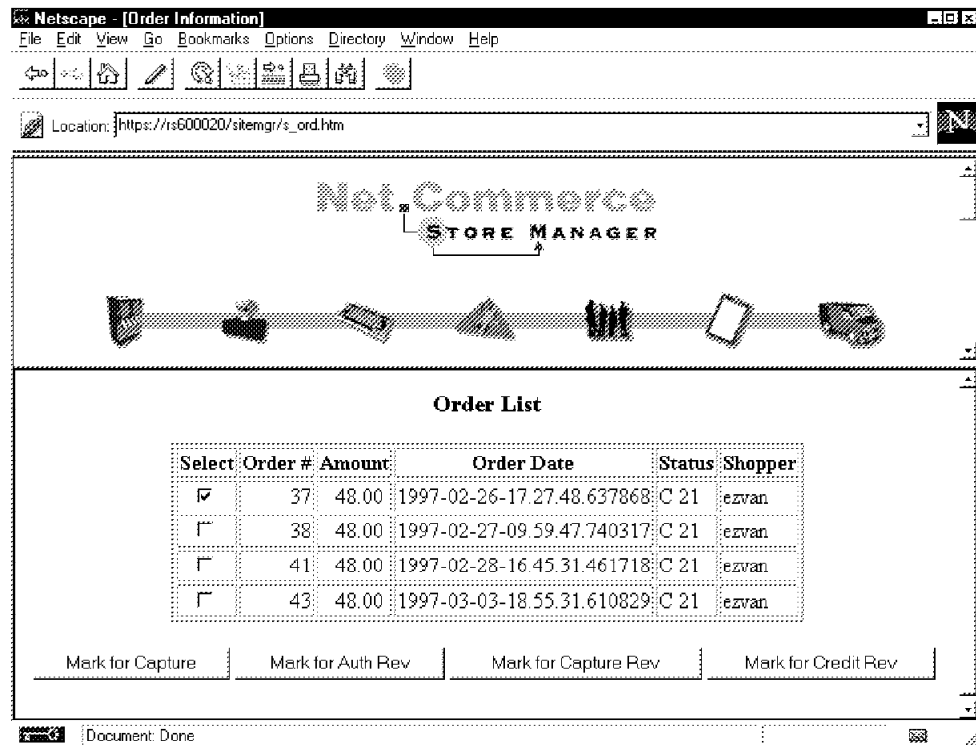


Figure 80. Marking for Capture

7.2.4 Sample Acquirer

When you first set up your Net.Commerce SET Extension installation you will probably not want to connect to the real acquirer payment gateway. For this reason, a sample acquirer is provided with the Net.Commerce SET Extension package to help you in setting up and testing your environment. This acquirer wait processes SET requests and composes SET responses. It simulates what would occur at an acquiring bank during the processing of the different requests.

The executable for this sample acquirer is
`/usr/lpp/NetCommerce/bin/acqdaemon_mod.exe.`

As for a real acquirer, the sample acquirer needs its own certificate to authenticate itself. The key file containing the certificate is `key.db`, and should be kept in a key ring directory. Furthermore, it keeps personal data in another private directory. The paths of these directories are passed as arguments to the command that launches the sample acquirer. In our case we used the following directory paths:

```
/usr/lpp/NetCommerce/set/data/acqdatabase and  

/usr/lpp/NetCommerce/set/data/pdudata.
```

Lastly, the sample acquirer uses the `libcms.a` library; so that the `LIBPATH` variable should be set up before running the sample executable.

An example script to start up the sample acquirer is shown in Figure 81 on page 119.

```
#!/bin/ksh
##
## start Net.Commerce SET sample acquirer
##

export LIBPATH=$LIBPATH:/usr/lpp/NetCommerce/lib
AcqCmsPath=/usr/lpp/NetCommerce/set/data/acqdatabase
PduPath=/usr/lpp/NetCommerce/set/data/pdudata

/usr/lpp/NetCommerce/bin/acqdaemon_mod.exe $AcqCmsPath $PduPath 8888 s
```

Figure 81. Acquirer StartUp Sample Script

The arguments to the sample acquirer are:

- /usr/lpp/NetCommerce/set/data/acqdatabase is the path of the directory containing the key file
- /usr/lpp/NetCommerce/set/data/pdudata is the path of the private directory to keep the personal data
- 8888 is the port on which the acquirer listens and waits for the merchant requests

The sample acquirer must be running before the Net.Commerce SET Extension Daemon is started, so that the ncsetd daemon finds the acquirer and can request the payment gateway certificate.

In order for the merchant to communicate with the sample acquirer, the record in the SETACQUIRER table must contain:

- The host name of the machine running the sample acquirer in the SETAHOSTNAME field
- The listening port (8888) in the SETAHOSTPORT field

7.3 Configuring the Payment System

To configure the payment system on your merchant server, you have to go through the following steps:

1. Get your merchant certificates and store them in your key file
2. Customize the SET tables
3. Customize the macros assigned to order tasks
4. Test

7.3.1 Key Generation and Certificate Request

In the early version of Net.Commerce SET Extension that we used for this project, the procedure for generating merchant key pairs and requesting certificates did not yet exist. We generated a set of keys and certificates on the CA machine for testing purposes, which then had to be manually copied to the system running IBM Registry for SET.

The final versions of the merchant and CA products will have streamlined processes for key generation and signature, similar to the cardholder process. However, we have described the process as it existed during our project in

8.10.1, "Producing Test Certificates for Merchant and Payment Gateway" on page 165.

No matter what mechanism is used to arrive at it, the final result of the key generation and signature process is that the merchant system will have a key ring directory, containing four files:

- key.db
- keypair.db
- crl.db
- bci.db

key.db is the merchant's key database. It contains the two key pairs for the merchant (signature key and key-exchange key) and the certificates for them, signed by the Merchant CA. These certificates contain the complete authentication chain, made up of the certificates for the root, brand and merchant CAs.

Net.Commerce SET Extension looks for its key database in the directory indicated by the CmsPath environment variable (see Figure 76 on page 112).

7.3.2 SET Tables Configuration

To customize the SET tables:

1. Log in as the instance owner.
2. Connect to the database: db2 connect to winstead

Several tables need to be configured in order to suit your configuration. The records in these tables contain information related to:

- The brands accept by the merchant
- Acquirers used by the merchant
- Authorization codes returned by the acquirers

7.3.2.1 SETMERCH Table Configuration

This table should contain one record per merchant working with the SET protocol. Before you fill in this table with the pertinent information, check the existence and the current content of the table:

```
db2 "select * from setmerch"
```

To configure the SET merchant table, you will need the merchant reference number as stored in the Net.Commerce merchant table. To get this information:

```
db2 "select * from merchant"
```

The merchant reference number is stored in the MERFNBR field.

At this point, you have two options if the content of the database does not correspond to your configuration:

1. To delete the records in your database: db2 "delete * from setmerch"
2. To modify the existing record(s)

Let us choose the first option:

```
db2 "delete * from setmerchant"
```

```
db2 "insert into setmerchant values (1, 'N', 'Y', 'ITS0', '1234', 840, -2)"
```


where:

- 1 is the ITSO store merchant reference number.
- 'N' is for not doing the capture during the payment request processing.
- 'Y' is for doing the authorization during the payment request processing.
- 'ITSO' is the merchant identification as defined in the certificate request form.
- '1234' is the merchant category (currently not used).
- 840 is the country currency code, that is Dollars for the ITSO store.
- -2 is the exponential factor for the currency.

7.3.2.2 SETACQUIRER Table Configuration

The SETACQUIRER table should contain one record per acquirer operating for a unique brand. Before you fill in this table with the pertinent information, check the current content of the table:

```
db2 "select * from setacquirer"
```

and clean the table if necessary:

```
db2 "delete * from setacquirer"
```

Before adding a record to define your acquirer, you have to decide what will be the reference number for the acquirer. This reference number must be unique. When you have gathered all the information related to your acquirer, such as the acquirer's business hours, type:

```
db2 "insert into setacquirer values (1, 'rs600020.itso.ral.ibm.com',  
      8888, 5, 'N', 'N', 6, 23, 30, 3, 1800, -1, 24)"
```

where:

- 1 is the acquirer reference number. The first one in this example.
- 'rs600020.itso.ral.ibm.com' is the host name of the acquirer of the brand in question.
- 8888 is the port number on which the acquirer is listening.
- 5 is the maximum concurrent connections supported by the acquirer.
- 'N' indicates that the acquirer is not off on Saturdays.
- 'N' indicates that the acquirer is not off on Sundays.
- 6 indicates that business starts each day at 6 a.m.
- 23 indicates that business stops at 11 p.m.
- 30 is the time between transaction replies.
- 3 is the number of transaction replays allowed.
- 1800 is time between transaction retrys.
- -1 is the number of transaction retries allowed.
- 24 is the minimum wait before first retry of transaction.

7.3.2.3 SETBRAND Table Configuration

The SETBRAND table should contain one record per merchant and per brand accepted by the merchant. Before you fill in this table with the pertinent information, check the current content of the table:

```
db2 "select * from setbrand"
```

and clean the table if required:

```
db2 "delete * from setbrand"
```

To insert a record in this table, type:

```
db2 "insert into setbrand values (1, 1, 'N', 'Pass')"
```

where:

- 1 is the merchant reference number for the ITSO store.
- 1 is the acquirer reference number for the Pass brand acquirer (which you chose in the previous step).
- 'N' indicates that the certificate of the acquirer is not yet in the database.
- 'Pass' is the brand name.

Note: The third field must be set to N when you add a record in the database. This field will be updated automatically by the Net.Commerce SET Extension Daemon when it has received the acquirer certificate.

7.3.2.4 SETACQAUTHRESP Table Configuration

The SETACQAUTHRESP table contain the declarations of the response codes returned by the acquirer when an authorization is requested. One record per authorization code and per acquirer must be added in this table.

The information related to the returned authorization code should be obtained from the acquirer prior to the configuration. In our lab environment, the codes returned by the Pass brand acquirer are:

- '000' for a successful authorization
- '001' for a rejected authorization

Note that separate entries are required for authorization response codes and authorization reversal responses codes, even if the response code is the same.

To define the response codes:

```
db2 "insert into setacqauthresp values ( 1, '000', 'AUTH_SUCCESS', NULL )"
db2 "insert into setacqauthresp values ( 1, '001', 'AUTH_REJECTED', NULL )"
db2 "insert into setacqauthresp values ( 1, '000', 'AUTH_REV_SUCCESS', 'R' )"
db2 "insert into setacqauthresp values ( 1, '001', 'AUTH_REV_REJECTED', 'R' )"
db2 "insert into setacqauthresp values ( 1, '000', 'AUTH_REV_SUCCESS', 'R' )"
db2 "insert into setacqauthresp values ( 1, '001', 'AUTH_REV_REJECTED', 'R' )"
```

where:

- 1 is the merchant reference number.
- '000' and '001' are the response codes.
- 'AUTH_SUCCESS', 'AUTH_REJECTED', 'AUTH_REV_SUCCESS' and 'AUTH_REV_REJECTED' are the response actions.
- NULL denotes that the response code is generated for an authorization request and 'R' is the response code generated for an authorization reversal request.

7.3.3 Macro Customization

The Net.Commerce package comes with a set of macros to work with order commands. These macros are assigned to specific tasks as shown in Table 2.

Default Macro Name	Task Name	Macro Description
orderlstp.d2w	ORD_LST_PEN	Macro to list all orders pending for a given user
orderlstc.d2w	ORD_LST_COM	Macro to list all past/completed orders for a given user
orderdsp.d2w	ORD_DSP_PEN	Macro to display a given pending order for a given user
orderdspc.d2w	ORD_DSP_COM	Macro to display a given past/completed order for a given user
orderacpt.d2w	ORD_OK	Macro to acknowledge the acceptance of an order just completed

These macros are not SET oriented in that they were not developed to take care of the peculiarities of the SET extension. That is why it is necessary to customize these macros:

- To initiate the SET transaction when the customer purchases something
- To request information from the SET database tables

Before detailing the customization that you will have to do, let's summarize the order process:

1. The shopper adds products and items to his or her shopping cart and provides shipping information. When the shopper displays the order for the first time, and only then, the order is created in the database.
2. When the order is displayed, it is locked automatically. If the shopper makes changes to the order after displaying it, the order becomes unlocked.
3. The order can be placed when the order is unlocked.

The purchase occurs actually when the order is placed. This is the step of the process that needs to be modified in order to initialize the SET transaction.

7.3.3.1 Pending Order Details Macro Customization

The store page on which the customer places the order is the store page displayed by the macro assigned to the ORD_LST_PEN task, that is, by default, orderdsp.d2w. The function of this macro is to display the details of a pending order and to allow the customer to place the order. In other words, the store page displayed includes a **Purchase** button that links to the process URL as shown in Figure 82 on page 124.

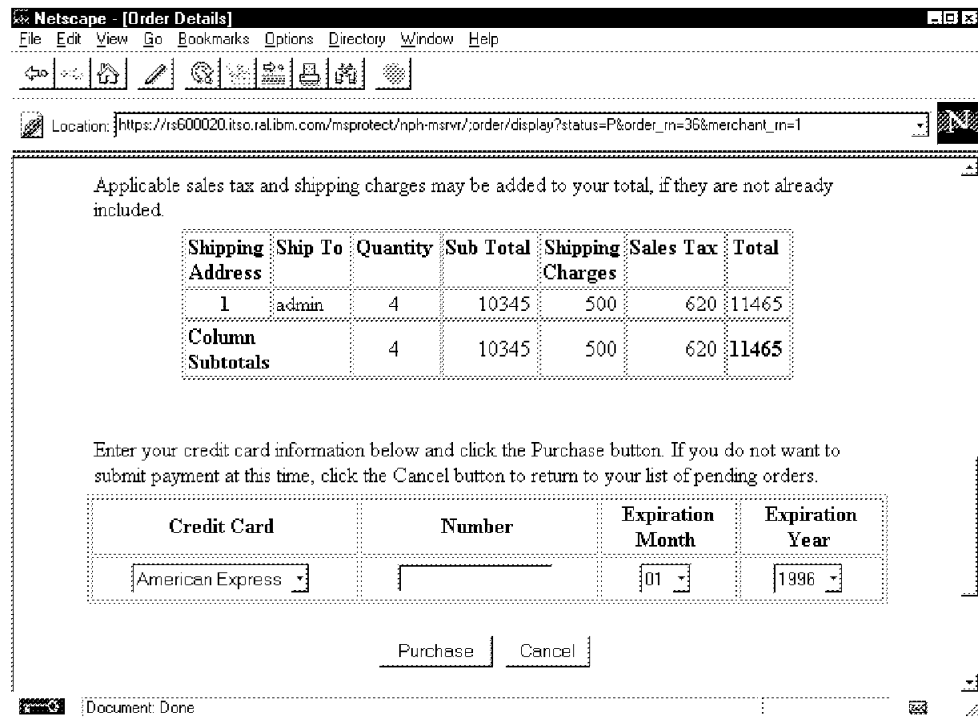


Figure 82. Order Details Store Page

The orderdsp.d2w macro provided with the Net.Commerce package links to the process command as follows:

```
<form action="/cgi-bin/nph-msrvr;/order/process">
  <input type=hidden name="order_rn" value="$(order_rn)">
  <input type=hidden name="merchant_rn" value="$(V_merrf)">
```

This has to be replaced by the wakeup command to initialize the SET transaction.

Furthermore, it no longer makes sense to give the user the choice of the type, number and expiration date of the payment card, because the SET extensions will handle that information.

Let us then modify the default macro to suit the SET peculiarities and to display a customized store page as shown in Figure 83 on page 125.

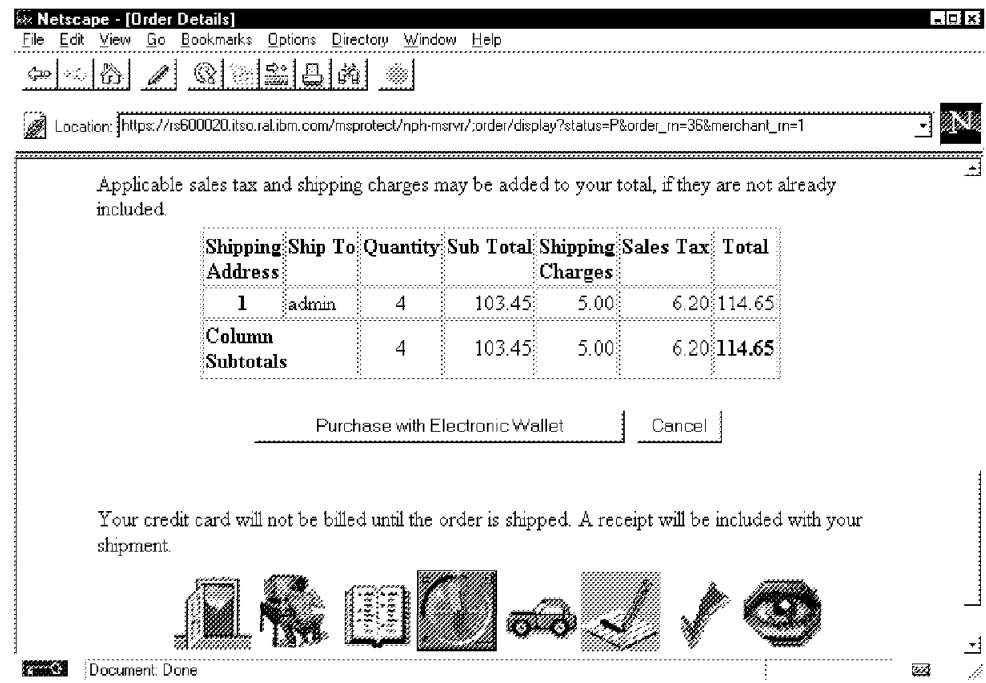


Figure 83. ITSO Corner Order Details Page

To customize the store page:

1. Copy the Net.Commerce SET Extension orderdsp.d2w macro, as follows:
cp orderdsp.d2w /usr/lpp/NetCommerce/macro/en_US/winstead/itso_orderdsp.d2w
2. Edit the macro file.
3. Assign the new macro to the ORD_DSP_PEN task.

The modifications in the default macro file, for the ITSO Corner Store are the following:

- The background tile is changed, as for all the store pages.
- The fields corresponding to the card description are removed.
- The URL assigned to the submit button is that to wake up the Net.Commerce SET Extension Daemon.

The critical part of the new macro file is shown in Figure 84 on page 126.

```

</p>
<p>
%EXEC_SQL(subtot_prc)
%EXEC_SQL(tot_items)
%EXEC_SQL(tot_prc)
<p>
<p>
<form action="/cgi-bin/nph-pay/wakeup"> 1
    <input type=hidden name="order_rn" value="$(order_rn)">
    <input type=hidden name="merchant_rn" value="$(V_merrf)">
    <input type=hidden name="SPSMSG" value="1">
<center>
<table>
<tr>
<td>
        <input type=submit value="Purchase with Electronic Wallet">
</form>
</td>
<td><form action="/cgi-bin/nph-msrvr;/order/list">
        <input type=hidden name="status" value="P">
        <input type=submit value="Cancel">
</form></td>

```

Figure 84. itso_orderdsp.d2w Macro Source File Extract

Note:

1 This is the reference to the CGI program to trigger the SET transaction. To assign this macro file to the ORD_DSP_PEN task, log in to the Net.Commerce Administrator and go to the Task Management menu. Fill in the Task Management form as shown in Figure 85 on page 127.

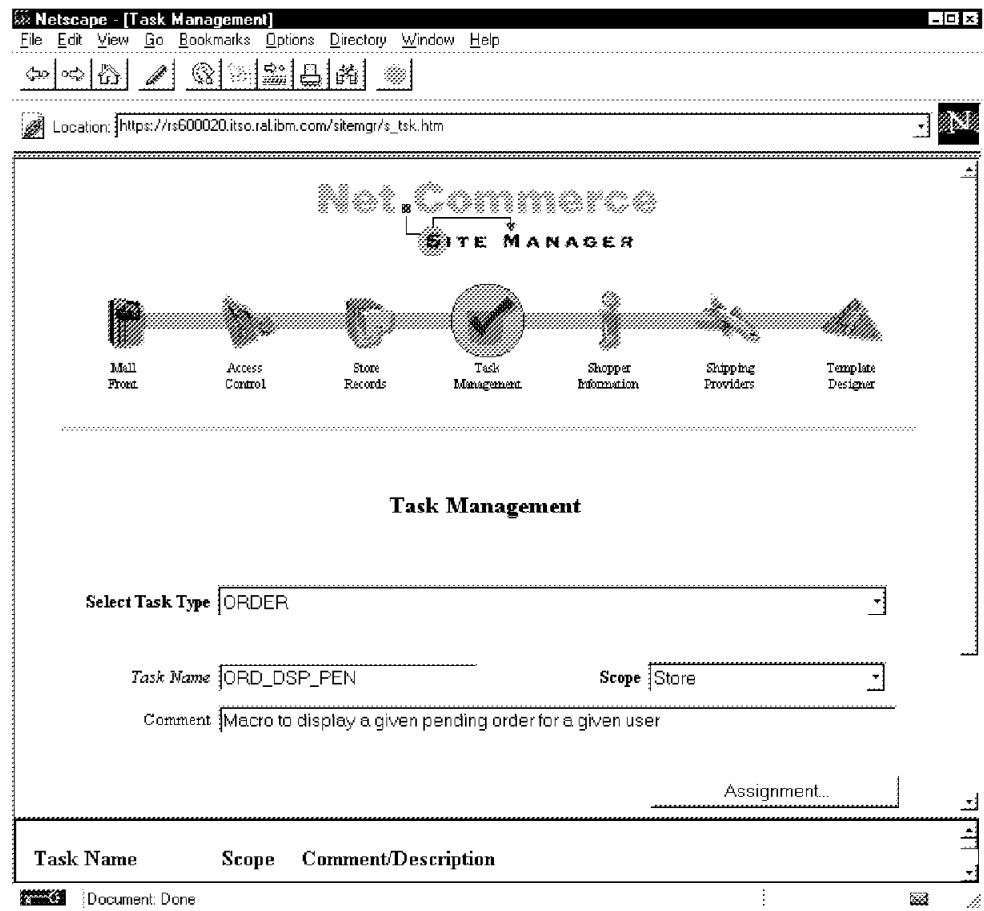


Figure 85. Task Management Form

1. Select the Order task type.
2. Click on the ORD_DSP_PEN in the list in the bottom frame.
3. Change the task Scope to Store.
4. Click on **Save**.
5. Click on **Assignment**.

On the assignment form:

1. Select the store from the drop-down list.
2. Enter the macro file name, that is itso_orderdsp.d2w.
3. Click on **Save**.

These steps are shown in Figure 86 on page 128.

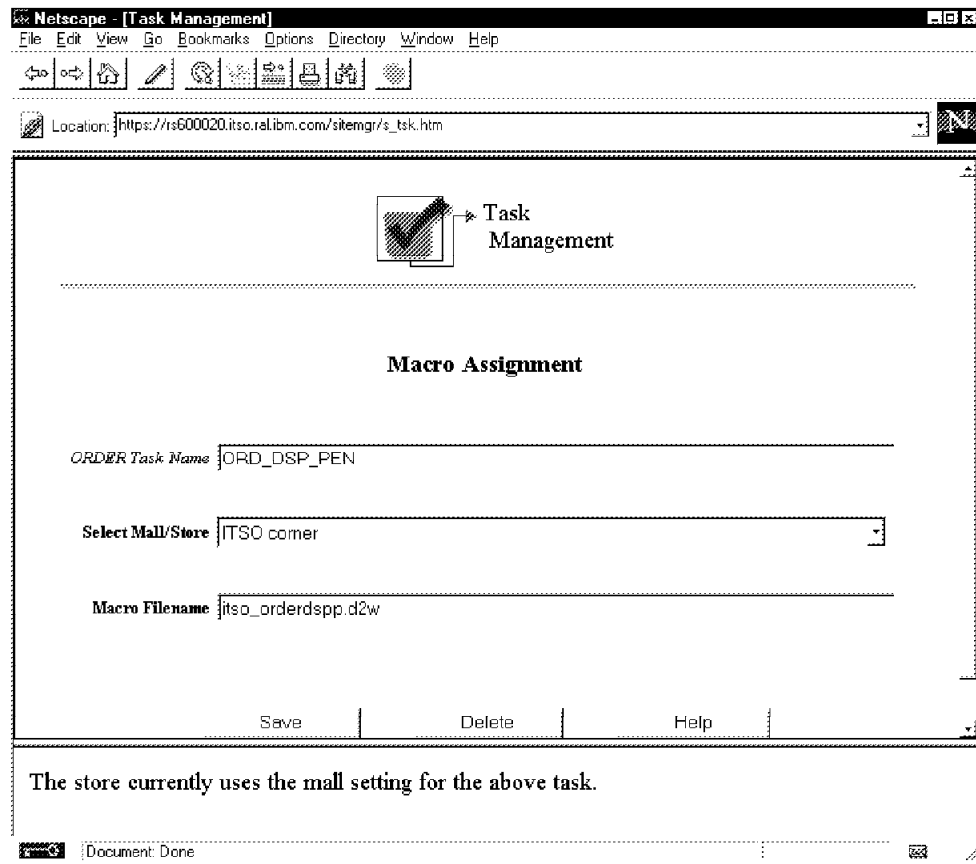


Figure 86. Macro Assignment Form

You would modify the macros assigned to the ORD_LST_PEN, ORD_LST_COM, ORD_DSP_COM or ORD_OK in the same way. We did not find any specific reason to change the macros to list all orders or to display a given pending order. But you will likely modify, as we did, the macros to acknowledge the acceptance of an order and to display a given/past completed order.

The macro to acknowledge an order just completed would need to refer to the fact that the payment has been authorized but that your credit card will not be charged until the capture is done. Depending on the type of goods you sell on your merchant server, the capture will occur when the goods are shipped or immediately. This store page is a simple acknowledgment and the customization of the corresponding macro is a pure formality.

The macro to display a past/completed order needs to be modified, since the sample macro provided with the Net.Commerce package refers to fields in the database that are updated only with the standard order process.

We customized the macro in order to give the customer some SET-related information, as follows:

1. cp orderdsp.d2w /usr/lpp/NetCommerce/macro/en_US/winstead itso_orderdsp.d2w
2. Edit /usr/lpp/NetCommerce/macro/en_US/winstead/itso_orderdsp.d2w
3. Assign the new macro to the ORD_DSP_COM task for the ITSO store.

The order details page for the ITSO store is shown in Figure 87 on page 129 and the source code of the modified sections in the macro file are shown in Figure 88 on page 130 and Figure 89 on page 132.

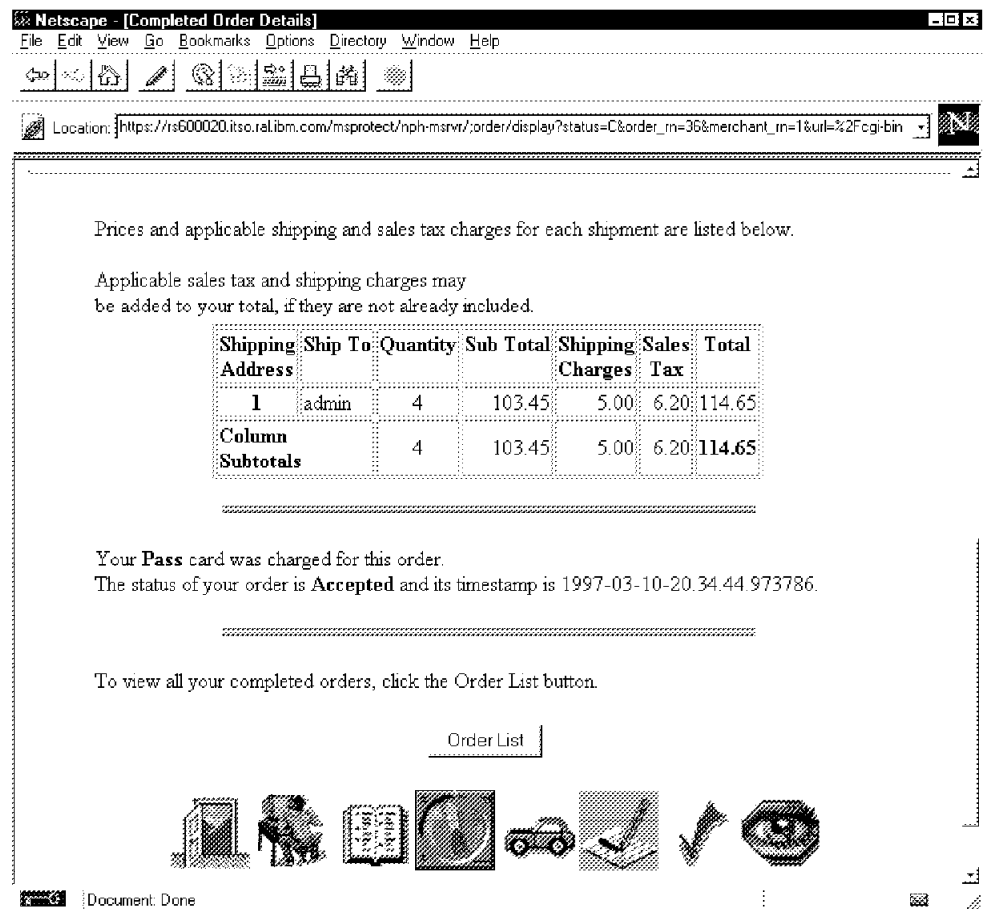


Figure 87. Order Details: ITSO Store

On the ITSO Order Details Page, the customer will find specific information such as the brand of the card charged for the order, the timestamp and the current status of the order. This information is extracted from the SET tables in the database.

```

%SQL (paym) {
    select setsbrandid, setustmp, case setsstatcode 1
        when 7 then 'Accepted'
        when 8 then 'Accepted'
        when 9 then 'Completed'
        when 10 then 'Refused'
        when 21 then 'Accepted' 2
    end
    from setstatus, setauth 1
    where setsornbr=$(order_rn) and setsmenbr=$(merchant_rn)
    and setuornbr=$(order_rn) and setumenbr=$(merchant_rn)

    %SQL_REPORT {
        %ROW {
            <p>
            <p>
            <center>
            <table width=600>
            <tr>
            <td>
            Your <b>$(V1)</b> card was charged for this order.<br>
            The status of your order is <b>$(V3)</b>
            and its timestamp is $(V2).
            </td>
            </tr>
            </table>
            </center>
            %}
        %}

    %SQL_MESSAGE {
        100: {
            </table>
            <p><b>No payment method information could be found for this
            order.</b>
            %}
        default: {
            </table>
            <font size=+3><b>Database Error:</b><p>
            A database error occurred when attempting to
            get the status order from the database. Please contact the
            merchant server administrator.</font>
            %}
        %}
    %}
}

```

Figure 88. itso_orderdspc.d2w Macro File - Extract #1

Notes:

1 This select statement reads from the SETSTATUS and SETAUTH tables the fields related to the order:

- SETSBRANDID contains the brand of the payment card.
- SETUSTMP contains the timestamp of the SET transaction.

- SETSTATCODE contains the current status code of the SET transaction.

2 The text detailing the current status of the SET transaction is evaluated on the status code stored in the SETSSTATCODE field in the SETSTATUS table. See in Table 3 for more information on the status codes. In this sample macro file, we do not deal with all of the status codes.

Code	Status
1	Payment Init Request Received
2	Payment Request Received
3	Authorization Ready
4	Authorization Sent
6	Authorization Response Code Unknown
7	Marked for Capture
8	Capture in Progress
9	Capture Completed
10	Capture Failed
11	Mark for Credit
12	Credit in Progress
13	Credit Completed
14	Credit Failed
15	Marked for Credit Revision
20	Authorization Manual
21	Ready for Capture
25	Marked for Authorization Revision
30	Authorization Pending
31	Authorization Rejected
32	Capture Closed
33	Capture Unknown
35	Marked for Capture Revision
40	Timer Wait. Transaction scheduled to occur later.

```

%HTML_REPORT {
<html>
<head>
  <META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1996 01:01:01 GMT">
  <title>Completed Order Details</title>
</head>
<body background="/winstead/quebra_ca.gif" bgcolor="#FFFFFF">

%EXEC_SQL(InitSQL)
%include "$(MailHeader)"

%EXEC_SQL(merchant)
%EXEC_SQL(currency)
%EXEC_SQL(ship_to)
<p>
<hr>
<p>

%EXEC_SQL(subtot_prc)
%EXEC_SQL(tot_items)
%EXEC_SQL(tot_prc)
<p align=center><strong></strong></p>

%EXEC_SQL(paym) 3

<p align=center><strong></strong></p>

<center>
<table width=600>
<tr>
<td>
To view all your completed orders, click the Order List button.
</td>
</tr>
</table>
</center>

<center>
<form action="/cgi-bin/nph-msrvr;/order/list">
  <input type=hidden name="status" value="C">
  <input type=hidden name="url" value="/cgi-bin/nph-msrvr;/order
/display?status=$(V_status)&order_rn=$(order_rn)
&merchant_rn=$(merchant_rn)">
  <input type=submit value="Order List">
</form></center>

%include "$(MailFooter)"

</body>
</html>

```

Figure 89. itso_orderdspc.d2w Macro File - Extract #2

Notes:

3 Here is the call to the SQL routine detailed in Figure 88 on page 130.

7.3.3.2 Failed Order Macro

As described in “SET Order Failure Task” on page 114, a new task is introduced by the Net.Commerce SET Extension. This task and the corresponding macro are described in Table 4.

Default Macro Name	Task Name	Macro Description
ord_failed.d2w	ORD_SET_FAILED	Failure URL for SET transactions

The macro to display the Order Failure page for your store or mall should be written and implemented as for the other store pages.

The Order Failure page of the ITSO store was designed to return to the customer the text field explaining the reason of the failure as shown in Figure 90.

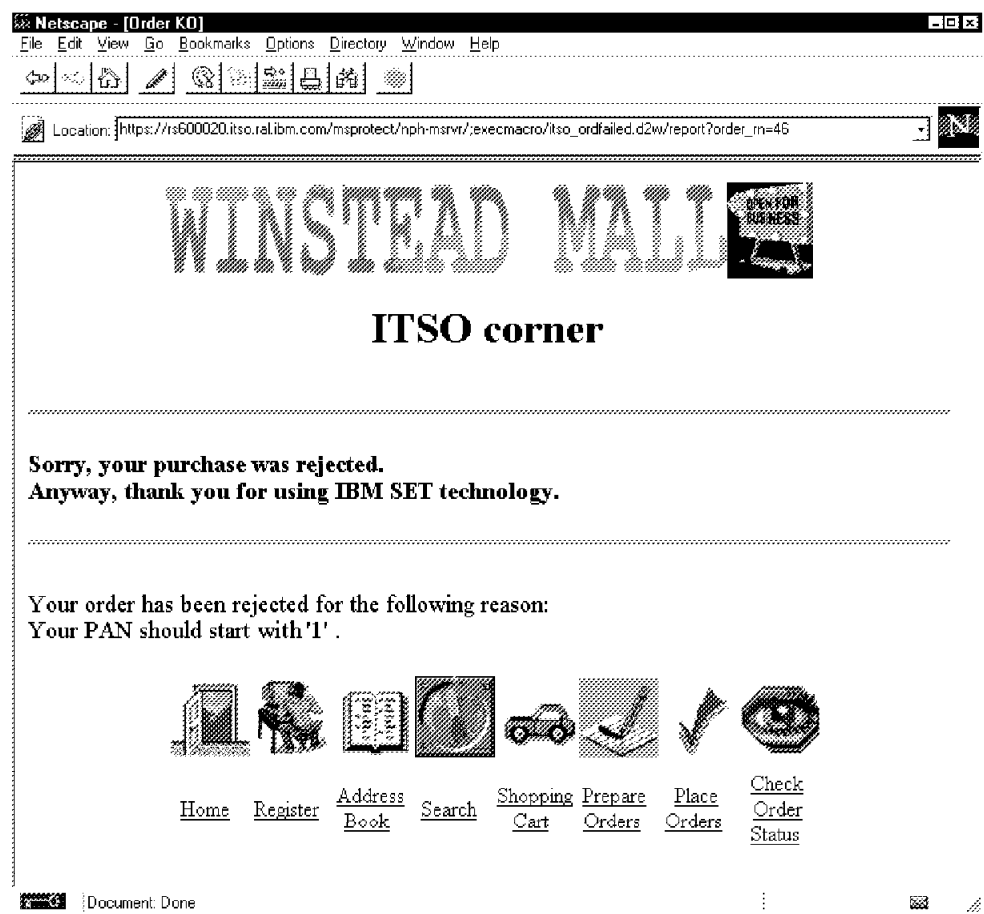


Figure 90. ITSO Order Failure Page

The reason why this order was rejected is that the Personal Account Number of the card does not start with a '1'. This was an arbitrary restriction that we implemented in the acquirer payment gateway as a method to generate a failure case. The explanation is actually sent by the acquirer to the merchant as the

result of the authorization request (the payment gateway exit code that does this is shown in Figure 139 on page 200).

To implement the ITSO Order Failure Page:

1. Create and edit:
/usr/lpp/NetCommerce/macro/en_US/winstead/itso_ordfailed.d2w
2. Assign the new macro to the ORD_SET_FAILED task for the ITSO store.

The source code of the ITSO Order Failure Page is listed in Figure 91 on page 135.

```

%{=====
The sample Templates, HTML and Macros are furnished by IBM as simple
examples to provide an illustration. These examples have not been
thoroughly tested under all conditions. IBM, therefore, cannot guarantee
imply reliability, serviceability or function of these programs. All
programs contained herein are provided to you "AS IS".

The sample Templates, HTML and Macros may include the names of individuals,
companies, brands and products in order to illustrate them as completely as
possible. All of these are names are fictitious and any similarity to the
names and addresses used by an actual business enterprise is entirely
coincidental.

(C) Copyright IBM Corp. 1995, 1996.
=====}%}

%define {
%}

%SQL(InitSQL){

    select mthead, mtfoot, mtheadbase from mall

    %SQL_REPORT{
        %ROW{
            @DTW_assign( MallHeader, V1)
            @DTW_assign( MallFooter, V2)
            @DTW_assign( MallBase, V3)
        }
    }

    %SQL_MESSAGE{
        default: { %} :continue %}

%}

%SQL(reason) {
    select seturesprea from setauth where setuornbr = $(order_rn) 1
    %SQL_REPORT {
        %ROW{
            @DTW_assign( Reason, V1 ) 2
        }
    }
%}

%SQL(merchant) {
    select mestname from orders, merchant where
        orrfnbr = $(order_rn) and
        ormenbr = merfnbr

    %SQL_REPORT {
        %ROW {
            <p align=center><font size=6>
            <strong>$(V_mestname)</strong><br>
            <br>
            </font><hr>
        }
    }

    <p>
    <h3>
    Sorry, your purchase was rejected.<br>
    Anyway, thank you for using IBM SET technology.
    </h3>
    <p>
    <hr><font size=5><br>
        </font>
        <font size=4>
        Your order has been rejected for the following reason:<br>
        $(Reason).<br> 3
    </p>
    %}
%}

```

Figure 91 (Part 1 of 3). itso_ordfailed Macro File

```

%SQL_MESSAGE {
  100: {
    <p><b>
    No Merchant Name could be found for this order.</b>
    %}

    default: {
      <font size=+3><b>Database Error:</b><p>
      A database error occurred when attempting to
      add your information to the database. Please
      contact the merchant server administrator.
      </font><BR>Returned an error code of ${SQL_CODE}
    %}
  %}
%}

%SQL(nextorder) {
select orrfnbr, ormenbr
from ORDER_PEND
where orshnbr=(select shrfnbr from shopper
                where shlogid='${SESSION_ID}')
AND orstat='P'
order by orrfnbr

%SQL_REPORT {
  %ROW {
    %if ("$(ROW_NUM)" == "1")
    @DTW_assign(nextorrf, V_orrfnbr)
    %endif
  %}

  <center>
  <table cellspacing=0 cellpadding=0>
  <tr>
  <td align=center valign=bottom>
  <form action="/cgi-bin/nph-msrvr;/order/display">
    <input type=hidden name="status" value="P">
    <input type=hidden name="order_rn" value="$(nextorrf)">
    <input type=submit value="Process Next Order">
  </form>
  </td>
  <td align=center valign=bottom>
  <form action="/cgi-bin/nph-msrvr;/order/list">
    <input type=hidden name="status" value="P">
    <input type=submit value="View All Pending Orders"></p>
  </form>
  </td>
  </tr>
  </table>
  </center>
  %}

%SQL_MESSAGE{
  -100: {
    %} : continue

  100: {
    %} : continue
}

```

Figure 91 (Part 2 of 3). itso_ordfailed Macro File


```
        default: {
            <font size=+3><b>Database Error:</b><p>
            A database error occurred when attempting to
            add your information to the database. Please
            contact the merchant server administrator.
            </font><BR>Returned an error code of $(SQL_CODE)
        }
    }
}

%HTML_REPORT {
<html>
<head>
  <META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1996 01:01:01 GMT">
  <title>Order KO</title>
</head>
<body background="/winstead/quebra_ca.gif" bgcolor="#FFFFFF">

%EXEC_SQL(InitSQL)
%include "$(MallHeader)"
%EXEC_SQL(reason) 4
%EXEC_SQL(merchant) 4
%EXEC_SQL(nextorder)
%include "$(MallFooter)"

</body>
</html>
}
```

Figure 91 (Part 3 of 3). *itso_ordfailed Macro File*

Notes:

- 1** The SETURESPREA field in the SETAUTH table contains the text accompanying the authorization response. The text should contain, in case of rejection, the explanation of what was wrong.
- 2** This SQL code is only for setting the Reason variable.
- 3** This is where the Reason variable is used.
- 4** Note that the SQL routine to evaluate the Reason variable is called prior to the SQL routine that displays the core of the message.

Chapter 8. The Certificate Server

As we described in the first part of this book, the SET protocol relies on certificates to validate all the parties in a transaction. The IBM product that provides the certification function is IBM Registry for SET.

IBM Registry for SET provides a certificate management infrastructure for cardholders, merchants, and acquirers to facilitate secure payments over the Internet using SET protocols. Through the use of IBM Registry for SET, acquiring and issuing banks will be able to provide the critical element of authentication to their merchants and cardholders. IBM Registry for SET allows the card issuer to issue certificates to their cardholders and acquirer organizations to issue certificates to their merchants to establish their identities within the SET structure.

8.1 IBM Registry for SET Components: Overview, Interfaces and Interactions

IBM Registry for SET runs on a server to create and administer certificates for use with the SET protocol. It can function as a Cardholder Certificate Authority (CCA), Merchant CA (MCA), or a Payment Gateway CA (PCA). IBM Registry for SET can operate as individual CAs (for example, a Cardholder CA and a Merchant CA) on separate machines or be configured to allow multiple CAs to run simultaneously on a single machine.

IBM Registry for SET supports online requests using protocols defined by the SET specification for HTTP transmission with approval of the request accomplished through a customer-tailored interface. This interface allows for the certificate request to be processed immediately or deferred for further review by the approver application. If approval is deferred, subsequent inquiry is required to determine the status of the certificate request.

There are three main components to IBM Registry for SET:

1. The Administrator
2. The Approver
3. The IBM Registry for SET Server

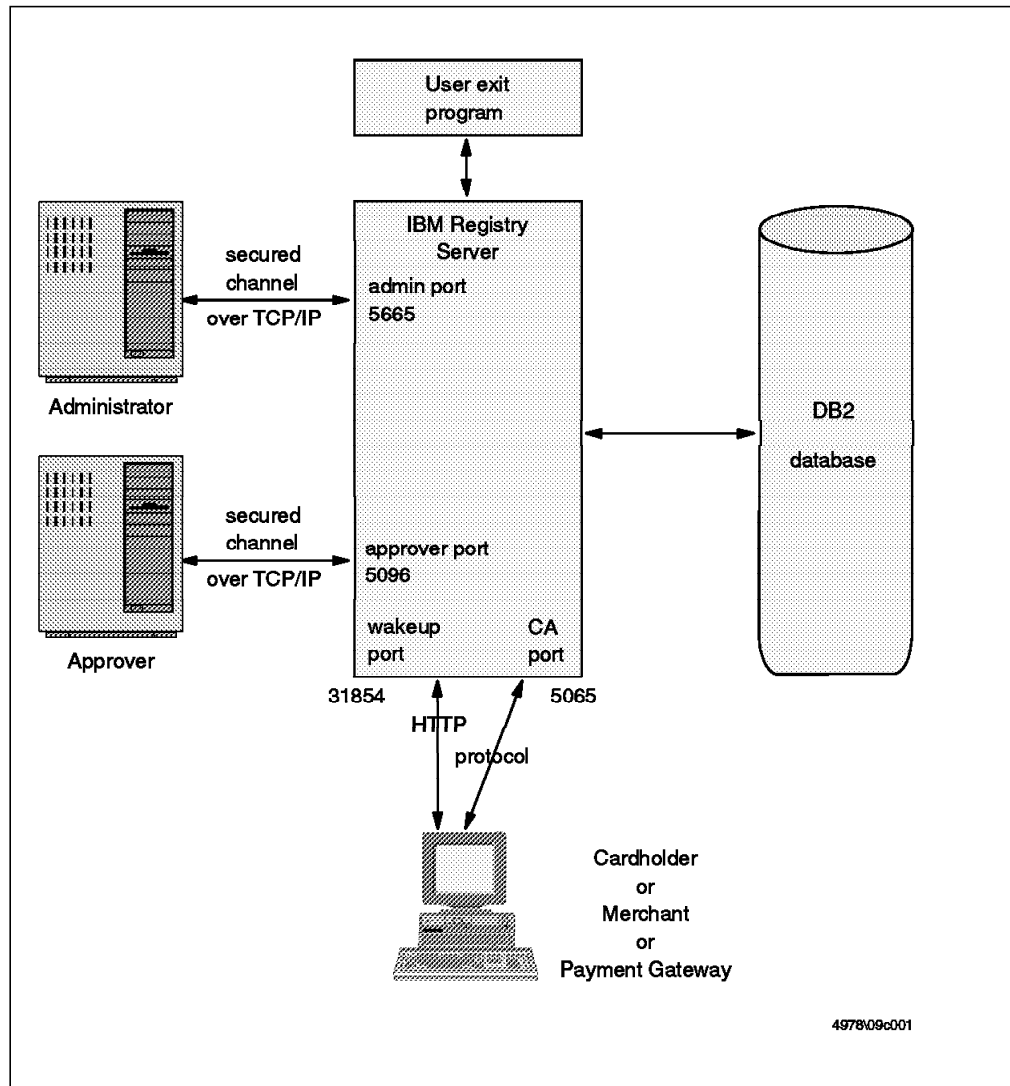


Figure 92. IBM Registry for SET Interfaces and Interactions

8.1.1 The Administrator

The administrator is the application and associated interface for operational management of the server and security management of password and keys.

The administrator application, *setadmin*, is a graphical user interface, developed in the Java language. *setadmin* communicates with the IBM Registry for SET server over TCP/IP on a secured channel. The server listens on a dedicated port; the port number is set up in the server configuration file.

The communication between the administrator application and the IBM Registry for SET server conforms to the traditional client/server scheme over TCP/IP. In other words, the administration application could be run on any AIX machine in the network. To run *setadmin* on a different AIX machine, you only need to copy the Java virtual machine (interpreter) code and the Java class files for the application.

8.1.2 The Approver

The approver is the application and associated interface for receiving and validating of SET certificate requests. Like the administrator application, the approver application, `setapprove`, is a graphical user interface, also developed in the Java language.

The IBM Registry for SET server listens on another dedicated port to wait for approver connections. As in the case of `setadmin`, you can run `setapprove` on a remote system.

8.1.3 The IBM Registry for SET Server

The server processes the SET certificate requests, signing, and issuing the request either online or offline.

The IBM Registry for SET server is implemented as a daemon, listening on different ports and waiting for requests from its partners: administrators, approvers and, of course, cardholders, merchants and acquirers. Cardholder, merchant and payment gateway clients connect to the wakeup port to initiate the process of receiving certificates. The HTTP protocol is used for this initiation, but when the clients submit their certificate requests they connect to a different port and use a SET-specific protocol.

The `setca` daemon uses a relational database (DB2 only in the current version) to store transaction data related to its cardholder, merchant, acquirer, approver, and administrator functions. Because this database contains confidential information, access is restricted.

The server is designed to allow immediate online SET certificate requests; this is possible thanks to authorization user exits. These user exits are invoked whenever an approval is required for a cardholder certificate. If the exit determines that it is possible to make an immediate decision about issuing the certificate, it processes it directly.

8.2 Cardholder Registration Scenario

Before installing and configuring the IBM Registry for SET package, let us examine the flow of a cardholder registration. This will help in understanding how to set up the parameters as described in 8.5.2, “Modifying the Configuration File” on page 149.

In 4.4.1, “Cardholder Certification Process” on page 56, the cardholder certification process is described as defined in the SET specification. This process starts with the initiate request sent by the cardholder to the certificate authority server. This request is actually sent by the CommercePOINT Wallet application, which needs to be initiated in some way.

The Figure 93 on page 142 shows you the full flow for the case where a WWW server is used to initiate the process.

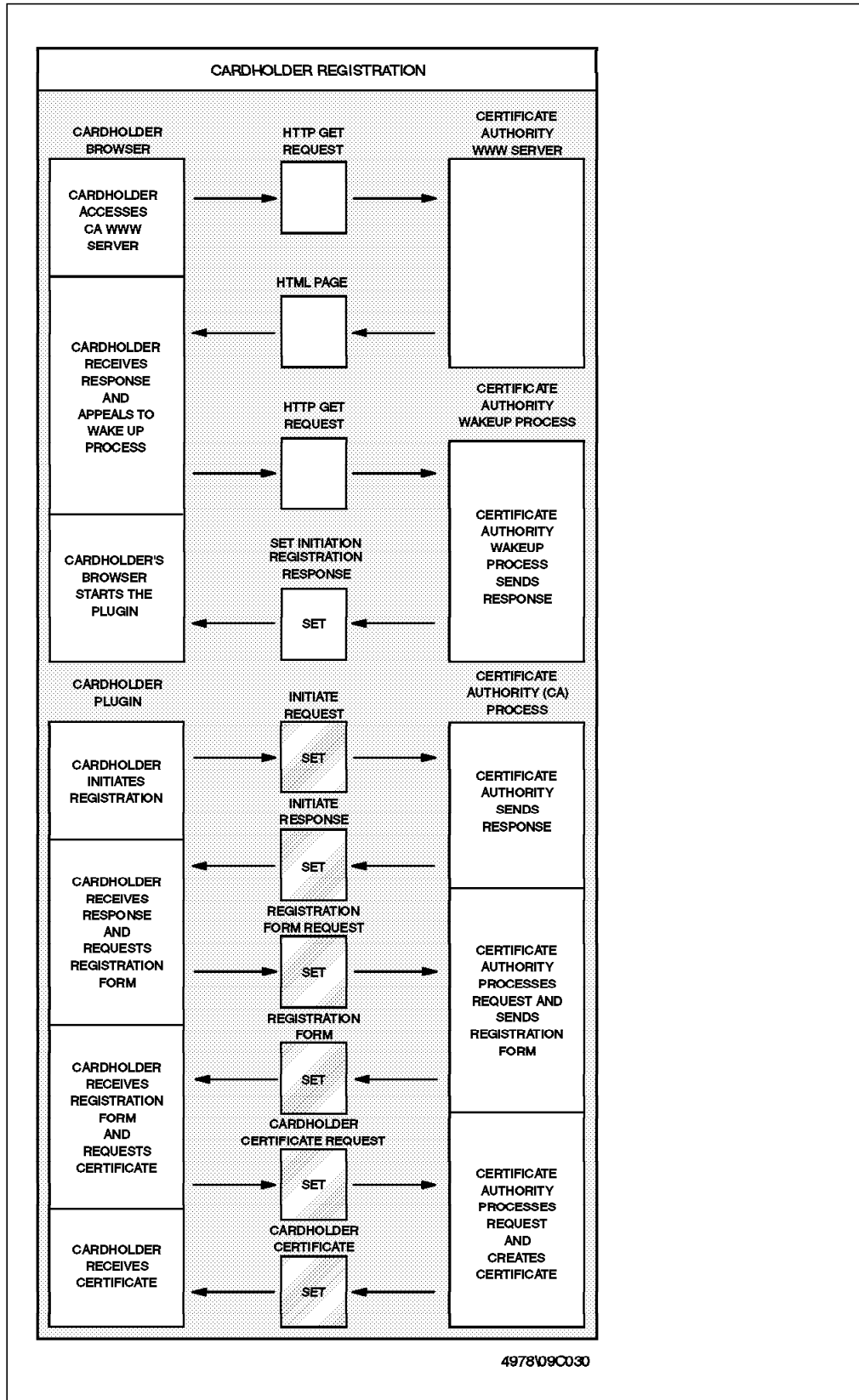


Figure 93. Cardholder Registration Flow

The different steps in this flow are:

1. The cardholder accesses, with his browser, a page on the CA WWW server. This means that a GET HTTP request is sent to the WWW server.
2. The WWW server, actually an HTTP daemon, responds with an HTML page including a reference to the wakeup server. The URL for this wakeup server includes the name of the server, the port on which the wakeup server is listening and an identifier that helps the wakeup server to distinguish the CCA (remember that the IBM Registry for SET server could be hosting several CCAs). When configuring, this identifier is also named the relative URL.
3. The cardholder, by clicking on the link on the HTML page, accesses the wakeup server.
4. The wakeup server parses the request (a plain GET HTTP request) to get the identifier. It builds a response message with a MIME type of set-registration-initiation. This initiation message includes, among other information, the URL for the CA server.
5. The browser receives the MIME message and starts the helper corresponding to the set-registration-initiation MIME type, that is the CommercePOINT Wallet application.
6. Now the handshake has completed and the browser plug-in module has a connection with the CA server. The SET protocol can therefore begin, so the flow goes on as we described in 4.4.1, "Cardholder Certification Process" on page 56.

8.3 Preparing the System

There are some prerequisites that you must satisfy to ensure that the IBM Registry for SET installation will be successful. The prerequisites are:

- RS/600 machine with 128 MB of RAM (or more)
- AIX Version 4.2
- IBM DATABASE 2 Version 2.1.0
- 25 MB of free disk space

8.3.1 Checking System Memory

The amount of RAM can be obtained with the following command:

```
bootinfo -r
```

It displays the amount of real memory in kilobytes.

8.3.2 Checking the AIX System Level

The version of AIX can be checked easily with the following command:

```
oslevel
```

The bos.iconv.ucs.com LPP (Licensed Program Product) should be installed. It comes with the AIX system, but is not necessarily installed on your system. To check the presence of the LPP:

```
ls|pp -l bos.iconv.ucs.com
```

8.3.3 Checking IBM DATABASE 2 Installation

The status of the IBM DATABASE 2 LPPs can be listed with the following command:

```
lslpp -l 'db2*'
```

The DB2 prerequisites are:

- db2_02_01.db2.rte
- db2_02_01.client
- db2_02_01.clp

8.3.4 Checking the Free Disk Space

The installation procedure will install files in several different directories:

- /usr/bin
- /usr/lpp/setca
- /usr/lib
- /home

These directories, depending on your environment, may belong to the same file system or different file systems. It is therefore difficult to give a specific rule for the prerequisites regarding disk space. Anyway, the installation procedure will check it for you and will extend the file systems if the corresponding option in the installation menu allows it.

8.4 Installing the IBM Registry for SET Product

To install the CA Server, follow these steps:

1. Type `su root` to log on as the root user. This type of user is allowed unrestricted ability to access and modify any part of the operating system.
2. On the command line, enter `smitty install_latest`. The SMIT Install and Update from LATEST Available Software menu appears, as shown in Figure 94 on page 145.


```

                                Install and Update from LATEST Available Software

Type or select a value for the entry field.
Press Enter AFTER making all desired changes.

* INPUT device / directory for software      [Entry Fields]
                                             []

F1=Help      F2=Refresh      F3=Cancel      F4=List
F5=Reset      F6=Command      F7=Edit      F8=Image
F9=Shell      F10=Exit      Enter=Do

```

Figure 94. Install and Update Software from LATEST Available (1 of 2)

3. In the Input device / directory for software field, type the name of the device or directory in which the installation image of the IBM Registry for SET product is available.
4. Then press Enter. The Install and Update from LATEST Available detailed menu is displayed, as shown in Figure 95 on page 146.

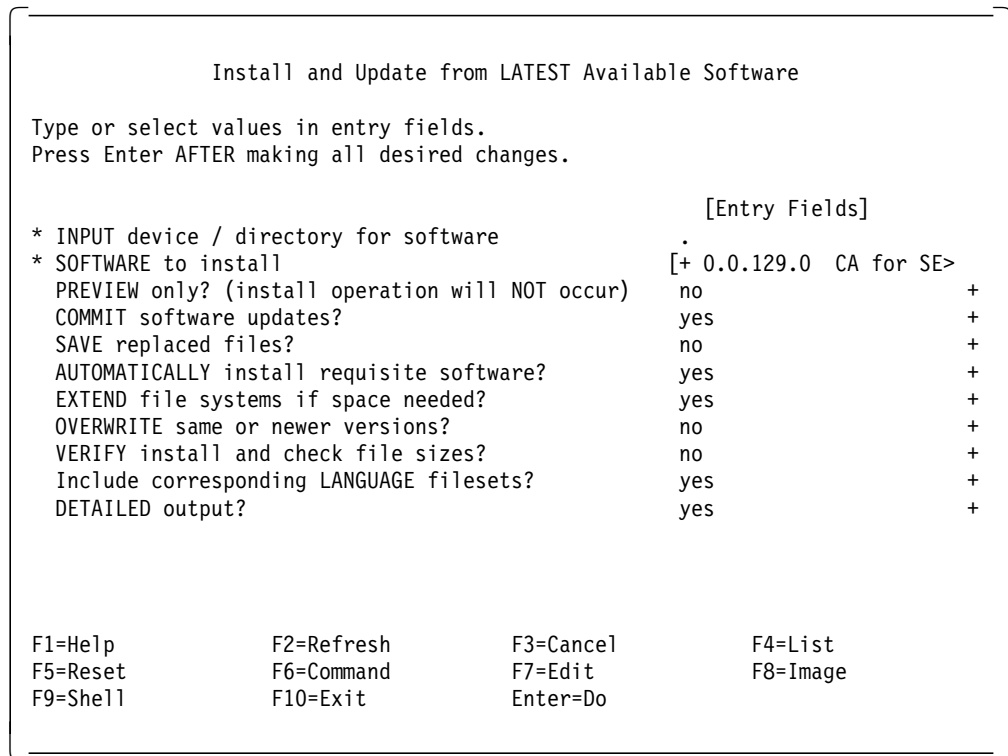


Figure 95. Install and Update from LATEST Available Software (2 of 2)

5. Press F4 in the the SOFTWARE to install field to see a list of products.
6. Select the IBM Registry for SET product in the list.
7. Change the DETAILED output field to Yes (press Tab to toggle from No to Yes) and press Enter.
8. A confirmation message appears. Press Enter.

The Command Status window appears indicating that the installation of the IBM Registry for SET product has started. The installation completes when the Command field at the top of the window changes from Running to OK.

Note: Depending on your machine configuration, this might take some time to install.

As suggested by the warning message at the end of the installation process, we encourage you to verify that your paging space is at least 192 MB. For information about the current size of the paging space, type:

```
lsps -a
```

You have now installed the IBM Registry for SET system.

8.4.1 Post Checking

Upon installation, a new user was created on your system and new files were installed. Now is the time to get familiar with this new environment!

The executable files, the Java interpreter and classes, the message catalogs and even more files are installed in your AIX system. To view a list of installed files and their directories, run the following command:

```
ls|pp -f setca
```

where setca is the name of the IBM Registry for SET package. Table 5 on page 147 summarizes the location of the different files according to their type.

File Type	File Location
IBM Registry for SET Java Classes	/usr/lpp/setca/java/setca
Generic Java Classes	/usr/lpp/setca/java/classes
Java Executable and Shared Objects	/usr/lpp/setca/java/bin
IBM Registry for SET Executables	/usr/bin
IBM Registry for SET Configuration File and User Exit Samples	/usr/lpp/setca/samples
IBM Registry for SET Include File	/usr/include
IBM Registry for SET Message Catalogs	/usr/lib/nls/msg/en_US
IBM Registry for SET Database Creation Scripts	/usr/lpp/setca/scripts
CMS Library	/usr/lib
IBM Registry for SET DB2 Database	/home/nrsetca

A user group called nrsetca is created on the AIX system and a new user ID, nrsetca, is created belonging to this new group. The nrsetca user is the user that will be used to run the server. It is also the instance owner of the IBM Registry for SET database in DB2.

The tables in the brand new DB2 database record all the certificate requests, the approver and administrator identities, and the approver and administrator actions. The tables are:

- REGFORM
- CCATRANSLOG
- CCATRANSHISTORY
- CCACERTIFICATES
- MPCATRANSLOG
- MPCATRANSHISTORY
- MPCACERTIFICATES
- ADMINID
- ADMINCMD
- CERTSERIAL

For security purposes, access to the database is restricted. It can only be accessed by the DB2 administrator and root user IDs.

A minimal configuration file, /etc/setca.conf, is created. It will help you to customize IBM Registry for SET to suit your specific needs.

8.5 Configuring the CA Server

The IBM Registry for SET product is now installed on your AIX system. Before starting the server, you need to configure it, by performing the following steps:

- Plan your customization
- Modify your configuration file
- Get and install your CA certificate

If you intend to allow online certification, you will also need to develop and integrate your own user exit, but this customization is not required before you start the server for the first time.

8.5.1 Planning for Configuration

Before configuring your certificate server, you should determine the number and the types of CAs you will be supporting. The server can operate as a cardholder CA (CCA), a merchant CA (MCA), a payment gateway CA (PCA), or any combination of them.

In our test environment, we invented a credit card brand called *Pass* for which the IBM Registry for SET server generates certificates. It acts as a CCA, MCA and PCA. For now, we will configure the server to issue the certificates offline. We will see in 8.10.3, "Implementing a User Exit" on page 166 how to modify the configuration to issue the cardholder certificates online.

In order to store the files related to the IBM Registry for SET server, we suggest you create a dedicated directory:

```
mkdir -p /usr/local/setca
```

This directory will host, for example, the file containing the policy text, the executables of the user exits, and other related files.

Prior to configuring the IBM Registry for SET server, you need also to identify the location (URL) of:

- The brand logo image file
- The card (issuer) logo image file
- The certificate issuance HTML page
- The certificate rejection HTML page
- The end-user cancellation HTML page

These image files and HTML pages must be accessible at all times via the HTTP protocol on a Web server.

In our test environment, we chose to host the Web server on the same system as the IBM Registry for SET server, but in a real-life environment you would use a separate system, ideally in a separate network so as to minimize exposure of the CA server to attack. The software used to build our WWW server is IBM Internet Connection Secure Server. The HTML pages and image files are stored in the `/usr/lpp/setca/www` directory and we added the following statement in the `/etc/httpd.conf` configuration file to direct requests to it:

```
Pass /CA/* /usr/local/setca/www/*
```

The brand logo and the card logo are the same. The URL is: `http://rs600019.itso.ral.ibm.com/CA/pass.gif` and the image file is `/usr/local/setca/www/pass.gif`. We created three simple HTML pages to indicate to the requester (cardholder, merchant or payment gateway) the certificate issuance, certificate rejection or cancellation. The HTML pages are stored respectively as `/usr/local/setca/www/success.html`, `/usr/local/setca/www/failure.html`, and `/usr/local/setca/www/cancel.html`.

8.5.2 Modifying the Configuration File

The configuration file for the IBM Registry for SET server is `/etc/setca.conf`. As we stated in 8.4.1, “Post Checking” on page 146, a minimal configuration file was created during the installation process. This file needs to be modified before starting your server for the first time. A sample configuration file was also installed during the installation step that is included in the `/usr/lpp/setca/samples` directory and can be used as a basis for your own configuration file.

The format of the configuration file is trivial:

keyword value

where value is built up of one or more words.

All lines starting with a pound sign (#) are considered as comments.

The configuration file contains two types of information:

1. Global values
2. Definition of the server instance(s)

8.5.2.1 IBM Registry for SET Global Values

In most cases, you will keep intact this section of the original configuration file. On our test system, as shown in Figure 96, the only value that we added to the original file is `KEYDB.PATH`.

```
#
# SET/CA configuration file
# Pass brand CA for Cardholder, Merchant and Payment Gateway
#
#
# SETCA Server global values
#
caport 5065
adminport 5665
approverport 5096
wakeupport 31854
groupid nrsetca
userid nrsetca
DBBindPath /home/nrsetca/
DB2INSTANCE nrsetca
#
KEYDB.PATH /usr/local/setca/ 1
```

Figure 96. Global Values in IBM Registry for SET Configuration File

Note:

- 1** We added this statement because we chose to store the key file in the `/usr/local/setca` directory as are all files specific to the local implementation of the IBM Registry for SET product.

8.5.2.2 Cardholder Certificate Authority Server Definition

In our test environment, the IBM Registry for SET server generates cardholder certificates for the Pass brand.

The statements in the configuration file to set up this CCA are shown in Figure 97.

```
##
## CCA Server - Pass brand
##
CANAME cca-server
CATYPE CCA
URL      /get-certificateCCA 1
rqst.timeout 60 2
QUERY.URL http://rs600019.itso.ral.ibm.com:5065/get-certificateCCA 3
SUCCESS.URL http://rs600019.itso.ral.ibm.com:80/CA/success.html
FAILURE.URL http://rs600019.itso.ral.ibm.com:80/CA/failure.html
CANCEL.URL http://rs600019.itso.ral.ibm.com:80/CA/cancel.html
BRAND.ID Pass 4
BRAND.LOGO.URL http://rs600019.itso.ral.ibm.com:80/CA/pass.gif
CARD.LOGO.URL http://rs600019.itso.ral.ibm.com:80/CA/pass.gif
POLICY.TEXT /usr/local/setca/policy.txt 5
REASON.TEXT This is a reason for service denial 6
FORM Title
FORM Given Name
FORM Surname
FORM Sex
FORM Address
FORM City 7
FORM State
FORM Country
FORM Postal Code
REFERRAL mailto:customer-service@ca.itso.ral.ibm.com 8
#ACFILE /usr/local/setca/bin/authchk 9

#
#THREADS.MIN 1 10
#THREADS.MAX 256 11
#DNSLOOKUP NO 12
```

Figure 97. Cardholder CA Definition in Configuration File

Note:

1 /get-certificateCCA is the relative URL to identify this CA server. The IBM Registry for SET server can operate as a CA for different entities. This relative URL allows the server to identify the CA being used.

2 The request timeout is set to 1 minute (60 seconds). This means that the server will process a request for 1 minute and kill the connection if it is not complete in this period.

3 http://rs600019.itso.ral.ibm.com:5065/get-certificateCCA is the absolute URL for this CA server. This is actually a combination of the fully qualified domain name of the host system, the CA port set up in the global values and the relative URL defined previously (see **1**).

4 This is the brand ID as it is displayed to the end user. In our test environment, the brand is Pass.

5 This is the absolute path to a file containing the policy text. The length of the policy text should not exceed 4 KB. In our test environment, the policy text file is /usr/local/setca/policy.txt.

6 This text is the brand- specific reason to be displayed to the end entity, in this case the cardholder, if the registration form is not available.

7 Title, Given Name, Surname, Sex, Address, City are the fields that compose the registration form. The information collected in the registration form will be used by the approver to determine if a certificate should be delivered to the requester or not.

8 This e-mail address is a referral location, that is the location (URL or e-mail address) where the customer can find more information when required.

9 In this first step, the certificates are issued offline. This statement is therefore commented out.

10 The minimum number of connections created when the server is started is 1 by default. But you can specify any value you want.

11 The maximum number of server connections allowed at one time can be tuned. The default value is 256 and the upper limit is 1024.

12 This statement specifies the DNSLookup status. The default is off, meaning that the reverse name lookup for the hostname of the client's IP address will not be done.

8.5.2.3 Merchant and Payment Gateway Certificate Authority Server Definitions

In our test environment, the IBM Registry for SET server operates also as a merchant and payment gateway CA.

The statements to configure these two CAs are shown in Figure 98 on page 152. They are pretty similar to the statements used to declare the cardholder CA.

```

##
## MCA Server - Pass brand
##
CANAME mca-server
URL      /get-certificateMCA 1
ACFILE  /usr/local/setca/bin/authchk
CATYPE  MCA
RQST.TIMEOUT  60
QUERY.URL  http://rs600019.itso.ral.ibm.com:5065/get-certificateMCA 1
SUCCESS.URL http://rs600019.itso.ral.ibm.com:80/CA/success.html
FAILURE.URL http://rs600019.itso.ral.ibm.com:80/CA/failure.html
CANCEL.URL http://rs600019.itso.ral.ibm.com:80/CA/cancel.html
POLICY.TEXT /usr/local/setca/policy.txt
BRAND.LOGO.URL http://rs600019.itso.ral.ibm.com:80/CA/pass.gif
BRAND.ID Pass
CARD.LOGO.URL http://rs600019.itso.ral.ibm.com:80/CA/pass.gif
REASON.TEXT This is a reason
FORM Merchant Locality
FORM Gross Revenue
FORM Average Employee Age
FORM Average Tax Paid
REFERRAL mailto:customer_service@ca.itso.ral.ibm.com

##
## PCA Server - Pass brand
##
CANAME pca-server
URL      /get-certificatePCA 2
#ACFILE  /usr/local/setca/bin/authchk
CATYPE  PCA
#RQST.TIMEOUT  360
#THREADS.MIN  1
#THREADS.MAX  256
#DNSLOOKUP  NO
QUERY.URL  http://rs600019.itso.ral.ibm.com:5065/get-certificatePCA 2
SUCCESS.URL http://rs600019.itso.ral.ibm.com:80/CA/success.html
FAILURE.URL http://rs600019.itso.ral.ibm.com:80/CA/failure.html
CANCEL.URL http://rs600019.itso.ral.ibm.com:80/CA/cancel.html
BRAND.LOGO.URL http://rs600019.itso.ral.ibm.com/pass.gif
BRAND.ID Pass
CARD.LOGO.URL http://rs600019.itso.ral.ibm.com/pass.gif
POLICY.TEXT /usr/local/setca/policy.txt
REASON.TEXT This is a reason
FORM Gateway Name
FORM Bank
FORM Financial Institute
FORM Number of Transactions
REFERRAL mailto:customer_service@ca.itso.ral.ibm.com

```

Figure 98. Merchant and Payment Gateway Definitions in setca.conf

Note:

1 The Merchant CA has its own URL that distinguishes it from the other CAs hosted on the same IBM Registry for SET system.

2 The Payment Gateway CA has just another URL.

8.6 Installing the CA Certificate

The IBM Registry for SET server has its own certificate so that the cardholder, the merchant or payment gateway can make sure that the CA server is authenticated. The IBM Registry for SET certificate is issued by the brand (association) CA. In fact, IBM Registry for SET is not limited to acting as the lowest layer of the CA hierarchy; it can be used to generate Root and Brand keys as well as acting as a CA for cardholders, merchants and acquirers. However, the standard facility it provides for key storage uses a password-protected database on a disk file. This may not be adequate for the higher security demanded for Root and Brand keys.

In IBM Registry for SET, the CA key database is a key file located in the directory defined by the KEYDB.PATH directive in the configuration file.

8.6.1 Key and Certificate Hierarchy Required for SET

At this point, before describing how we set up the key hierarchy for our test environment, it is worth reviewing exactly what we are trying to achieve. Each entity in the SET protocols needs one or more public key pairs. These key pairs are used for a number of purposes:

- For digitally signing messages
- For encrypting other keys (that is, for sending symmetric keys securely)
- For signing public key certificates

For a key pair to be usable it has to have a certificate, attesting to its authenticity. The certificate has to be signed by a CA and the chain of CA signatures has to be traceable all the way up to the root CA.

Figure 99 on page 154 shows the minimum set of key databases, key pairs, and certificates for *one* cardholder to make a purchase from *one* merchant using *one* brand of credit card.

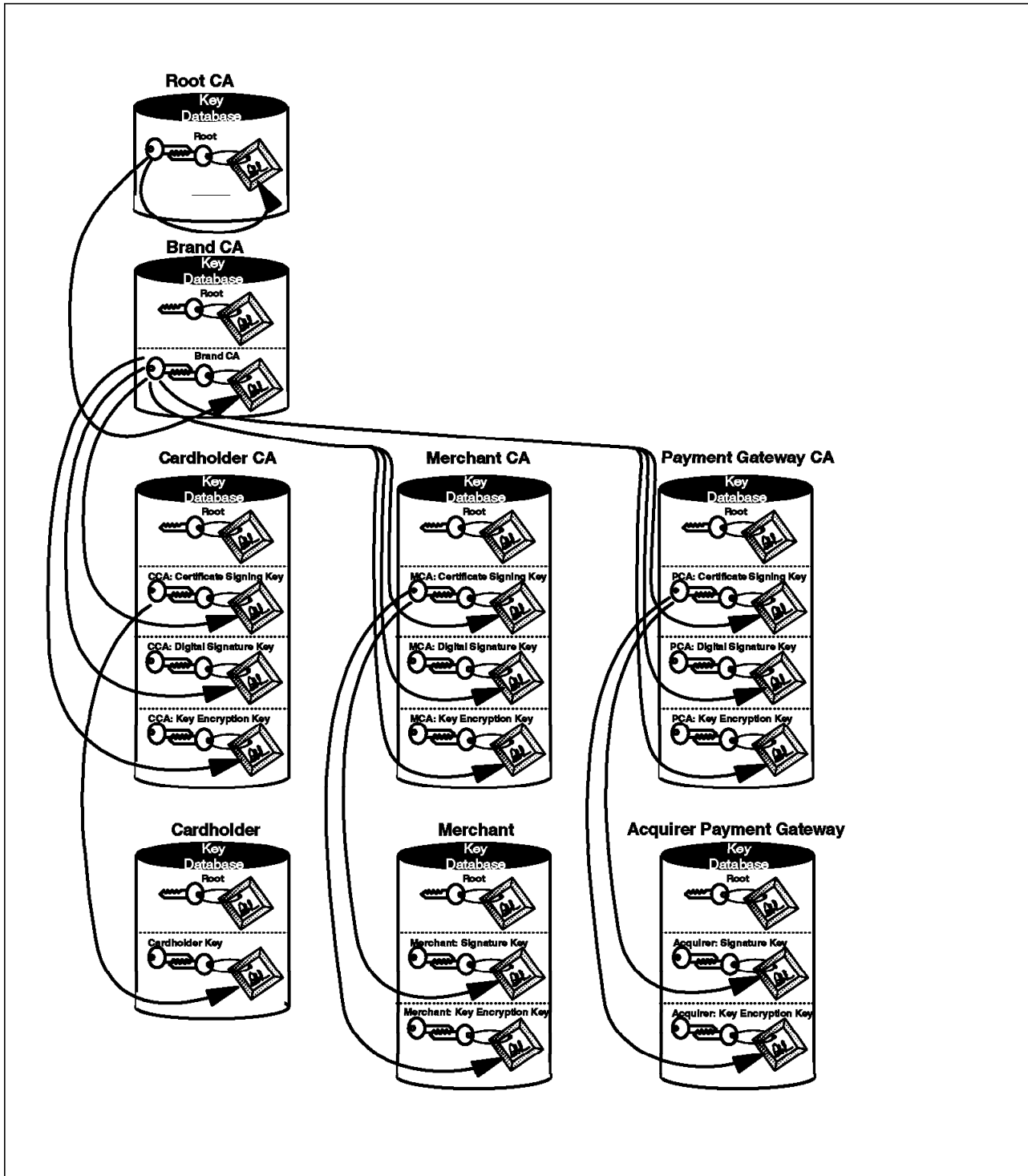


Figure 99. Hierarchy of Key Databases, Keys, and Certificates for SET

The figure shows a number of features:

- Each entity has a key database, containing some key pairs. The key pair is always generated locally and the private key never moves from the database after generation.
- The public key of each key pair has a certificate attached, which has been signed by the CA immediately above it in the hierarchy (the signatures are indicated by arrows in the diagram). The exception to this is the Root public

key which has a *self-signed* certificate, that is, one that has been signed by the private key from the Root key pair.

- All of the key databases except the Root have a copy of the Root public key and certificate installed. The entity needs this whenever it is presented with a certificate for validation. Each certificate contains a chain of signatures that can be traced back to the Root, so the Root certificate is needed to check that the chain is correct and unbroken. The need for checking starts when the entity first requests a certificate (it has to know that the CA signing its own certificate can trace a line of trust back to the Root). Therefore, each key database has to have the Root certificate pre-installed before it can start generating keys and requesting certificates.

8.6.2 Setting Up a CA Hierarchy for Test Purposes

Normally you will be installing a CA into an existing hierarchy, so the details of setting up the key database will depend on the brand rules and procedures. However, for our project we needed to set up the whole hierarchy, including the Root keys.

IBM Registry for SET provides the `setca_certmgr` command for managing key databases (functions such as generating key pairs and signing certificates). Figure 100 shows the initial prompt and the Root CA option menu.

```
# setca_certmgr
Enter CA type:
1. Root CA, 2. Brand CA, 3. End Entity CA. 1

    M A I N    M E N U

0. Exit
1. Generate key pairs and certificate requests
2. Replace certificate requests with certificates
3. Generate self-signed certificates (RCA only)
4. Generate certificates for a lower level CA (RCA and BCA only)
5. Generate CRLs (RCA and BCA only)
6. Generate BCIs (BCA only)
```

Figure 100. Initial Prompt and Menu for `setca_certmgr`

A.1, “Using `setca_certmgr` to Create a Certificate Hierarchy” on page 283 describes the sequence of actions to create a complete hierarchy of CA key databases using `setca_certmgr`.

Creating the hierarchy manually in this way is time consuming, so as an alternative, IBM Registry for SET provides a command, `setbrandca`, that generates all of the key databases in one operation. This is what we used to create the environment for our project.

8.6.2.1 Using setbrandca to Create a Test CA Hierarchy

setbrandca creates four key databases, as shown in Table 6.

Database file	Description
rootcakey.db	This is the Root CA key database.
rootcakey0.db	A key database containing the Root certificate but without the private key. This is used to seed the database for lower-level entities, as described in 8.6.1, "Key and Certificate Hierarchy Required for SET" on page 153. You have to copy this file to the cardholder, merchant and payment gateway before generating keys and certificates.
brandcakey.db	This is the Brand CA key database.
eecakey.db	This is the <i>end entity</i> key database. The setbrandca program creates a single CA for cardholders, merchants and payment gateways. This database contains all three sets of keys and certificates (CCA, MCA and PCA).

To create all of these key databases:

1. Create a directory:

```
mkdir -p /usr/local/setca
```

 and change directory

```
cd /usr/local/setca
```

 (the files are created in the current directory).
2. Enter setbrandca (the executable file is /usr/bin/setbrandca)
3. Follow the prompts. We show a sample dialog with setbrandca in A.2, "Using setbrandca to Create a Certificate Hierarchy" on page 289.

Note

At the time we were working with the setbrandca alpha code, two things were hard coded:

1. The key database passwords must be abc123
2. The cmpca certificate sign label has to be cmpca_certsign

4. Copy the end entity key database to the directory defined by the KEYDB.PATH directive in the configuration file. In our case we defined this directory to be /usr/local/setca, so we simply had to rename the database file to key.db (command: mv eecakey.db key.db).

Once you have run setbrandca, you can check the contents of the key databases using the cmsutil command. A.3, "Displaying the Contents of a CA Key Database" on page 291 shows an example of the output of cmsutil for the end entity (combined CA server) key database.

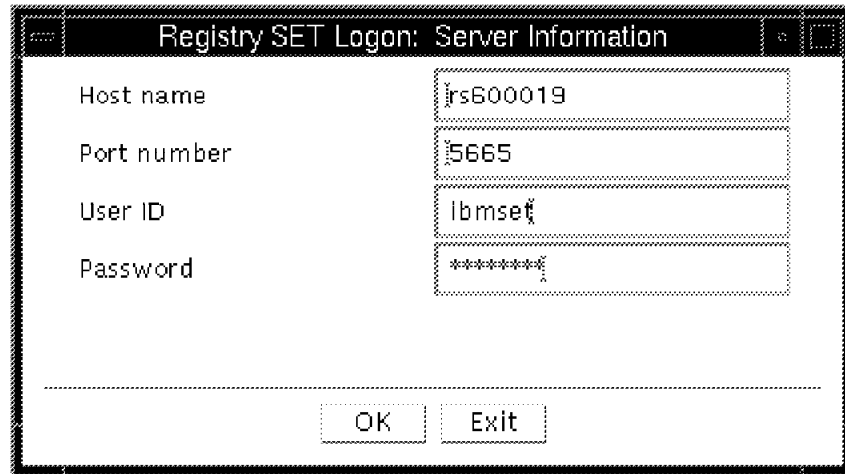
8.7 Starting the CA Server

Your IBM Registry for SET server is now ready to run. It's time to start it! The IBM Registry for SET server should be started with the nrsetca ID or whichever ID as long as the user ID belongs to the nrsetca group. Actually, after the installation process, the only members of the nrsetca group are root and nrsetca. To start the server:

1. Log on as nrsetca
2. Run setca

At start up time, the server reads the configuration file and gets ready to be activated. The server will not accept certificate requests or approval actions until it is activated.

The IBM Registry for SET server is started by the administrators using the setadmin interface. To start the administration tool, run the setadmin command. The first panel, as shown in Figure 101 will prompt for an administrator ID and password.



The image shows a terminal window titled "Registry SET Logon: Server Information". It contains four input fields: "Host name" with the value "rs600019", "Port number" with the value "5665", "User ID" with the value "ibmset", and "Password" with the value "*****". At the bottom of the window are two buttons labeled "OK" and "Exit".

Figure 101. setadmin Logon Panel

In this panel, you will fill in the following information:

1. The hostname of the system on which the IBM Registry for SET server is running; this is usually the local system, but the administration tool could be started from any AIX system in the network as explained in 8.1.1, "The Administrator" on page 140.
2. The port number on which the IBM Registry for SET server is listening; 5665 if you kept the default value.
3. The administrator ID.
4. The password.

The first time the server is started after installation, the only valid account is the *ibmset* ID, with password *ibmadmin*. As you will see later, the administrator account can be edited. We encourage you to change the password for the *ibmset* ID or simply to delete this account after creating your own accounts.

After login, the main menu is displayed as shown in Figure 102 on page 158.

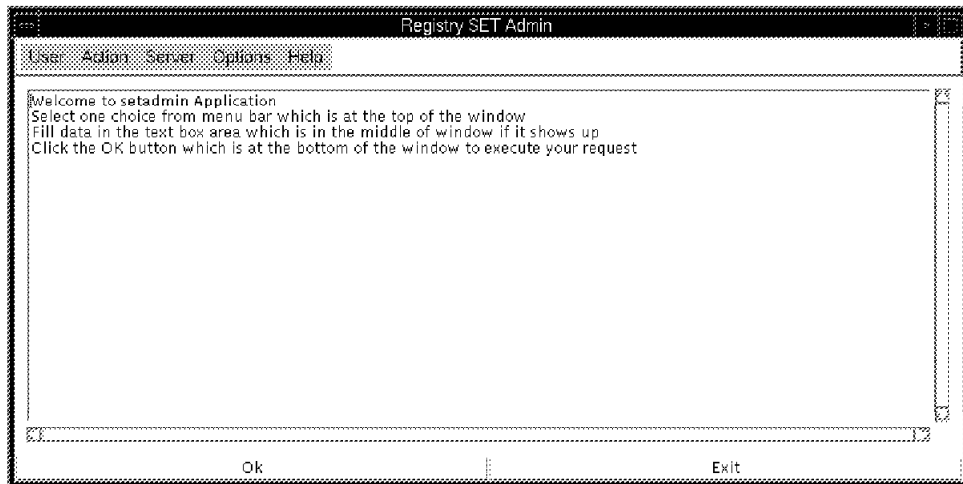


Figure 102. setadmin Main Menu

You can now activate your server:

1. Select the **Activate Admin** entry in the Action pop-up menu.
2. Enter your key file password (CMS password) when prompted as shown in Figure 103.
3. Click on **OK**.

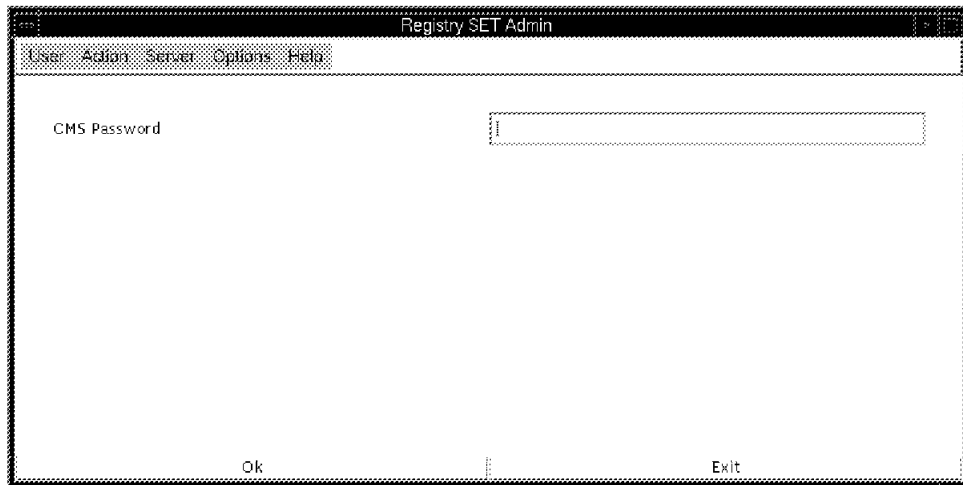


Figure 103. Administration Activation - Password Panel

The administration interface will be activated and the result of the command displayed in the main window. You can now:

- Create or modify administrator and approver accounts.
- Activate the CA server itself to allow it to receive certificate requests for the end entity.
- Activate the approver function to allow approvers to process the certificate requests.

To activate the CA server:

1. Select the **Activate Server** entry in the Action pop-up menu.
2. Click on **OK**.

To activate the approver function:

1. Select the **Activate Approver** entry in the Action pop-up menu.
2. Click on **OK**.

Your IBM Registry for SET server is now ready to receive certificate requests and the approvers are allowed to use the approver application to process the requests.

8.8 Creating Administrator and Approver Accounts

Before creating new administrator or approver accounts, we encourage you to change the password for the setibm administrator account, as follows:

1. Select the **Change User Password** entry in the User menu.
2. Click on **OK**.
3. A new menu is displayed as shown in Figure 104.

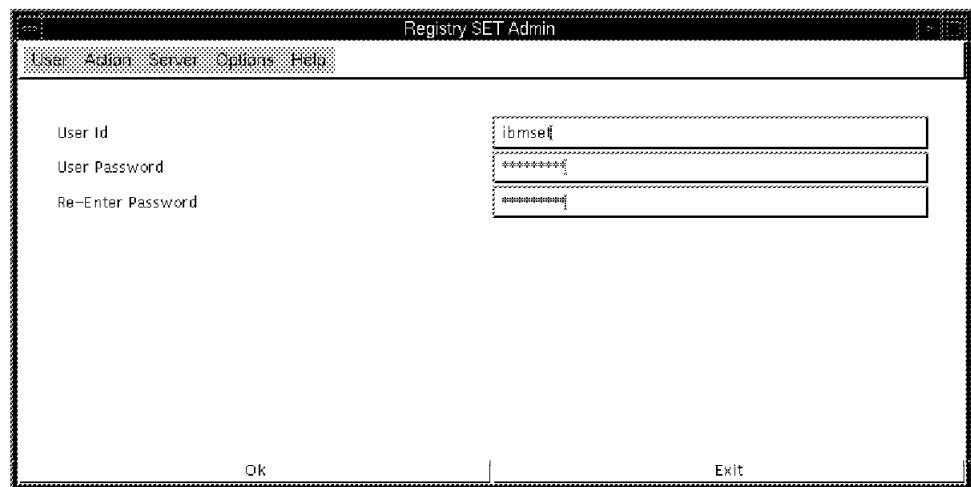


Figure 104. Changing ibmset User Password

- a. Type `ibmset` in the User ID field.
- b. Type the new password.
- c. And re-type the new password for confirmation.
- d. Click on **Ok**.

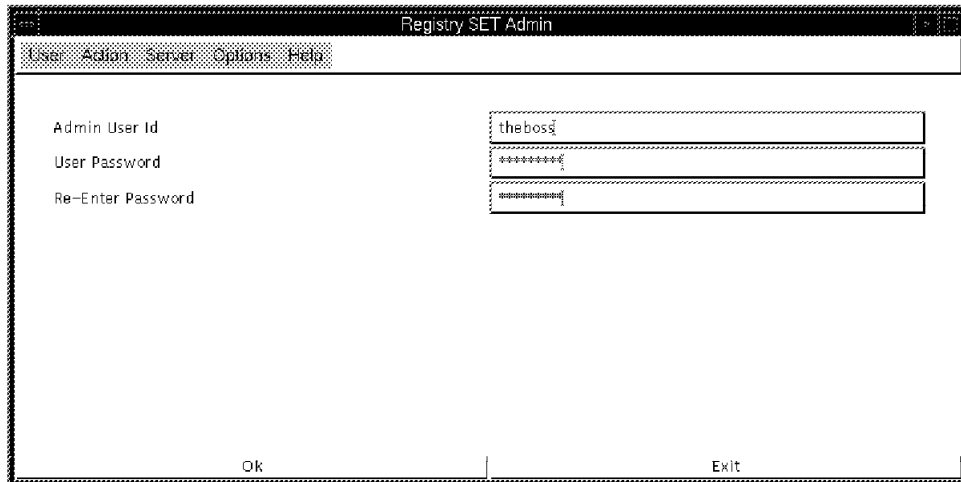
A success message will be displayed.

An alternative for this security issue would be to create as many administrator accounts as you need, as explained in 8.8.1, "Creating a New Administrator Account" on page 160, and to remove this initial account when you are sure of the usability of the new accounts.

8.8.1 Creating a New Administrator Account

As on any system, it is always a good idea for each administrator to have a unique ID. Let us now create an additional administrator account:

1. Select the **Add Admin User and Password** entry in the User menu.
2. Click on **OK**.
3. A new menu is displayed as shown in Figure 105.



The screenshot shows a dialog box titled "Registry SET Admin" with a menu bar containing "User", "Admin", "Server", "Options", and "Help". The dialog contains three input fields: "Admin User Id" with the text "the boss", "User Password" with masked characters "*****", and "Re-Enter Password" with masked characters "*****". At the bottom, there are "OK" and "Exit" buttons.

Figure 105. Creating an Additional Administrator Account

- a. Type in the administrator ID in the user ID field.
- b. Type in the password.
- c. Re-type the password for control.
- d. Click on **OK**.

A new administrator ID has been added that can:

- Activate the administration server.
- Activate/deactivate the approver server.
- Activate/deactivate the CA server.
- Create/modify/delete administrator and approver accounts.

The administrator details are kept in the ADMINID database table.

8.8.2 Creating an Approver Account

An administrator for IBM Registry for SET is responsible for keeping the system running and maintaining it. This is very different from an *approver*, who is responsible for deciding which public key certificates to sign.

To create a new approver account:

1. Select the **Add Approver User and Password** entry in the User menu.
2. Click on **OK**.
3. A new menu is displayed as shown in Figure 106 on page 161.

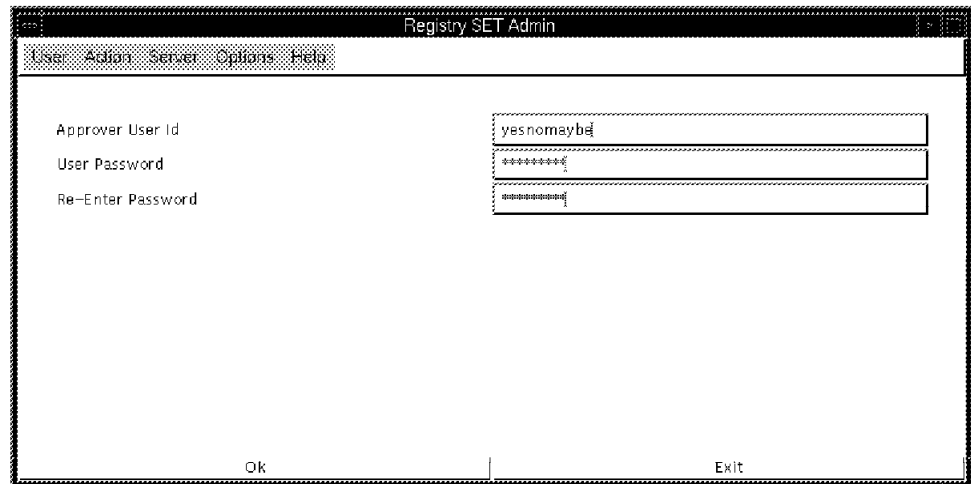


Figure 106. Creating an approver account

- a. Type in the approver ID.
- b. Type in the password.
- c. Re-type the password for control.
- d. Click on **OK**.

A new approver ID has been added that will be allowed to process the certificate requests.

8.9 Creating a CCA WWW server

In 8.5.1, "Planning for Configuration" on page 148, it was established that hosting a Web server on your CA system, or any other system, would be useful to serve brand image files and HTML pages. There is another function that this Web server can perform if you choose to install it: it can serve one or several HTML pages that will help the cardholders initiate their certificate requests.

8.9.1 Creating an HTML Page that References the Wakeup Server

Actually, the cardholder cannot initiate the request for a public key certificate on its own. The CardClnitReq request, which is the first request sent by the cardholder to the CA, contains data that the cardholder cannot guess, such as the brand ID. The cardholder needs help in formatting this data before it can pass it to the CA server in a POST HTTP request. In fact, a helper is built into the IBM Registry for SET server. This is the role of the server when it receives a request on the wakeup port.

The HTTP request that must be sent to the server on the wakeup port is a plain GET request:

```
GET /relative-URL
```

where relative-URL is the URL specified in the URL statement for this CA server in the configuration file.

The IBM Registry for SET server will respond with a registration initiation message. The registration initiation message for a cardholder CA has the contents shown in Figure 107 on page 162.

```
MIME-VERSION: 1.0
CONTENT-TYPE: text/plain
CONTENT-LENGTH: 0
X-SET-WAKEUP-TYPE: CardCInitReq
X-SET-SET-URL: http://rs600019.itso.ral.ibm.com:5065/get-certificateCCA
X-SET-QUERY-URL: http://rs600019.itso.ral.ibm.com:5065/get-certificateCCA
X-SET-SUCCESS-URL: http://rs600019.itso.ral.ibm.com:80/CA/success.html
X-SET-FAILURE-URL: http://rs600019.itso.ral.ibm.com:80/CA/failure.html
X-SET-CANCEL-URL: http://rs600019.itso.ral.ibm.com:80/CA/cancel.html
X-SET-BRAND: Pass <http://rs600019.itso.ral.ibm.com:80/CA/pass.gif>
```

Figure 107. Registration Initiation Message Example

The MIME-encapsulated form of the above message, as sent by the CA server, would be as shown in Figure 108.

```
HTTP/1.0 200 Request Complete
DATE: Fri Mar 7 09:24:24 1997
SERVER: SETCA/0.1
CONNECTION: close
MIME-VERSION: 1.0
CONTENT-TYPE: application/set-registration-initiation
CONTENT-LENGTH: 530
CONTENT-TRANSFER-ENCODING: binary

MIME-VERSION: 1.0
CONTENT-TYPE: text/plain
CONTENT-LENGTH: 0
X-SET-WAKEUP-TYPE: CardCInitReq
X-SET-SET-URL: http://rs600019.itso.ral.ibm.com:5065/get-certificateCCA
X-SET-QUERY-URL: http://rs600019.itso.ral.ibm.com:5065/get-certificateCCA
X-SET-SUCCESS-URL: http://rs600019.itso.ral.ibm.com:80/CA/success.html
X-SET-FAILURE-URL: http://rs600019.itso.ral.ibm.com:80/CA/failure.html
X-SET-CANCEL-URL: http://rs600019.itso.ral.ibm.com:80/CA/cancel.html
X-SET-BRAND: Pass <http://rs600019.itso.ral.ibm.com:80/CA/pass.gif>
```

Figure 108. MIME-Encapsulated Registration Initiation Message Example

When the cardholder's Web browser receives this MIME message, it will start the helper associated with the application/set-registration-initiation MIME type. If the cardholder's browser is correctly set up, the browser should start the CommercePOINT Wallet application that will work on the MIME message.

The home page developed for the test server is indeed minimal but it gives an example of an HTML page referring to the wakeup port of a CCA server.



Figure 109. CCA Home Page

The source code of this minimal HTML is listed in Figure 110.

```
<HTML>
<HEAD>
<TITLE>
ITSO Root Certificate Authority
</TITLE>
</HEAD>
<BODY>
<H1>Welcome to ITSO Certificate Authority Server</H1>
<H2>Pass Brand - Certificates</H2>
<A HREF="pass-policy.html">Get information on our policy
<P>
<A HREF="http://rs600019:31854/get-certificateCCA">
Request your CardHolder Certificate</A>
</BODY>
</HTML>
```

Figure 110. CCA Home Page Source Code

8.9.2 Replacing the Wakeup Process with a CGI Script

In fact, the response sent by the IBM Registry for SET server when you access the wakeup port could be sent by a CGI script. This alternative could be used in the following cases:

- You want to limit the IP ports accessible on your IBM Registry for SET server.
- You want to wake up the CommercePOINT Wallet application but you don't have any CA server available.

The response message sent by the CGI script must be exactly as shown in Figure 107 on page 162 and it must be encapsulated as a MIME message with the set-registration-initiation type.

An example of such a CGI script is listed in Figure 111 on page 164. This script is a Korn shell script and therefore is designed to run on a UNIX system.

```
#!/bin/ksh
#
# This CGI script sends to the requester the same information that the
# wakeup server would send if it was solicited, that is a MIME message
# with the application/set-registration-initiation type.

cat << !! 1
MIME-Version: 1.0 2
Content-Type: application/set-registration-initiation

MIME-Version: 1.0 3
X-SET-WakeUp-Type: CardCInitReq
Content-Type: text/plain
Content-Length: 0
X-SET-SET-URL: http://rs600019.itso.ral.ibm.com:5065/get-certificateCCA
X-SET-Query-URL: http://rs600019.itso.ral.ibm.com:5065/get-certificateCCA
X-SET-Success-URL: http://rs600019.itso.ral.ibm.com/CA/success.html
X-SET-Failure-URL: http://rs600019.itso.ral.ibm.com/CA/failure.html
X-SET-Cancel-URL: http://rs600019.itso.ral.ibm.com/CA/cancel.html
X-SET-Brand: Pass <http://rs600019.itso.ral.ibm.com/CA/pass.gif>
!!
```

Figure 111. CGI Script to Wake Up the Cardholder Plug In Module

Note:

- 1** This cat command will send to standard output all the text lines following it in the script until the line that reads "!!".
- 2** This is the MIME header to be processed by the browser.
- 3** This is the content of the message. It will be passed as is to the helper, that is to the CommercePOINT Wallet application. And CommercePOINT Wallet expects a MIME message.

Let's name this CGI script ccainit. It must be stored in a directory with other CGI scripts. For example:

```
/usr/lpp/internet/server_root/cgi-bin/ccainit
```

and the URL for this script will be:

```
http://<WWW server hostname>/cgi-bin/ccainit
```

A simple access to this URL should start your CommercePOINT Wallet application and make it ready to request a certificate.

8.10 Signing Certificates with IBM Registry for SET

The SET certification flows, as described in Chapter 4, "SET Certification" on page 51, all have the same overall shape. The SET participant (cardholder, merchant or payment gateway) communicates with the CA, which provides proof of identity. The participant then generates a certificate request and sends it securely to the CA. If the request is acceptable, the CA signs it and returns the completed certificate.

At the time of our project, only the cardholder process was completely implemented by IBM Registry for SET. For the merchant and payment gateway certificates we used a test program, `mpgcert`, to drive the certificate signing process. In this section we briefly describe how we used `mpgcert` and then go into more detail on the options for cardholder certification.

8.10.1 Producing Test Certificates for Merchant and Payment Gateway

To request a merchant certificate using `mpgcert` when the IBM Registry for SET server is running:

1. Create a new directory:

```
mkdir /usr/local/setca/merchant
```

2. Set the `SETCAKEY_PATH` environment variable to the path of this new directory:

```
export SETCAKEY_PATH=/usr/local/setca/merchant
```

3. Copy the root key seed database to this directory:

```
cp /usr/local/setca/rootcakey0.db /usr/local/setca/merchant/key.db
```

4. Copy the other database seed files (provided with IBM Registry for SET) to the same directory:

```
cp /home/nrsetca/keypair.db /usr/local/setca/merchant/keypair.db
cp /home/nrsetca/cr1.db /usr/local/setca/merchant/cr1.db
cp /home/nrsetca/bci.db /usr/local/setca/merchant/bci.db
```

5. Execute the merchant key generation tool:

```
mpgcert localhost 5065
```

where 5065 is the CA IP port (see Figure 92 on page 140 and the `CAPORT` directive in the configuration file, `setca.conf`).

6. Respond to the prompts, as follows:

- Merchant or Payment Gateway Flow? Enter (M/P). Reply **M**.
- Enter the CMS password. Reply **abc123**. This is the password for the key database, which was entered when we created the root seed database previously.
- Enter the brand ID. Reply with the name of your card brand, **Pass** in our sample case. This is part of the *common name* of the Brand CA and it appears in the Brand CA certificate with the label `BrandID`. See A.3, “Displaying the Contents of a CA Key Database” on page 291 for an example.
- Enter the Merchant URL. Reply **get-certificateMCA**. This is the URL used to trigger the merchant signature process. It is the same as specified by the `URL` directive in the configuration file, `setca.conf`.
- Enter the Merchant ID. This can be any name of your choice, but it must match the ID in the `SETMERCH` table on the merchant system (see 7.3.2.1, “`SETMERCH` Table Configuration” on page 120).
- Input the merchant BIN. This is an identifier for the brand that is used to tie together the merchant and its associated acquirer. It can be any number (for example, **12345**) as long as it matches the value entered for the acquirer payment gateway.
- Fill in the registration form details with the merchant name and address information.

7. mpgcert then sends the certification request to IBM Registry for SET and, if it is approved, will receive it into the merchant database (key.db file).

Note that this assumes that online approval is enabled. We will show in 8.10.3, “Implementing a User Exit” a sample authchk user exit to perform this. A more likely case for merchant and payment gateway certificates is that the request will be stored and processed manually later. IBM Registry for SET provides the setapprove command to manage pending requests and action them. You would then need to re-enter mpgcert using the query option (-q flag on the command line) to receive the signed certificate. This function was not fully operational in the early version that we used.

The process to request a gateway certificate using mpgcert is very similar to the merchant process.

Once you have created the key database and ancillary files, copy the complete directory contents (/usr/local/setca/merchant, in the merchant example, above) to the merchant or payment gateway system. The target directories are specified in each case by the value of the \$CmsPath environment variable. This is defined in different ways for the merchant and payment gateway implementations:

- On the merchant server, we recommend setting it in the initial startup script for the SET extensions. See the example in Figure 76 on page 112.
- On the payment gateway, it is defined in the .spgrc profile file, which is in /tsm/tsmspg. By default the value is set to /tsm/tsmspg/database.

8.10.2 Generating Cardholder Certificates

We show an example of requesting a cardholder certificate in Chapter 6, “The Cardholder’s View” on page 85. In summary, the process is:

1. Make sure that IBM Registry for SET is running.
2. Make sure that the cardholder machine has a key database with the root certificate installed. In the case of a test environment, copy /usr/local/setca/rootcakey0.db to \NetCommerce\Data\key.db on your cardholder machine.
3. Using your browser, request the certificate-request URL from the IBM Registry for SET server (in our case, http://<setca server>:31854/get-certificateCCA)

Normally the cardholder certificate will be automatically signed online. In order to make this work you have to define a user exit in IBM Registry for SET which will approve online requests.

8.10.3 Implementing a User Exit

In order to process certificate requests online, a user exit should be implemented to capture data about the request and supply immediate decisions about approving, denying, or delaying a certificate request.

A user exit is not mandatory when you start your IBM Registry for SET server for the first time. If you commented out the ACFILE directive in the configuration file, no user exit is invoked (see Figure 98 on page 152).

Through its application program interface, IBM Registry for SET provides an exit for online approval function. This exit can be any customer-written code conforming to the API.

8.10.3.1 Developing the User Exit

The user exit should be written in C or C++. The interface definition file `authchk.h` contains all you need to write your own user exit:

- Return codes
- Structure definitions
- Entry point prototypes

Samples are provided with the IBM Registry for SET package. You can use them as a starting point for your own user exit. But, in any case, you should apply the following rules:

1. Make sure your exit is thread safe.
2. Use the `ac_malloc` routine to allocate storage passed back to the server.

The sample listed in Figure 112 on page 168 was used in our test environment. This sample illustrates the rules listed above and gives hints to writing your own user exit. The policy implemented in this user exit is simple: all certificate requests are approved!

```

/*
 * ITSO AuthChk user exit
 *
 * This user exit approves all requests
 *
 * Debug messages are logged into a file
 */

#include "authchk.h"

#define LOGFILE "/usr/local/setca/log/authchk.log"

long AuthChk( long exit, void * in, void ** out,
              void * (* ac_malloc)(size_t size))
{
    FILE *fd;
    long rc;

    fd = fopen( "/usr/local/setca/log/authchk.log", "a" );

    if ( fd )
        fprintf( fd, "Entering authchk (%d) ...\n", exit );

    switch ( exit )
    {
        struct _out_CardCInit *CCIp;
        struct _out_Cert      *Cp;

    case AUTHCHK_X_CARDCINIT:
        *out = ac_malloc( sizeof(out_CardCInit) ); 1
        CCIp = *out;
        CCIp->version = ((struct _in_CardCInit *)in)->version;
        CCIp->action = AUTHCHK_ACTION_OK; 2
        fprintf( fd, "CARDCINIT (%ld)\n", CCIp->version );
        rc = AUTHCHK_OK; 3
        break;

    case AUTHCHK_X_REGFORMREQ:
        fprintf( fd, "REGFORMREQ\n" );
        fprintf( fd, "\tCA URL : [%s]\n",
                ((struct _in_RegForm *)in)->CA_URL );
        fprintf( fd, "\tCA name : [%s]\n",
                ((struct _in_RegForm *)in)->CA_Name );
        fprintf( fd, "\tPAN : [%s]\n",
                ((struct _in_RegForm *)in)->PAN );
        rc = AUTHCHK_OK; 3
        break;

    case AUTHCHK_X_ME_AQCINITREQ:
        fprintf( fd, "ACQINITREQ\n" );
        rc = AUTHCHK_OK; 3
        break;
    }
}

```

Figure 112 (Part 1 of 3). AuthChk Exit Sample Source Code


```

case AUTHCHK_X_CERTREQ:
    *out = ac_malloc( sizeof(out_Cert) );
    Cp = *out;
    fprintf( fd, "CERTREQ\n" );
    fprintf( fd, "\tCA URL : [%s]\n",
              ((struct_in_Cert *)in)->CA_URL );
    fprintf( fd, "\tCA name : [%s]\n",
              ((struct_in_Cert *)in)->CA_Name );
    fprintf( fd, "\tPAN : [%s]\n",
              ((struct_in_Cert *)in)->PAN );
    fprintf( fd, "\tPAN expr : [%s]\n",
              ((struct_in_Cert *)in)->PANexpr );
    fprintf( fd, "\tmerchant BIN : [%s]\n",
              ((struct_in_Cert *)in)->merchantBIN );
    fprintf( fd, "\tmerchant ID : [%s]\n",
              ((struct_in_Cert *)in)->merchantID );
    fprintf( fd, "\tacquirer BIN : [%s]\n",
              ((struct_in_Cert *)in)->acquirerBIN );
    fprintf( fd, "\tAcct Identification : [%s]\n",
              ((struct_in_Cert *)in)->AcctIdentification );
    Cp->version = ((struct_in_Cert *)in)->version;
    Cp->action = AUTHCHK_ACTION_APP; 4
    rc = AUTHCHK_OK; 3
    break;

case AUTHCHK_X_CERTINQREQ:
    fprintf( fd, "CERTINQREQ\n" );
    rc = AUTHCHK_OK; 3
    break;

case AUTHCHK_X_RESPONSE:
    fprintf( fd, "RESPONSE\n" );
    rc = AUTHCHK_OK; 3
    break;

case AUTHCHK_X_ADMIN:
    fprintf( fd, "ADMIN\n" );
    rc = AUTHCHK_OK; 3
    break;

case AUTHCHK_X_ERROR:
    fprintf( fd, "ERROR\n" );
    rc = AUTHCHK_OK; 3
    break;

case AUTHCHK_X_APPROVER:
    fprintf( fd, "APPROVER\n" );
    rc = AUTHCHK_OK; 3
    break;

case AUTHCHK_X_INIT:
    fprintf( fd, "INIT\n" );
    rc = AUTHCHK_OK; 3
    break;

```

Figure 112 (Part 2 of 3). AuthChk Exit Sample Source Code

```

        case AUTHCHK_X_SHUTDOWN:
            fprintf( fd, "SHUTDOWN\n" );
            rc = AUTHCHK_OK; 3
            break;

        default:
            rc = AUTHCHK_UNKEXIT;
            break; 3
    }
    fflush( fd );
    close( fd );
    return rc;
}

```

Figure 112 (Part 3 of 3). AuthChk Exit Sample Source Code

Note:

- 1** ac_malloc() routine is used to allocate storage passed back to the server.
- 2** The version action code must always be set in the output structure.
- 3** The return code must be AUTHCHK_OK, AUTHCHK_UNKEXIT, AUTHCHK_BADDATA or AUTHCHK_ERROR.
- 4** The action code for a certification request must be AUTHCHK_ACTION_APP, AUTHCHK_ACTION_DEN or AUTHCHK_ACTION_OK if the certificate is respectively approved, denied or deferred.

To create the loadable module for this sample:

```

xlc_r -qcpluscmt -c authchk.c
xlc_r -o authchk authchk.o -eAuthChk

```

where AuthChk is the name of the entry point in the C file.

And make sure to give execute permission to the nrsetca user ID.

8.10.3.2 Integrating the User Exit

The user exit is configured in the /etc/setca.conf file. In the stanza for the CA, a declaration indicates the filename of the user exit:

```
ACFile /usr/local/setca/bin/authchk
```

If this modification is done when the server is running, you will have to stop and restart the server to validate it.

Chapter 9. The Payment Gateway

The CommercePOINT Gateway is the IBM application that provides the acquirer payment gateway function of SET. The payment gateway is a very specialized function, so this product is more a set of code elements and toolkits than a ready-to-run application. In this chapter we will describe CommercePOINT Gateway and show examples of configuring and extending it.

9.1 CommercePOINT Gateway: Overview, Interfaces and Interactions

CommercePOINT Gateway functions as an intelligent router for incoming and outgoing messages to and from the Internet. It interfaces between legacy credit card processing systems and the SET protocol. It checks the validity of data received in incoming SET messages and provides user exits to easily customize message formats to legacy systems.

Legacy

Use of the word legacy in this context is rather unfortunate, in that it implies that the systems we are connecting to are being retired. In fact, SET is just another input to the systems, which are very much alive and well. However, "legacy" seems to have become common usage for the bankcard networks in respect to SET, so we will continue to use it here.

The CommercePOINT Gateway has four main components:

1. The Payment Gateway Transaction Monitor (PGTM)
2. The Payment Gateway Application
3. The Payment Gateway Listener
4. The Payment Gateway Legacy Application

Figure 113 on page 172 shows the different components and the interfaces between them. Note that we have omitted the firewall from this diagram, but there would certainly be a screening firewall between the Internet and the Payment Gateway Listener and another firewall between the Payment Gateway Transaction Monitor and the legacy gateway.

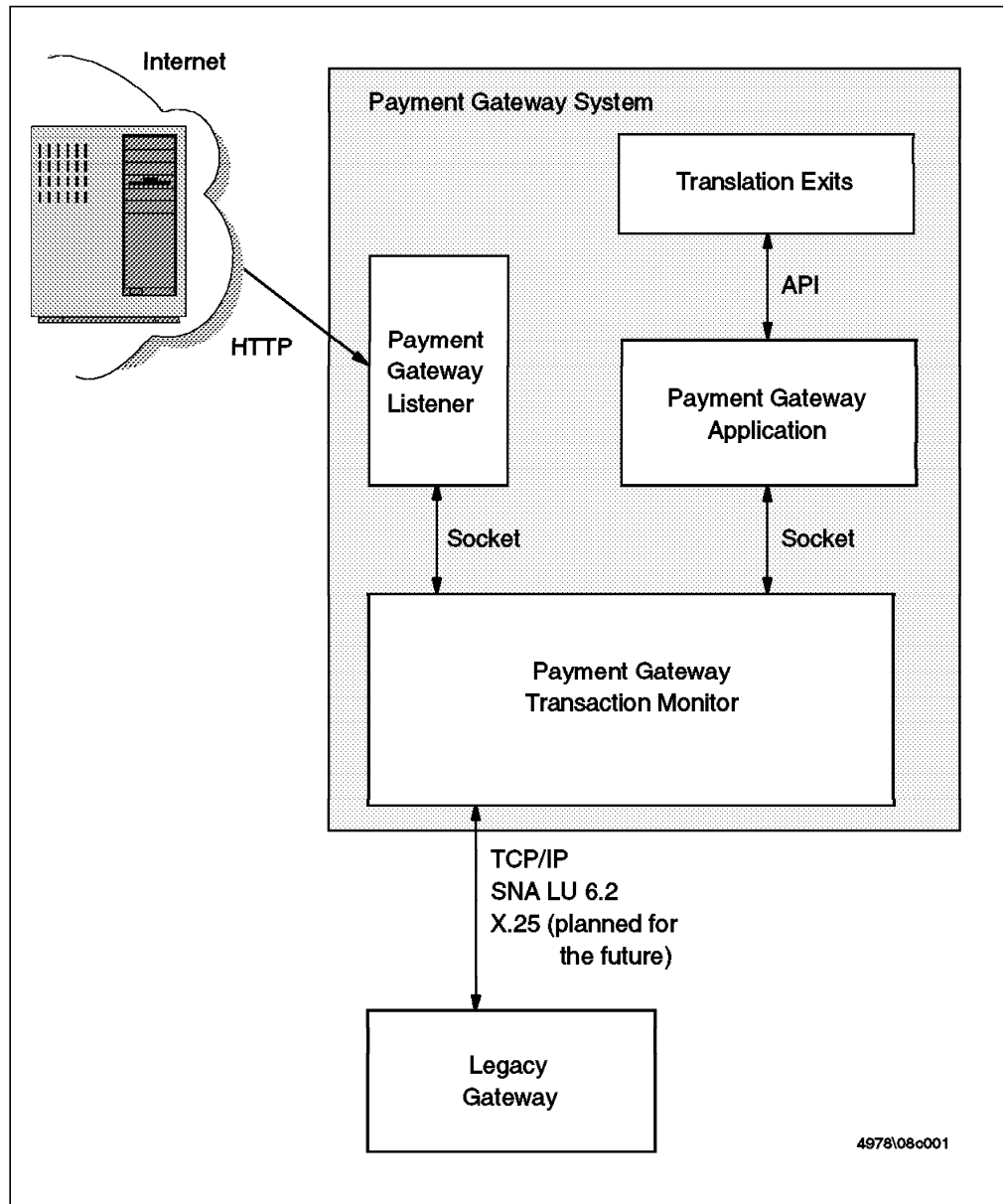


Figure 113. CommercePOINT Gateway Interfaces and Interactions

9.1.1 The Payment Gateway Transaction Monitor

The Payment Gateway Transaction Monitor (PGTM) is the underlying infrastructure of the gateway and a system designed to enable the development of communication-independent applications. It allows applications located on different platforms and using different communication methods to interact with one another by providing conversion between both protocols and data format when specified by the user. To do so, the PGTM provides several services such as:

- Registration services
- Interface services
- Mapping services
- Monitoring services

- Interconnect services
- Trace services
- Audit log services
- Messaging services
- Translation services
- Encryption services
- Store and Forward Services

Actually, the PGTM is an advanced router between applications.

The PGTM support interfaces to the merchant servers, the acquirer host processor and the certificate authorities. The PGTM currently supports local connections (sockets), TCP/IP and SNA LU 6.2 for the communications with the applications. X.25 support is planned in the future.

In practice, the PGTM is implemented as a group of processes (daemons) such as:

- saf_daemon
- init_router
- local_client, tcp_conn_client
- unix_connect_server, tcp_connect_server

9.1.2 Payment Gateway Application

PGTM provides routing services, but there is a set of core functions that all SET implementations need in addition to the specialized interface applications for other networks. These functions are provided by the Payment Gateway Application.

Also Known As a TSM

CommercePOINT Gateway is constructed from a number of building blocks, based on more generic transaction processing code. The main element of this is called a Transaction Service Manager (TSM). You will find the term TSM used extensively in place of the real application titles in the configuration dialogs.

The Payment Gateway Application is implemented as a PGTM application. As such, it utilizes PGTM services to perform all communication functions with merchant servers, legacy hosts and certificate authorities.

The Payment Gateway Application performs the following functions:

- Authenticates the merchant.
- Verifies the acquirer/merchant relationship.
- Decrypts the payment instructions from the cardholder.
- Validates that the cardholder certificate matches the account number used in the purchase.
- Validates consistency between merchant's authorization request and the cardholder's payment instruction data.

- Provides end-user exits with the appropriate data fields from the SET message.
- Responds to the merchant with a SET response.

In the PGTM environment, the Payment Gateway Application is a server that listens for requests from the other participants in the SET protocol and delivers the services listed above to them.

The Payment Gateway Transaction Monitor and the Payment Gateway Application communicate over UNIX sockets, so they must both reside on the same AIX system.

In practice, the Payment Gateway Application is implemented as a daemon: spg, which is started by a super daemon: pgmaster.

The end-user exits or translation exits should be customized for the acquirer environment. Their function is to support message conversion between SET messages and the acquirer's legacy messages.

9.1.3 The Payment Gateway Listener

The Payment Gateway Listener is the entry point to the CommercePOINT Gateway for the merchants. The Payment Gateway Listener listens on a dedicated IP port and collects HTTP requests from the Internet. Its role is to present these HTTP requests to the Payment Gateway Application.

In the PGTM environment, the Payment Gateway Listener is a client and the Payment Gateway Application is the server that will service the SET requests collected by the Payment Gateway Listener application. In practice, the Payment Gateway Listener is implemented as a daemon: shttpsrv, which is started by the super daemon pgmaster.

9.1.4 The Payment Gateway Legacy Application

The fourth component of the CommercePOINT Gateway is the Payment Gateway Legacy Application. The Payment Gateway Legacy Application is a PGTM application. The role of this application is to serve the requests sent by the Payment Gateway Application, to process them and to send the responses back to the Payment Gateway Application. The translation exit in the Payment Gateway Application formats the messages before sending them to the Payment Gateway Legacy Application so you can think of the exit as an extension of the function of this component, converting the formats to allow it to act as a gateway between the PGTM environment and the acquirer host processor.

Since this application is dependent on the legacy system, it cannot be generic. Each acquirer implementation will have to develop its own Payment Gateway Legacy Application variant.

As a PGTM application, the communication between the PGTM and the Payment Gateway Legacy Application can be established over socket, TCP/IP, LU 6.2 and X2.5 in the future.

9.1.5 Payment Authorization Scenario

To illustrate the relationship between the different components implied in the payment gateway system, let us examine the payment authorization flow between the merchant and the acquirer.

A representation of the different steps in this flow is shown in Figure 114.

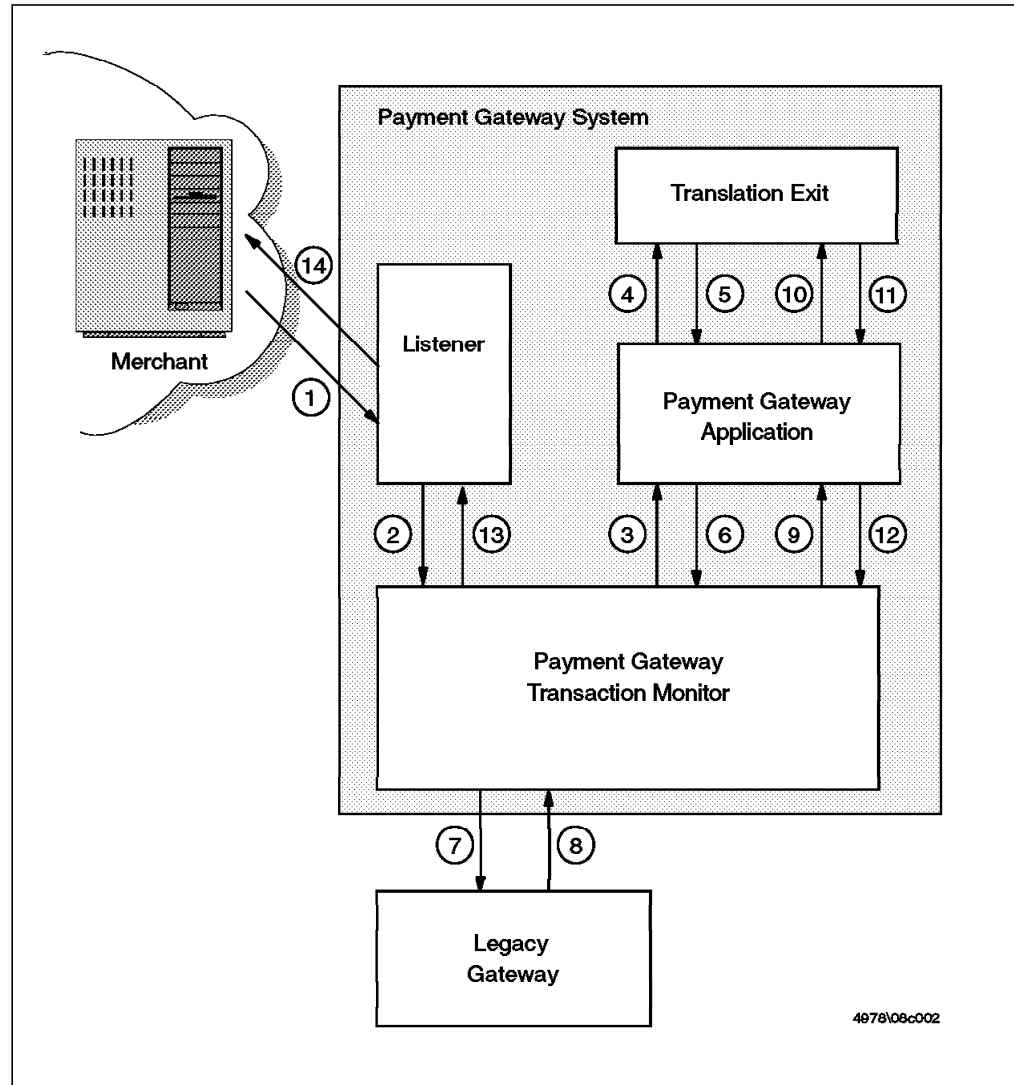


Figure 114. CommercePOINT Gateway Interfaces and Interactions

1 The merchant sends a payment authorization request to the acquirer. The request is sent over TCP/IP and the destination port is the one defined in the record in the SETACQUIRER table. The request data conforms to the SET specification and are transmitted using the HTTP protocol.

2 On the acquirer system, the Payment Gateway Listener is permanently waiting for requests on a dedicated IP port. This listener is responsible for extracting the core data from the incoming HTTP requests and determining the final destination for the request. When the request is reformatted and the ID of the Payment Gateway Transaction Monitor destination application is identified, the listener submits this request to the Payment Gateway Transaction Monitor.

The Payment Gateway Listener maintains the connection with the merchant for the duration of the process.

3 The Payment Gateway Transaction Monitor routes the request to the Payment Gateway Application according to the destination application ID.

4 The Payment Gateway Application analyzes the request, decrypts the SET message, and validates whatever needs to be validated, such as cardholder and merchant certificates. When the SET message is analyzed and validated, the Payment Gateway Application invokes the translation exit corresponding to the SET message type.

5 The translation exit routine does any message conversion required and passes back to the Payment Gateway Application the modified message along with a return code. At this point of the flow, the message is a request as opposed to a response. There should be two different processes in the translation exit, according to the nature of the message: request or response. The translation here is therefore from an incoming SET request to its equivalent banking system message.

6 The Payment Gateway Application gets back the message from the translation exit and sends this message to the Payment Gateway Legacy Application through the Payment Gateway Transaction Monitor.

7 The Payment Gateway Transaction Monitor routes the message to the Payment Gateway Legacy Application according to the destination application ID. As for any job submitted to the Payment Gateway Transaction Monitor, the job is traced and logged.

8 The Payment Gateway Legacy Application analyzes the request message, does any processing required to determine the answer to be sent back to the merchant. This processing includes any connection to a legacy host system, etc. The response is formatted to be suitable for the Payment Gateway Application and its translation exit processing. The response is sent to the Payment Gateway Application through the Payment Gateway Transaction Monitor that will record the transaction.

9 The Payment Gateway Transaction Monitor routes the message from the Payment Gateway Legacy Application to the Payment Gateway Application according to the destination application ID.

10 The Payment Gateway Application receives the response message from the Payment Gateway Legacy Application and invokes the translation exit.

11 The translation exit routine is responsible for translating the banking system response to its equivalent SET response. The SET response is sent back to the Payment Gateway Application along with a return code.

12 The Payment Gateway Application does any encryption and signing required using the gateway public key certificates (see Chapter 3, "SET Payment Protocols" on page 17). The result is a properly formatted SET response. The Payment Gateway Application then submits the SET response to the Payment Gateway Transaction Monitor in order to route it to the Payment Gateway Listener.

13 The Payment Gateway Transaction Monitor routes the response message to the Payment Gateway Listener according to the destination application ID.

14 The Payment Gateway Listener receives the response message from the Payment Gateway Application and sends it to the merchant with which the IP connection is still established. When the HTTP response message is sent, the connection is closed.

9.2 Preparing the System

In this section we walk through the steps needed to get the basic payment gateway functions working.

9.2.1 Verifying Prerequisites

The Payment Gateway requires a platform with the following hardware and software requirements:

- RISC System/6000
- 500 MB hard disk space
- AIX V4.1.4+
- DB2/6000 V2.1.1+
- C Set++ Run Time Libraries V3.1.4.2
- ADSM V2.1.0.4
- SNA Server/6000

Note: ADSM is used for backup support by PGTM, but is optional. If ADSM is not used, the database must be managed by other means. Similarly, SNA Server/6000 is only needed if the gateway will communicate with a bankcard network that uses SNA protocols.

9.2.2 Planning the Installation

The installation procedure will create new file systems on your system:

- /tsm, the transaction manager file system
- /tsm/tsm/log/audit, the audit file system

The two file systems will be created in the same volume group and require at least 50 MB. Before starting the installation procedure, you need to decide in which volume group you want these file systems to be created. To determine the list of volume groups on your system, use the `lsvg` command, and to get detailed information on a volume group, for example `rootvg`, enter `lsvg rootvg`. The `FREE PPs` field gives information on the free space in the given volume group, `PP` being the Physical Partitions (usually 1 PP = 4 MB).

The Payment Gateway Transaction Monitor uses a DB2 database to store different kinds of information, ranging from configuration data to transaction logs. Before creating the database and the tables in it, you need to designate an owner and an administration group for the DB2 instance.

In our test environment, the configuration was established as follows:

- The volume group for the Payment Gateway Transaction Monitor file systems is `rootvg`.

- The administration group for the DB2 instance is instpgw.
- The owner of the DB2 instance is instpgw and its home directory is /home/instpgw.

These values will be used in the example commands in the following sections.

9.3 Installing the Payment Gateway

The Payment Gateway uses the `installp` provided by AIX to install the system software. `Installp` installs Licensed Program Products (LPPs). An LPP is composed of packages that contain individually installable entities called filesets.

The Payment Gateway LPP name is `spg`. The LPP image for the Payment Gateway contains three packages and each package contains individually installable entities as follows:

- `spg`
 - `spg.tmgr`** PGTM system software
 - `spg.appl`** Payment Gateway Application system software
- `spg.msg.{lang}`
 - `spg.msg.{lang}.tmgr`** PGTM message catalogs
 - `spg.msg.{lang}.appl`** Payment Gateway Application message catalogs
- `spg.ps.{lang}`
 - `spg.ps.{lang}.tmgr`** PGTM PostScript documents
 - `spg.ps.{lang}.appl`** Payment Gateway Application PostScript documents

The installation process will create all required system resources. The system resources include:

- User IDs
- Groups
- File systems
- `/etc/inittab` entries
- `/etc/services` entries

Prior to installing the system, ensure that DB/2 has been installed and verify that all `spg` system resources are removed from the system. `Installp` is very sensitive and if the resources that it tries to create are already contained on the system, it will fail.

Note: Do not attempt to stop the installation once it has begun. If the installation is stopped using `Ctrl-C` or any other mechanism, operating system resources may be left in an inconsistent state. If you determine that you do not want to install the system during the install, let the installation complete and deinstall the system using the `installp` command as follows:

```
installp -u spg > out 2>&1
```

Installation Steps

1. Log in as root
2. If necessary, uncompress and untar the CommercePOINT Gateway image as follows:

```
zcat spg.<version>.<release>.<maintenance>.<fixlevel>.tar.Z | tar -xvpf -
```

This will generate the following files, which can be installed using installp:

- spg
- spg.msg.{lang}
- spg.ps.{lang}

3. Before executing the installation command, you need to define the TMVOLGRP environmental variable. This variable should contain the name of the volume group in which the new file systems will be created. See 9.2.2, “Planning the Installation” on page 177 for more information on this. Export the TMVOLGRP environmental variable. For example, if the chosen volume group is rootvg and the shell is the Korn Shell:

```
export TMVOLGRP=rootvg
```

and install the system using installp from the command line as follows:

```
installp -aX -d <device_containing_image> spg > output 2>&1
```

where device_containing_image is the name of the device on which the package is available, possibly a file.

The output file will contain all output generated by installp. It can be used to verify that the image was installed correctly.

4. Set passwords for all user IDs created by the image. The user IDs created are:
 - tsm (PGTM)
 - tsmspg (payment gateway user ID)
 - tsmadmin

9.4 Configuring the Payment Gateway

Note

The example commands in the following section are those used in our test environment. The configuration parameters are detailed in 9.2.2, “Planning the Installation” on page 177. Please replace the configuration parameters with your own values whenever they appear in the commands.

Before configuring the payment gateway, you need to configure DB2.

9.4.1 Create Server DB2 Instance

This section describes how to create the server DB2 instance, assuming that the PGTM is implemented as a server only and not as a client/server configuration.

1. Log in as root.
2. Using smit crjfs create three journaled file systems:
 - /home/instpgw for the instance with at least 12 MB

- /home/instpgw/dbtsmdb for the database with at least 128 MB.
- /home/instpgw/logtsmdb for the logs with at least 64 MB.

Note: Logs should be on a different physical device from the database. If the archive logs are kept locally the file system size should be increased.

Note: It is recommended that all file systems be created with the automatic mount at restart option enabled.

3. Mount the instance owner file system:

```
mount /home/instpgw
```

4. Create the database and log directories:

```
mkdir /home/instpgw/dbtsmdb  
mkdir /home/instpgw/logtsmdb
```

5. Mount the database and log file systems:

```
mount /home/instpgw/dbtsmdb  
mount /home/instpgw/logtsmdb
```

6. Create a DB2 system administration group for the instance owner:

```
mkgroup instpgw
```

7. Using the smitty mkuser command, create the Server Instance Owner user ID with the following attributes:

- a. Primary group equal to the DB2 System Administration group created above (instpgw).
- b. Enter a home directory if you do not want to take the default.
- c. Login equal to true (it should be the default).
- d. Rlogin equal to true (also the default).

8. Set the Server Instance Owner user ID's password:

```
passwd instpgw
```

9. Change the owner privileges for the file systems created above. Set the owner to the Server Instance Owner user ID and the group to the DB2 system administration group, as follows:

```
chown instpgw:instpgw /home/instpgw  
chown instpgw:instpgw /home/instpgw/dbtsmdb  
chown instpgw:instpgw /home/instpgw/logtsmdb
```

10. Create a DB2 for server instance:

```
/usr/lpp/db2_02_01/instance/db2icrt instpgw -a server
```

where instpgw is the server instance name.

11. Enter DB2 license information into /usr/lib/netls/conf/nodelock.

12. Add group tsmdb to the Instance Owner's list of groups by editing the attributes for this user:

```
smitty chuser
```

13. Log in as instance owner.

14. Set up the DB2 environmental variable by updating file db2profile located in directory ~instpgw/sqllib (that is, /home/instpgw/sqllib, in our example), as follows:

- a. Add line LIBPATH=\${LIBPATH}:/usr/lpp/db2_02_01/lib

- b. Add line `export LIBPATH`
 - c. Change `DB2DBDFT=SAMPLE` to `DB2DBDFT=`
15. Invoke `db2profile` from within the instance owner `.profile` script:
 - a. Create `~instpgw/.profile` file if it does not already exist
 - b. Add the following line:

```
. /home/instpgw/sql1lib/db2profile
```
 16. Log in again as instance owner to allocate `db2profile`.
 17. Start the DB2 server instance:

```
db2 start database manager
```

9.4.2 Create DB2 Database and Database Objects

This section describes how to create and initialize the DB2 database used by the PGTM.

1. Log in as instance owner `instpgw`.
2. Change to directory `/tsm/tsm/db_utils`.
3. Execute script `create_tsm_database` as follows:

```
./create_tsm_database INSTPGW /home/instpgw off /tsm/tsm
```

where:
 - `INSTPGW`=server instance name
 - `/home/instpgw`=server instance owner home directory
 - `off`=the database configuration value for option `userexit` (valid values are `on` or `off`). In this sample configuration, no `userexit` executable is used to archive or retrieve log files or to manage the location of the archived log files.
 - `/tsm/tsm`=home directory of `tsm` user ID

The script writes a log to directory `/tsm/tsm/db_utils/db_log`.

Note: Verify that the database is started prior to performing this step.

Note: Enter the password for the server instance owner when prompted for a password.

4. Stop and restart the DB2 server instance:

```
db2 stop database manager
db2 start database manager
```
5. Log in as the PGTM user ID (`tsm` in our case).
6. Edit `~tsm/.dbrc` to change the path for `db2profile`.
7. Log off and back on with the `tsm` user ID so that `db2profile` is run.
8. Execute the `/tsm/tsm/tsm_utils/create_tsm_tables` script.
9. Log in as the instance owner (`instpgw`).
10. Back up the database offline:

```
mkdir ~instpgw/backup
cd ~instpgw/backup
backup database TSMDB
```

9.4.3 PGTM Environmental Configuration

As the PGTM is comprised of a variety of processes, each communicating with the others through message queues and shared memory, configuration information must be maintained to allow these processes to effectively communicate with one another. To do this, a set of configuration files is used, along with environmental variables, to indicate which processes are started and the resources they use. The configuration files, located in /tsm/tsm/config, are:

- local_config_file
- global_config_file
- tsm_base_processes
- tran_table_config
- tsm_subsystems

The environment the PGTM and its processes run under is established by a set of initialization files:

- .tsmrc
- .dbrc
- .kshrc

These files are responsible for establishing environmental variables, paths and aliases for the PGTM process. .tsmrc is used to configure most of the variables and paths. .dbrc is used for configuration of PGTM's DB2 instance and usage of ADSM. .dbrc is invoked at login time by the .tsmrc script. It is therefore important that .tsmrc is called by the standard shell initialization script, .kshrc.

Configuration steps

1. Log in as the PGTM user ID (tsm).
2. Verify the DB2 instance is properly set in file .dbrc.
3. If you are not using ADSM, modify the .dbrc file as follows:

```
#CRON_FILE=/tsm/tsm/tsm_utils/tsmCronADSM
CRON_FILE=/tsm/tsm/tsm_utils/tsmCron
```

This change prevents the PGTM from utilizing ADSM to manage its DB2 tables.

4. Initialize PGTM environmental variables by logging off and logging back on. This will export all variables and aliases used by the PGTM.
5. Modify /tsm/tsm/config/tsm_base_processes so that only the required servers and clients are started. This can be done by placing a "#" before the clients and servers that are not needed. The file is broken up into groups of processes to make it easier to determine which processes are which. The file is separated into rows with two columns, which are separated by a <tab>. The columns are:

- PGTM user ID to start the process
- Startup program and its arguments. The path to the program is specified relative to the home directory of the user ID starting the program.

Each process has the same first two parameters and then is followed by communication-specific parameters. The base PGTM process is started as:

```
<process_name> <number of processes> <smid group> <program parms...>
```

where:

- <number of processes>** How many copies of the process are created to handle the process function.
- <smid group>** The set of shared memory tables used, from those defined in tran_table_config. This parameter should not be changed unless groups are added or deleted.

Here is an example of the file:

```
#
# Servers
#
tsm    bin/base/unix_connect_server 1 3 1.0
tsm    bin/base/tcp_connect_server 1 2 1.0
#
# Router
#
tsm    bin/base/init_router 1 2
#
# Clients
#
tsm    bin/base/local_client 1 3 1.0
tsm    bin/base/tcp_conn_client 1 2 1.0
#
# Services
#
tsmadmin bin/base/admin_daemon 1 0
tsm     bin/base/saf_daemon 1 1
```

unix_connect_server and local_client should always be included; all other clients and servers should be included based on the communication needs of the PGTM and the machine.

6. Modify the size and number of the shared memory tables, if necessary, to meet memory restrictions and volume needs. This is done by modifying the /tsm/tsm/config/tran_table_config file. This file contains the layouts of the shared memory tables used by the PGTM processes. The tables must be configured so that they can handle any size transaction that may be received by the process. In addition, the table must be configured with the appropriate number of rows so that concurrent transactions can be handled to meet the volume demands of the system.

The file is structured as a list of numbers of rows and row sizes. Each group of tables is separated by a -1 -1 entry. The group numbers start at 0 and increase by 1 when a -1 -1 entry is found. Hence, the file listed below has 5 table groups, numbered 0 - 4, and group 3 has two tables while the others have only 1. The first table is configured to have 50 rows, each with a length of 1024 bytes.

```
50 1024
-1 -1
50 4000
-1 -1
200 4000
-1 -1
200 16000
50 64000
-1 -1
50 4000
```

7. Modify the /tsm/tsm/config/tsm_subsystems file to run only those subsystems required. This can be done by placing a “#” before any subsystem that is not needed.

There are programs that are extensions to the base PGTM router. They are distinct from the base processes because they typically use the PGTM or are used by the PGTM and should be included in tsm_subsystems.

Here is an example of the file:

```
#Header Map
#tsm    bin/init_header_map  bin/terminate_header_map

#Payment Gateway
tsmspg  bin/init_spg 2    bin/stop_spg
```

The file contains three columns, separated by tabs. It is important that each entry be tab separated and that only 1 tab separates each entry.

The values for each column must contain:

- User ID to start the program under
- Path from the specified user’s home directory to the startup program along with any command line arguments
- Path from the specified user’s home directory to the termination program along with any command line arguments

When the PGTM is started, all of the initialization programs are run. Terminating the PGTM invokes the termination programs. The subsystems are terminated before any other PGTM process and are started last. If subsystems have dependencies on one another they should be configured so that the subsystem that is dependent on another is added later in the file.

8. Initialize the local configuration file used by the PGTM to store information used by PGTM processes to determine how to access the local application directory (LAD) and how to communicate with one another:

```
cd /tsm/tsm
localConfigDrive
```

9. Verify that the file /tsm/tsm/config/local_config_file exists and has a length greater than zero.

10. Initialize the global configuration file by running the following command:

```
globalConfigDrive <num_appls>
```

where num_appls is the base number of applications to allocate space for in the LAD. We recommend 72 as a default, but this number may be adjusted downward if memory is limited.

This file contains information that is global across all PGTMs. The file name is stored in the local configuration file. Therefore, you need to ensure that localConfigDrive is run before running globalConfigDrive.

11. Verify that the file /tsm/tsm/config/global_config_file exists and has a length greater than zero.

12. Copy or link the libraries in /tsm/tsm/lib/lib_prod to the /usr/lib directory.

This is required, because the libraries are dynamically linked to PGTM programs that look in the /usr/lib directory for these files.

```
cd /tsm/tsm/lib/lib_prod
cp -p lib* /usr/lib
```


13. Add an entry for starting DB2 to inittab if it does not already exist.

```
mkitab "db2:2:wait:rc.db2 >/tmp/db2.out 2>&1"
```

where rc.db2 is a shell script to start DB2. Figure 115 shows an example of rc.db2.

```
#!/bin/ksh

if [[ ! -f "/home/instpgw/sql1lib/db2profile" ]]
then
    echo The db2profile file does not exist in
    echo /home/instpgw/sql1lib
    echo DB2 cannot be started
fi
. /home/instpgw/sql1lib/db2profile
su - instpgw -c "db2start"
```

Figure 115. Sample rc.db2 Startup Script

The entry in the /etc/inittab file must be before the inittm entry that initializes the PGTM. Also, it needs to be configured with wait specified (as specified in the mkitab command example above) to prevent other systems, such as the PGTM, from attempting to use DB2 before it is initialized. You can manipulate inittab entries using the chitab command.

14. Register the payment gateway application. This can be done by running the andRegister command and selecting **Register Applications**. Note that this configuration program refers to the application as a Transaction Service Manager, or TSM (see Figure 116, Figure 117 on page 186 and Figure 118 on page 186).

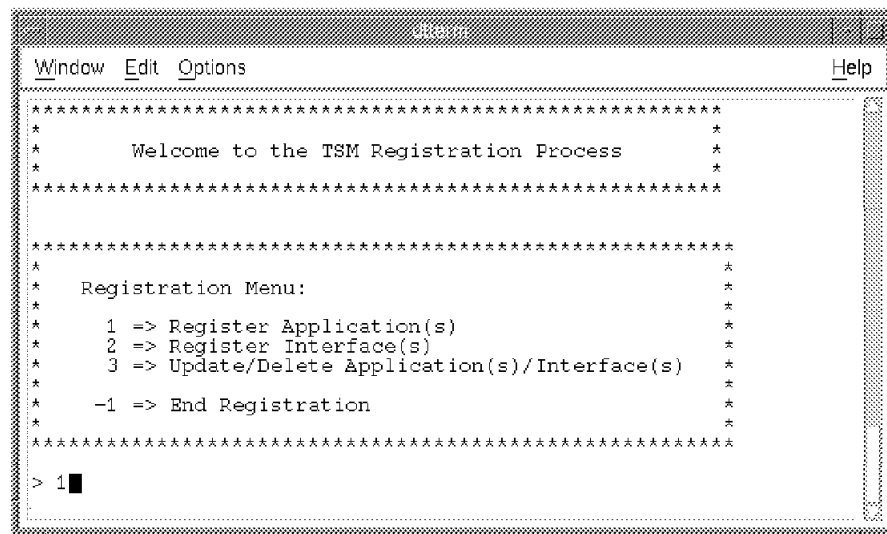


Figure 116. Register TSM

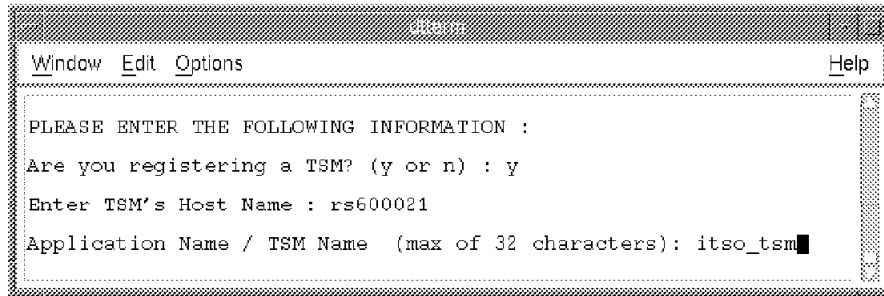


Figure 117. Register TSM (Continued)

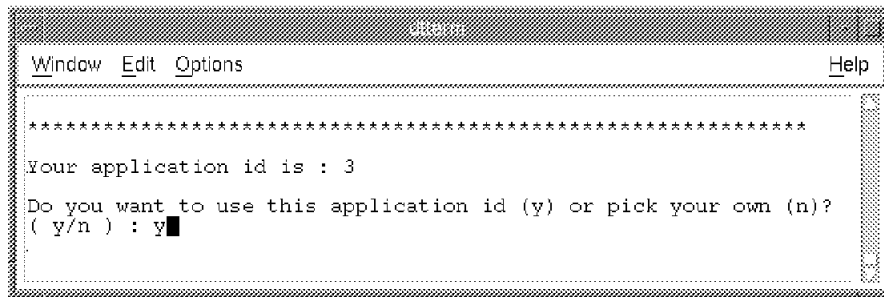


Figure 118. Register TSM (Continued). Note that in this example the application ID is 3, because we already registered one TSM and one Payment Gateway application before this TSM. When you run andRegister for the first time, the application ID for the TSM is 1.

9.4.4 Payment Gateway Application Registration

This section describes the registration process for payment gateway applications. To do this, first log in as user ID tsm spg.

If installing for the first time, create the payment gateway database with the following command:

```
run_su tsm /tsm/tsm/tsmspg/bin/setup_SPGdb /tsm/tsmspg tsmdb
```

If this is not the first installation, bind the payment gateway database, as follows:

```
run_su tsm /tsm/tsm/tsmspg/bin/bind_spgdb_access /tsm/tsmspg tsmdb
```

9.4.4.1 Register Payment Gateway

The registration process is invoked using the andRegister program, as for the TSM registration. andRegister initially displays the screen shown in Figure 116 on page 185. Select option 1 (Register Applications(s)) and then do the following:

1. After selecting option 1 the following message will be displayed:

```
PLEASE ENTER THE FOLLOWING INFORMATION:
```

```
Are you registering a TSM? (y or n) :
```

Reply n.

2. Enter a name for the application:

```
Application Name / TSM Name (max of 32 characters): setpgw
```

This name is a user-defined name that is used to easily identify the application on the statistics screen provided by the PGTM. The internal

identifier, used by the PGTM, for an application is the Application ID. The application ID will be chosen later in the registration process.

3. Enter the application timeout value:

Timeout value: 30

The timeout value is the maximum amount of time, in seconds, that the PGTM will wait for a response from an application.

4. Select the PGTM header version.

Enter Header Version(1-2): 2

The PGTM supports two version of headers. This parameter registers the header version that the application accepts.

5. The next parameter is the default response exit.

Would you like to register a Default Response Exit? (y or n): n

The default response exit is a user-written exit that is used by the PGTM to return a default response to a client when it does not receive a normal response from the application.

6. Answer the question that asks a user to register a translation exit:

Would you like to register a Translation Exit? (y or n): n

If you choose **y**, a path name must be entered along with the format for the translation.

The PGTM provides an exit that performs ASCII-to-EBCDIC translation. The full path of the exit is /tsm/tsm/bin/ascii_ebcdic. This translation exit is used by the PGTM to translate the transaction to the desired format.

7. The next question refers to the store and forward facility of the PGTM:

Do you want to use the TSM Store and Forward Facility? (y or n): n

The store and forward facility is designed to store transactions that are on their way to an application and resend them at a later time. The transactions are stored in the DB2 database.

If you choose **y**, the Resend Interval and Maximum Retries values must be provided.

8. The next parameter requested is a network ID flag:

Does your application require the network ID of the client? (y or n): y

The PGTM has the capability to capture the network address of the client that sent a transaction into the system.

9. Determine if the PGTM will capture audit records for the application:

Do you want audit records captured? (y or n): n

The PGTM provides a facility that captures audit records for an application and stores them in its DB2 database.

10. The next parameter refers to the status of the application when the PGTM is initialized.

Initial status of application (0=Inactive; 1=Active): 1

Applications can be initialized in an active or an inactive state. Applications that are initialized as inactive must be manually activated once the PGTM is initialized in order to have transactions routed to them. If an application is

initialized as active, transactions will be routed to the application as they arrive.

11. Choose the application ID.

Your application ID is: 2

Do you want to use this application ID (y) or pick your own (n)? (y/n): y

The application ID is used by the PGTM to uniquely identify the application within the PGTM system. All application IDs must be unique.

9.4.4.2 Register the Payment Gateway Interface

Once the base parameters are registered for a payment gateway application, the communication interfaces can be registered. The communication interface parameters determine the protocol that the PGTM will use to communicate with the application. They consist of base interface parameters and communication protocol parameters.

1. Run the andRegister program. The screen shown in Figure 116 on page 185 will be displayed.
2. Choose option 2 to register an interface (see Figure 119).

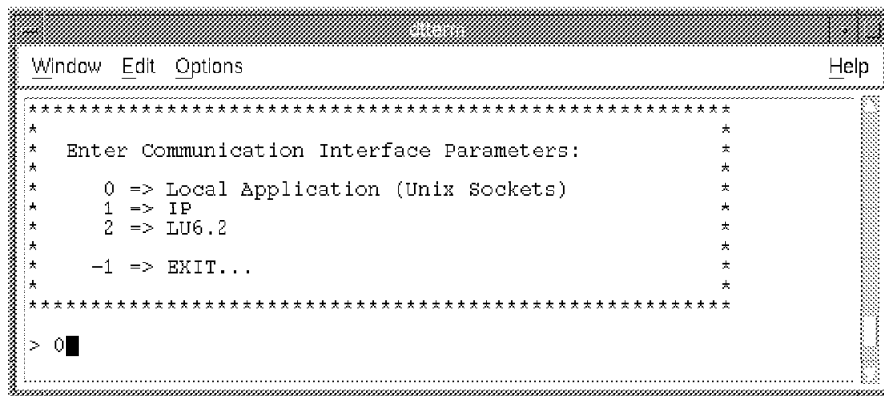


Figure 119. Register Payment Gateway Interface

3. Choose **Local Application (Unix Sockets)**.

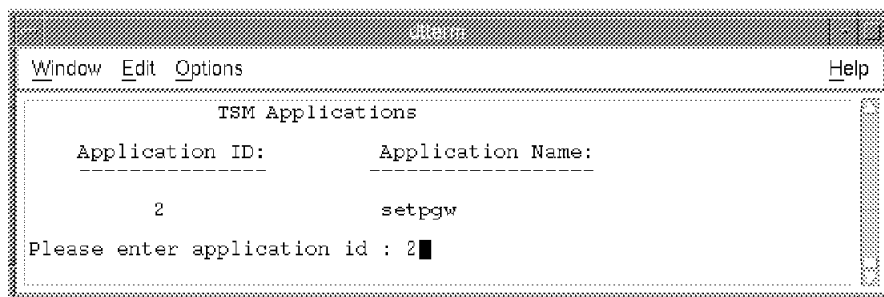


Figure 120. Register Payment Gateway Interface

4. Enter the application ID allocated to the application.
5. Enter the path name of a UNIX socket.
Please enter path name of Unix Socket: /tsm/tsmspg/socket/receiveSocket
6. Determine the status of the interface at PGTM initialization time:
Enter Interface Status (Active=1, Inactive=0): 1

Like an application, an interface can initially be active or inactive. If it is initialized as inactive, it must be manually changed to active before transactions will flow to it.

7. Enter the interface priority. Interfaces can be registered at different priorities. Interface at a lower priority will not be used unless all of the interfaces at all higher priorities are inactive. This allows an interface to be registered as a backup interface. All interfaces within the same priority level will be load balanced.

Enter Interface Priority (0-3, 0=Highest Priority): 0

8. The next parameter to register is a translation exit. This is a user-provided exit that translates the transaction into the format of the application hosting machine. When the response is received from the application, the exit performs the reverse of the request transaction.

Would you like to register a Translation on Exit? (y or n): n

If you enter **y**, you must also enter a data format and the file path of the exit.

9. Determine if the PGTM will strip off the PGTM header before it transmits the transaction to the application.

Will your application accept a TSM header? (y or n): y

If you answer **n**, the PGTM will strip off the header and transmit only the data portion of the transaction to the application.

9.4.4.3 Verify that the Payment Gateway Is Registered

By selecting option 3 from the main registration menu, you can check that the payment gateway is registered correctly. Figure 121 through Figure 127 on page 192 show the sequence of displays.

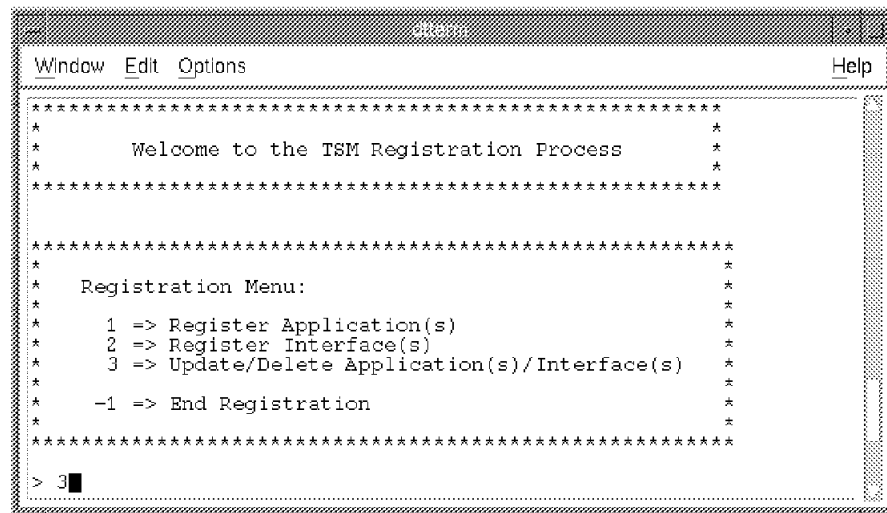


Figure 121. andRegister Main Menu, Select Option 3

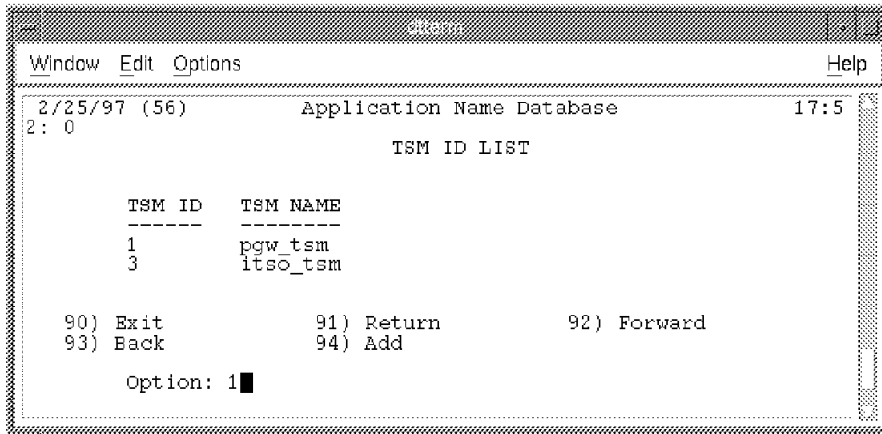


Figure 122. Two TSMs Registered, Select The Payment Gateway, pgw_tsm

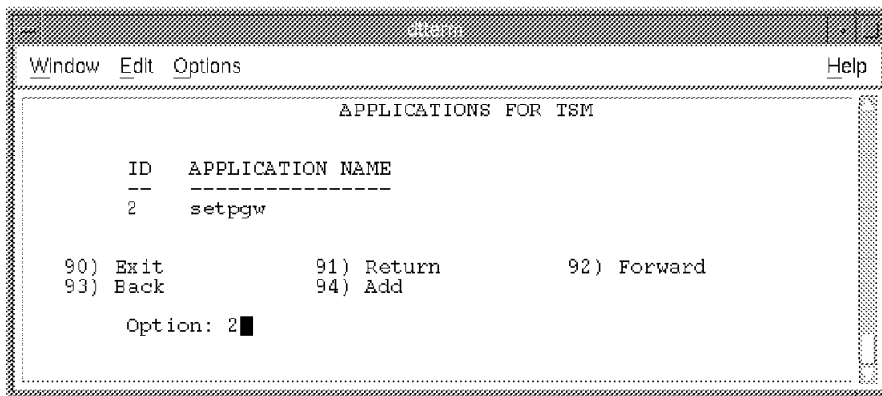


Figure 123. One Application Registered to the TSM, Select It

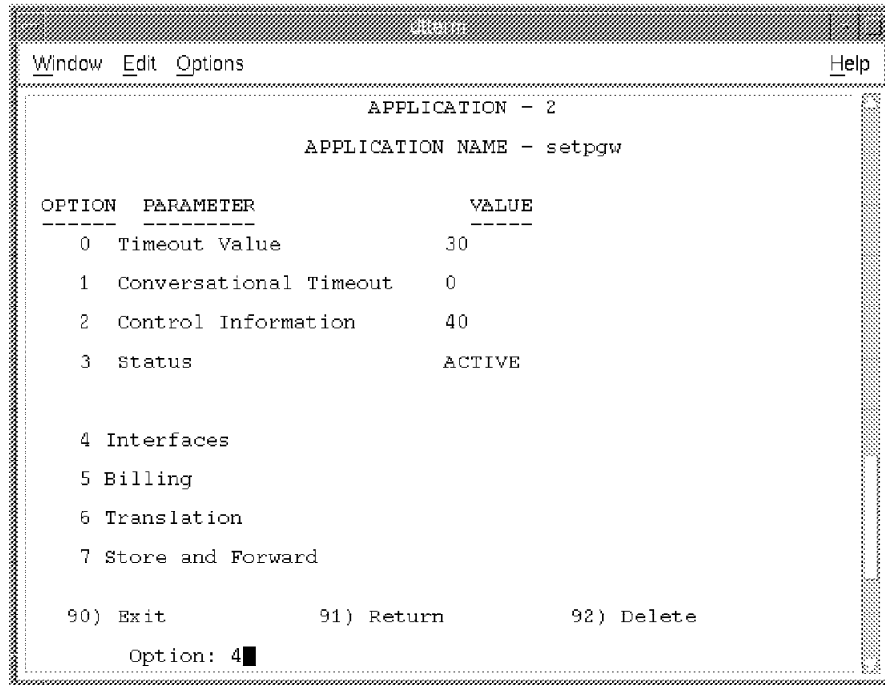


Figure 124. Initial Application Details Screen, Select Interfaces

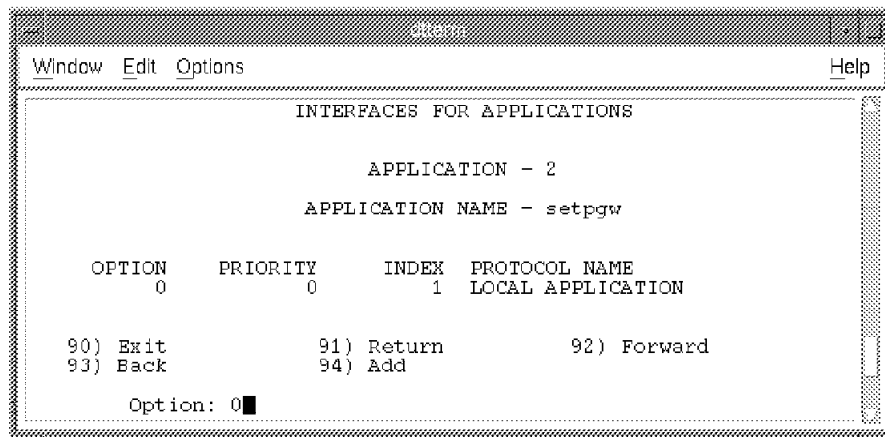


Figure 125. One Local Interface Defined

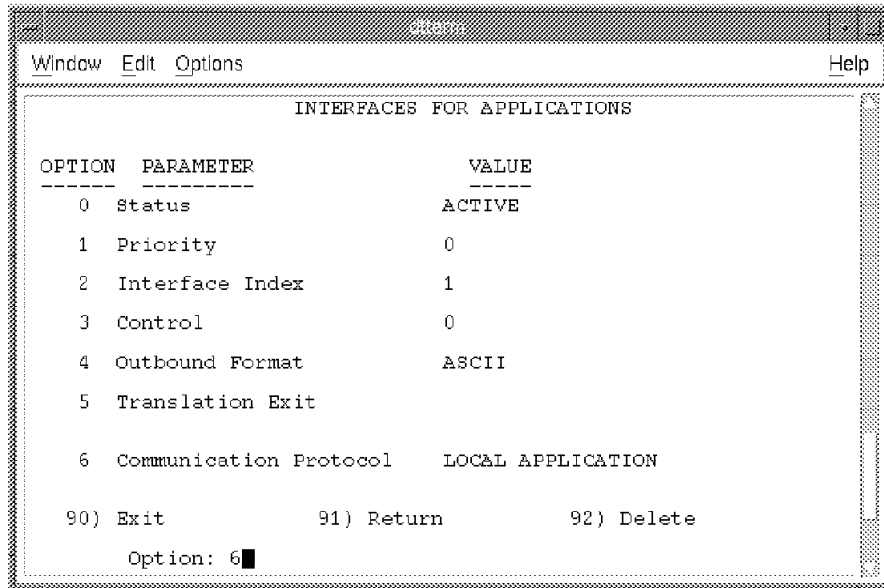


Figure 126. Interface Details, Select Communications Protocol

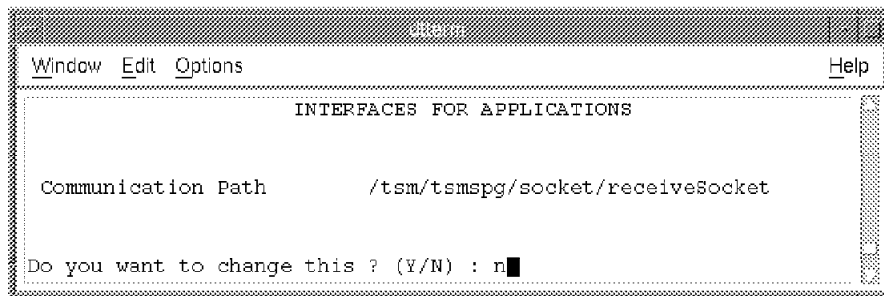


Figure 127. UNIX Socket Definition for Local Application

9.4.5 Configure the Payment Gateway

Now that we have configured the application router, the next step is to configure the listener process, which is the payment gateway as it appears to the other SET parties.

The payment gateway listener can be configured by using the pgconfig command located in the /tsm/tsmspg/bin directory. pgconfig is used to add a new payment gateway, change the configuration of an existing payment gateway, or delete an existing payment gateway. Figure 128 on page 193 shows the pgconfig main menu.

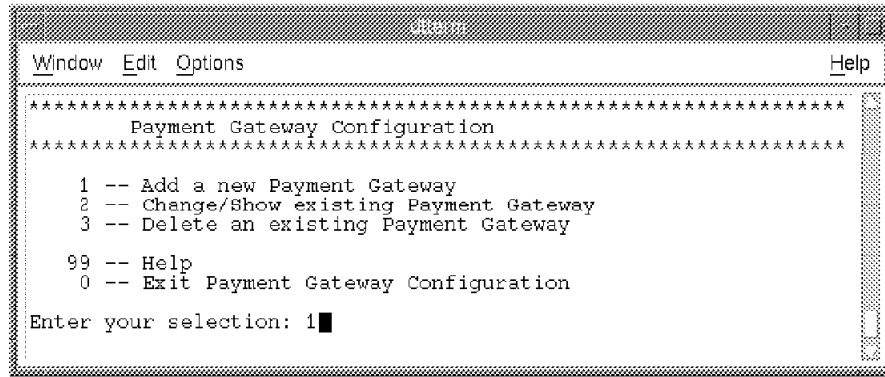


Figure 128. pgconfig Main Menu

9.4.5.1 Add a Payment Gateway Listener

Before adding a payment gateway, you must select a TCP/IP port. A payment gateway communicates with merchant servers using a TCP/IP port that you must define. Refer to 9.1.3, “The Payment Gateway Listener” on page 174 for more information on this TCP/IP port. Your /etc/services file should contain an up-to-date list of all TCP ports in use. Select a port that is not currently in use (we recommend using a number higher than 10000). Update /etc/services to add the TCP port that the payment gateway will be using.

In the lab environment, we chose TCP/IP 12345, so the entry in /etc/services is:

```
12345/tcp pgwy-listener # Payment Gateway Listener
```

You have to step through a sequence of pgconfig screens to update the port definition. Select 1 (**Add a new Payment Gateway**) from the pgconfig main menu and then step through the sequence shown in Figure 129 through Figure 133 on page 195.

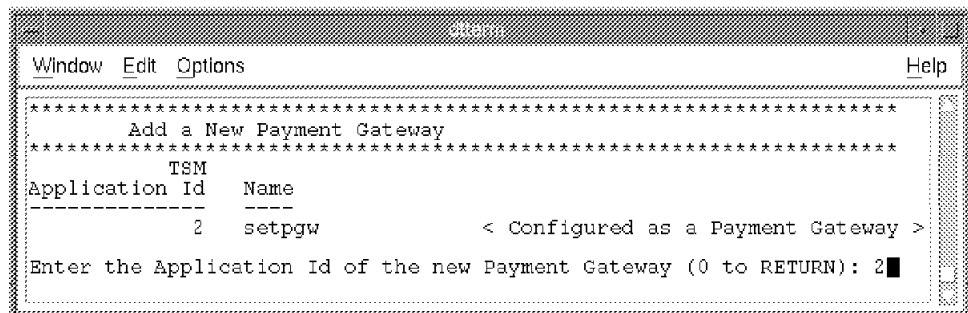


Figure 129. Add a Payment Gateway. Select the payment gateway application you added before.

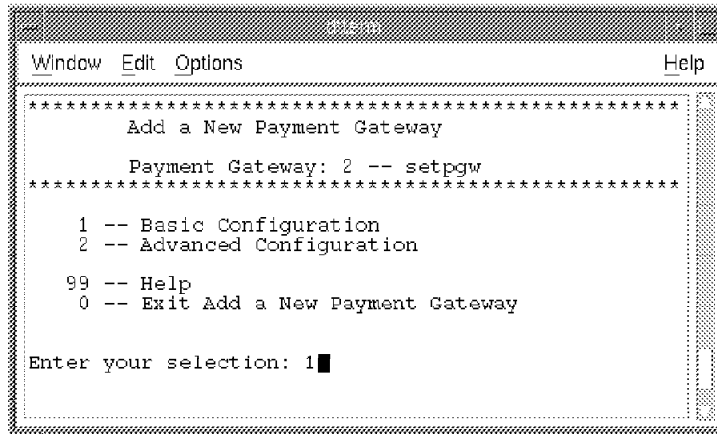


Figure 130. Select 1 for Basic Configuration

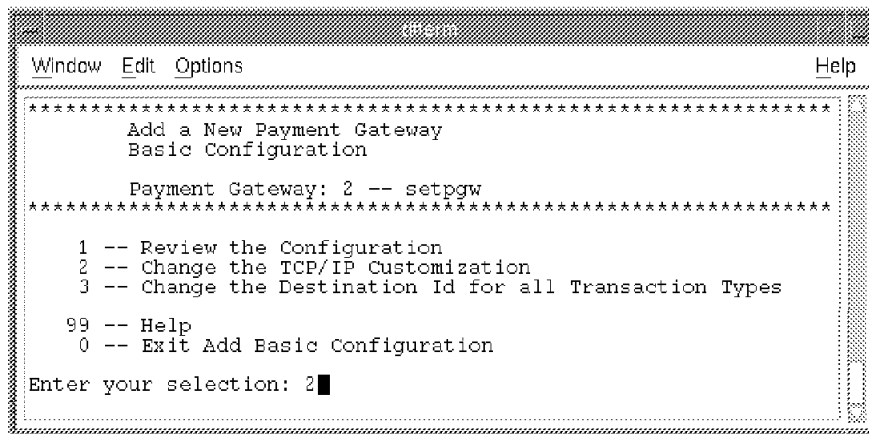


Figure 131. Select 2 to Update the TCP/IP Configuration

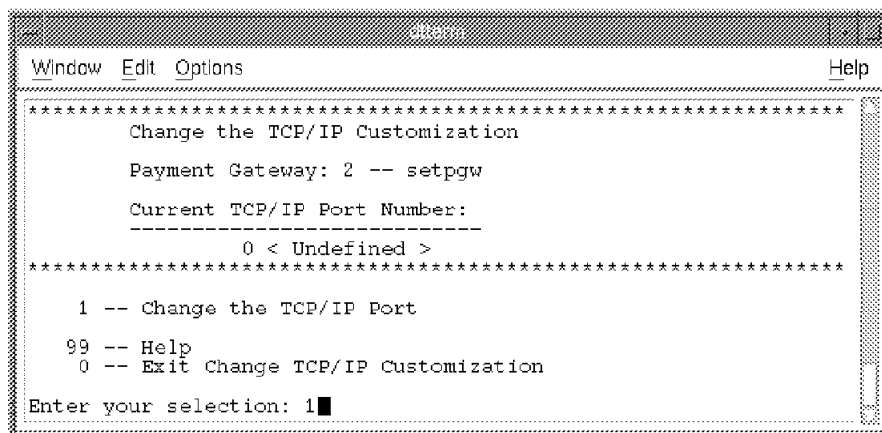


Figure 132. Select 1 to Change the TCP/IP Port

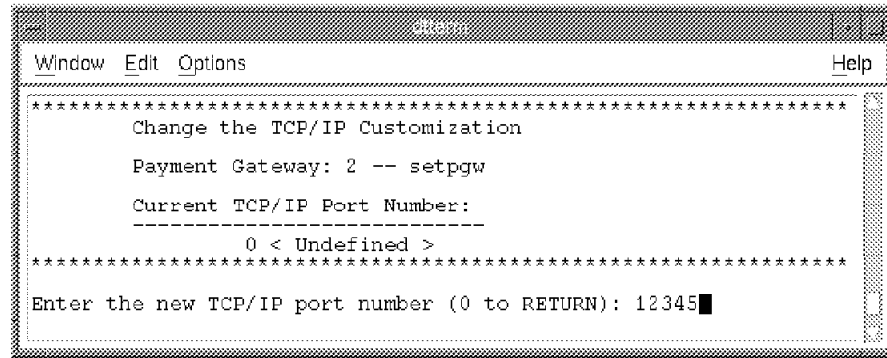


Figure 133. Enter Your Chosen Port Number

If your configuration is valid, you will receive a message that confirms that the payment gateway has been added to the configuration.

The payment gateway listener can be started automatically when PGTM is initialized and stopped automatically when PGTM is terminated by updating the /tsm/tsm/config/tsm_subsystems file as follows:

```
userid start_prog parm stop_prog
```

where:

- userid is the ID under which program execution will be started.
- start_prog is the path and file name of the program that starts the payment gateway listener.
- parm is the payment gateway application ID.
- stop_prog is the path and filename of the program that stops the payment gateway listener.

For example, if the application ID is 2 and the payment gateway user ID is tsmspg, the entry in the tsm_subsystems file would be:

```
tsmspg bin/init_spg 2 bin/stop_spg
```

All entries in the file must be separated by a single tab (\t) character.

Note: In the tsm_subsystems file provided with the product, a default entry is provided to define the startup and shutdown procedure for the Payment Gateway Application. You will have to edit this entry to replace the application ID with the ID returned by andRegister when you registered the Payment Gateway Application.

9.4.5.2 Change a Payment Gateway

Once you have added a payment gateway, you can change the configuration of your payment gateway at any time. Changes that you make to the configuration will not take effect until the next time you restart the payment gateway.

Select option 2 from the pgconfig main menu.

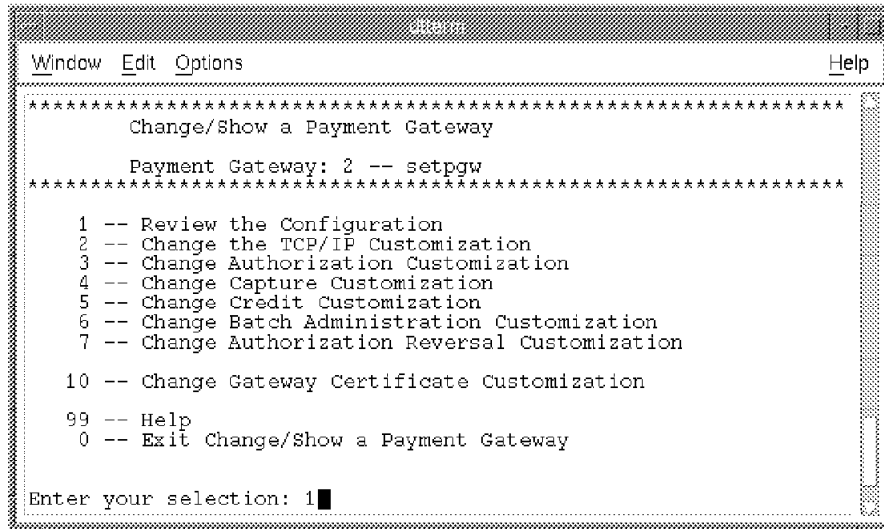


Figure 134. Changing a Payment Gateway

9.4.5.3 Delete a Payment Gateway

Once you have added a payment gateway, you can delete it at any time. A payment gateway that is deleted while it is running will continue running until it is stopped.

Select option 3 from the pgconfig main menu.

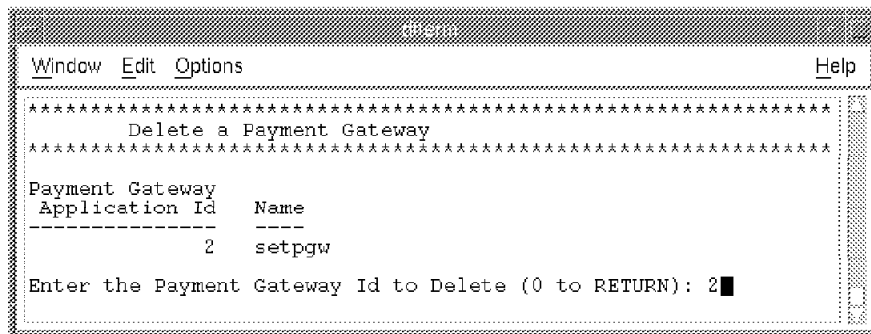


Figure 135. Deleting a Payment Gateway

9.4.6 PGTM Initialization and Termination

Initialization and termination of the PGTM is done through a variety of scripts and programs. The scripts start and stop each of the PGTM processes and ensure that each is started under the appropriate user ID. In addition to starting the programs, these scripts manage the resources used by the PGTM.

9.4.6.1 PGTM Initialization

After the PGTM has been installed and configured, the PGTM can be started under the tsm identity using the command:

```
init_tsm
```

This command is responsible for:

1. Creating the resources used by the PGTM processes

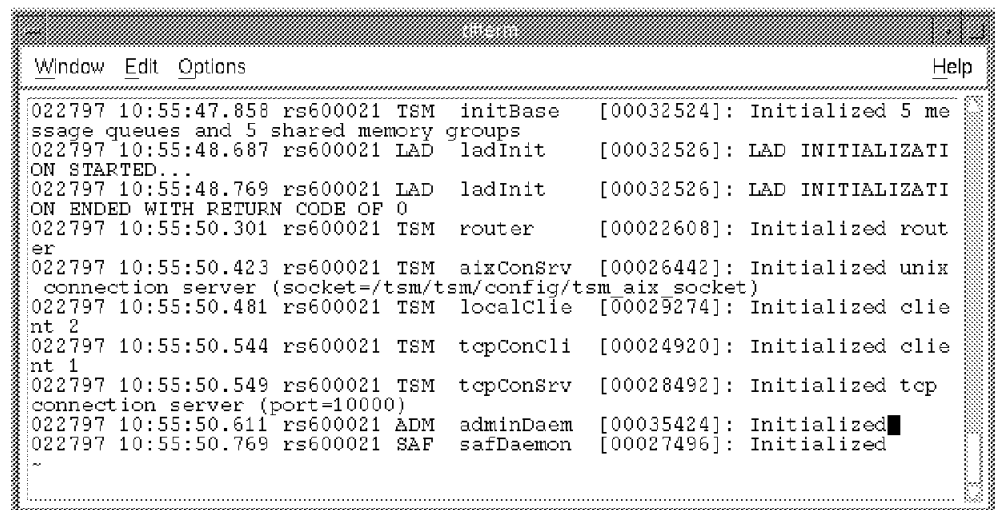
2. Initializing a statistics file for the current day if one does not already exist
3. Forming the local application directory (LAD) from the Application Name/Database (AN/D)
4. Scheduling routine maintenance tasks, such as loading audit files and backing up statistics and log files
5. Starting the PGTM processes specified in /tsm/tsm/config/tsm_base_processes
6. Starting the PGTM subsystems specified in /tsm/tsm/config/tsm_subsystems
7. Ensuring that if the PGTM is already active, it is terminated to avoid conflicts between systems using different LADs and different system resources

9.4.6.2 Verifying PGTM Initialization

Once the `init_tsm` command has been run, the status of the PGTM can be verified through a variety of methods. The simplest method is to examine the PGTM's system log file, `/tsm/tsm/log/tsm_sys_log`.

Note: If this file is not present, verify that the environmental variable `TSM_SYS_LOG_FILE_NAME` is specified in `.tsmrc`.

As each component of the system is initialized, it records its initialization status in the system log. All components should have an initialization status of 0 or indicate they are initialized. An example of what should be in the system log after a successful initialization is shown in Figure 136.



```
022797 10:55:47.858 rs600021 TSM initBase [00032524]: Initialized 5 me
ssage queues and 5 shared memory groups
022797 10:55:48.687 rs600021 LAD ladInit [00032526]: LAD INITIALIZATI
ON STARTED...
022797 10:55:48.769 rs600021 LAD ladInit [00032526]: LAD INITIALIZATI
ON ENDED WITH RETURN CODE OF 0
022797 10:55:50.301 rs600021 TSM router [00022608]: Initialized rout
er
022797 10:55:50.423 rs600021 TSM aixConSrv [00026442]: Initialized unix
connection server (socket=/tsm/tsm/config/tsm_aix_socket)
022797 10:55:50.481 rs600021 TSM localClie [00029274]: Initialized clie
nt 2
022797 10:55:50.544 rs600021 TSM tcpConCli [00024920]: Initialized clie
nt 1
022797 10:55:50.549 rs600021 TSM tcpConSrv [00028492]: Initialized tcp
connection server (port=10000)
022797 10:55:50.611 rs600021 ADM adminDaem [00035424]: Initialized
022797 10:55:50.769 rs600021 SAF safDaemon [00027496]: Initialized
~
```

Figure 136. Example of PGTM System Log

To determine if the LAD is active, run the command `tsmStats`. If this command provides a list of applications, the LAD is properly initialized.

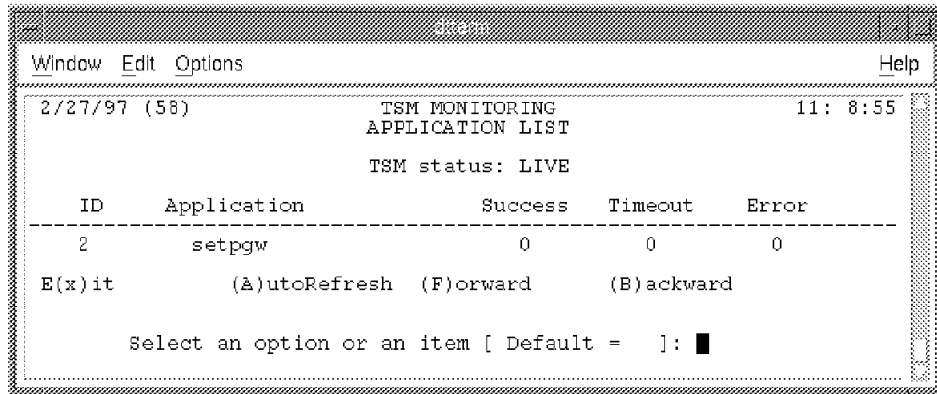


Figure 137. Checking that LAD is Active

9.4.6.3 PGTM Termination

It is important to shut down the PGTM cleanly. To terminate the PGTM, log in as the tsm user and enter the command:

```
terminate_tsm
```

This command performs the reverse of init_tsm. It terminates all of the subsystems and then the PGTM processes. Once these are shut down, the LAD and resources used by the PGTM are removed.

As with initialization, log messages indicate the termination status of each process (see Figure 138).

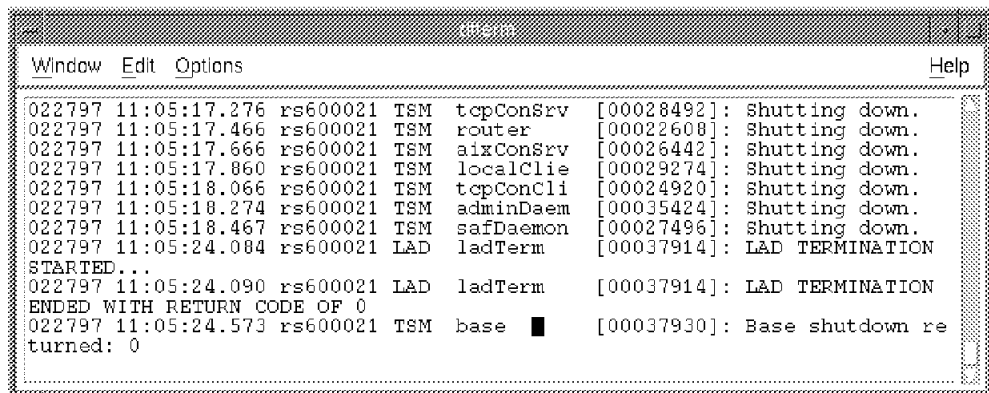


Figure 138. Example of PGTM System Log at Termination

9.5 Customizing the Translation Exit

The Secure Electronic Transaction protocol is intended to be used with many different existing banking systems. Each of these systems has its peculiarities and especially its own set of messages to communicate with.

As explained in 9.1.2, "Payment Gateway Application" on page 173, the Payment Gateway Application provides all the routines to do the following:

- Translate SET requests from merchants into simplified requests.

- Invoke user exits to translate the simplified requests into requests respecting the banking system message format.
- Invoke the user exit to translate the response from the banking system into a simplified response.
- Translate the simplified response into a full SET response and send it back to the merchant.

Due to their nature, the user exits depend on the bankcard network system and it is the responsibility of the acquirer to adapt them in each case. The translation exits are implemented as a set of routines stored in a shared library. This shared library is named `libIBM_PayGate.a` and is loaded at run-time. It must be located in a directory in the payment gateway's `LIBPATH`. The same user exit is invoked when a request is being sent to the legacy application and also when the legacy application returns a response. The user exit code has a number of entry points, each related to a specific payment gateway function.

A skeleton for user exits is provided with the product. It is located in `/tsm/tsmspg/exit/source`. This directory includes a makefile as well as the C source code.

The following section describes the implementation of a set of translation exits that we used in our lab environment.

9.5.1 Sample Translation Exit Code

Figure 139 on page 200 lists the sample translation exit code. We discuss the numbered lines following the listing.

```

//=====
// IBM SET Payment Gateway v1.0 - Payment Agent Translation Exit Skeletons
//
// Please make a copy of this file prior to making modifications
//
// IMPORTANT:
//   Do not delete any of the functions that are supplied with this file.
//   They are required in order for the gateway to work properly.
//
// This file contains the skeletons for the transaction translation
// functions that are executed by the payment agents. Each function
// initially returns a value "unsupported." It is the customer
// responsibility to fill in actual processing function where it is required.
//
// General programming notes:
//   These exits will be run in a threads environment. It is
//   extremely important to ensure that the code in these functions
//   is thread-safe and re-entrant. They also must be compiled and linked
//   with a thread-safe version of the C or C++ Compiler (cc_r for C,
//   x1C_r for C++).
//
//   Storage allocated by the exits for legacy requests and info baskets
//   must be obtained using malloc or calloc subroutines.
//
//   Storage allocated by the exits will be deallocated by the payment
//   gateway in the following situations:
//     Exit allocates a legacy request and exits with the PG_EXITRC
//       set to PG_EXITRC_ASCII_LEGACY_RQST or PG_EXITRC_EBCDIC_LEGACY_RQST
//     Exit allocates an info basket and exits with the PG_EXITRC
//       set to PG_EXITRC_SET_INFOB.
//
//   Important note: the payment gateway will not deallocate any
//   storage pointed to by the dataptr when the PG_EXITRC
//   is PG_EXITRC_NOT_SUPPORTED, PG_EXITRC_INTERNAL or PG_EXITRC_UNKNOWN.
//   If the exit has allocated storage prior to exiting with
//   PG_EXITRC_NOT_SUPPORTED, PG_EXITRC_INTERNAL or PG_EXITRC_UNKNOWN,
//   it is the exit's responsibility to free that storage. Otherwise,
//   the exit will introduce a memory leak into the system (this will
//   eventually cause the payment gateway itself to crash as a result
//   of lack of available storage. The recovery procedure would be
//   to restart the payment gateway).

```

Figure 139 (Part 1 of 11). Payment Gateway Legacy Application Sample Source Code


```

// The exit should never free any storage provided as input from the
// payment gateway.
//
// ibmset.h contains descriptions of the structures used in the
// contexts and info baskets. If a structure element listed in
// ibmset.h is followed by a comment indicating that the element
// is optional or NULL, you may set the structure's address to 0
// instead of allocating storage for it if you have nothing to
// store inside the structure. However, if the structure element
// listed in ibmset.h is **not** followed by a comment that indicates
// that the structure is optional or NULL, you **must** allocate
// the structure and set the field to the allocated structure's
// address even if you have nothing to store inside the structure.
//
// Example: in the AuthResInfo basket, there is a structure
// called acqCardMsgData which contains three optional fields
// acqCardText, acqCardUrl, and acqCardPhone. Even if you do
// not have any information to store in acqCardText, acqCardUrl
// or acqCardPhone, you must allocate an acqCardMsgData structure
// and place its address inside the info basket.
//
//
//=====
// The functions are designed to handle the appropriate request/response
// pair. They are defined as follows:
//
// Function Name                               SET Transaction Type
// -----
// Xlate_AuthReqRes                             Authorization
// Xlate_CapReqRes                              Capture
// Xlate_CredReqRes                             Credit
// Xlate_BatchAdminReqRes                       Batch Administration
// Xlate_RevReqRes                              Authorization Reversal (*)
// Xlate_CapRevReqRes                           Capture Reversal (*)
// Xlate_CredRevReqRes                          Credit Reversal (*)
//
// (*) denotes those transaction pairs not supported in payment gateway
// release 1.0
//=====
// The function parameters are defined as follows:
//
// indicator - flag describing type of transaction to being passed to exit for
// processing.
//
// context - application data from the inbound SET request
//
// dataptr - general buffer variable used to pass transaction data to/from
// the translation exit.
//
//          = NULL (for inbound SET request)
//          = legacy response (for inbound legacy response)
//          = NULL (for gateway error)
//          = legacy request (for outbound legacy request)
//
// data_length - size of outbound legacy request or inbound legacy response
//              (ignored when PG_EXITRC = PG_EXITRC_SET_INFOB)
//
// legacy_applid - TSM application ID override value
//
// gateway_rc - gateway return code from processing of legacy request
//              (see header_codes.H TRANS_RETURN_CODES for a listing
//              of possible return codes)
//
//
// For more information, refer to the IBM_PayGate.H file or the appropriate
// Payment Gateway documentation.
//=====
// Maintenance Log:
//

```

Figure 139 (Part 2 of 11). Payment Gateway Legacy Application Sample Source Code

```

// Version      Date      Developer      Description
//-----
// 00          09/09/96    Wiesman       Initial Release
// 01          01/03/97    KARENU        Only use "C" (EXTTYPE)
//              for externs
//              when using C++ compiler,
//              pass in generic AcqPayCtx
//              rather than specific
//              request contexts
// 02          03/13/97    Ezvan         ITSO sample
//=====
#include "IBM_PayGate.H"

#define LOG_FILE "/tmp/userexit.log" 1

//-----
// THIS FUNCTION IS REQUIRED BY THE GATEWAY AND MUST NOT BE DELETED FROM
// THIS FILE
//-----
// Name: Xlate_AuthReqRes
// Process Authorization request and response transactions
//
// Context: AcqPayCtx
// AuthRequestCtx = &(context->msgctx.u.authPairCtx.authReqCtx)
// Info Basket: AuthResInfo
// Required fields for AuthResInfo info basket:
//   authResPayload
//   authHeader
//   authAmt
//   currency
//   amount
//   amtExp10
//   respCode
//   responseData
//   acqCardMsgData (cannot be NULL, structure must be allocated)
//
// For additional details about the context and the info basket,
// see ibmset.h
//-----
extern EXTTYPE
PG_EXITRC Xlate_AuthReqRes( PG_INDICATOR indicator,
                          AcqPayCtx * context,
                          void ** dataptr,
                          short * data_length,
                          long * legacy_applid,
                          TRANS_RETURN_CODES gateway_rc)
{
    // define local variables; P = ptr variable; L = local copy 2

    AuthRequestCtx *authReqCtxP;
    AuthResInfo *authResInfoP;
    AuthHeader *authHeaderP;
    int authReqAmtL;
    int authReqCurL;
    int authReqExpL;
    char PanL[20];
    char *merAcqBinP;
    FILE *fd;
    PG_EXITRC rc;

    fd = fopen( LOG_FILE, "a" ); 1
    fprintf( fd, "\nEntering Xlate_AuthReq ... \n" ); 1
    if ( indicator == PG_INDICATOR_REQUEST ) { 3
        fprintf( fd, "Authorization Request !\n" );
    }
}

```

Figure 139 (Part 3 of 11). Payment Gateway Legacy Application Sample Source Code

```

if (authPairCtx_chosen)
    authReqCtxP = &(context->msgctx.u.authPairCtx.authReqCtx);
else
    fprintf( fd, "IBMMessagecontext is %h\n", context->msgctx.choice );

fprintf( fd, "\nGeneral debugging info\n\n" );
fprintf( fd, "piState is %d\n", authReqCtxP->piState );

if ( authReqCtxP->authToken == NULL )
    fprintf( fd, "enveloped data ptr null\n" );

if ( authReqCtxP->panToken == NULL )
    fprintf( fd, "pantoken ptr is null\n" );
else {
    strcpy( PanL, authReqCtxP->panToken->pan );
    fprintf( fd, "PAN is %s\n", PanL );
    fprintf( fd, "Card expr %s\n", authReqCtxP->panToken->cardExpiry );
}

fprintf( fd, "\nAuthorization data\n\n" );

if ( authReqCtxP->panData == NULL )
    fprintf( fd, "pandata ptr is null\n" );
else {
    strcpy( PanL, authReqCtxP->panData->pan );
    fprintf( fd, "PAN is %s\n", PanL );
    fprintf( fd, "Card expr %s\n", authReqCtxP->panData->cardExpiry );
}

if ( authReqCtxP->authReqPayload == NULL )
    fprintf( fd, "AuthReq payload ptr null\n" );
else {
    authReqAmtL = authReqCtxP->authReqPayload->authReqAmt.amount;
    authReqCurl = authReqCtxP->authReqPayload->authReqAmt.currency;
    authReqExpl = authReqCtxP->authReqPayload->authReqAmt.amtExp10;

    fprintf( fd, "amount to be authorized is %d\n", authReqAmtL );
    fprintf( fd, "amount exp10 is %d\n",
        authReqCtxP->authReqPayload->authReqAmt.amtExp10 );
}

fprintf( fd, "\nCertificate data\n\n" );

fprintf( fd, "Merchant Name is %s\n ",
    context->merchantCertificateExtensions->merName.u.merName_printableString );

fprintf( fd, "Merchant ID is %s\n ",
    context->merchantCertificateExtensions->merID.u.MerchantID_printableString );

merAcqBinP = strdup( context->merchantCertificateExtensions->merAcquirerBIN);

fprintf( fd, "Merchant\\Acquirer BIN is %s\n ", merAcqBinP );
fprintf( fd, "Merchant\\Acquirer BIN is %s\n ",
    context->merchantCertificateExtensions->merAcquirerBIN );

fprintf( fd, "BrandID is %s\n", context->brandID->u.BrandID_printableString );

// Allocate storage for SET Auth. Response Msg. 4
authResInfoP = (AuthResInfo*) calloc(1, sizeof(AuthResInfo));

authResInfoP->authResPayload =
    (AuthResPayload*) calloc(1, sizeof(AuthResPayload));

authResInfoP->acqCardMsgData =
    (AcqCardMsgData*) calloc(1, sizeof(AcqCardMsgData));
authHeaderP = &(authResInfoP->authResPayload->authHeader);

```

Figure 139 (Part 4 of 11). Payment Gateway Legacy Application Sample Source Code

```

// fill auth. response fields needed by the merchant 5

authResInfoP->tokenOpaque = NULL;
authResInfoP->batchID = NULL;

fprintf( fd, "\n" );

if ( PanL[ 0 ] != '1' ) // fail this authorization if first digit <> 1
{
    authResInfoP->authRetNum = 0;
    authResInfoP->acqCardMsgData->acqCardURL =
        (URL) strdup("http://rs600019.itso.ra1.ibm.com\n");

    authHeaderP->authAmt.amount = 0;
    authHeaderP->authAmt.amtExp10 = 0;

    fprintf( fd, "generating bad authorization\n" );

    // respCode (see SETACQAUTHRES table on merchant host)
    strcpy( authHeaderP->respCode, "001" );
    authHeaderP->authAmt.currency = 0;

    strcpy( authResInfoP->authResPayload->authHeader.responseData.
        authValCodes.authCode, "999999" );
    authResInfoP->authResPayload->authHeader.responseData.respReason =
        (RespReason)strdup( "Your PAN should start by '1'\n" );
    authResInfoP->authResPayload->authHeader.responseData.bit_mask =
        authValCodes_present;
}
else // good authorization 7
{
    fprintf( fd, "\nauthorizing requested amount\n" );

    authResInfoP->acqCardMsgData->acqCardURL =
        (URL) strdup("http://rs600019.itso.ra1.ibm.com\n");
    authResInfoP->authRetNum = 69;

    authHeaderP->authAmt.amount = authReqAmtL;
    authHeaderP->authAmt.currency = authReqCurL;
    authHeaderP->authAmt.amtExp10 = authReqExpL;
    authResInfoP->authResPayload->authHeader.responseData.respReason =
        (RespReason)strdup( "Because we are a nice bank !\n" );
    strcpy( authHeaderP->respCode, "000" );
}

// end filling of response fields

// Pass addr of SET response structure back to PGW

*dataptr = (void*) authResInfoP; 8

fprintf( fd, "return number %d\n", authResInfoP->authRetNum );
fprintf( fd, "issuing bank URL? %s\n",
    authResInfoP->acqCardMsgData->acqCardURL );

fprintf( fd, "auth amount %d\n",
    authResInfoP->authResPayload->authHeader.authAmt.amount );
fprintf( fd, "response code %s\n",
    authResInfoP->authResPayload->authHeader.respCode );

rc = PG_EXITRC_SET_INFOD; 9

} // end SET indicator
else {
    fprintf( fd, "Authorization Response !\n" );
    rc = PG_EXITRC_NOT_SUPPORTED;
}

```

Figure 139 (Part 5 of 11). Payment Gateway Legacy Application Sample Source Code

```

        fprintf( fd, "\nReturning from Xlate_AuthReqRes (%d)\n\n", rc );
        fflush( NULL ); // flush everything
        fclose( fd );
        return( rc );

} // Xlate_AuthReqRes
//-----
// THIS FUNCTION IS REQUIRED BY THE GATEWAY AND MUST NOT BE DELETED FROM
// THIS FILE
//-----
//
// Name: Xlate_CapReqRes
// Process Capture request and response transactions
//
// Context: AcqPayCtx
// CapRequestCtx = &(context->msgctx.u.capPairCtx.capReqCtx)
// Info Basket: CapResInfo
// Required fields for CapResInfo info basket:
//   capResSeq
//   next
//   value
//     transIDs
//     localID_C
//     xID
//     pReqDate
//     language
//     capResPayload
//     capCode
//     capAmt
//     currency
//     amount
//     amtExp10
//     captureControl (cannot be NULL, structure must be allocated)
//     messageIDs (cannot be NULL, structure must be allocated)
//
// For additional details about the context and the info basket,
// see ibmset.h
//-----
extern EXTTYPE
PG_EXITRC Xlate_CapReqRes( PG_INDICATOR      indicator,
                          AcqPayCtx *      context,
                          void **          dataptr,
                          short *          data_length,
                          long *           legacy_applid,
                          TRANS_RETURN_CODES gateway_rc)
{
    // Local variables

    CapRequestCtx      *capReqCtxP;
    RRTags              *capRRTagsP;
    CapSeq_             *capReqSeqP;
    ClearCapToken      *clearCapTokenP;
    CapResSeq_         **capResSeqPtr;
    CapResSeq_         *capResSeqP;
    CapResItem         *capResItemP;
    CapResInfo         *capResInfoP;
    CapResPayload      *capResPayloadP;
    CapPayload         capPayloadL;
    AuthResPayload     authResPayload;
    AuthHeader         authHeader;
    CapPayload         capPayload;
    MerchantID         merchantID;
    TransIDs           *transIDsP;
    int                 capAmtL;
    int                 capCurl;
    int                 capExpL;
    FILE                *fd;
    PG_EXITRC          rc;

```

Figure 139 (Part 6 of 11). Payment Gateway Legacy Application Sample Source Code

```

fd = fopen( LOG_FILE, "a" );
fprintf( fd, "\nEntering Xlate_CapReqRes ...\n" );

if ( indicator == PG_INDICATOR_REQUEST ) {
    fprintf( fd, "Capture Request !\n" );

    capReqCtxP = &(amp;context->msgctx.u.capPairCtx.capReqCtx);

    if ( capReqCtxP->capRRTags == NULL )
        fprintf( fd, "RRTags is null\n" );

    clearCapTokenP = (*capReqCtxP->clearCapTokSeq) -> value;
    capReqSeqP = *capReqCtxP->capSeq;
    capPayloadL = capReqSeqP->value->capPayload;

    capRRTagsP = capReqCtxP->capRRTags;
    merchantID = capRRTagsP->merTermIDs.merchantID;

    authResPayload = capPayloadL.CapPayload_authResPayload;
    authHeader = authResPayload.authHeader;

    capAmtL = capPayloadL.capReqAmt.amount;
    capExpL = capPayloadL.capReqAmt.amtExp10;
    capCurl = capPayloadL.capReqAmt.currency;

    fprintf( fd, "CapPayload bit mask is: %x\n", capPayloadL.bit_mask );
    fprintf( fd, "AuthResPayload bit mask is: %x\n", authResPayload.bit_mask );

    if ( clearCapTokenP->panToken->pan == NULL )
        fprintf( fd, "PanToken pointer is NULL\n" );
    else
        fprintf( fd, "PAN is %s\n", clearCapTokenP->panToken->pan );

    if ( clearCapTokenP->capTokenData == NULL )
        fprintf( fd, "capTokenData pointer is NULL\n" );
    else
        fprintf( fd, "auth amount is %d\n",
            clearCapTokenP->capTokenData->authAmt.amount );

    fprintf( fd, "CapPayload bit mask is %x\n", capPayloadL.bit_mask );
    fprintf( fd, "Requested Capture amount is %d\n", capAmtL );
    fprintf( fd, "Requested Capture currency is %d\n", capCurl );
    fprintf( fd, "Amount authorized is %d\n", authHeader.authAmt.amount );
    fprintf( fd, "Auth response code is %s\n", authHeader.respCode );
    fprintf( fd, "Merchant ID is %s\n", merchantID.u.MerchantID_printableString );

    // allocate storage for the CAPTURE RESPONSE structure CapResInfo
    capResInfoP = (CapResInfo*) calloc( 1, sizeof(CapResInfo) );

    capResInfoP->messageIDs = (MessageIDs*)
        calloc( 1, sizeof(MessageIDs) );

    capResInfoP->captureControl = (CaptureControl*)
        calloc( 1, sizeof(CaptureControl) );

    capResPayloadP = (CapResPayload*)
        calloc( 1, sizeof(CapResPayload) );

    capResSeqP = (CapResSeq_*) calloc( 1, sizeof(CapResSeq_) );

    capResSeqPtr = (CapResSeq_**) calloc( 1, sizeof(CapResSeq_*) );

    capResItemP = (CapResItem*) calloc( 1, sizeof(CapResItem) );

    transIDsP = (TransIDs*) calloc( 1, sizeof(TransIDs) );

    // set contents of SeqPtr to point to ResSeqP & set ptr to ResSeqPtr
    *capResSeqPtr = capResSeqP;
    capResInfoP->capResSeq = capResSeqPtr;

```

Figure 139 (Part 7 of 11). Payment Gateway Legacy Application Sample Source Code

```

        /*
        * Certain variables in the SET structure must be supplied in
        * order to prevent coredumps in the PGW.
        */

        memcpy(transIDsP,&capReqSeqP->value->transIDs,sizeof(TransIDs));
        capResPayloadP->batchID = NULL;

        // set capture code to successful; amount to amount requested

        capResPayloadP->capCode = success;
        capResPayloadP->capAmt.amount = capAmtL;
        capResPayloadP->capAmt.currency = capCurL;
        capResPayloadP->capAmt.amtExp10 = capExpL;

        fprintf( fd, "amount captured = %d\n",capResPayloadP->capAmt.amount );
        fprintf( fd, "capcode is %d success = 1\n",capResPayloadP->capCode );

        // set up the rest of the CapResInfo structure

        capResItemP->capResPayload = capResPayloadP;
        capResItemP->transIDs = transIDsP;
        capResSeqP-> value = capResItemP;

        // pass addr of CapResInfo structure back to PGW

        *dataptr = (void*) capResInfoP;
        rc = PG_EXITRC_SET_INF0B;

    } // end SET indicator

    else {
        rc = PG_EXITRC_NOT_SUPPORTED;
    }

    fprintf( fd, "\nReturning from Xlate_CapReqRes (%d)\n\n", rc );
    fflush( NULL );
    fclose( fd );
    return( rc );

} // Xlate_CapReqRes

//-----
// THIS FUNCTION IS REQUIRED BY THE GATEWAY AND MUST NOT BE DELETED FROM
// THIS FILE
//-----
//
// Name: Xlate_CredReqRes
// Process Credit request and response transactions
//
// Context: AcqPayCtx
// CreditRequestCtx = &(context->msgctx.u.creditPairCtx.creditReqCtx)
// Info Basket: CreditResInfo
// Required fields for CreditResInfo info basket:
// messageIDs (cannot be NULL, structure must be allocated)
// creditResSeq
// next
// value
// transIDs
// localID_C
// xID
// pReqDate
// language

```

Figure 139 (Part 8 of 11). Payment Gateway Legacy Application Sample Source Code

```

//          capRevOrCredResPayload
//          capRevOrCredCode
//          capRevOrCredActualAmt
//          currency
//          amount
//          amtExp10
//          captureControl
//
// For additional details about the context and the info basket,
// see ibmset.h
//-----
extern EXTTYPE
PG_EXITRC Xlate_CredReqRes( PG_INDICATOR      indicator,
                          AcqPayCtx *        context,
                          void **            dataptr,
                          short *            data_length,
                          long *             legacy_applid,
                          TRANS_RETURN_CODES gateway_rc)
{
    return( PG_EXITRC_NOT_SUPPORTED );
}

//-----
// THIS FUNCTION IS REQUIRED BY THE GATEWAY AND MUST NOT BE DELETED FROM
// THIS FILE
//-----
//
// Name: Xlate_BatchAdminReqRes
// Process Batch Administration
//
// Context: AcqPayCtx
// BatchAdminRequestCtx =
//          &(context->msgctx.u.batchAdminPairCtx.batchAdminReqCtx)
// Info Basket: BatchAdminResInfo
// Required fields for BatchAdminResInfo info basket:
//          batchID
//
// For additional details about the context and the info basket,
// see ibmset.h
//-----
extern EXTTYPE
PG_EXITRC Xlate_BatchAdminReqRes( PG_INDICATOR      indicator,
                                  AcqPayCtx *        context,
                                  void **            dataptr,
                                  short *            data_length,
                                  long *             legacy_applid,
                                  TRANS_RETURN_CODES gateway_rc)
{
    return( PG_EXITRC_NOT_SUPPORTED );
}

//-----
// THIS FUNCTION IS REQUIRED BY THE GATEWAY AND MUST NOT BE DELETED FROM
// THIS FILE
//-----
//
// Name: Xlate_RevReqRes
// Process Authorization Reversal request and response transactions
//
// Context: AcqPayCtx
// Info Basket: to be determined
//-----
extern EXTTYPE
PG_EXITRC Xlate_RevReqRes( PG_INDICATOR      indicator,
                          AcqPayCtx *        context,
                          void **            dataptr,
                          short *            data_length,
                          long *             legacy_applid,
                          TRANS_RETURN_CODES gateway_rc)
{
    return( PG_EXITRC_NOT_SUPPORTED );
}

```

Figure 139 (Part 9 of 11). Payment Gateway Legacy Application Sample Source Code


```

//-----
// THIS FUNCTION IS REQUIRED BY THE GATEWAY AND MUST NOT BE DELETED FROM
// THIS FILE
//-----
//
// Name: Xlate_CapRevReqRes
// Process Capture Reversal request and response transactions
//
// Context: AcqPayCtx
// Info Basket: to be determined
//-----
extern EXTTYPE
PG_EXITRC Xlate_CapRevReqRes( PG_INDICATOR      indicator,
                             AcqPayCtx *      context,
                             void **          dataptr,
                             short *          data_length,
                             long *           legacy_applid,
                             TRANS_RETURN_CODES gateway_rc)
{
    return( PG_EXITRC_NOT_SUPPORTED );
}

//-----
// THIS FUNCTION IS REQUIRED BY THE GATEWAY AND MUST NOT BE DELETED FROM
// THIS FILE
//-----
//
// Name: Xlate_CredRevReqRes
// Process Credit Reversal request and response transactions
//
// Context: AcqPayCtx
// Info Basket: to be determined
//-----
extern EXTTYPE
PG_EXITRC Xlate_CredRevReqRes( PG_INDICATOR      indicator,
                              AcqPayCtx *      context,
                              void **          dataptr,
                              short *          data_length,
                              long *           legacy_applid,
                              TRANS_RETURN_CODES gateway_rc)
{
    return( PG_EXITRC_NOT_SUPPORTED );
}

//-----
// THIS FUNCTION IS REQUIRED BY THE GATEWAY AND MUST NOT BE DELETED FROM
// THIS FILE
//-----
//
// Name: Xlate_CertReqRes
// Process Gateway Certificate requests
//
// Context: AcqPayCtx
// Info Basket: CertResInfo
// Required fields for CertResInfo info basket:
// returnCertToMerch
//
// Operation:
// Unless this function is customized, all requests for gateway certificates
// from a merchant with valid certificates will be honored.
//
// For additional details about the context, see ibmset.h
// For additional details about the info basket, see IBM_PayGate.H
//-----
extern EXTTYPE
PG_EXITRC Xlate_CertReqRes( PG_INDICATOR      indicator,
                           AcqPayCtx *      context,
                           void **          dataptr,
                           short *          data_length,
                           long *           legacy_applid,
                           TRANS_RETURN_CODES gateway_rc)

```

Figure 139 (Part 10 of 11). Payment Gateway Legacy Application Sample Source Code

```

{
    FILE          *fd;

    fd = fopen( LOG_FILE, "a" );
    fprintf( fd, "Entering Xlate_CertReqRes ...\n" );
    CertResInfo* infoB = ( CertResInfo *) calloc (1, sizeof(CertResInfo));
    if( !infoB ) {
        fprintf( fd, "Xlate_CertReqRes: Allocation failure - Abort\n" );
        fprintf( fd, "Returning from Xlate_CertReqRes with PG_EXITRC_INTERNAL\n" );
        fflush( NULL );
        fclose( fd );
        return PG_EXITRC_INTERNAL;
    }

    infoB->returnCertToMerch = PGTrue;
    *dataptr = (void* ) infoB;
    fprintf( fd, "Returning from Xlate_CertReqRes with PG_EXITRC_SET_INFOB\n" );
    fflush( NULL );
    fclose( fd );
    return PG_EXITRC_SET_INFOB;
}

```

Figure 139 (Part 11 of 11). Payment Gateway Legacy Application Sample Source Code

Note:

1 The exit is a good point to write additional logging and debugging information. In our case we simply wrote diagnostic messages to a log file. In a real environment you may want to do something more sophisticated, separating information that must be kept for auditing purposes from program diagnostics, for example.

2 This is the start of one of the function calls (or entry points) within the exit code, Xlate_AuthReqRes. As for all of the exit functions, this function is called twice:

- When an authorization request arrives and is routed to a legacy application
- When the authorization response returns from the legacy application

All of the functions have similar behavior. They differ only in the type of request that they handle.

On the initial input the information from the SET request is provided in a data structure of type AcqPayCtx. The exit is provided with a pointer field (variable name dataptr in this example), which points to a buffer that is used to pass data in and out of the exit:

- When the payment gateway first calls the exit the pointer is null.
- When the exit passes control to the legacy application, the buffer contains the request format expected by the legacy application.
- When the exit is called again, after the legacy application has responded, the buffer contains the response from the legacy application.
- When the exit passes the response back to the payment gateway, the buffer contains a SET response structure (a data structure of type AuthResInfo, defined in header file ibmset.h).

3 The exit receives an indicator that tells it at which point it is being invoked (as a request from the payment gateway, or as a response (in

ASCII or EBCDIC format) from the legacy application, or as the result of an error). Our sample exit does not actually call a legacy application, so the indicator will always be PG_INDICATOR_REQUEST.

4 Here we see the exit starting to build the data buffer that it uses for output. In our simple case the output is not sent to a legacy application, but instead is sent straight back to the payment gateway as a SET response.

5 Some of the information in the response buffer is converted into information fields for use by the merchant.

6 Our sample exit is coded to refuse authorization for any credit card with a personal account number (PAN) that starts with a “1.” This section of the exit fills in the fields for the rejection response, including the reason text that is forwarded by the merchant to the cardholder (see 7.3.3.2, “Failed Order Macro” on page 133 for an example of how Net.Commerce SET Extension uses this information).

7 If the PAN does not start with a 1, the exit authorizes the payment (this is, indeed, a friendly bank!). This section of code copies the requested authorization amount into the response structure and sets the response code to 000.

8 The exit sets the returned data pointer to refer to the buffer it has built. In this case it does not need to fill in the data length, because the payment gateway knows how big the AuthResInfo structure is. If the exit was passing data to, or receiving it from, a legacy application, the data length field would need to be specified.

9 The function returns with a PG_EXITRC_SET_INFOB return code that indicates that it has completed processing and the request does not have to be passed on to a legacy application.

Part 3. Installing and Configuring Net.Commerce

Chapter 10. Installing Net.Commerce

In order to show examples of the SET payment processes we created an online mall and store using Net.Commerce. Although it is not directly part of the SET payment process, the IBM SET Merchant code runs as an extension to the Net.Commerce server. We therefore describe the installation of Net.Commerce in this chapter and we describe the customization of the online store in Chapter 11, "Creating an Online Store with Net.Commerce" on page 241.

10.1 Preparing the System

Before installing the Net.Commerce product, you must decide if it will be installed on one or more computers. If you decide to install on one computer, it will contain the Net.Commerce Server and Net.Commerce database code. If you decide to install on more than one computer, one will contain the Net.Commerce server code and the other the Net.Commerce database code. See the *Net.Commerce Installation and Operations Guide* for more information about the advantages of installing the Net.Commerce on one or more than one computer.

10.1.1 AIX System

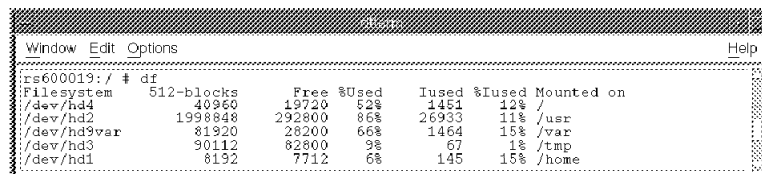
Follow these steps to ensure a successful installation:

- Check to see if the amount of free space in the /home directory is sufficient.
- Check to see if the amount of paging space is sufficient.
- Create a DB2 group.
- Create a DB2 instance owner.

10.1.1.1 Checking the Size of the /home Directory

The /home directory is used for the database; its minimum size must be 40,960 blocks of free space. If you experience problems creating your database using Net.Commerce, check the amount of free space in the /home directory as follows:

1. On a command line, enter df.
2. Check the Free field to see if you have at least 40,960 blocks in the /home directory. If you do, go to 10.1.1.3, "Checking the Paging Space Size" on page 217. Otherwise, go to 10.1.1.2, "Increasing the Size of the /home Directory" on page 216.



```
rs600019:/ # df
Filesystem      512-blocks    Free %Used    Used %Used Mounted on
/dev/hd4         40960      19720   52%     1451   12% /
/dev/hd2      1998948      292800   15%     26933   11% /usr
/dev/hd9war       81920      28200   34%     1464   15% /var
/dev/hd3          90112      82800   92%         67    1% /tmp
/dev/hd1          8192       7712    6%       145   15% /home
```

Figure 140. Checking the Size of the /home Directory

10.1.1.2 Increasing the Size of the /home Directory

You must be logged on as the root user to increase the size of the /home directory; you can use the command line or the graphical interface:

Using the Command Line

On the command line, type `chfs -a size='xxxx' /home`

where:

`xxxx = <old-size>+40960-<old-free-size>`

`<old-size>` is the current size of the directory and

`<old-free-size>` is the current free space of this directory.

Using the Graphical Interface

1. On the command line, type `smit`.
2. Select **System Storage Management (Physical & Logical Storage)** from the System Management main menu.
3. Select **File Systems**.
4. Select **Add / Change / Show / Delete File Systems**.
5. Select **Journaled File Systems**.
6. Select **Change / Show Characteristics of a Journaled File System**.
7. Select **/home**.
8. You will see a screen like the one shown in Figure 141.

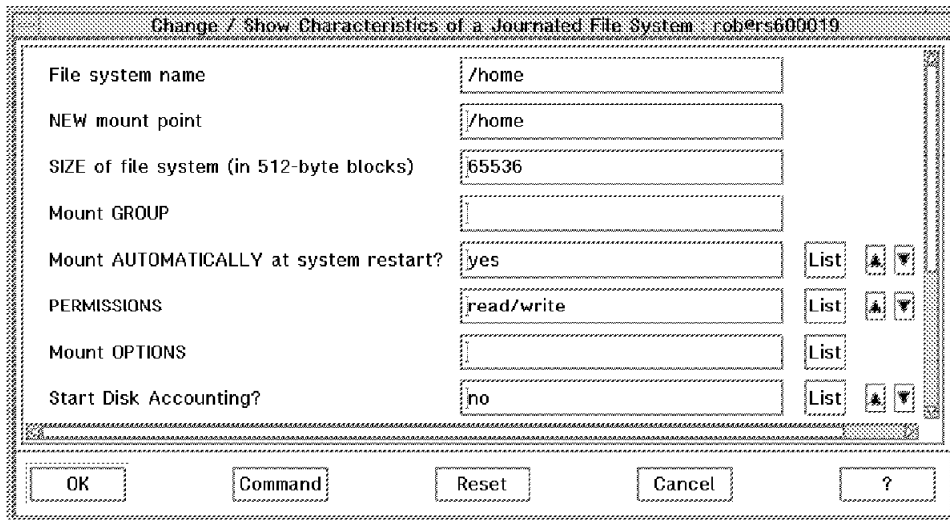


Figure 141. Increasing the Size of the /home Directory

9. Calculate the size that you need for your /home directory by using this formula:

`size = <old-size>+40960-<old-free-size>`

where:

`<old-size>` is the current size of the directory

`<old-free-size>` is the current free space of this directory

10. Type the number that you calculated in the SIZE of file system (in 512-byte blocks) field.
11. Press Enter.

When the top window changes from the status of Running to OK, the command has been successful.

12. Press F12 or click on **Cancel** several times until the System Management main menu disappears.

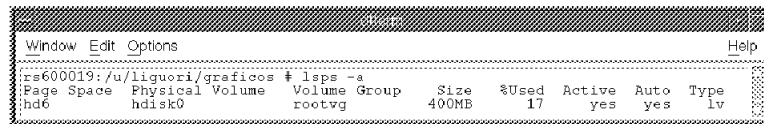
You should check the size of the paging space.

10.1.1.3 Checking the Paging Space Size

You must have at least 64 MB of paging space. To check the paging space size, you can use the command line or the graphical interface:

Using the Command Line

1. On the command line, type `lsp -a`.



```
rs600019:/u/liguori/graficos # lsp -a
Page Space  Physical Volume  Volume Group  Size  %Used  Active  Auto  Type
hd6         hdisk0             rootvg        400MB  17     yes    yes   lv
```

Figure 142. Checking the Paging Space Size

2. Check the Size field to be sure that you have at least 64 MB of space. If you do, go to 10.1.1.5, "Creating a DB2 System Administration Group" on page 220. If you don't, go to 10.1.1.4, "Increasing the Paging Space Size" on page 218.

Using the Graphical Interface

1. On the command line, type `smit`.
2. Select **System Storage Management (Physical & Logical Storage)** from the System Management main menu.
3. Select **Logical Volume Manager**.
4. Select **Paging Space**.
5. Select **List All Paging Spaces**.
6. You will see a screen like the one shown in Figure 143 on page 218.

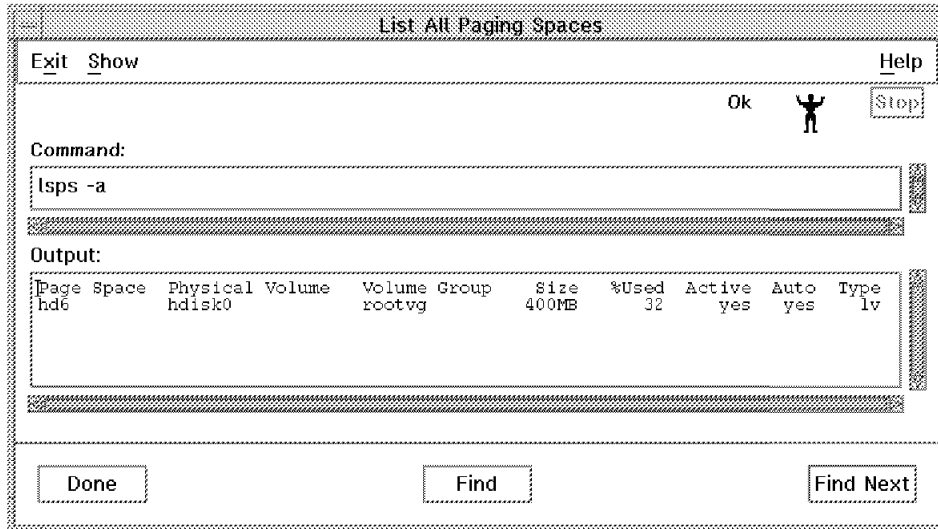


Figure 143. Checking the Paging Space Size

Check the Size field if you have at least 64 MB of space. If you do, go to 10.1.1.5, “Creating a DB2 System Administration Group” on page 220. If you don’t, go to 10.1.1.4, “Increasing the Paging Space Size.”

7. Press F12 or click **Cancel** several times until the System Management main menu disappears.

10.1.1.4 Increasing the Paging Space Size

To increase the size of the paging space, you can do one or both of the following:

- Increase the size of an existing paging space.
- Create a new paging space.

Increasing the Size of an Existing Paging Space

1. On the command line, type `smit`.
2. Select **System Storage Management (Physical & Logical Storage)** from the System Management main menu.
3. Select **Logical Volume Manager**.
4. Select **Paging Space**.
5. Select **Change / Show Characteristics of a Paging Space**.
6. Select the paging space that you want to activate.
7. Calculate the number of additional partitions. Subtract the number of the total size of all active partitions from 64, and divide by 4.
8. You will see a screen like the one shown in Figure 144 on page 219.

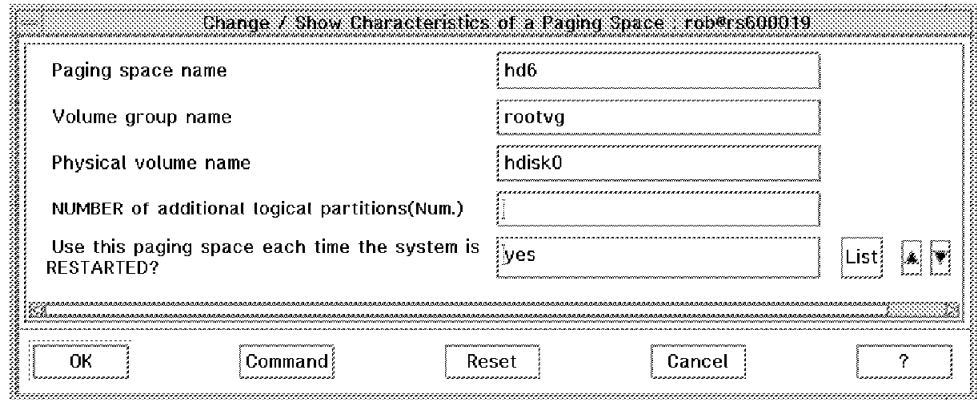


Figure 144. Increasing the Size of an Existing Paging Space

Type the result that you got in the last step in the NUMBER of additional logical partitions field.

9. Press Enter.

When the top window status changes from Running to OK, the command has been successful.

10. Press F12 or click on **Cancel** several times until the System Management main menu disappears.

Creating a New Paging Space

1. On the command line, type smit.
2. Select **System Storage Management (Physical & Logical Storage)** from the System Management main menu.
3. Select **Logical Volume Manager**.
4. Select **Paging Space**.
5. Select **Add Another Paging Space**.
6. Select the **Volume group name**.
7. You will see a screen like the one shown in Figure 145.

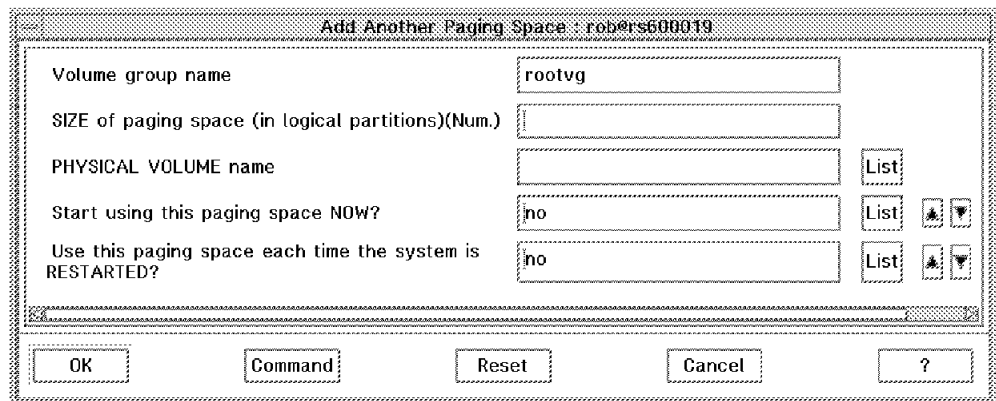


Figure 145. Creating a New Paging Space

8. Type the number of partitions in the SIZE of paging space (in logical partitions) field. These are 4 MB partitions, so to calculate the size of your

- new paging space, subtract the existing paging space total size from 64 and divide the result by 4.
9. Select **yes** in the Start using this paging space NOW? field.
 10. Select **yes** in the Use this paging space each time the system is RESTARTED? field.
 11. Press Enter. When the top window status changes from Running to OK, the command has been successful.
 12. Press F12 or click on **Cancel** several times until the System Management main menu disappears.

10.1.1.5 Creating a DB2 System Administration Group

You must create DB2 groups, called sysadm1 and dbadmin1, to administer the Net.Commerce database. To create a DB2 system administration group, called sysadm1, follow these steps.

Using Command Line: On the command line, type `mkgroup -'A' sysadm1`.

Using the Graphical Interface

1. On the command line, type `smit`.
2. Select **Security & Users** from the System Management main menu.
3. Select **Groups**.
4. Select **Add a Group**.
5. You will see a screen like the one shown in Figure 146.

The screenshot shows a graphical user interface window titled "Add a Group : rob@rs600019". The window contains several input fields and buttons. The first field is "Group NAME" with the text "sysadm1" entered. Below it is "ADMINISTRATIVE group?" with "false" entered and a "List" button to its right. The next field is "Group ID(Num.)" which is empty. Below that is "USER list" with an empty field and a "List" button to its right. The final field is "ADMINISTRATOR list" with an empty field and a "List" button to its right. At the bottom of the window, there are five buttons: "OK", "Command", "Reset", "Cancel", and "?".

Figure 146. Creating a DB2 Group

6. Type `sysadm1` in the Group NAME field.
7. Press Enter. When the top window status changes from Running to OK, the command has been successful.
8. Press F12 or click on **Cancel** several times until the System Management main menu disappears.

Using the Command Line: To create another group, called `dbadmin1`, type `mkgroup -'A' dbadmin1` on the command line.

Using the Graphical Interface

1. Follow the steps 1 through 4 from 10.1.1.5, "Creating a DB2 System Administration Group."
2. Type dbadmin1 in the Group NAME field.
3. Press Enter. When the top window status changes from Running to OK, the command has been successful.
4. Press F12 or click on **Cancel** several times until the System Management main menu disappears.

10.1.1.6 Creating a DB2 Instance Owner User ID

You must create a DB2 user ID to administer the Net.Commerce database. To create a DB2 instance owner user ID, called inst1, follow these steps:

1. On the command line, type smit.
2. Select **Security & Users** from the System Management main menu.
3. Select **Users**.
4. Select **Add a User**.
5. You will see a screen like the one shown in Figure 147.

Field	Value	Buttons
* User NAME	inst1	
User ID(Num.)		
ADMINISTRATIVE USER?	true	List, ▲, ▼
Primary GROUP	sysadm1	List
Group SET	sysadm1, dbadmin1	List
ADMINISTRATIVE GROUPS		List
Another user can SU TO USER?	true	List, ▲, ▼
SU GROUPS		List
HOME directory	/home/inst1	
Initial PROGRAM		

Buttons: OK, Command, Reset, Cancel, ?

Figure 147. Creating a DB2 User ID

6. Type inst1 in the User NAME field.
7. Type sysadm1 in the Primary GROUP field.
8. Type sysadm1, dbadmin1 in the Group SET field.
9. Type /home/inst1 in the HOME directory field.
10. Press Enter. When the top window status changes from Running to OK, the command has been successful.
11. Press F12 or click on **Cancel** several times until the System Management main menu disappears.

10.1.1.7 Changing the Password

To change the password for the user ID inst1, do the following steps:

1. Select **Security & Users** from the System Management main menu.
2. Select **Users**.
3. Select **Change a User's Password**.
4. Type inst1 in the User NAME field. Press Enter.
5. Type a password in the New password: prompt. Press Enter.
6. Type the same password again in the Re-enter the new password prompt. Press Enter.
7. Press F12 or click on **Cancel** several times until the System Management main menu disappears.

See *Net.Commerce Installation and Operations Guide* for further details about the above-listed procedures.

10.1.2 Windows NT

Net.Commerce for Windows NT requires a platform that meets minimum hardware requirements and software requirements. The following lists reflect the minimum configurations recommended by IBM.

10.1.2.1 Hardware Requirements

- Pentium P166 IBM-compatible Personal Computer or higher, with a graphics-capable monitor
- 64 MB minimum of random access memory (RAM)
- 100 MB minimum of free disk space
- CD-ROM drive
- Mouse or compatible pointing device
- A local area network (LAN) adapter that is supported by the TCP/IP protocol

10.1.2.2 Software Requirements

- Windows NT Server V3.51 or later
- DB2 V2.1.2 or later
- IBM Internet Connection Secure Server (ICSS) V4.1.1
- A Web browser, such as Netscape Navigator 3.0, that is:
 - SSL-compliant
 - Java-enabled and JavaScript-enabled
 - Capable of supporting tables and frames
 - Capable of handling cookies

10.2 Installing Net.Commerce

Once DB2 is up and running, you can go ahead and install Net.Commerce itself.

10.2.1 AIX System

To install the Net.Commerce system, follow these steps:

1. Log on as the root user.
2. Insert the CD-ROM into the CD-ROM drive.
3. On the command line, enter `smit`.
4. Select **Software Installation and Maintenance** from the System Management main menu.
5. Select **Install and Update Software**.
6. Select **Install and Update from LATEST Available Software**.
7. In the INPUT device / directory for software files, type: `/dev/cd0`
8. Click on **OK**.
9. A confirmation message appears. Click on **OK**.
10. When the top window status changes from Running to OK, the installation has been completed. To ensure that the installation has been successful, scroll the Command Status window to the Installation section, and check to see if it shows a status of Success and Already installed.
11. Click on **Done**.
12. Select **Exit SMIT** or press F10 to end the installation process.

You have now installed Net.Commerce System.

10.2.2 Windows NT

Net.Commerce for Windows NT includes all the components you need, including those that are also available as separate products (DB2 and the Internet Connection Secure Server).

The graphical user interface (GUI) for Windows NT V3.51 and Windows NT V4.0 are different. The installation instructions in this chapter describe how to install the Net.Commerce System on Windows NT V3.51 only.

Important

Prior to installing the Net.Commerce System, review the README file for any late changes to procedures that have been described in this book. The README file is in the root directory on the Net.Commerce CD-ROM.

10.2.2.1 Pre-Installation Procedures

Before installing IBM Net.Commerce on Windows NT, you must create a new Windows NT user ID to be used in some of the remaining installation steps.

Creating a Windows NT User ID

1. In the Welcome window, type your administrator user ID and password.
2. Double-click on the **Administrative Tools** icon that is located on your desktop.

3. Double-click on the **User Manager** icon. The User Manager window will appear.
4. Select **User**, and then **New User** from the menu bar. The New User window will appear.
5. Type the new user ID that you want to create and its password.
6. Click on **Groups**. The Group Memberships window will appear.
7. Select **Administrators** from the Not member of field.
8. Click on **Add** in the Group Memberships window. Note that Administrator moves from the Not member of field to the Member of field.
9. Click on **OK**. The New User window will reappear.
10. Click on **Add**. All the fields in the New User window will now be blank.
11. Click on **Close**. Note that your new user ID will now appear in the User Manager window.
12. To close the User Manager window, double-click on the small icon in the top left corner.
13. To log off the administrator user ID, first activate the Program Manager window. Then click **File** on the menu bar, and **Logoff** on the pull-down menu. Windows NT will present the Welcome window.
14. To log on to the user ID that you have created, type the ID and password in the appropriate fields and click on **OK**.

You are now ready to install the Net.Commerce System.

10.2.2.2 Installing the Net.Commerce System

Follow these steps to install the Net.Commerce System:

1. Log on to the administrator user ID.
2. Put the Net.Commerce Installation CD-ROM into the CD-ROM drive and on the command line, type:

```
<b>:\NetCommerce
```


where is your CD-ROM drive.
3. To run the setup.exe, type SETUP.

10.2.2.3 Automating Server Startup

After you have installed the Net.Commerce System, you must:

- Ensure that the Internet Connection Secure Server starts automatically.
- Ensure that the database starts automatically.

Starting Internet Connection Secure Server Automatically: To automatically have the Internet Connection Secure Server start whenever the Net.Commerce System starts, follow these steps:

1. From the Program Manager, open the **Main** folder, then double-click on the **Control Panel** icon.
2. Double-click on the **Services** icon.
3. In the Services window, select **IBM Internet Connection Server** and click on **Start up**.

4. In the Services window, select **Automatic** for a Startup Type. Select the **System Account** for a Log On As and click on **OK**.
5. In the Services window, click on **Start**. The IBM Internet Connection Server will appear with a status of Started and a startup of Automatic.
6. Close the Services window, the Control Panel group, and the Main folder by double-clicking on the **System Menu** icon in the upper left corner of each window.

Starting DB2 Security Server Automatically: To automatically have the DB2 Security Server start whenever the Net.Commerce System starts, follow these steps:

1. From the Program Manager, open the **Main** folder, then double-click on the **Control Panel** icon.
2. Double-click on the **Services** icon.
3. In the Services window, select **DB2 Security Server** and click on **Startup**.
4. In the Services window, select **Automatic** for a Startup Type. Select the **System Account** for a Log On As and click on **OK**.
5. In the Services window, click on **Start**. The DB2 Security Server will appear with a Status of Started and a Startup of Automatic.
6. Close the Services window, the Control Panel group, and the Main folder by double-clicking on the **System Menu** icon in the upper left corner of each window.

Creating a DB2 Instance Owner: The following steps for creating a DB2 instance owner should be done as part of the installation procedures:

1. Double-click on the **DB2 for Windows NT** icon.
2. Double-click in the **DB2 Command Window**.
3. Type db2ilist.
4. If your Windows NT user name does not show up, follow these steps:
 - a. Type db2icrt <instance_owner_user_ID>.
 - b. To ensure that your new instance owner has been created, type db2ilist

Starting DATABASE 2 Automatically: To automatically have the DATABASE 2 start whenever the Net.Commerce System starts, follow these steps:

1. From the Program Manager, open the **Main** folder, then double-click on the **Control Panel** icon.
2. Double-click on the **Services** icon.
3. In the Services window, select **DB2-<instance_owner_userID>** and click on **Startup**.
4. In the Services window, select **Automatic** for a Startup Type. Select the **System Account** for a Log On As, and click on **OK**.
5. In the Services window, click on **Start**. The DB2-<instance_owner_userID> will appear with a status of Started and a startup of Automatic.
6. Close the Services window, the Control Panel group, and the Main folder by double-clicking on the **System Menu** icon in the upper left corner of each window.

10.3 Post-Installation Steps

After completing the installation there are a number of other steps before you can go ahead with building an online store on your Net.Commerce system.

10.3.1 Building Security Key Rings

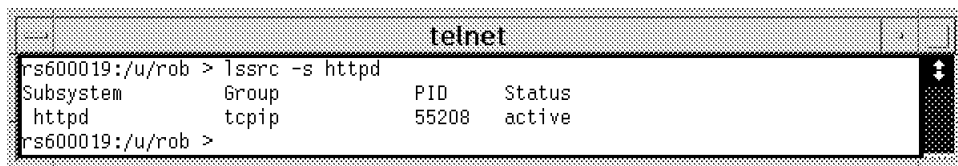
When you go online to sell products or services, the Net.Commerce server will be handling sensitive user information. SET will protect the portion of this information involved in credit card payments, but that is not the only data that needs to be protected. For example, you need your customers to enter name and address data, and you would like them to enter additional data, such as their earnings, age, likes and dislikes, etc. Net.Commerce uses Secure Sockets Layer (SSL) to protect this kind of information and to assure the user of the authenticity of the server site. (SSL is described in Appendix C, "Secure Sockets Layer" on page 305.)

SSL requires that the server has a public key pair and a certificate, so that the browser can check its authenticity. IBM Internet Connection Secure Server provides facilities for generating key pairs and requesting certificates. You can create a self-signed certificate for demo purposes, or request a secure server certificate signed by a recognized certifying authority. Self-signed keys are not secure. They should *not* be used in customer transaction situations. Use the self-signed keys only as we did, for test and development systems.

10.3.1.1 Creating a Self-Signed Certificate

To create a self-signed security server certificate, follow these steps:

1. Check to see if your IBM Internet Connection Secure Server is running. On AIX you do this by typing: `lssrc -s httpd`. You should see that the httpd subsystem is active, as shown in Figure 148. On NT, check that it is running by selecting **Control Panel** from the Start menu and then double-clicking on the **Services** icon.



```

telnet
rs600019:/u/rob > lssrc -s httpd
Subsystem      Group          PID    Status
httpd          tcpip         55208  active
rs600019:/u/rob >

```

Figure 148. Checking If httpd Is Running

If httpd is not running, on AIX type: `startsrc -s httpd` to start the daemon. On NT, click on **Start** in the Services list dialog.

2. Start your Web browser and disable all caching and any kind of proxy or SOCKS server.
3. Enter the following URL to access your IBM Internet Connection Secure Server Configuration page:

`http://<your_node_name>`

where `<your_node_name>` is a fully qualified server name, such as `rs600020.itso.ral.ibm.com`.

4. Click on **Configuration and Administration Forms**.

5. Type the user name and password. If you did not change the user name and password after installing the IBM Internet Connection Secure Server, the default user name is webadmin and the default password is webibm. Click on **OK**.
6. From the main Configuration Web page, scroll through the form and click on **Create Keys**.
7. Select a certificate type of **Other** and click on **Apply**.



Figure 149. Creating Keys

8. Type the key name of servkey and the key ring file name of /usr/lpp/internet/server_root/serv.kyr. If you want, you can choose different names for the key and the key ring file.
9. Change the size of the server key ring to the highest key setting that is available.

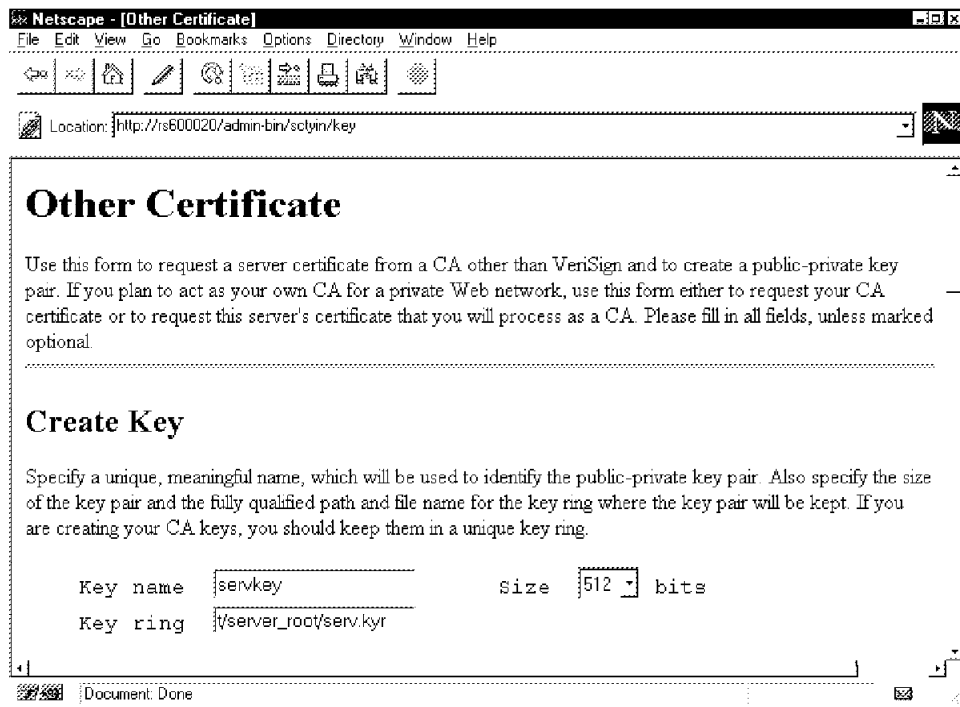


Figure 150. Creating a Key

10. Type a key ring password of your choice. This password will be used when you change the default key in the key ring, or when you receive certificates into that key ring.
11. Check the **Automatic login** box.

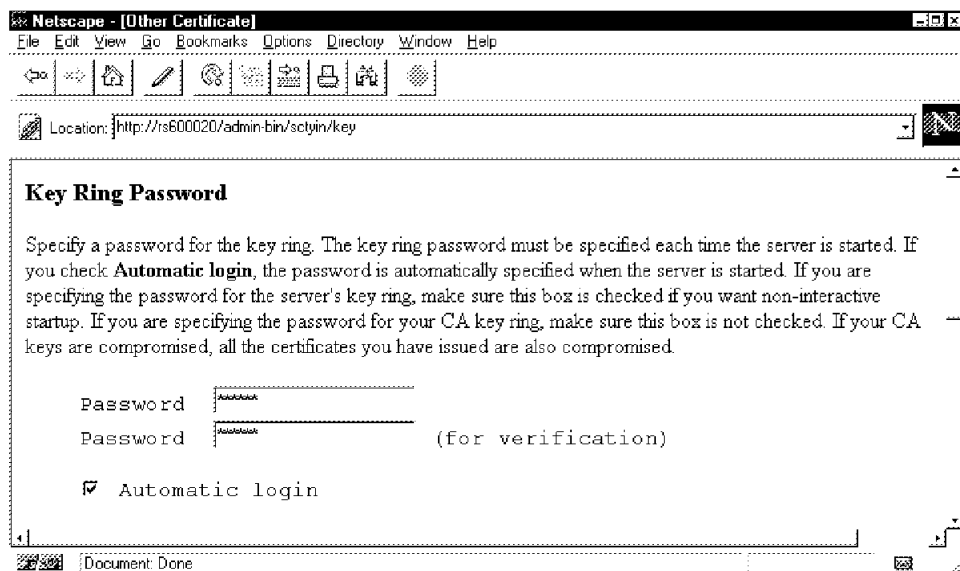


Figure 151. Creating a Key

12. In the fields that are provided, type the name of your server and address of the organization on which this server resides. Use a fully qualified server name of <your_node_name>, such as rs600020.itso.ral.ibm.com. Be sure that all the fields, even the optional ones, have been completed.

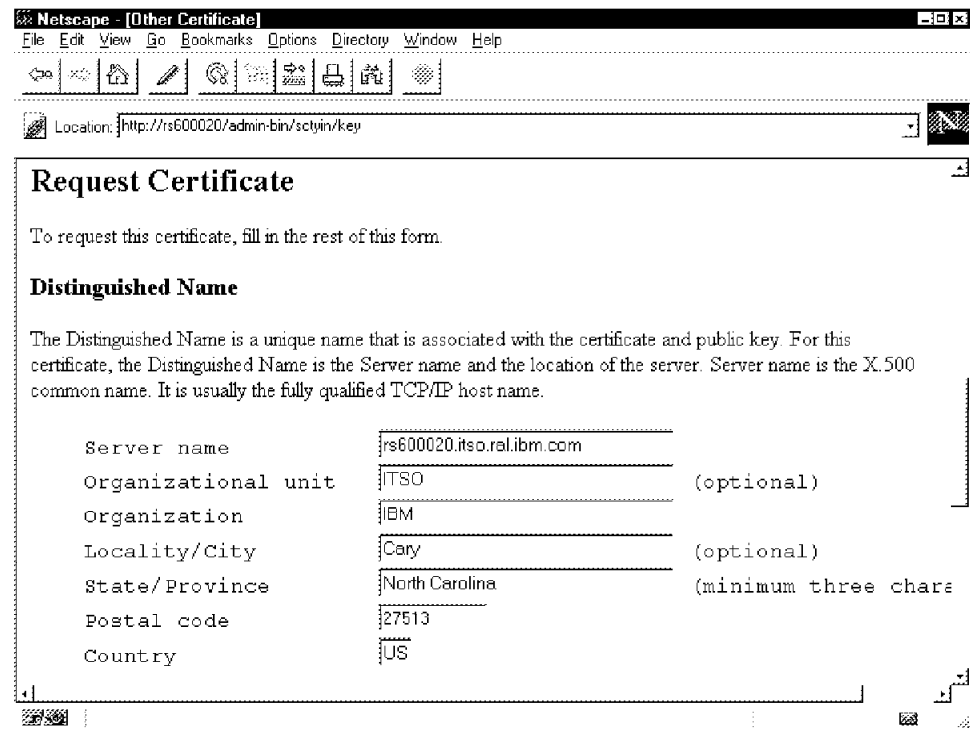


Figure 152. Creating a Key

13. Click on **Don't Mail**.
14. In the Save certificate request to file field, type `/usr/lpp/internet/server_root/servreq.txt` and click on **Apply**. If you want, you may choose a different name for the requested certificate.

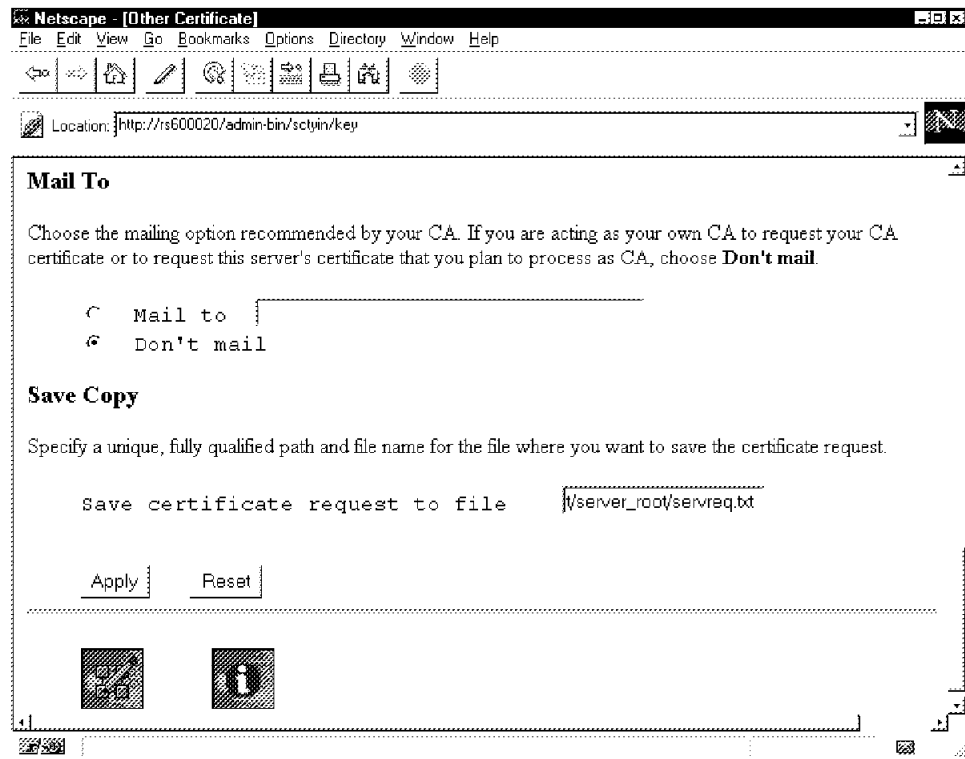


Figure 153. Creating a Key

15. You will see a confirmation screen, indicating that you have successfully created your public/private key pair and certificate request.

10.3.1.2 Changing the Current Ring

Next, you must tell the Web server to start to use your new key ring.

1. Return to the Configuration and Administration Forms page by clicking **Configuration Page** at the bottom of the form.
2. Scroll down the list and select **Security Configuration**.
3. Select **serv.kyr** in the Key Rings field.
4. Click on **Set selected key ring as current key ring**.
5. Click on **Apply**.

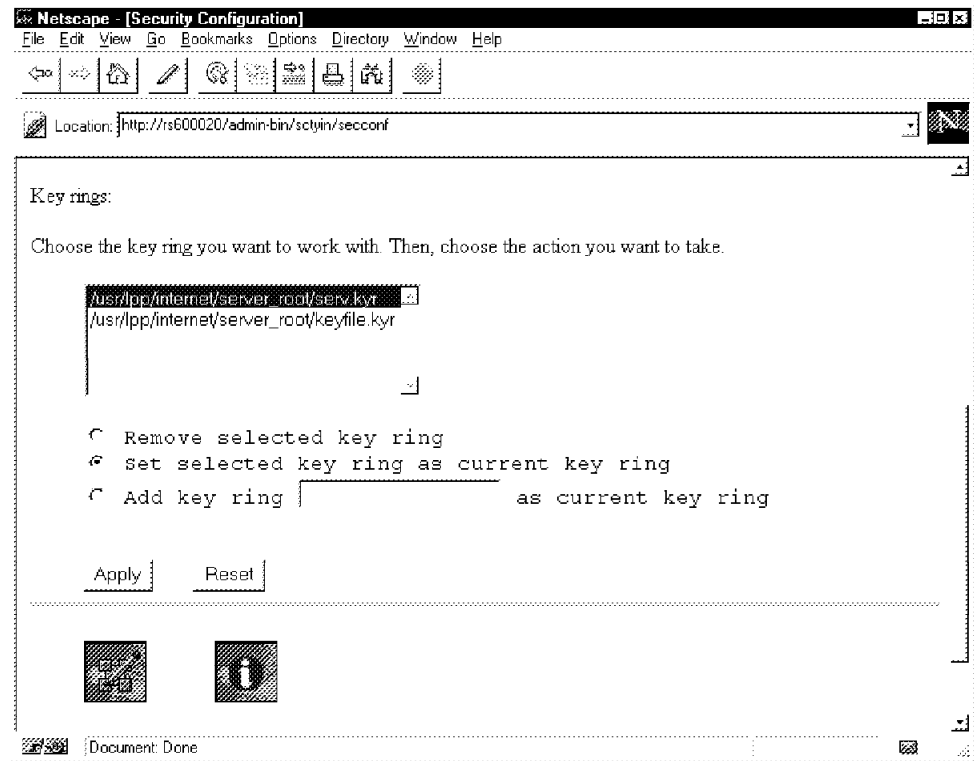


Figure 154. Changing the Current Ring

10.3.1.3 Receiving the Certificate for The Server

The private key in your key ring file is not usable until it is united with the matching public key certificate.

1. Return to the Configuration and Administration Forms page by clicking on **Configuration Page**.
2. From the main Configuration Web page, scroll through the form and click on **Receive Certificate**. The Receive Certificate page is displayed as in Figure 155 on page 232.
3. Type the fields as follows:
 - a. Type `/usr/lpp/internet/server_root/servreq.txt` as the name of the file that contains the certificate.
 - b. Type `/usr/lpp/internet/server_root/serv.kyr` as the key ring.
 - c. Type the password that you used to create the server key ring above.
 - d. Click on **Apply**. You will see another confirmation screen, indicating that the certificate has been successfully received.

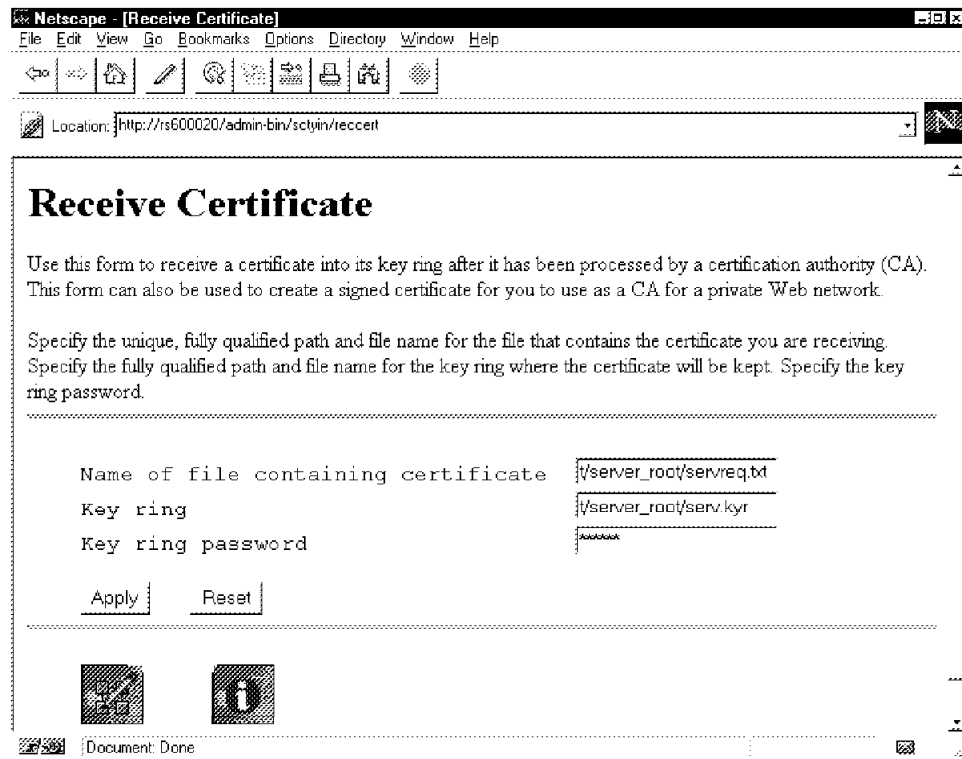


Figure 155. Receiving the Certificate

4. Return to the Configuration and Administration Forms page by clicking on **Configuration Page**.
5. Scroll through the page and select **Key Management**.
6. Type the key ring password as shown in Figure 156 on page 233.
7. Select **Designate Trusted Root Keys**.

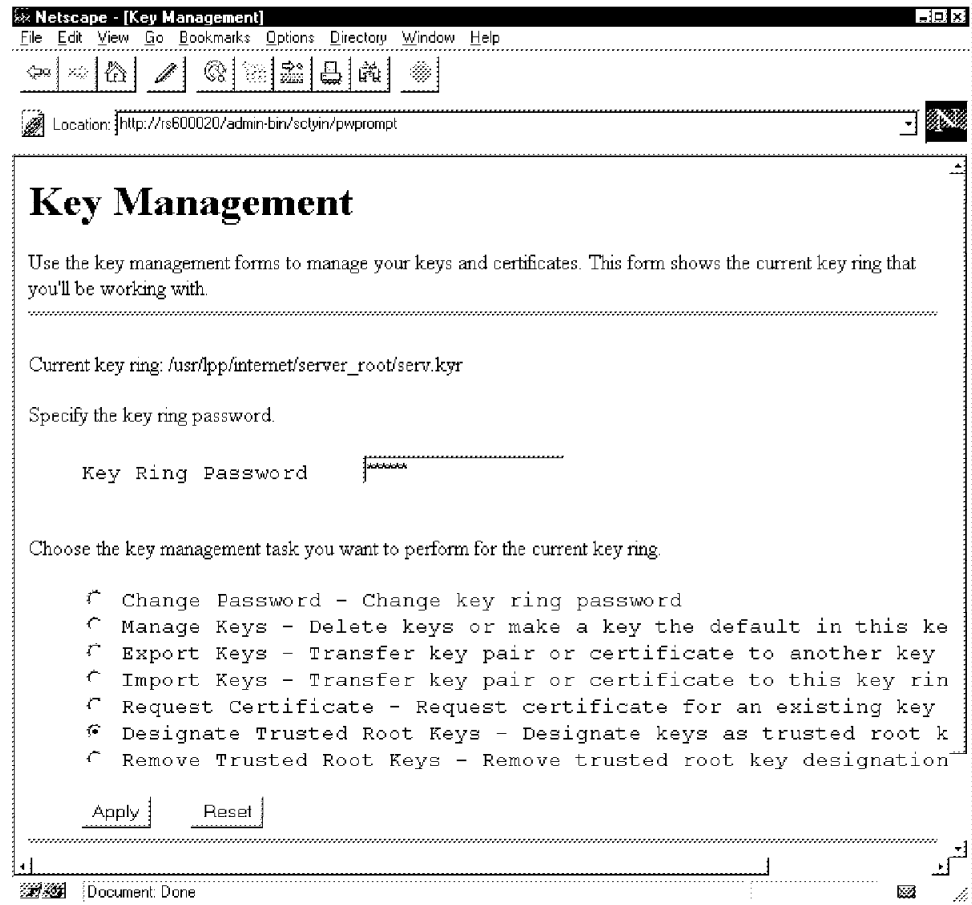


Figure 156. Receiving the Certificate

8. Click on **Apply**. A new page is displayed as in Figure 157 on page 234.
9. You should see the servkey name in the list. Click on **Apply**.

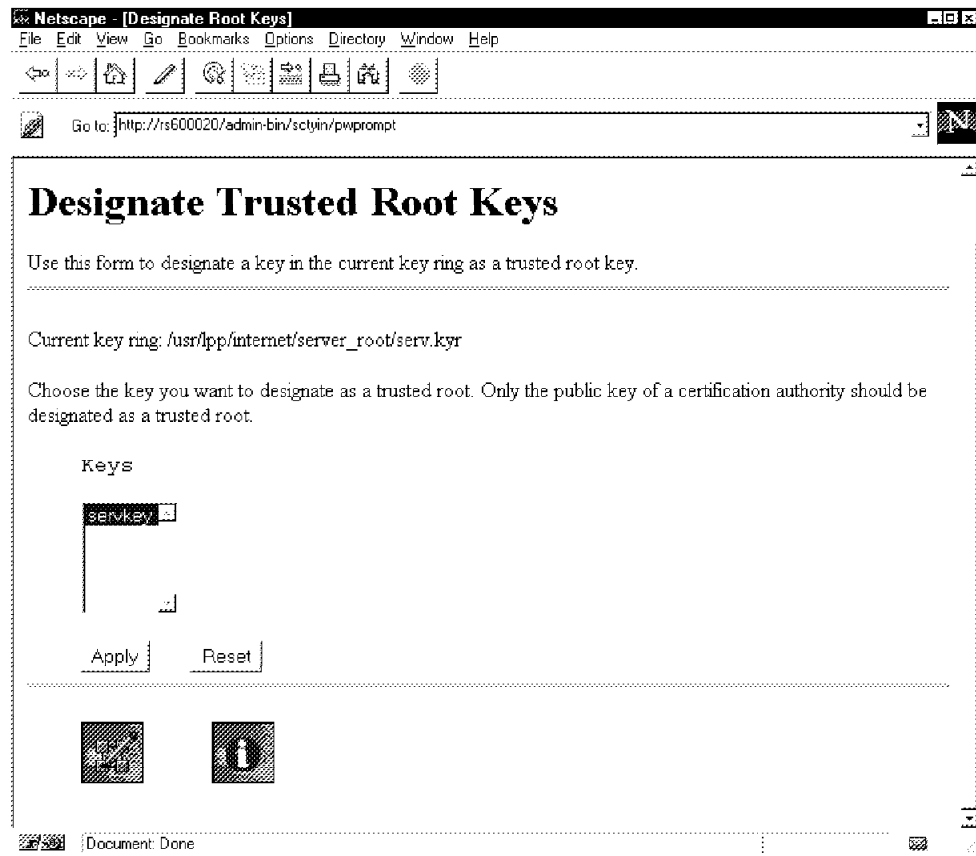


Figure 157. Receiving the Certificate

10. Another confirmation page is displayed. Select **Configuration Page** to return to the main Configuration Web page.
11. Stop your IBM Internet Connection Secure Server by logging on as root user and typing `stopsrc -s httpd`. Press Enter.
12. Start your IBM Internet Connection Secure Server by logging on as root user and typing `startsrc -s httpd`. Press Enter.
13. To test the key ring, on the command line of your browser type `https://<your_node_name>`. If your key has been set up correctly, you will see several informational windows concerning your secure connection.
14. Enable cache and proxy or SOCKS servers from your Web browser.

10.3.1.4 Creating a Secure Server Certificate Signed by a Certifying Authority

To create a secure server certificate signed by a certifying authority, follow these steps:

1. Follow steps 1 through 6 of 10.3.1.1, "Creating a Self-Signed Certificate" on page 226.
2. If you are going to request a certificate from Verisign, select a certificate type of **Verisign (Secure Server Certificate)** and click on **Apply**.
3. Follow the steps 8 through 12 of 10.3.1.1, "Creating a Self-Signed Certificate" on page 226.

4. Mail the certificate to the certificate authority. You will also have to provide additional proof of identity via letter. Check with your certifying authority for their specific requirements. Look at the Verisign secure server process at <http://www.verisign.com/ibm/index.html> as an example of a typical certification mechanism.
5. Follow step 14 of 10.3.1.1, "Creating a Self-Signed Certificate" on page 226.

10.3.1.5 Changing the Current Ring

Next, you must tell the Web server to start to use your new key ring.

1. Return to the Configuration and Administration Forms page by clicking on **Security Configuration**.
2. Select **serv.kyr** in the Key Rings field.
3. Click on **Set selected key ring as current key ring**.
4. Click on **Apply**.

10.3.1.6 Receiving the Certificate for This Server

When you receive the secure server certificate, follow these steps:

1. Save the e-mail as an ASCII text file, called serv.crt.
2. Use a file transfer program (FTP) to copy files to the /usr/lpp/internet/server_root directory on your server.
3. Enter the following URL to access the server administration forms:
`http://<your_node_name>`
where <your_node_name> is a fully qualified server name, such as rs600020.itso.ral.ibm.com.
4. Click on **Configuration and Administration Form**.
5. Type the default user name webadmin and default password webibm. (If you have changed the user name or password, use the new ones.) Click on **OK**.
6. From the main Configuration Web page, scroll through the form and click on **Receive Certificate**.
7. Type the fields as follows:
 - a. Type /usr/lpp/internet/server_root/serv.crt as the name of the file that contains the certificate.
 - b. Type /usr/lpp/internet/server_root/serv.kyr as the key ring file.
 - c. Type the password you used to create the server key ring above.
 - d. Click on **Apply**. You will see another confirmation screen, indicating that the certificate has been successfully received.
8. Return to the main Configuration Web page. Click on **Configuration Page** at the bottom of the page.
9. Select **Key Management** at the bottom of the page.
10. Type the server key ring password and select **Manage Keys**. Click on **Apply**.
11. You will see a list of the keys in the key ring. Select **servkey** and **Set as default**. Click on **Apply**.
12. Follow the steps 11 through 14 from 10.3.1.3, "Receiving the Certificate for The Server" on page 231.

10.3.2 Verifying That the Installation and Configuration Is Successful

1. To launch the default store front page, from your Web browser, type the following URL:
`http://<your_host_name>/ncsample/base1.htm`
2. To test registration:
 - a. From your Web browser, type the following URL:
`http://<your_host_name>/cgi-bin/nph-msrvr;/register/form`
 - b. Fill in the information in the bold fields and click on **Submit**.
Note: This may take a while to respond. Be patient.
3. To test shopping cart:
 - a. From your Web browser, type the following URL:
`http://<your_host_name>/msprotect/nph-msrvr;/shopcart/display`
 - b. Use admin as the user ID and admin as the password for this user ID.
 - c. Quit.
 - d. Restart your Web browser.
4. To launch the Net.Commerce Administrator:
 - a. From your Web browser, type the following URL:
`http://<your_host_name>/ncadmin/`
 - b. Use admin as the user ID and admin as the password.
5. To launch the Net.Commerce Template Designer:
 - a. Click on **Site Manager** in the Net.Commerce Administrator.
 - b. Click on **Template Designer**.
 - c. Click on **Load**. The Template Designer will then load.
 - d. Click on **File**, then **New**.
 - e. Click on the **Add a Text Box** icon.
 - f. Click and drag a box in the working area.
 - g. Double-click on the **Text Box Object**.
 - h. Click on the **Database** menu item to ensure that items are listed in this menu.

10.3.3 Controlling the Net.Commerce Components

When you are configuring Net.Commerce you may have to stop and restart it several times. This is complicated by the fact that it is comprised of several components, some of which depend on other components to be active first.

10.3.3.1 Controlling IBM Internet Connection Secure Server on AIX

The IBM Internet Connection Secure Server is implemented as a subsystem. You can, therefore, start and stop it by typing these standard commands for subsystem operations:

```
startsrc -s httpd
stopsrc -s httpd
```

These commands require root authority.

After IBM Internet Connection Secure Server is installed, it is started automatically at boot time by the `/etc/rc.httpd` shell script, which is invoked by the init process as a result of a new entry labeled `rchttpd` in the `/etc/inittab` file.

The configuration for the IBM Internet Connection Secure Server is controlled by one configuration file, `/etc/httpd.conf`. Normally, you will not need to directly edit this file, because it is maintained automatically by the different installation and configuration dialogs. If you want to edit it yourself, refer to the *Internet Connection Servers V4.1 for AIX: Up and Running!* documentation.

10.3.3.2 Controlling IBM Internet Connection Secure Server for Windows NT

The IBM Internet Connection Secure Server is implemented as an NT service. Therefore, you can start and stop it from the Services panel of the Control Panel folder.

The IBM Internet Connection Secure Server is started automatically at boot time because the related service has the startup option set to automatic.

The configuration file for the IBM Internet Connection Secure Server is `\\WINNT\httpd.cnf`. This file is maintained automatically by the different installation and configuration scripts. If you want to edit it yourself, you can refer to the *Internet Connection Servers for Windows NT: Up and Running!* documentation.

You see in the following sections how the IBM Internet Connection Secure Server interacts with Net.Commerce Server and Net.Commerce Administrator.

10.3.3.3 Controlling IBM DATABASE 2 for AIX

On AIX, IBM DATABASE 2 is run as a set of daemons running in the background. These daemons are independent, in that they are not controlled by the Subsystem Resource Controller.

There are two options to start up and shut down the Net.Commerce database on AIX:

1. The first option is to log on as either the root user or the instance owner of the Net.Commerce database, and type:
 - `db2start` to start the database
 - `db2stop` to stop the database
2. The second option is to use the Net.Commerce Server Manager interface. The Database Management menu lets you start up or shut down the Net.Commerce database.

The URL to access the Server Control menu of Net.Commerce Server Manager is: `http://<your_node_name>:4444/admin/nccconfig/database`. This supposes, of course, that the Net.Commerce Server Manager is currently active.

IBM DATABASE 2 is started automatically at boot time by the `/etc/rc.netc` shell script, which is invoked by the init process (see the `rcnetc` entry in the `/etc/inittab` file).

10.3.3.4 Controlling IBM DATABASE 2 for Windows NT

On Windows NT, IBM DATABASE 2 is implemented as a pair of services:

- DB2 - <instance_owner_userID>
- DB2 - Security Server

There are two options to start up and shut down the Net.Commerce database on Windows NT:

1. The first option is to open **Services** in the Control Panel. Select **DB2 - Security Server service**, and click on either **Start** or **Stop**. Then select **DB2 - <instance_owner_userID> service** and click again on **Start** or **Stop**.
2. The second option is similar to the second option for the AIX platform.

The two services are started up automatically at boot time, since the startup option for both of them is set to automatic.

Except for maintenance operations, such as backup and table reorganization, you will probably never access the databases using DB2 or SQL commands. Customers access the database indirectly through the IBM Internet Connection Secure Server and Net.Commerce Server, and merchants access it through the IBM Internet Connection Secure Server and Net.Commerce Administrator. There are further details on how this operates in the following sections.

10.3.3.5 Controlling Net.Commerce Server

The number of Net.Commerce Server Daemons and the IP port used to communicate with Net.Commerce Server are configured at installation time or may be modified later via the System Configuration menu of Net.Commerce Server Manager. These parameters correspond respectively to the Port Number and Number of Processes field of the form.

The configuration of Net.Commerce Server is stored in a configuration file:

- /etc/mserver.conf under AIX
- \WINNT\MSERVER.INI on Windows NT

Refer to Chapter 11, "Updating the mserver Configuration File" in the *IBM Net.Commerce Customization Guide and Reference* for further details on this configuration file.

The Net.Commerce Server Daemon is implemented as a daemon on AIX and as a service on Windows NT.

Starting and Stopping the Net.Commerce Server Daemon on AIX: There are three options to start and stop the Net.Commerce Server Daemon on AIX:

1. The first option is to use the Net.Commerce Server Manager interface. The Server Control menu lets you start up and shut down the Net.Commerce Server Daemon.

The URL to access the Server Control menu of Net.Commerce Server Manager is: `http://<your_node_name>:4444/admin/ncconfig/process`.

Note that the Net.Commerce Server Manager must be currently running.

2. The second option is to start and stop the Net.Commerce Server Daemon like any process on UNIX:

Log on as the instance owner of the database, and type:

```
/usr/lpp/NetCommerce/bin/mserverd
```

To stop the daemon(s), you have to identify the process identifier(s) and kill the process(es) manually, by typing:

```
ps -ef | grep mserverd  
kill -9 <process_id>
```

3. The third option is to start the Net.Commerce Server Daemon using the /etc/rc.netc shell script. This requires the root authority. This will also start the database, if it is not yet started.

To stop the daemon(s), you will use the same process described in the second option.

Net.Commerce Server is started automatically at boot time by the /etc/rc.netc shell script which is invoked by the init process (see the rcnetc entry in the /etc/inittab file).

Starting and Stopping the Net.Commerce Server Daemon on Windows NT:

There are two options to start the Net.Commerce Server Daemon:

1. The first option is identical to the first option for the AIX platform.
2. The second option is to open the **Services** window in the Control Panel, select **Net.Commerce service** and click either the **Start** or the **Stop** button.

10.3.3.6 Controlling the Net.Commerce Server Manager

Net.Commerce Server Manager is an HTTP server listening on a specific port (4444 by default). This HTTP server is actually another instance of the IBM Internet Connection Secure Server with a specific configuration. This second instance of the IBM Internet Connection Secure Server cannot be started as a subsystem on AIX or as a service on Windows NT. Therefore, it is started as a mere daemon on AIX and a mere task on Windows NT. Access to this Web server requires an administrator account. Note that, as a security measure, Net.Commerce Server Manager is not started at boot time.

Unless you want to restrict access to the Net.Commerce Server Manager from a particular machine or subnetwork, there is no reason to modify the configuration file. However, if you want to look at the configuration file, it is /usr/lpp/NetCommerce/server/bin/httpd.conf.\$LANG on the AIX platform.

Net.Commerce Server Manager on AIX: There are two shell scripts to start and stop Net.Commerce Server Manager:

```
/usr/lpp/NetCommerce/server/bin/start_admin_server  
/usr/lpp/NetCommerce/server/bin/stop_admin_server
```

The configuration file of Net.Commerce Server Manager is in /usr/lpp/NetCommerce/server/bin directory. The name of the file is httpd.conf.\$LANG, where \$LANG represents the LANG environment variable.

Net.Commerce Server Manager on Windows NT: Net.Commerce Server Manager is started by clicking on the **Administrator Web server** icon in IBM Net.Commerce Server and Administrator folder. Net.Commerce Server Manager is started as a task. The corresponding window provides information about connection statistics, error messages, etc. Closing this window simply shuts down Net.Commerce Server Manager.

Chapter 11. Creating an Online Store with Net.Commerce

To create an online store, you need to follow these steps:

1. Configure the basics using Net.Commerce Server Manager.
2. Create and configure the mall and some global definitions using the site manager.
3. Create and configure the store using the store manager.

In this chapter we illustrate these steps by showing how we created the store used in the SET examples found elsewhere in this book. The examples here refer to the AIX version of Net.Commerce, but in general the same process would apply to other platforms.

11.1 Configuring the Basics of Net.Commerce

The first step is to start the Net.Commerce Server Manager. Log on as root, and type the following command:

```
/usr/lpp/NetCommerce/server/bin/start_admin_server
```

You will receive some confirmation messages.

To access the Net.Commerce Server Manager follow these steps:

1. Start your Web browser. Disable the following items:
 - All caching
 - Proxy servers
 - SOCKS servers

2. Type the following URL in your Web browser:

```
http://<your_host_name>:4444
```

where <your_host_name> is a fully qualified server name, such as rs600020.itso.ral.ibm.com. This is the name that you have assigned to the Net.Commerce Server.

3. Fill in the fields with the user name and password of the IBM Internet Connection Secure Server administrator. After the product has initially been installed, the default user name and password are: webadmin and webibm respectively.

You will be presented with the Net.Commerce Server main page, as shown in Figure 158 on page 242.

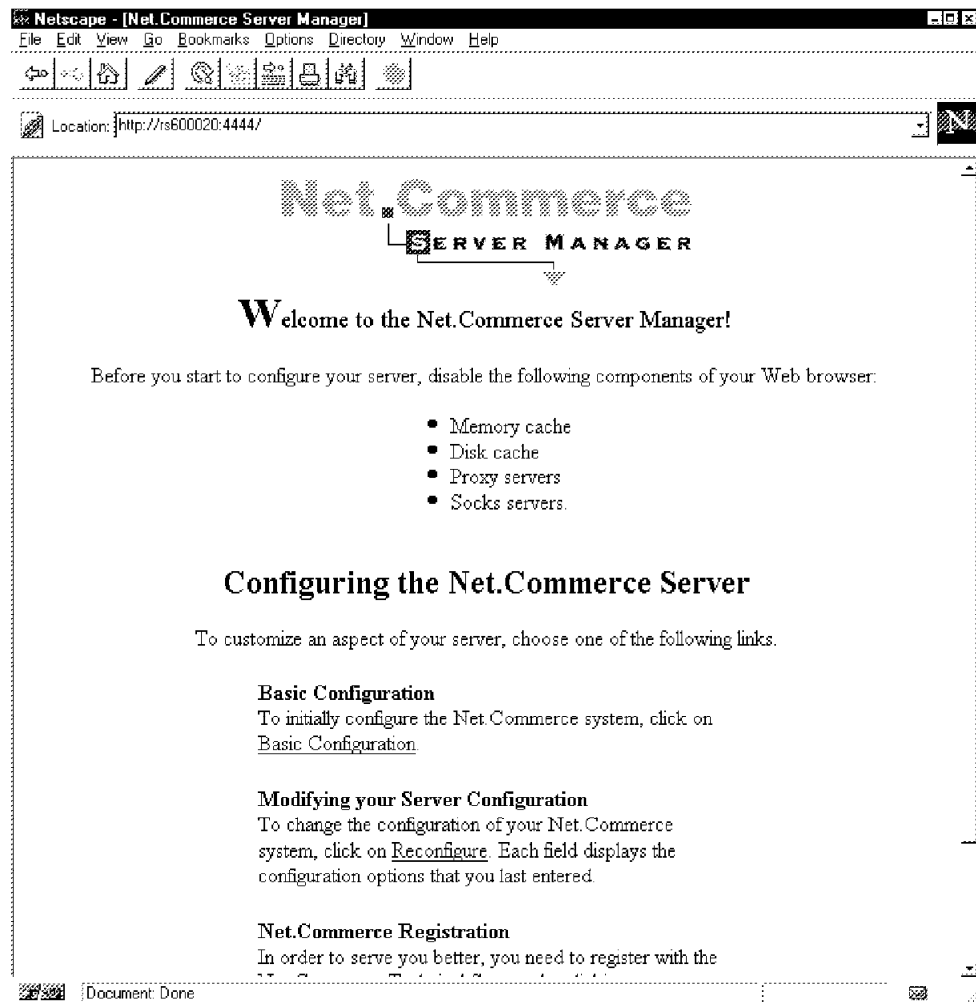


Figure 158. Net.Commerce Server Manager Main Page

4. On the Net.Commerce Server Manager main page, select **Basic Configuration**. If you have already configured Net.Commerce and want to change the configuration, click on **Reconfigure**.
5. Select **First Step System Configuration** at the bottom of the page. This link will guide you in configuring your system by ensuring that the step-by-step instruction sequence is done in the correct order. The sequence is shown graphically at the base of each screen. You will be able to configure:

- a. System Configuration

You will receive a form with most of the fields filled in. There are three fields to be completed in order to create your own mall or store. They are:

- **Database Name**

Type the name of the database in which you will place all the data about your mall or store. For the Winstead mall we chose a database name of *winstead* (see Figure 159 on page 243).

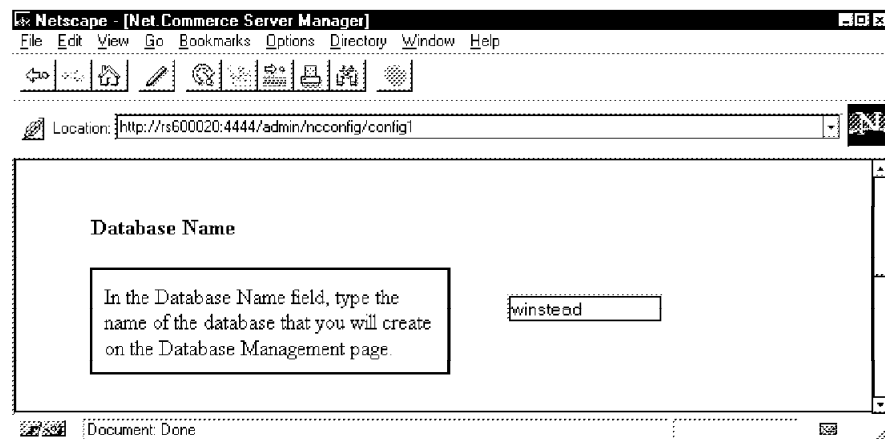


Figure 159. Specifying the Database Name

- **Database Administrator**

Type the DB2 Instance Owner user ID that you created before the installation of Net.Commerce. The default user ID is inst1.

- **Administrator Password**

Type the DB2 Instance Owner password that you chose when you created the DB2 Instance Owner user ID, and then re-enter it for confirmation in the second field.

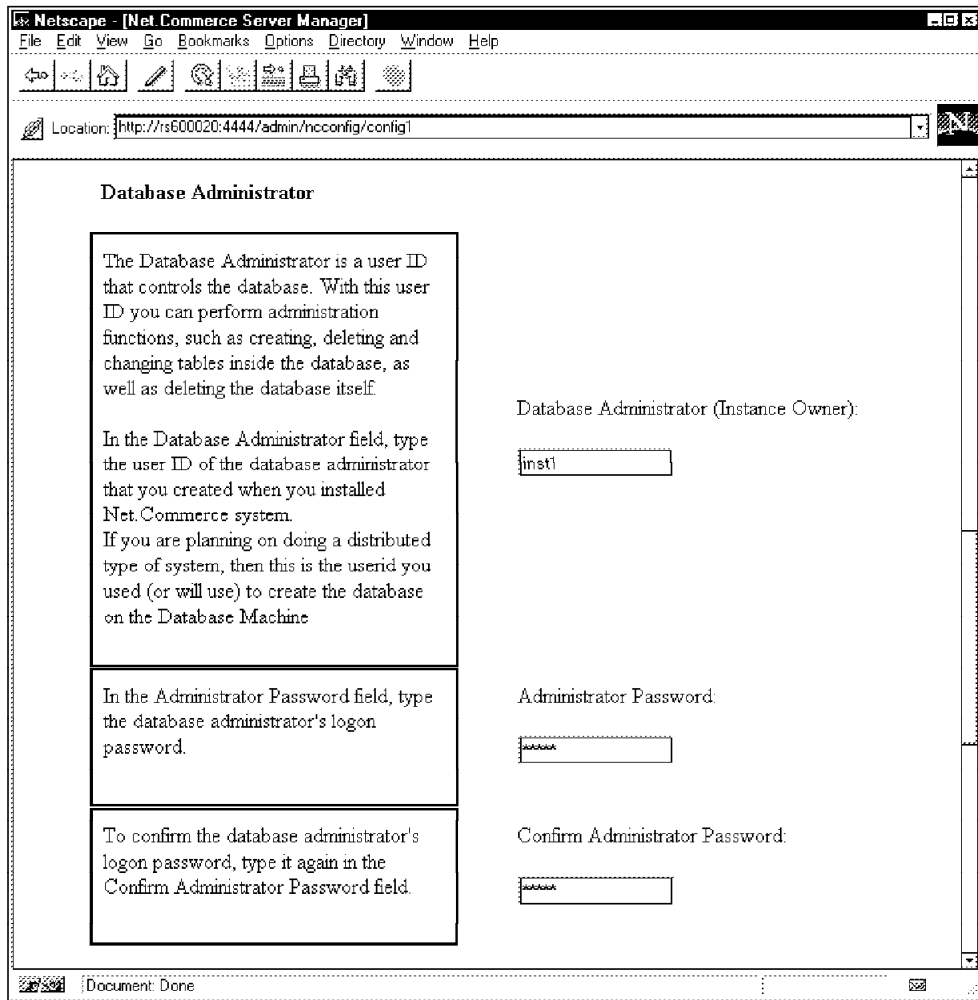


Figure 160. Administrator User ID and Password

When you have completed all the fields, click on **Submit Form**. A status page will appear, indicating that the configuration has been successful. If you have problems, check to see if all the fields have been filled in, or see *Net.Commerce Basics: Open for Business*.

- Click on **Next Step Access Control**.

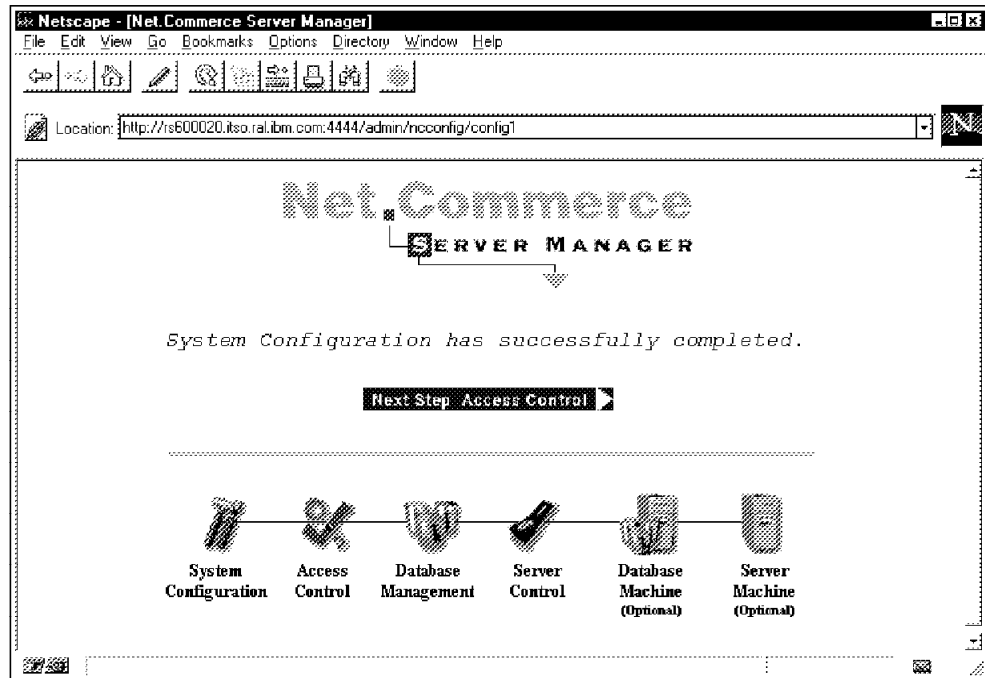


Figure 161. Confirmation Page

b. Access Control

In this form you have the chance to change the organization of the Net.Commerce system, by setting the paths for macros, admin programs, passwords, configurations, HTML and CGI-BIN files. In our case we did not have any need to change any of the defaults so we did not change these fields.

- When you have completed all the fields, click on **Submit Form**. A status page will appear, indicating that the configuration has been successful. If you have problems, check to see if all the fields have been filled in, or see *Net.Commerce Basics: Open for Business*.
- Click on **Next Step Database Management**.

c. Database Management

In this step you can start and stop the database, create the database with the tables that Net.Commerce uses and, if necessary, remove the database (see Figure 162 on page 246).

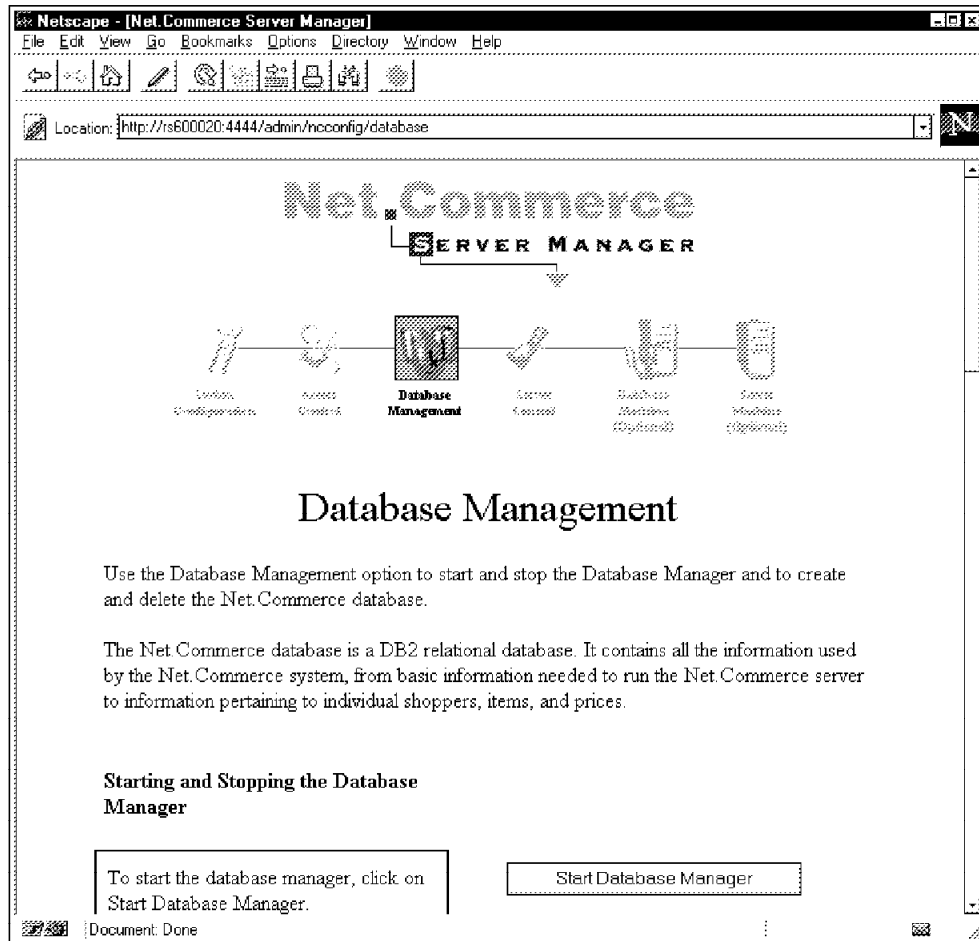


Figure 162. Database Management

If you are creating a new mall or store and you do not have any existing data for it, choose **Regular creation** and click on **Create**. This builds the database tables without inserting any store definitions.

If you are migrating, or already have the data necessary for your mall or store, choose **Custom creation** and complete the fields with the path and name of the script file of the data that will fill the tables. Click on **Create**. A status page will appear, indicating that the configuration has been successful. If you have problems, check to see if all the fields have been filled in, or see *Net.Commerce Basics: Open for Business*.

- Click on the **Server Control** icon.

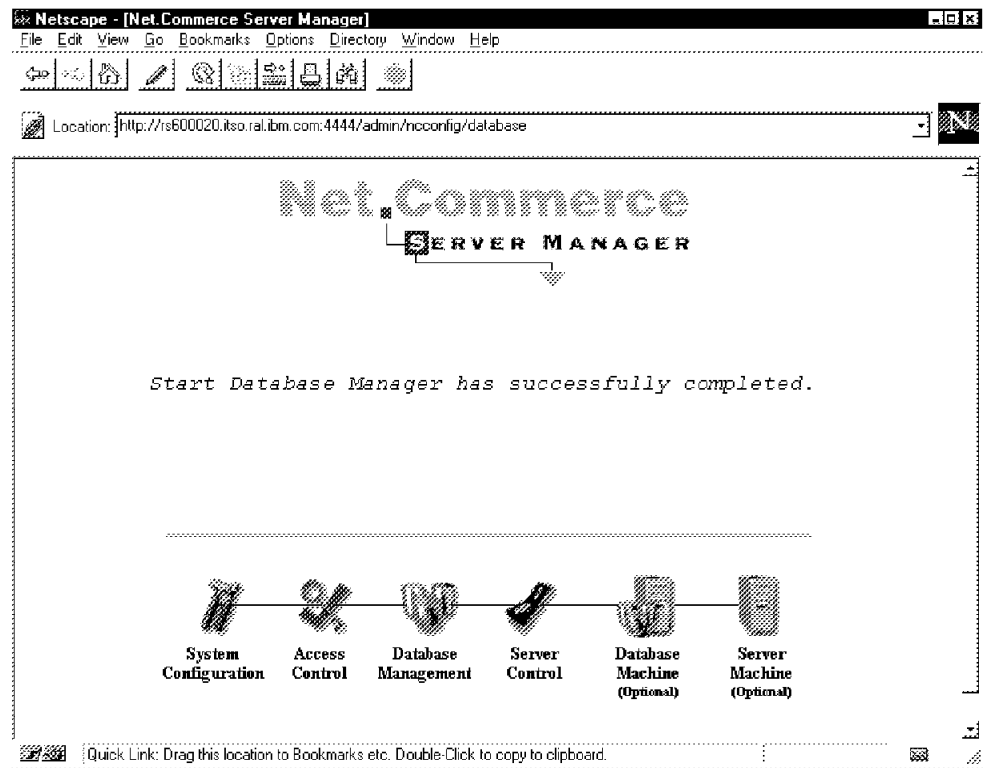


Figure 163. Confirmation Page

d. **Server Control**

You will be able to start, stop and refresh the Net.Commerce Server.

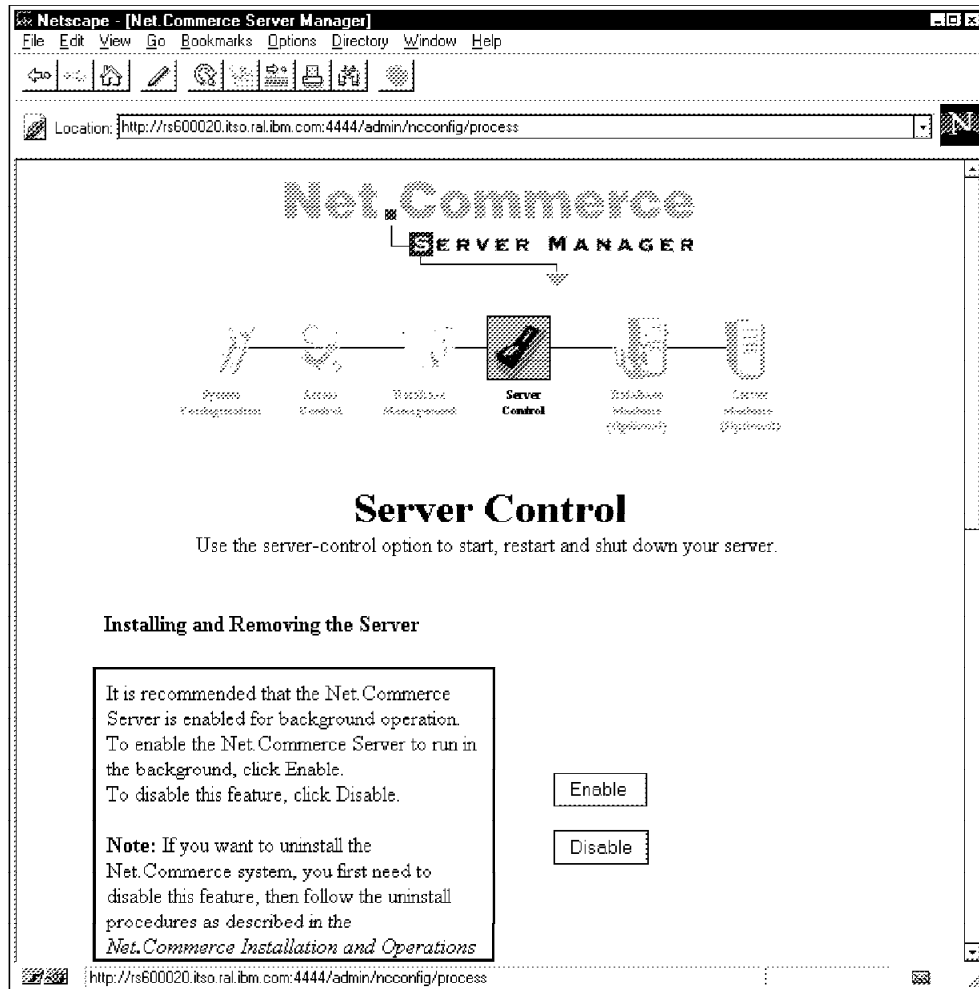


Figure 164. Server Control Form

11.1.1 Changing the Database

In the System Configuration form, fill in the Database Name field with the name of the database that will be used to record the data of your mall or store. If you want to create another mall, you should create another database with new tables. Use the Database Management form to do this. Follow these steps:

1. If Net.Commerce is running, shut it down by clicking on:
 - a. **Reconfigure** on the Net.Commerce Server Manager main panel.
 - b. **System Control**.
 - c. **Shut Down** at the bottom of this form.

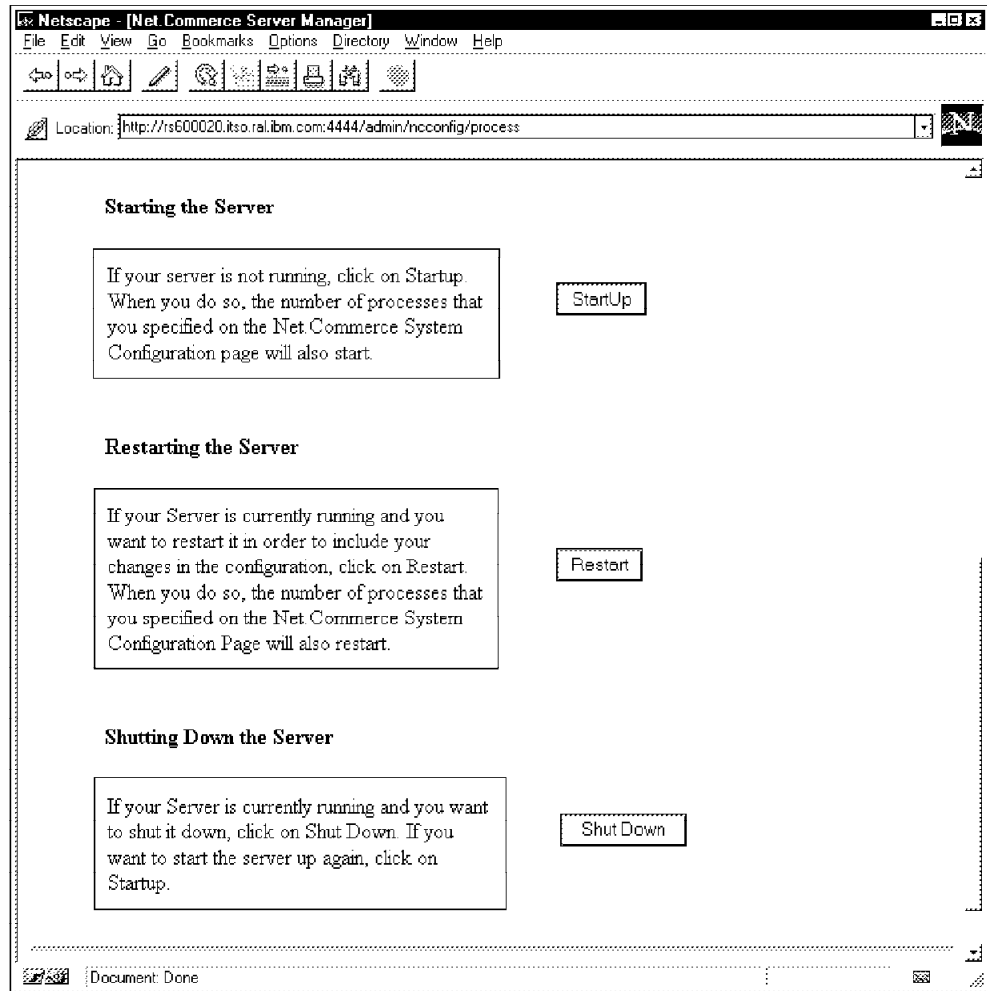


Figure 165. Shutting Down the Server

A status page will appear, indicating that the shutdown has been successful.

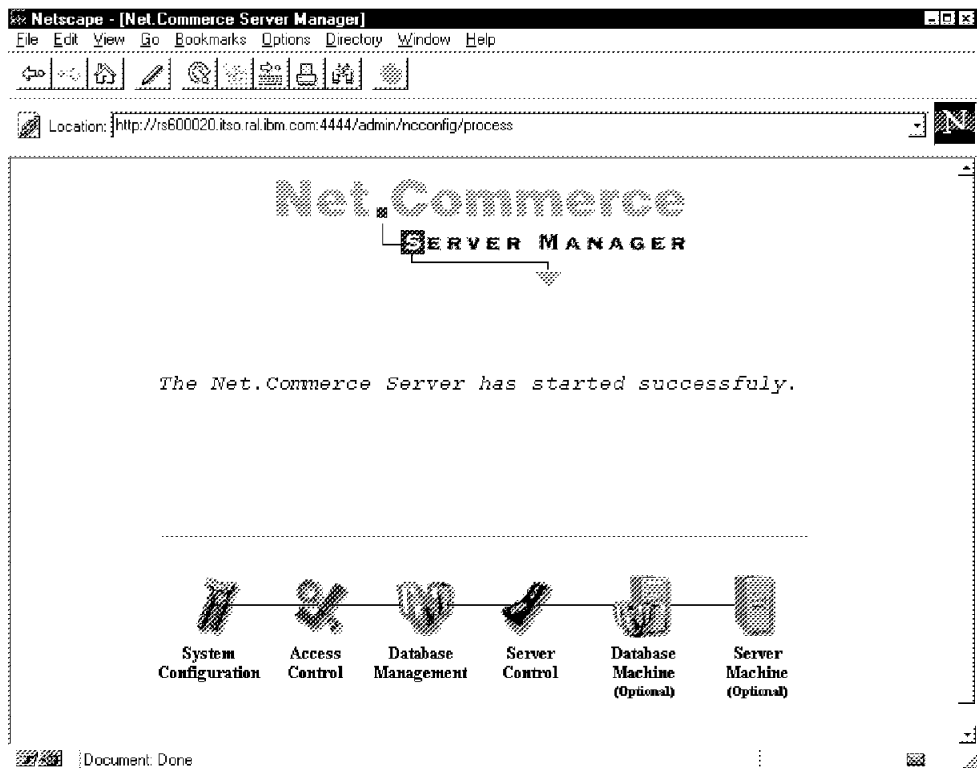


Figure 166. Shutting Down the Server

2. Click on the **Database Management** icon.
3. Create your own Database using the **Regular creation** or **Custom creation** option.

Note

If you have already created the database but it is not set as the working database, go to the System Configuration form and fill in the Database Name field with the name of your database.

Note

After you have created a new database, it will be set automatically as the working database.

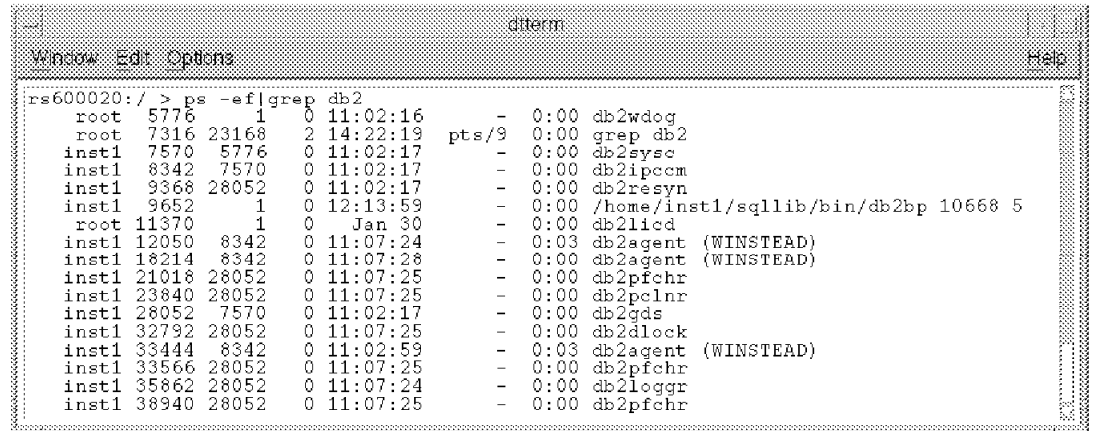
4. Stop DB2 by clicking on **Stop Database Management** in the Database Management form. A status page will appear, indicating that DB2 has stopped.
5. Check in the System Configuration form to see if the Database Name field has been filled in with the name of the database that you have created. Check in the file /etc/mserver.conf to see if there is the following statement:
`MS_DBNAME <your_database_name>`
where <your_database name> is the name of the database that you create for your mall or store.
6. Start DB2 by clicking on **Start Database Management** in the Database Management form. A status page will appear, indicating that DB2 has been started.

7. Start the Net.Commerce Server by clicking on **StartUp** in the Server Control form.

Check to see if Net.Commerce is now working with your new database, by typing:

```
ps -ef|grep
```

You will see a daemon db2agent, followed by the name of your database.



```
rs600020:/ > ps -ef|grep db2
root      5776      1      0 11:02:16      - 0:00 db2wdog
root      7316    23168      2 14:22:19 pts/9    0:00 grep db2
inst1     7570     5776      0 11:02:17      - 0:00 db2sysc
inst1     8342     7570      0 11:02:17      - 0:00 db2ipccm
inst1    93668    28052      0 11:02:17      - 0:00 db2resyn
inst1     9652      1      0 12:13:59      - 0:00 /home/inst1/sql1lib/bin/db2bp 10668 5
root     11370      1      0   Jan 30      - 0:00 db2l1cd
inst1    12050     8342      0 11:07:24      - 0:03 db2agent (WINSTEAD)
inst1    18214     8342      0 11:07:28      - 0:00 db2agent (WINSTEAD)
inst1    21018    28052      0 11:07:25      - 0:00 db2pfchr
inst1    23840    28052      0 11:07:25      - 0:00 db2p1nr
inst1    28052     7570      0 11:02:17      - 0:00 db2qds
inst1    32792    28052      0 11:07:25      - 0:00 db2dlock
inst1    33444     8342      0 11:02:59      - 0:03 db2agent (WINSTEAD)
inst1    33566    28052      0 11:07:25      - 0:00 db2pfchr
inst1    35862    28052      0 11:07:24      - 0:00 db2loggr
inst1    38940    28052      0 11:07:25      - 0:00 db2pfchr
```

Figure 167. Checking to See If Net.Commerce Is Working with the Correct Database

Important

After you have finished configuring the basics of Net.Commerce, it is recommended that you stop the Net.Commerce Server Manager by typing: `/usr/lpp/NetCommerce/server/bin/stop_admin_server`. In this way, the daemon that controls the administration of Net.Commerce will be disabled, and your system will be secure.

11.1.1.1 Creating and Configuring the Mall

To create and configure a mall or store, you must use the Net.Commerce Administrator. It is a set of forms that will help you do all the necessary tasks to create and configure your mall or store. In our example, we created a mall, called Winstead Mall. Follow these steps to create your own mall.

Accessing the Net.Commerce Administrator

1. Start your Web browser.
2. Type the following URL in your Web browser:

```
http://<your_host_name>/ncadmin
```

where <your_host_name> is a fully qualified server name, such as `rs600020.itso.ral.ibm.com`. This is the name that you have assigned to the Net.Commerce Server.

3. Fill in the fields with your user name and password to administer Net.Commerce. Once the product has been installed, the default user name and password are `admin` and `admin`.

For security reasons, this password must be modified after the first access. Refer to *Access Control in Net.Commerce Basics: Open for Business*.

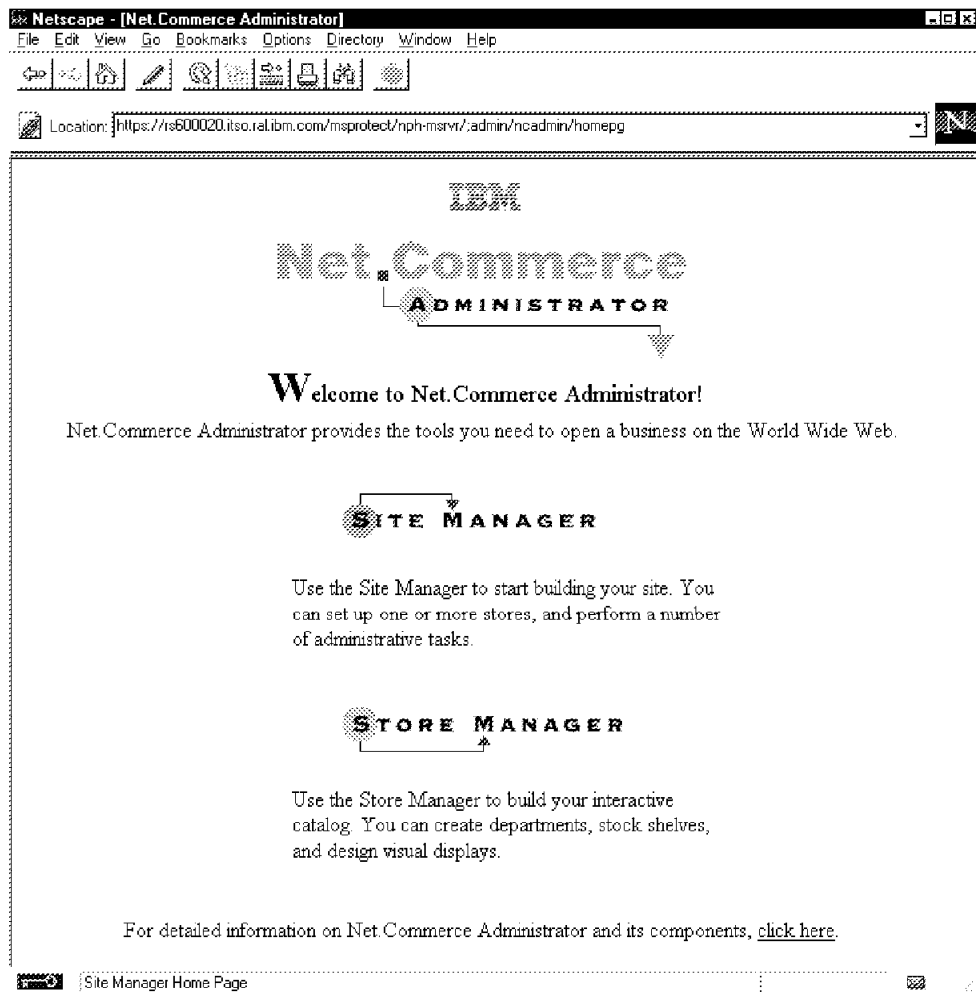


Figure 168. Net.Commerce Administrator Main Page

Note

All the Net.Commerce Administrator forms are secured using the SSL protocol. Everything you enter into the forms will be encrypted to prevent snooping.

4. Click on **Site Manager**. A screen will appear, showing the settings that can be customized, as shown in Figure 169 on page 253.

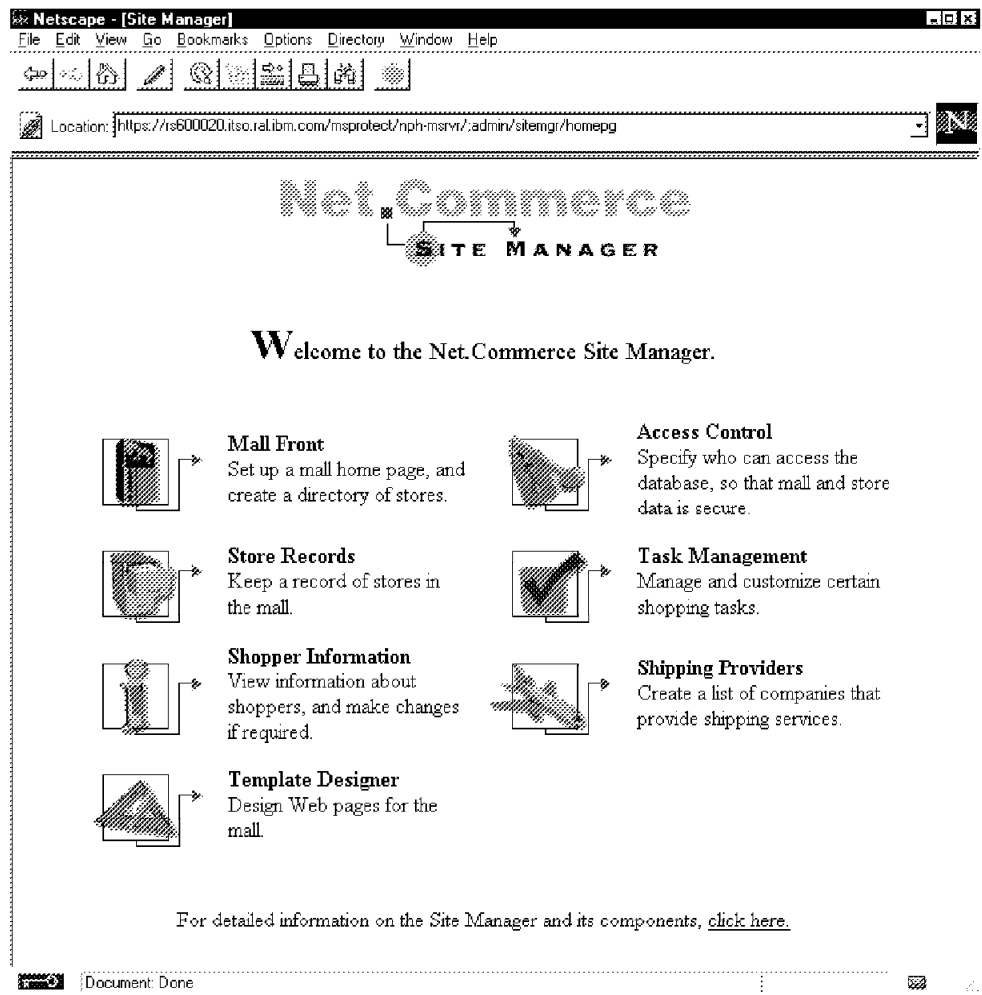


Figure 169. Site Manager Main Page

Creating the Mall Front: Before you begin the customization, you must create the home pages for your mall. They are:

- The front home page. This will be displayed when someone first accesses your mall. This home page must contain a header and a footer. Figure 170 on page 254 shows the HTML for our sample mall front. Note that you can place any kind of HTML coding here. In our case we have enhanced the user's experience with a JavaScript banner and animated GIF icons.

```

<HTML>
<HEAD>
<TITLE>Winstead Mall</TITLE>
</HEAD>

<BODY onLoad="timerONE=window.setTimeout('scrollit(30)',90);"
background="/winstead/quebra_ca.gif">
<SCRIPT LANGUAGE="JavaScript">
<!--
function scrollit(qik00){
var m1 = " Welcome to Winstead Mall";
var msg=m1;
var out = " ";
var c = 1;
if (qik00 > 20) {
qik00--;
var cmd="scrollit(" + qik00 + ")";
timerTwo=window.setTimeout(cmd,20);}
else if (qik00 <= 20 && qik00 > 0) {
for (c=0 ; c < qik00 ; c++) {out+=" ";} 1
out+=msg;
qik00--;
var cmd="scrollit(" + qik00 + ")";
window.status=out;
timerTwo=window.setTimeout(cmd,20);}
else if (qik00 <= 0)
{if (-qik00 < msg.length)
{out+=msg.substring(-qik00,msg.length);
qik00--;
var cmd="scrollit(" + qik00 + ")";
window.status=out;
timerTwo=window.setTimeout(cmd,20);}
else {
window.status=" ";
timerTwo=window.setTimeout("scrollit(50)",90);}}}

<!-- -->
</SCRIPT>
<CENTER>
<br><br>
</CENTER>

<CENTER>
<p> <h5> Welcome and thank you for visiting the most secure and modern
mall on the world!<BR>
Before you enter the mall please <B><a href="/cgi-bin/nph-msrvr/
;register/form">register</a></B> with us, by clicking on the register
icon at the top of the page.<BR> On a subsequent visit, you may
enter the mall as a<BR><A HREF="/msprotect/nph-msrvr;execmacro/
mall_dir.d2w/report">registered shopper</A> or as a 2
<A HREF="/cgi-bin/nph-msrvr;execmacro/mall_dir.d2w/report">
guest shopper</A>.
</h5><p><p><p>
</CENTER>

```

Figure 170 (Part 1 of 2). Mall Front Home Page

```

<CENTER>
<TABLE WIDTH=100>
<TR>
<TD WIDTH=160></TD>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/;
execmacro/mall_dir.d2w/report">
</A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/;register/
form"></A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/;execmacro
/adrbk.d2w/input">
</A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/;execmacro/
search.d2w/input">
</A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/;shopcart/
display">
</A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/;shipto/list">
</A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/;order/list?
status=P">
</A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/;order/
list?status=C">

</A>
</TR>
<TR>
<TD WIDTH=160></TD>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/;execmacro/
mall_dir.d2w/report">Home</A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/;register/
form">Register</A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/;execmacro/
adrbk.d2w/input">Address Book</A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/;execmacro/
search.d2w/input">Search</A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/;shopcart/
display">Shopping Cart</A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/;shipto/list">
Prepare Orders</A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/;order/
list?status=P">Place Orders</A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/;order/
list?status=C">Check Order Status</A>
</TR>
</TABLE>
</CENTER>
</BODY>
</HTML>

```

Figure 170 (Part 2 of 2). Mall Front Home Page

- 1** Java script
- 2** Calls to the registration routines
- 3** Links to other routines (footer)

The front home page must include the footer and the header, because this is the only page that is not created dynamically by Net.Commerce. Header and footer to be used with the other mall pages are inserted dynamically by Net.Commerce. You have to create the HTML coding for the header and footer sections, which could be the same as the front home page if you wish. Figure 171 on page 256 and Figure 172 on page 256 show the header and footer we used.

```

<CENTER>
<br><br>
</CENTER>

```

Figure 171. Mall Header

```

<center>
<TABLE WIDTH=100>
<TR>
<TD WIDTH=160></TD>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;execmacro/mall_dir.d2w/report"></A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;register/form"></A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;execmacro/adrbk.d2w/input"></A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;execmacro/search.d2w/input"></A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;shopcart/display"></A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;shipto/list"></A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;order/list?status=P"></A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;order/list?status=C"></A>
</TR>
<TR>
<TD WIDTH=160></TD>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;execmacro/mall_dir.d2w/report">Home</A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;register/form">Register</A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;execmacro/adrbk.d2w/input">Address Book</A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;execmacro/search.d2w/input">Search</A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;shopcart/display">Shopping Cart</A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;shipto/list">Prepare Orders</A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;order/list?status=P">Place Orders</A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;order/list?status=C">Check Order Status</A>
</TR>
</TABLE>
</center>

```

Figure 172. Mall Footer

Next we need to tell Net.Commerce about these new pages:

1. Click on **Mall Front**.
2. Fill in the Home Page, Header and Footer fields with the path and name of your pages. We recommend that you put your home pages inside the directory of your mall. To create this directory, type:
 - a. `cd /usr/lpp/NetCommerce/html/en_US`

- b. mkdir winstead
- c. To be sure that all files in the /winstead directory are accessible from a Web browser, you must add a PASS directive in the configuration of your Internet Connection Secure Server. Add the following directive in the /etc/httpd.conf:

Pass /winstead/* /usr/lpp/NetCommerce/html/en_US/winstead/*

This must be placed before the catch-all Pass statement (Pass /*).

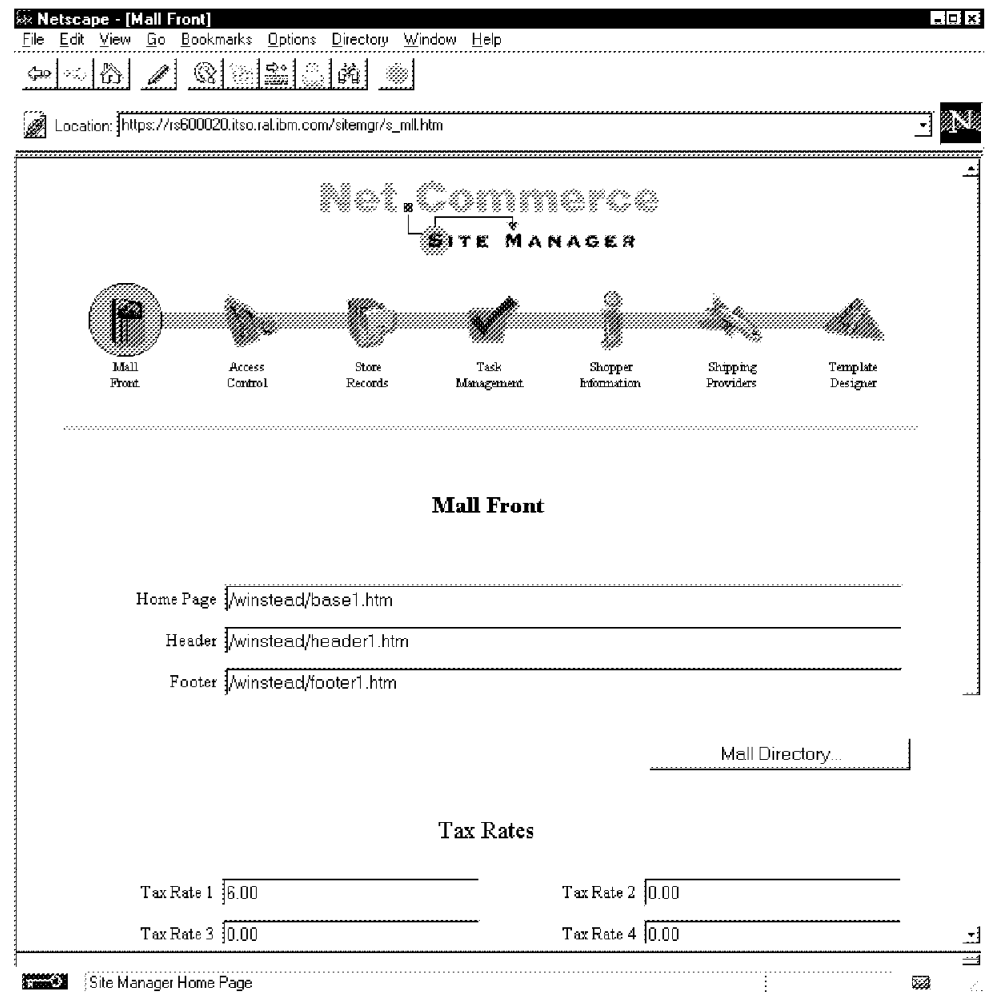


Figure 173. Mall Front Configuration

- 3. Click on **Mall Directory...** and create the categories of products and services that your mall offers.
 - a. Fill in the Categories field with the type of products and services that your mall offers, for example, books.
 - b. Click on **Create** and check to see if this new category appears below the Categories field.

In this form you are able to delete a category by selecting **Remove** in the Categories field.

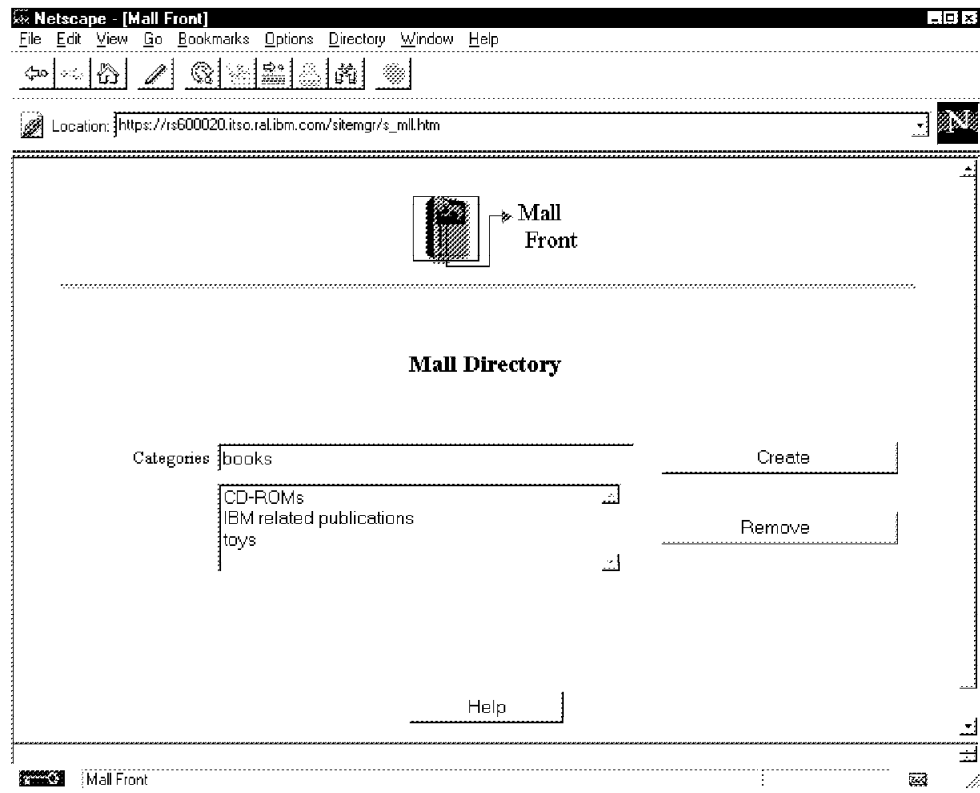


Figure 174. Defining Categories

- c. Click on the top icon to return to the Mall Front form.
- d. Click on **Save** at the bottom of the Mall Front form.

A status message will appear, indicating that the information has been successfully added to the database.

For details about the other fields in this form, see *Net.Commerce Basics: Open for Business*.

Configuring Access Control: Net.Commerce is designed to allow multiple administrators to create and maintain the mall and the stores in it. The Access Control is provided to you to create and modify the user IDs of the administrators.

Click on the **Access Control** icon. To create a site administer user, follow these steps:

1. Fill in all the fields that have bold letters.
2. For Site Authority? choose **Yes**.
3. Type a password and its confirmation.

The screenshot shows a Netscape browser window titled "Netscape - [Access Control]". The address bar displays "Location: https://rs600020.itso.ral.ibm.com/sitemgr/s_acs.htm". The main content area is titled "Access Control" and contains a form with the following fields and controls:

- Administrator's ID:** Text input field containing "malladm".
- Site Authority?:** Dropdown menu set to "Yes".
- Store Authorities...:** Button.
- Password:** Text input field.
- Password Confirmation:** Text input field.
- Title:** Dropdown menu.
- Last Name:** Text input field containing "Administrator".
- First Name:** Text input field containing "Joe".
- Middle Name/Initial:** Text input field containing "F".
- Challenge Question:** Text input field.
- Answer:** Text input field.
- Last Visit:** Text input field.
- Registration:** Text input field.
- Last Updated:** Text input field.
- Cancellation:** Text input field.

At the bottom of the form are five buttons: "Save", "Search", "Clear", "Delete", and "Help". The browser's status bar at the bottom shows "Document: Done".

Figure 175. Creating a Site Administrator User

4. Click on **Save**.

A status message will appear, indicating that the information has been successfully added in the database.

To create a store administrator user, follow these steps:

1. Fill in all fields that have bold letters.
2. For Site Authority? **No**.
3. Type a password and its confirmation.

The screenshot shows a Netscape browser window titled "Netscape - [Access Control]". The address bar displays "https://rs600020.itso.ral.ibm.com/sitemgr/s_acs.htm". The main content area is titled "Access Control" and contains a registration form. The form fields are as follows:

- Administrator's ID:
- Site Authority?: (dropdown menu)
- Store Authorities...:
- Password:
- Password Confirmation:
- Title:
- Last Name:
- First Name:
- Middle Name/Initial:
- Challenge Question:
- Answer:
- Last Visit:
- Registration:
- Last Updated:
- Cancellation:

At the bottom of the form are five buttons: , , , , and .

Figure 176. Creating a Store Administrator User

4. Click on **Save**.
A status message will appear, indicating that the information has been successfully added to the database.
5. Click on **Store Authorities**.
6. Click on the **All Stores** selection field. The name of the store for which the user you have created will be the administrator.
7. Click on **Add**.

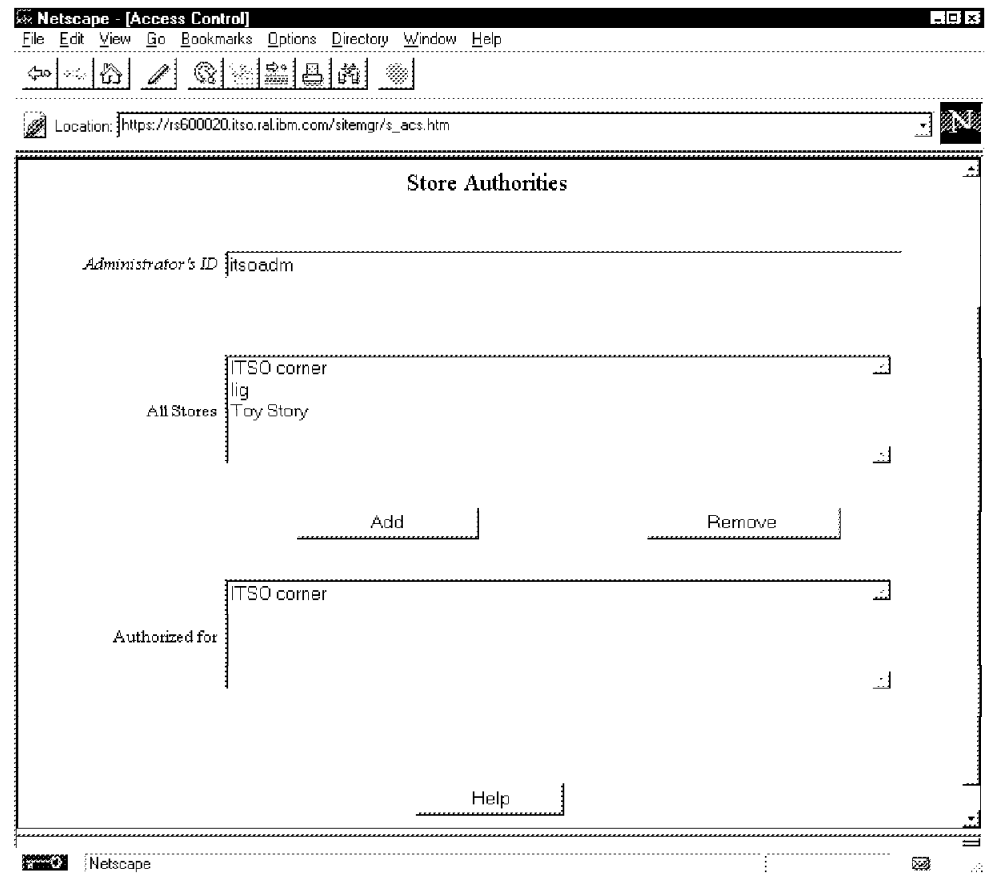


Figure 177. Creating a Store Administrator User

A status message will appear, indicating that the information has been successfully added to the database.

To modify user information, follow these steps:

- a. Fill in some of the fields with information about the user that you want to modify.
- b. Click on **Search**. (If you just click on Search without filling in the fields above, the status field at the bottom of the page will list *all* users.)
- c. From the status field, click on the name of the user to be modified, and the fields above will be filled in with the information about this user.
- d. Change the necessary fields. Click on **Save**.

A status message will appear, indicating that the information has been successfully modified in the database.

To delete user information, follow these steps:

- a. Fill in some of the fields with information about the user that is to be deleted.
- b. Click on **Search**. (If you just click on Search without filling in the fields above, the status field at the the bottom of the page will list *all* the users.)
- c. From the status field, click on the name of the user to be deleted. The information about this user will be deleted from the status fields.

- d. Click on **Delete**.

A status message will appear, indicating that the information has been successfully deleted from the database.

Configuring the Store Records: In this form, you can configure the name of the stores that will appear in your mall:

1. Click on the **Store Record** icon.

Fill in all the fields that have bold letters.

The screenshot shows a Netscape browser window titled "Netscape - [Store Records]". The address bar displays "Location: https://rs600020.itso.ral.ibm.com/sitemgr/s_rcd.htm". The main content area is titled "Store Records" and contains the following form fields:

- Store Name:** TSO corner
- Contact Information:**
 - Company Name:** International Technical Support Organisation
 - Phone Number:** 123.456.789
 - Contact's Job Title:** Manager
- Personal Information:**
 - Last Name:** McGyver
 - First Name:** John
 - Middle Name/Initial:** (empty)
 - Phone Number:** 999.999.999

At the bottom of the form, there are five buttons: Save, Search, Clear, Delete, and Help.

Figure 178. Creating Store Records

2. Click on **Save**.

A status message will appear, indicating that the information has been successfully added to the database.

To modify store records information, follow these steps:

1. Fill in some of the fields with information about the store that you want to modify.
2. Click on **Search**. (If you just click on Search without filling in the fields above, the status field will list *all* store records.)
3. From the status field at the bottom of the page, choose the store record to be modified by clicking on it. The fields above will be filled in with information about this store.
4. Change the necessary fields. Click on **Save**.

A status message will appear, indicating that the information has been successfully modified in the database.

To delete store records information, follow these steps:

1. Fill in some of the fields with information about the store that you want to be deleted.
2. Click on **Search**. (If you just click on Search without filling in the fields above, the status field at the bottom of the page will list *all* store records.)
3. From the status field, choose the store record to be deleted by clicking on it, and the fields above with information about this user will be deleted.
4. Click on **Delete**.

A status message will appear, indicating that the information has been successfully deleted from the database.

Configuring the Shipping Providers: In this form, you can configure the name and characteristics of each shipping provider.

Click on the **Shipping Providers** icon.

1. In the **Select** field, add New Carrier/ Shipping Mode.
2. Fill in all fields that have bold letters.

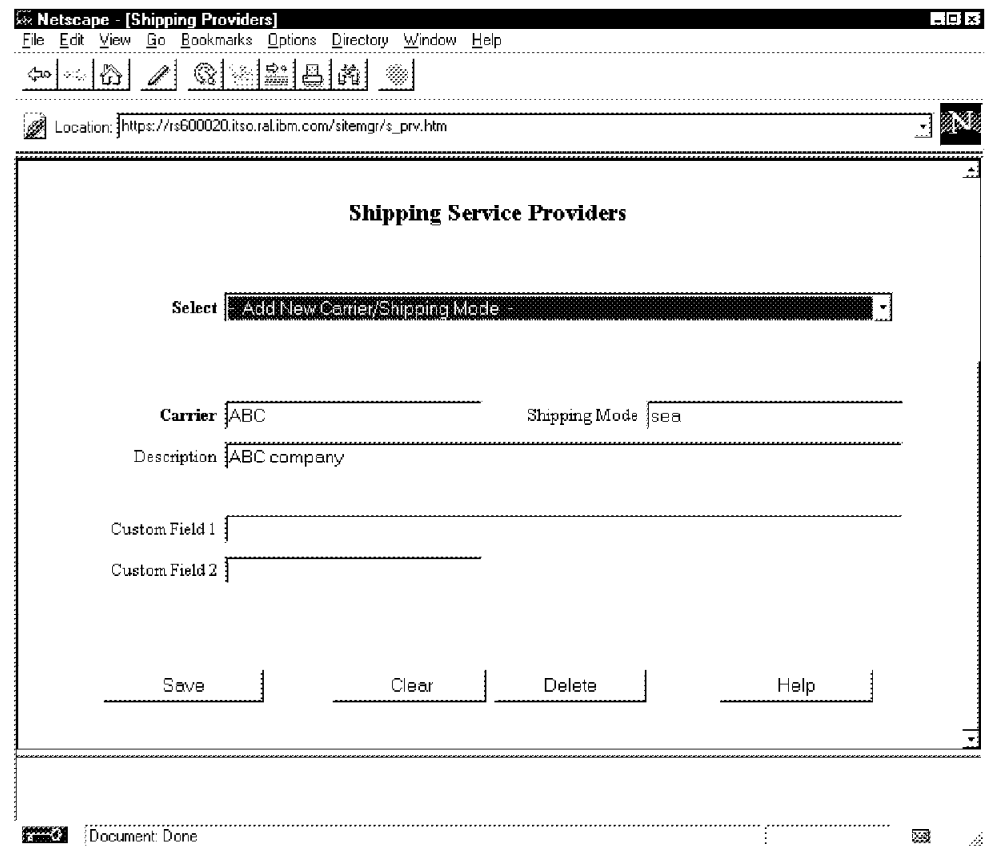


Figure 179. Creating a Shipping Provider

3. Click on **Save**.

A status message will appear, indicating that the information has been successfully added to the database.

To modify shipping provider information:

1. From the **Select** field, choose the name of the shipping provider. All the information about it will be filled in the fields below.
2. Modify the information.
3. Click on **Save**.

A status message will appear, indicating that the information has been successfully modified in the database.

To delete shipping provider information:

1. From the **Select** field, choose the name of the shipping provider. All the information about it will be filled in the fields below.
2. Click on **Delete**.

A status message will appear, indicating that the information has been successfully deleted from the database.

11.1.2 Creating and Configuring a Store

Inside the mall you find the stores. The next step is to create the individual store details. In our example, we created a store called ITSO Corner. Follow these steps to create your own store.

Click on **Store Manager** on the Net.Commerce Administrator main page. A screen will appear, showing all the settings that can be customized (see Figure 180 on page 265).

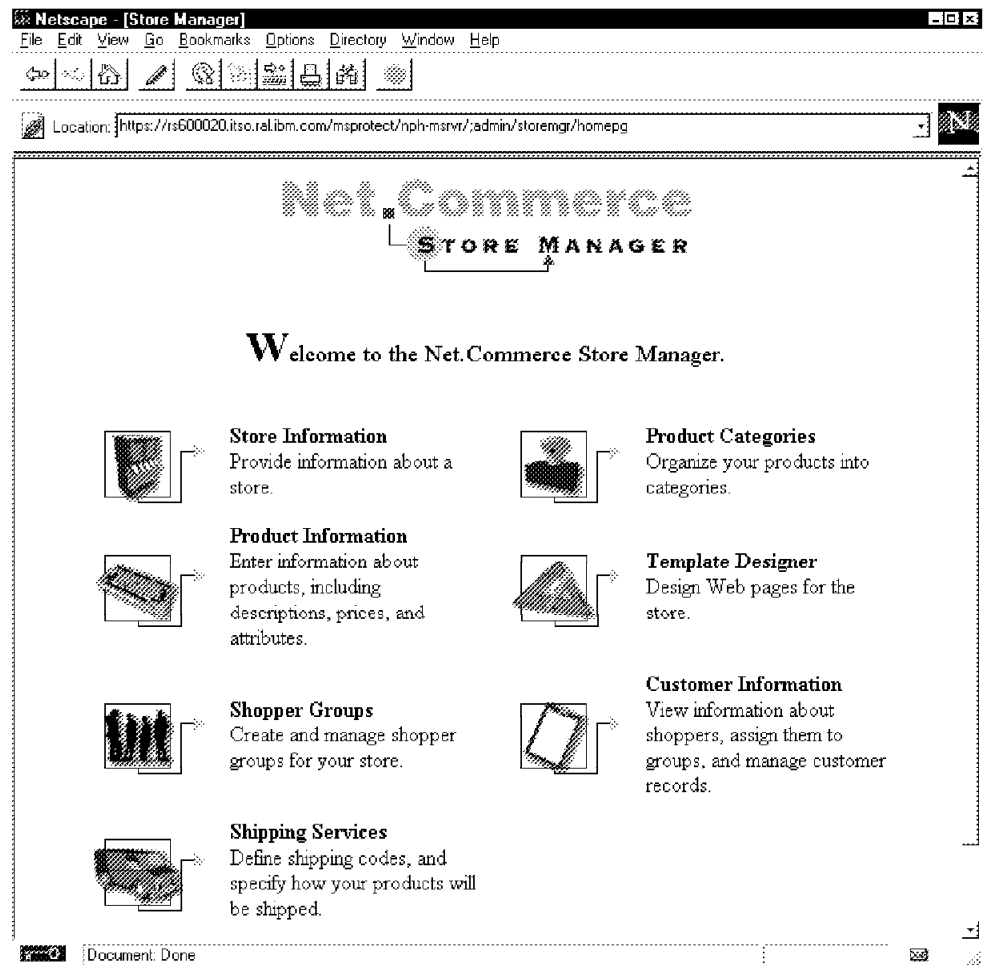


Figure 180. Store Manager Main Page

Before you begin the customization, you must create the home pages for your store. These are similar to the pages you previously created for the mall. The pages are:

- Store home page that will be displayed when someone accesses your store. This home page must contain a header and a footer.

```

<HTML>
<HEAD>
<title>ITSO Corner</title>
</HEAD>

<BODY background="/winstead/itso/back1.gif">

<center>
<br><br>
</center>

<center>
<p> <h5> Welcome to ITSO Corner!<BR>
Do you want to know about the latest technologies in the world?
You are at the right place!!<BR>
We have the most complete library of computer science and 1
we are the only shop that sells the famous IBM redbooks!<BR>
Choose the category and click on the button<BR>
</h5>
</center>

<CENTER>
<FORM Action="https://rs600020.itso.ra1.ibm.com/msprotect/
nph-msrvr/;execmacro/search2.d2w/report">
<INPUT TYPE=HIDDEN Name="SearchBy" Value="Category">
<SELECT NAME="ProductCategory">

        <OPTION SELECTED VALUE="IBM Application
Development">IBM Application Development
        <OPTION SELECTED VALUE="IBM AS/400">IBM
AS/400
        <OPTION SELECTED VALUE="IBM Networking and
Systems Management">IBM Networking and Systems Management
        <OPTION SELECTED VALUE="IBM Personal Systems 2
">IBM Personal Systems
        <OPTION SELECTED VALUE="IBM RISC System/6000
">IBM RISC System/6000
        <OPTION SELECTED VALUE="IBM System/390">
IBM System/390
        <OPTION SELECTED VALUE="IBM Transaction
Processing and Data Management">IBM Transaction Processing
and Data Management
</SELECT>

<INPUT TYPE="submit" VALUE="Click to Display Products">
</FORM><p>
</CENTER>

```

Figure 181 (Part 1 of 2). Store Front Home Page

```
<center>
<TABLE WIDTH=100>
<TR>
<TD WIDTH=160></TD>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;execmacro/mall_dir.d2w/report"></A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;register/form"></A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;execmacro/adrbk.d2w/input"></A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;execmacro/search.d2w/input"></A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;shopcart/display"></A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;shipto/list"></A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;order/list?status=P"></A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;order/list?status=C"></A>
</TR>
<TR>
<TD WIDTH=160></TD>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;execmacro/mall_dir.d2w/report">Home</A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;register/form">Register</A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;execmacro/adrbk.d2w/input">Address Book</A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;execmacro/search.d2w/input">Search</A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;shopcart/display">Shopping Cart</A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;shipto/list">Prepare Orders</A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;order/list?status=P">Place Orders</A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;order/list?status=C">Check Order Status</A>

</TR>
</TABLE>
</center>
</BODY>
</HTML>
```

3

Figure 181 (Part 2 of 2). Store Front Home Page

- 1 Welcome message
- 2 Options of products that the buyer can choose
- 3 Links to other routines (footer)

As in the case of the mall front, the store front home page is the only one that is not created dynamically by Net.Commerce, so it is a complete HTML page.

- The other pages are created dynamically, but you need to create header and footer coding for Net.Commerce to imbed into each page. Figure 182 on

page 268 and Figure 183 on page 268 show the header and footer that we created for the ITSO Corner.

```
<center>
<br><br>
</center>
```

Figure 182. Store Header

```
<center>
<TABLE WIDTH=100>
<TR>
<TD WIDTH=160></TD>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;execmacro/mall_dir.d2w/report"></A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;register/form"></A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;execmacro/adrbk.d2w/input"></A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;execmacro/search.d2w/input"></A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;shopcart/display"></A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;shipto/list"></A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;order/list?status=P"></A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;order/list?status=C"></A>
</TR>
<TR>
<TD WIDTH=160></TD>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;execmacro/mall_dir.d2w/report">Home</A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;register/form">Register</A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;execmacro/adrbk.d2w/input">Address Book</A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;execmacro/search.d2w/input">Search</A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;shopcart/display">Shopping Cart</A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;shipto/list">Prepare Orders</A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;order/list?status=P">Place Orders</A>
<TD WIDTH=55 ALIGN=CENTER><A HREF="/cgi-bin/nph-msrvr/
;order/list?status=C">Check Order Status</A>
</TR>
</TABLE>
</center>
```

Figure 183. Store Footer

11.1.2.1 Configuring the Store Information

In this form, you can configure all the information about the store.

Note

To create or modify the information about a store, you must be logged on as the store administrator or site administrator of this store.

Click on the **Store Information** icon.

1. From the Select field, choose the name of the store. All the available information about it will be filled in the fields below.
2. Fill in all fields that have bold letters.
3. Fill in the Store Home Page, Store Header File and Store Footer File fields, using the path and name of your home pages. We recommend that you put your home pages inside the directory of your store. To create this directory, type:

```
cd /usr/lpp/NetCommerce/html/en_US/winstead
mkdir itso
```

4. To be sure that all files in the /itso directory have been accessed, you must add a PASS directive in the configuration of your Internet Connection Secure Server. Add the following directive in the /etc/httpd.conf in the mapping rules group directive:

```
Pass    /itso/*    /usr/lpp/NetCommerce/html/en_US/winstead/itso/*
```

Netscape - [Store Information]
File Edit View Go Bookmarks Options Directory Window Help
Location: https://rs600020.itsc.re.ibm.com/storemgr/s_str.htm

Store and Merchant Information

Select Store: ITSO corner

Store Information

Store Name: ITSO corner
Store Description: Redbooks online store
Logo Thumbnail Image: /its0/logo.gif
Store Category: IBM related publications
Currency: Dollars

Store Front

Store Home Page: /its0/home.htm
Store Header File: /its0/header.htm
Store Footer File: /its0/footer.htm

Document: Done

Figure 184. Store Information

5. Click on **Save**.

A status message will appear, indicating that the information has been successfully added to the database.

To modify store information:

1. From the Select field, choose the name of the store. All the available information about it will be filled in the fields below.
2. Modify the information.
3. Click on **Save**.

A status message will appear, indicating that the information has been successfully modified in the database.

Creating and Configuring the Product Categories: In this form, you can create and configure the product categories that the store will offer.

Click on the **Product Categories** icon.

From the Select Store field, choose the store in which you will create the product categories.

The name of the store and the product categories (if they have already been created) will appear.

To create new product categories, follow these steps:

1. Click on **Add**.
2. Fill in all fields that have bold letters.

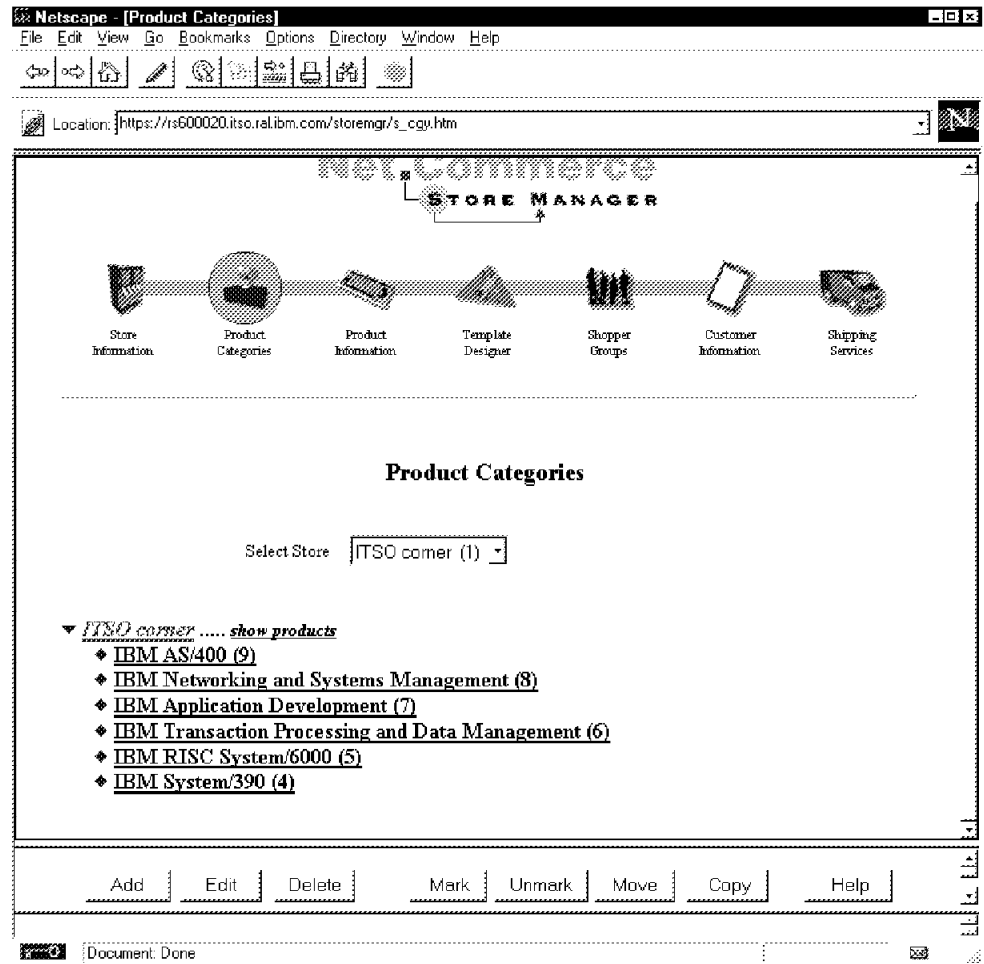


Figure 185. Product Categories

3. Click on **Save**.

A status message will appear, indicating that the information has been successfully added to the database.

To modify product categories, follow these steps:

1. Click on the product category that you want to modify.
2. Click on **Edit**.
3. Modify the necessary fields.
4. Click on **Save**.

A status message will appear, indicating that the information has been successfully updated in the database.

To delete product categories, follow these steps:

1. Click on the product category that you want to delete.
2. Click on **Delete**.

A status message will appear, indicating that the information has been successfully deleted from the database.

Configuring the Product Information: In this form, you can configure the product information that the store will offer. There are four forms to complete:

- General Product Information
Defines the dimensions and description of the product.
- Attributes
Defines the attributes of the product.
- Templates.
Defines which template is used to list each product.
- Price Information.
Defines the price of the product.

To create the product information, follow these steps:

1. Click on the **Product Information** icon.
2. Fill in all that have bold letters within the four forms.
3. Click on **Save**.

A status message will appear, indicating that the information has been successfully added to the database.

To modify the product information, follow these steps:

1. Fill in some of the fields with information about the product that you want to modify.
2. Click on **Search**. (If you just click on Search without filling in the fields above, the status field at the bottom of the page will list *all* products.)
3. From the bottom status field, choose the product to be modified by clicking on it. The fields above will be filled with the information about this product.
4. Change the necessary fields. Click on **Save**.

A status message will appear, indicating that the information has been successfully modified in the database.

To delete product information, follow these steps:

1. Fill in some of the fields with information about the product that you want to delete.
2. Click on **Search**. (If you just click on Search without filling in the fields above, the status field at the bottom of the page will list *all* products.)
3. From the bottom status field, choose the name of the user to be deleted by clicking on it. The fields above will be filled in with the information about this product.
4. Click on **Delete**.

A status message will appear, indicating that the information has been successfully deleted from the database.

Configuring the Shipping Service In this form, you can configure and associate the shipping mode with the stores. There are three forms to complete.

- **Shipping Service**
Defines the available shipping mode that each store supports.
- **Product Shipping Codes**
Defines the available calculation methods.
- **Shipping Details**
Associates the calculation methods with the available shipping modes.

To define the available shipping mode for each store, follow these steps:

1. Click on the **Shipping Service** icon.
2. Select the store from the Select Store field.
3. Select **Add New Supported Shipping Mode** from the Supported Shipping Mode field.
4. Select the available shipping mode that this store will support in the Shipping Carrier and Mode field.
5. Click on **Save**.

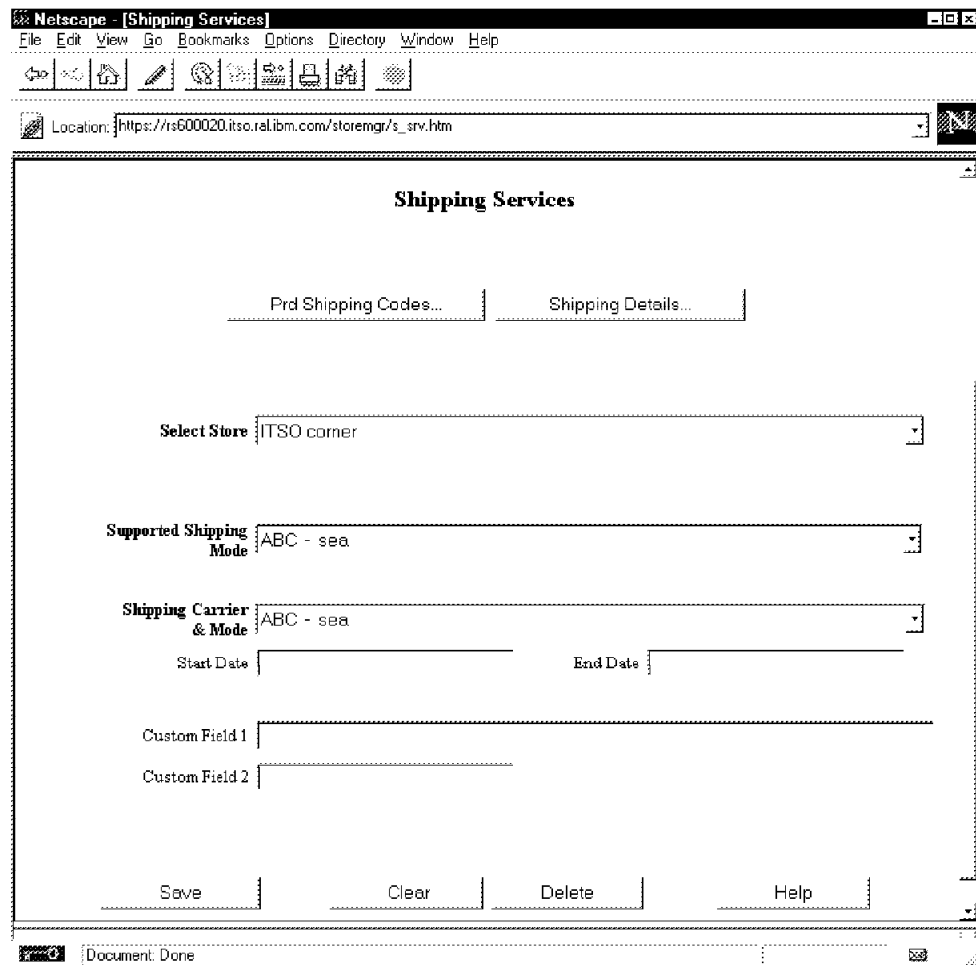


Figure 186. Shipping Service

A status message will appear, indicating that the information has been successfully updated in the database.

To modify the available shipping mode for each store, follow these steps:

1. Select the store from the Select Store field.
2. Select the shipping mode from the Supported Shipping Mode field. All the available information about it will be filled in the fields below.
3. Modify the necessary information.
4. Click on **Save**.

A status message will appear, indicating that the information has been successfully updated in the database.

To delete the available shipping mode of each store, follow these steps:

1. Select the store from the Select Store field.
2. Select the shipping mode from the Supported Shipping Mode field. All the available information about it will be filled in the fields below.
3. Click on **Delete**.

A status message will appear, indicating that the information has been successfully deleted from the database.

To create the calculation method, follow these steps:

1. Click on **Product Shipping Codes** on the Shipping Services page.
2. Select **Add New Code** from the Supported Product Shipping Code.
3. Fill in the name of the product shipping code in the Product Shipping Code field.
4. Choose the Calculation Method.
5. Click on **Save**.

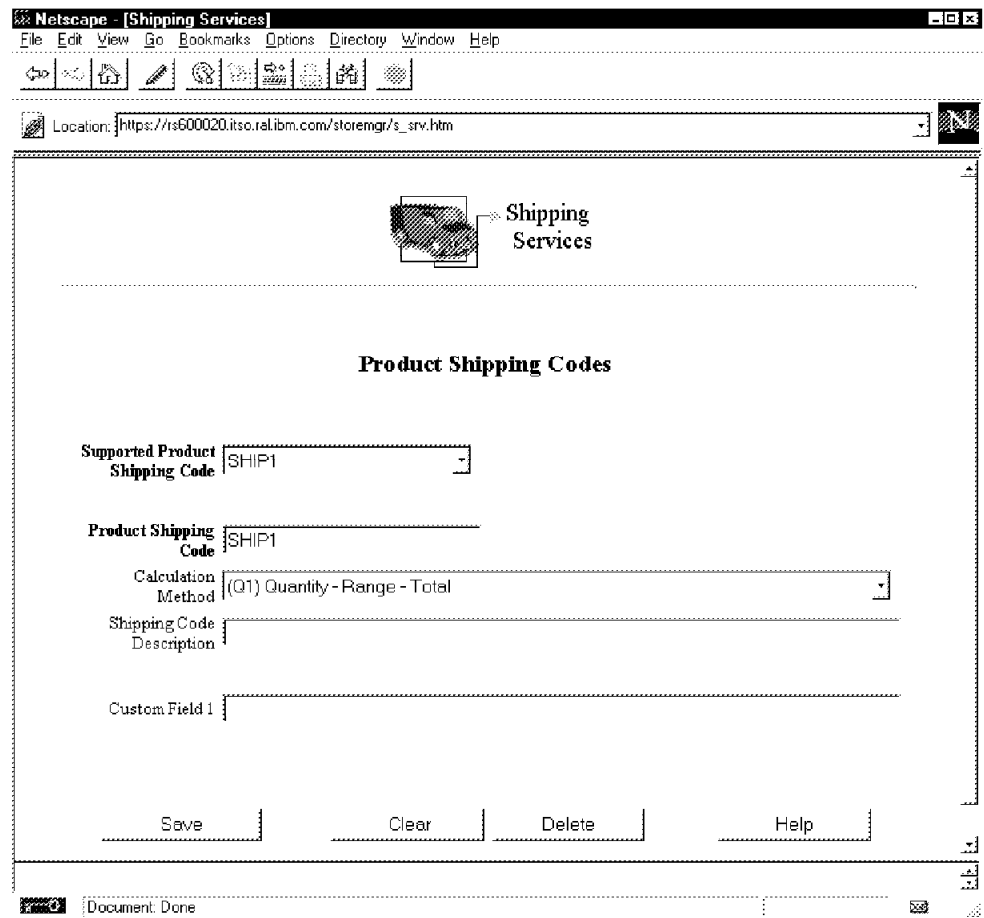


Figure 187. Calculation Method

A status message will appear, indicating that the information has been successfully updated in the database.

To modify the calculation method, follow these steps:

1. Select the shipping code from the Supported Product Shipping Code field. All the available information about it will be filled in the fields below.
2. Modify the necessary fields.
3. Click on **Save**.

A status message will appear, indicating that the information has been successfully updated in the database.

To delete the calculation method, follow these steps:

1. Select the shipping code from the Supported Product Shipping Code field.
All the available information about it will be filled in the fields below.
2. Click on **Delete**.

A status message will appear, indicating that the information has been successfully deleted from the database.

To create an association with the available calculation methods and the shipping codes, follow these steps:

1. Click on **Shipping Details** in the Shipping Services page.
2. Select the available product shipping code from the Product Shipping Code field.
3. Select the available shipping mode from the Shipping Mode field.
4. Fill in all fields that have bold letters.
5. Click on **Create**.

The screenshot shows a Netscape browser window titled "Netscape - [Shipping Services]". The address bar displays "https://rs600020.itso.ral.ibm.com/storemgr/s_srv.htm". The main content area is titled "Shipping Details" and contains a form with the following fields:

- Product Shipping Code**: SHIP1 (Q1)
- Shipping Mode**: ABC - sea (Effective from 01/23/1997 to 12/31/9999)
- Start Date**: 01/30/1997
- End Date**: 12/31/9999
- Range: minimum**: 1.00
- Range: maximum**: 20.00
- Shipping Charge (Total Cost)**: 5.00
- Shipping Rate (Unit Cost)**: [empty]
- Country**: [empty]
- Jurisdiction**: [empty]
- Custom Field 1**: [empty]
- Custom Field 2**: [empty]

At the bottom of the form, there are buttons for "Create", "Update", "Search", "Clear", "Delete", and "Help". The status bar at the bottom of the browser window shows "Document: Done".

Figure 188. Shipping Details

A status message will appear, indicating that the information has been successfully added to the database.

To modify the shipping details, follow these steps:

1. Fill in some of the fields with the shipping detail that you want to modify.

2. Click on **Search**. (If you just click on Search without filling in the fields above, the status field at the bottom of the page will list *all* shipping details.)
3. From the status field, choose the shipping detail to be modified by clicking on it. The fields above will be filled in with the information about this shipping detail.
4. Change the necessary fields. Click on **Update**.
A status message will appear, indicating that the information has been successfully modified in the database.

To delete shipping detail information, follow these steps:

1. Fill some of the fields with information about the shipping detail that you want to delete.
2. Click on **Search**. (If you just click on Search without filling in the fields above, the status field at the bottom of the page will list *all* shipping details.)
3. From the status field, choose the shipping detail to be deleted by clicking on it. The fields above will be filled in with the information about this shipping detail.
4. Click on **Delete**.
A status message will appear, indicating that the information has been successfully deleted from the database.

11.1.3 Customizing Your Macros

The Net.Commerce system provides the opportunity for you to customize most of the store and mall elements. Among these customizable elements are the macros. The modifications you can apply to the macros may be more or less complex depending on the nature of them. In fact, the modifications can range from the replacement of text, graphics or background on a store page to the addition of fields to the shopping cart. And the knowledge you need to perform these modifications increases the complexity, varying from HTML page design to SQL statements.

The Net.Commerce package comes with a set of macro samples that you can modify to suit your personal requirements. These macros are located in the `/usr/lpp/NetCommerce/macros/en_US/ncsample` directory.

11.1.3.1 Adopt a New Directory to Store the Macros

We recommend that you to keep your own set of macros in a distinct directory. And to let the Net.Commerce server know where your macros are stored, you need to modify the default configuration of the DB2/WWW interface. The configuration file for the DB2/WWW interface is the `/usr/lpp/internet/server_root/pub/db2www.ini` file and the directive to specify the path for the macro files is `MACRO_PATH`.

1. `mkdir /usr/lpp/NetCommerce/en_US/winstead`
2. `cp /usr/lpp/NetCommerce/en_US/ncsample/*
/usr/lpp/NetCommerce/en_US/winstead`
3. `edit /usr/lpp/internet/server_root/pub/db2www.ini`

A `db2www.ini` sample file is shown in Figure 189 on page 278. In this file, the path to the new macro directory has been added to `MACRO_PATH`.

```

MACRO_PATH /usr/lpp/NetCommerce/macro/en_US/winstead;/usr/lpp/NetCommerce/
macro/en_US/ncsample;/usr/lpp/NetCommerce/macro/en_US/demomall;/usr/lpp/
NetCommerce/macro/en_US
DB2INSTANCE inst1
BIND_FILE /usr/lpp/NetCommerce/bin/d2wsq1.bnd
INCLUDE_PATH /usr/lpp/NetCommerce/macro/en_US/ncsample;/usr/lpp/NetCommerce/
macro/en_US;/usr/lpp/NetCommerce/html/en_US/ncadmin/teeditor;/usr/lpp/
NetCommerce/html/en_US
HTML_PATH /usr/lpp/internet/server_root/pub
EXEC_PATH /usr/lpp/NetCommerce/macro/en_US/ncsample
ENVIRONMENT (DTW_SQL) /usr/lpp/NetCommerce/lib/dtwsq1shr.o ( IN DATABASE,
LOGIN, PASSWORD, TRANSACTION_SCOPE, ALIGN)
ENVIRONMENT (DTW_DEFAULT) /usr/lpp/NetCommerce/lib/dtwfuncshr.o ( )
ENVIRONMENT (DTW_APPLET) /usr/lpp/NetCommerce/lib/dtwapplshr.o ( )
ENVIRONMENT (DTW_ODBC) /usr/lpp/NetCommerce/lib/dtwodbcshr.o ( IN DATABASE,
LOGIN, PASSWORD, TRANSACTION_SCOPE, ALIGN)
ENVIRONMENT (DTW_REXX) /usr/lpp/NetCommerce/lib/dtwrexshr.o (OUT RETURN_CODE)
ENVIRONMENT (DTW_WEBREG) /usr/lpp/NetCommerce/lib/dtwregshr.o (OUT RETURN_CODE)
ENVIRONMENT (DTW_FILE) /usr/lpp/NetCommerce/lib/dtwffishr.o (OUT RETURN_CODE)
ENVIRONMENT (DTW_PERL) /usr/lpp/NetCommerce/lib/dtwperlshr.o (OUT RETURN_CODE)
DB2PATH

```

Figure 189. db2www.ini Sample File

Note that if you have different files with the same name in different directories, the system will use the first macro it finds in searching in the MACRO_PATH path. It is the reason why in this sample the

/usr/lpp/NetCommerce/macro/en_US/winstead directory comes before the /usr/lpp/NetCommerce/macro/en_US/ncsample directory.

And if you intend to change the macros on a per-store basis, you will have to give different names to the macros even if they correspond to the same task and you will have to then assign these new macros to the corresponding tasks, as described in 11.1.3.3, “Assigning Your Macro to Its Task” on page 279.

11.1.3.2 Modifying Your Macro

Giving you technical detail about macro customization is beyond the scope of the book. Nevertheless, we would like to show you how easy it is to dramatically change the look of your store.

The sample macros that come with the Net.Commerce display store pages with a uniform background that, in most cases, does not correspond to the image of your store you want to give to your customers. Changing the background of the pages is as easy as changing your mind about the color or the tile. Look at Figure 190 on page 279 to see the new look of the ITSO Corner change registration page after a minor modification in the corresponding macro.

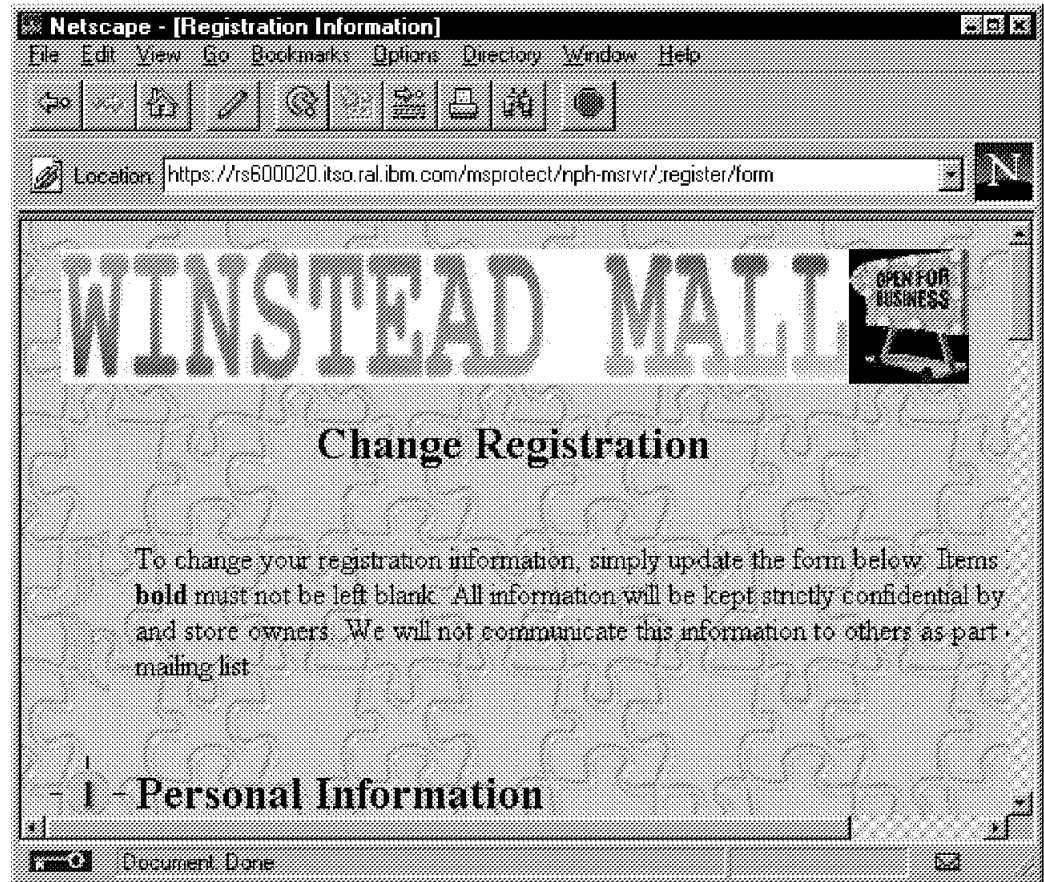


Figure 190. Change Registration Page

To change the background of all our store pages, we have slightly modified the macros. What used to be:

```
<BODY background="/ncsample/back1.gif">
```

has been changed to:

```
<BODY background="/winstead/quebra_ca.gif">
```

where /winstead/quebra_ca.gif refers to the store tile.

In the specific case of the change registration form, the macro that has been modified is msregupd.d2w. Its new file name is itso_msregupd.d2w and it is located in the /usr/lpp/NetCommerce/macro/en_US/winstead directory.

11.1.3.3 Assigning Your Macro to Its Task

Let's imagine now that you have several stores in your mall, with all different layouts for their pages. This mean that you will need different macros for your stores and will probably have a set of macros per store. Because of the way macros are searched for, that is using the MACRO_PATH in your DB2/WWW configuration file, you will need to give different names to the macros assigned to the same task but for different stores.

The Task Management function of the Net.Commerce Administrator creates a link between a task and the macro that implements the task. The customization can be performed on a mall or on the store. In our example, the customization is done on a store.

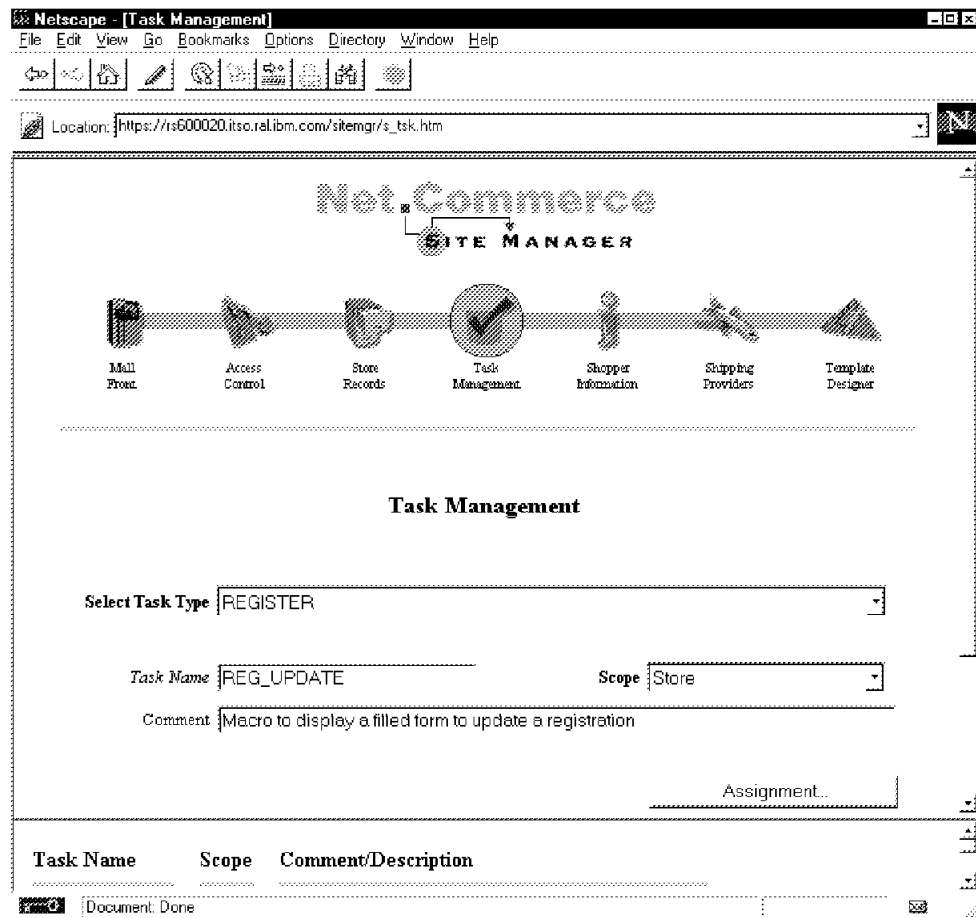


Figure 191. Selecting the Task for an Assignment

To assign a new macro to the update registration task (see Figure 191):

1. Start the task management menu.
2. Select the task type: **Register**.
3. In the bottom of the frame, select the desired task in the list: **REG_UPDATE**.
4. Change the task Scope to Store.
5. Click on the **Save** button
6. Click on the **Assignment** button.

A new frame will appear as shown in Figure 192 on page 281.

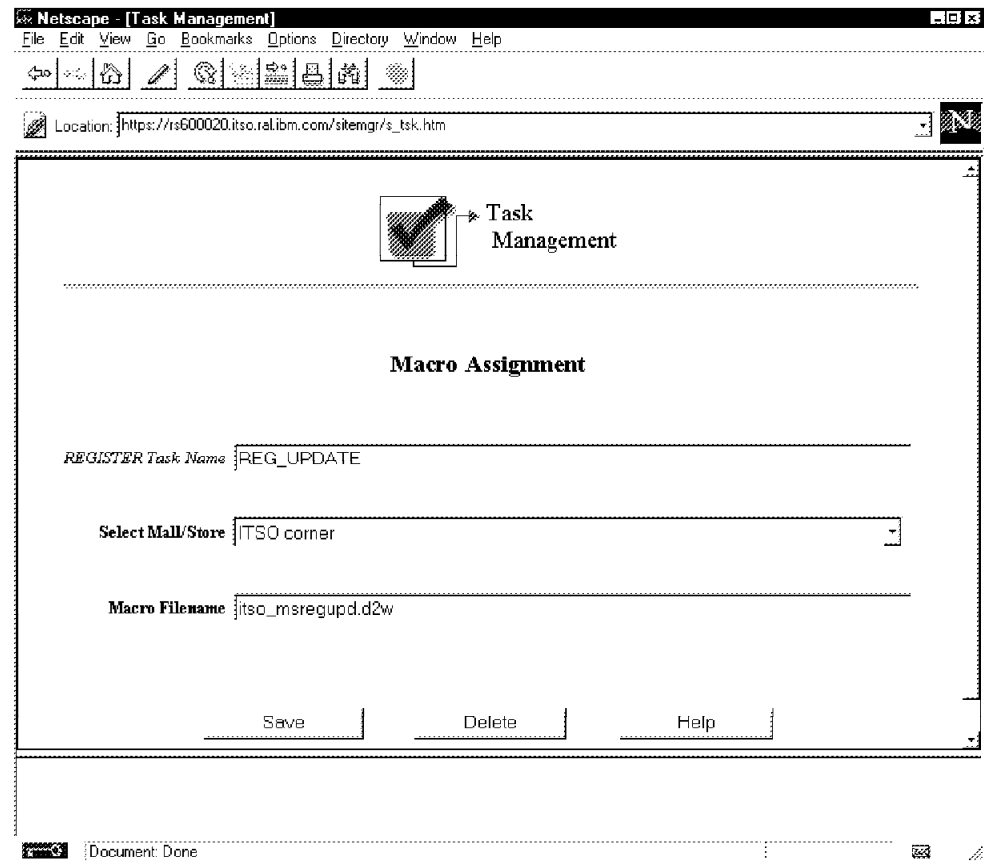


Figure 192. Assigning a Macro

To assign your own macro to your store:

1. Select the store to which the task applies: **ITSO Corner**
2. Enter the macro file name: `itsc_msregupd.d2w`.
3. Click on **Save**.

Appendix A. Certification Examples

This appendix contains examples of using the certification commands provided by IBM Registry for SET.

A.1 Using setca_certmgr to Create a Certificate Hierarchy

setca_certmgr gives you a simple menu-driven interface for generating keys, signing certificates and other CA functions. In this section we go through the sequence of steps to build a test CA hierarchy using the tool. In A.2, "Using setbrandca to Create a Certificate Hierarchy" on page 289 we show how to use the setbrandca shell script to perform the same action in a somewhat easier way. Each table of actions assumes that you have just started setca_certmgr from the AIX command line and that your current directory is where you want the key databases to be created.

A.1.1.1 Create a ROOT CA Key

	setca_certmgr Prompt	Response
1	1. Root CA, 2. Brand CA, 3. End-Entity CA	1
2	MAIN MENU	1 (Generate key pairs and certificate requests)
3	Country Name(3-letter ISO 3166 country code)	USA
4	Organization Name	MY_TEST_ROOT_CA
4	key pair DB filename [rca_keypair.db]	(press Enter)
5	key pair DB password	abc123 Note: you will see an error message because the key pair database file does not exist yet. This is normal.
6	Enter key usage	3 (CERTIFICATE_AND_CRL_SIGNATURE)
7	key pair record label [rca_certandcrlsign]	(press Enter)
8	Generate another key pair?	y
9	Enter key usage	3 (CERTIFICATE_AND_CRL_SIGNATURE)
10	key pair record label	rca_certandcrlsign_backup Note: this is the backup root key
11	Generate another key pair?	n

You now have a file called rca_keypair.db in the current directory, containing two key pairs (the Root key pair and the replacement Root key pair) and certificate requests for the two keys.

A.1.1.2 Create a Self-Signed Certificate for the Root Public Key

	setca_certmgr Prompt	Response
1	1. Root CA, 2. Brand CA, 3. End-Entity CA	1
2	MAIN MENU	3 (Generate self-signed certificates:(RCA only)
3	key pair DB filename [rca_keypair.db]	(press Enter) Note: same file as above

	setca_certmgr Prompt	Response
4	key pair DB password	abc123
5	key DB filename [rca_in_key.db]	(press Enter)
6	key DB password	abc123
7	key pair label for in_use public key	rca_certandcrlsign
8	Enter key usage	3 (CERTIFICATE_AND_CRL_SIGNATURE)
9	key pair label for backup public key	rca_certandcrlsign_backup
10	Enter certificate validity period (in days)	730
11	Enter private key usage period (in days)	730
12	Generate another certificate?	n

You now have a file called rca_in_key.db in the current directory. This contains the Root key certificates, signed by the root key itself. This file can be used as a starting point for key databases lower in the hierarchy, which require the Root certificate to be pre-installed for verification purposes (see 8.6.1, "Key and Certificate Hierarchy Required for SET" on page 153 for an explanation of this).

A.1.1.3 Merge the Root Certificates into the Root Key Database

	setca_certmgr Prompt	Response
1	1. Root CA, 2. Brand CA, 3. End-Entity CA	1
2	MAIN MENU	2 (Replace certificate requests with certificates)
3	key DB filename [rca_in_key.db]	(press Enter) Note: file containing the public keys and self-signed certificates
4	key DB password	abc123
5	key pair DB filename [rca_keypair.db]	(press Enter) Note: file containing the original key pairs and certificate requests
6	key pair DB password	abc123
7	key DB filename [rca_key.db]	(press Enter)
8	key DB password	abc123
9	Batch Processing: check all the key pairs...	y
10	MAIN MENU	0 (Exit)

You now have a file called rca_key.db in the current directory. This is the Root key database itself. It contains the key pairs and the self-signed certificates for them.

A.1.1.4 Create a Brand CA Key Database

Now we take on the role of a Brand CA, to generate a key database and certificate requests.

	setca_certmgr Prompt	Response
1	1. Root CA, 2. Brand CA, 3. End-Entity CA	2

	setca_certmgr Prompt	Response
2	MAIN MENU	1 (Generate key pairs and certificate requests)
3	Country Name(3-letter ISO 3166 country code)	USA
4	Organization Name	My_Test_Brand_CA
5	Common Name(= BrandId = BrandName[:ProductType])	Pass Note: this is the name of our fictional credit card brand
6	key pair DB filename [bca_keypair.db]	(press Enter)
7	key pair DB password	abc123
8	Enter key usage	3 (CERTIFICATE_AND_CRL_SIGNATURE)
9	key pair record label [bca_Pass_certandcrlsign]	(press Enter)
10	Generate another key pair?	n
11	MAIN MENU	0 (Exit)

You now have files called bca_keypair.db and bca_Pass_certandcrlsign.req in the current directory. The first of these is the Brand key database which should not leave the Brand CA system. The second is a certificate request file for the brand CA key. This has to be signed by the Root CA.

A.1.1.5 Signing the Brand CA Certificate

Now we switch back to the role of Root CA, to sign the certificate request just created.

	setca_certmgr Prompt	Response
1	1. Root CA, 2. Brand CA, 3. End-Entity CA	1
2	MAIN MENU	4 (Generate certificates for a lower-level CA)
3	key DB filename [rca_key.db]	(press Enter) Note: this is the database containing the key that we will use to do the signature, that is, the Root CA key database.
4	key DB password	abc123
5	issuer key record label	rca_certandcrlsign
6	key DB filename [bca_in_key.db]	(press Enter) Note: this is where the signed certificate will be placed
7	key DB password	abc123
8	certification request filename	bca_Pass_certandcrlsign.req Note: this is the file you created as the BCA, above
9	Enter key usage	3 (CERTIFICATE_AND_CRL_SIGNATURE)
10	Enter certificate validity period (in days)	365
11	Enter private key usage period (in days)	365
12	Generate another certificate?	n
13	MAIN MENU	0 (Exit)

You now have a bca_in_key.db file in the current directory. This contains the Brand public key certificate, signed by the Root CA.

A.1.1.6 Merge the Brand Certificate into the Key Database

Now we go back to the role of Brand CA to receive the certificate from the Root CA.

	setca_certmgr Prompt	Response
1	1. Root CA, 2. Brand CA, 3. End-Entity CA	2
2	MAIN MENU	2 (Replace certificate requests with certificates)
3	key DB filename [bca_in_key.db]	(press Enter) Note: this is the file from the Root CA, containing the signed certificate
4	key DB password	abc123
5	key pair DB filename [bca_keypair.db]	(press Enter) Note: this is the original Brand key pair database, containing the private keys and public key certificate request
6	key pair DB password	abc123
7	key DB filename [bca_key.db]	(press Enter) Note: this will be the complete brand CA key database, containing the key pair and certificate
8	key DB password	abc123
9	Batch Processing: check all the key pairs...	y
10	MAIN MENU	0 (Exit)

You now have a file called bca_key.db in the current directory. This is the final Brand key database for the Pass credit card brand.

A.1.1.7 Create an End Entity CA Key Database

Finally we reach the bottom of the CA hierarchy. We will create an end entity CA key database. "End entity" is the generic term for any of the lowest-level CAs, Merchant, Cardholder or Payment Gateway.

	setca_certmgr Prompt	Response
1	1. Root CA, 2. Brand CA, 3. End-Entity CA	3
2	MAIN MENU	2 (Generate key pairs and certificate requests)
3	Country Name	USA
4	Organization Name	My_Test_Merchant_CA
5	Common Name(= BrandId = BrandName[:ProductType])	Pass Note: this is the same brand ID as we used for the Brand CA before
6	key pair DB filename [eeca_keypair.db]	mca_keypair.db
7	key pair DB password	abc123

	setca_certmgr Prompt	Response
8	Enter certificate type	4 (MCA) Note: there are several certificate types listed at this point. Which you choose depends on the function that your CA will perform. For example, if you want to be both a Merchant CA and a Payment Gateway CA you would respond with 8 (MPCA). In this case we only want to set up a Merchant CA, so 4 (MCA) is appropriate.
9	Enter key usage	0 (DIGITAL_SIGNATURE)
10	key pair record label [mca_Pass_digisign]	(press Enter) Note: the end entity CAs need three key pairs (see 8.6.1, "Key and Certificate Hierarchy Required for SET" on page 153). This is the key that is used for signing authenticated messages.
11	Generate another key pair?	y
12	Enter certificate type	4 (MCA)
13	Enter key usage	1 (CERTIFICATE_SIGNATURE) Note: this is the key that is used for signing Merchants' certificates
14	key pair record label [mca_Pass_certsign]	(press Enter)
15	Generate another key pair?	y
16	Enter certificate type	4 (MCA)
17	Enter key usage	4 (KEY_AND_DATA_ENCIPHERMENT) Note: this is the key that is used for encrypting keys to send across the network.
18	key pair record label [mca_Pass_keyanddataenciph]	(press Enter)
19	Generate another key pair?	n
20	MAIN MENU	0 (Exit)

You now have four files in the current directory:

- mca_keypair.db. This contains the key pairs and certificate requests. It should not leave the Merchant CA system.
- mca_Pass_certsign.req. The certificate request for the certificate signing key.
- mca_Pass_digisign.req. The certificate request for the digital signature key.
- mca_Pass_keyanddataenciph.req. The certificate request for the key-encryption key.

The certificate requests must be sent to the Brand CA for signature.

A.1.1.8 Signing End Entity CA Certificates

Now we switch back to the role of Brand CA to sign the Merchant CA certificates.

	setca_certmgr Prompt	Response
1	1. Root CA, 2. Brand CA, 3. End-Entity CA	2

	setca_certmgr Prompt	Response
2	MAIN MENU	4 (Generate certificates for a lower level CA)
3	key DB filename [bca_key.db]	(press Enter) Note: this is the Brand CA key database
4	key DB password	abc123
5	issuer key record label	bca_Pass_certandcrsign
6	key DB filename [eeca_in_key.db]	mca_in_key.db
7	key DB password	abc123
8	certification request filename	mca_Pass_certsign.req Note: This is one of the three certificate request files created above.
9	Enter certificate type	4 (MCA)
10	Enter key usage	1 (CERTIFICATE_SIGNATURE)
11	Enter certificate validity period (in days)	365
12	Enter private key usage period (in days)	365
13	Generate another certificate?	y
14	certification request filename	mca_Pass_digisign.req
15	Enter certificate type	4 (MCA)
16	Enter key usage	0 (DIGITAL_SIGNATURE)
17	Enter certificate validity period (in days)	365
18	Enter private key usage period (in days)	365
19	Generate another certificate?	y
20	certification request filename	mca_Pass_keyanddataenciph.req
21	Enter certificate type	4 (MCA)
22	Enter key usage	4 (KEY_AND_DATA_ENCIPHERMENT)
23	Enter certificate validity period (in days)	365
24	Enter private key usage period (in days)	365
25	Generate another certificate?	n
26	MAIN MENU	0 (Exit)

You now have one more file in the current directory, mca_in_key.db. This contains the signed certificates that can be returned to the Merchant CA and merged with the key pairs.

A.1.1.9 Merge the End Entity Certificates into the Key Database

Finally we assume the role of the Merchant CA once more, to receive the certificates that the Brand CA has signed for us.

	setca_certmgr Prompt	Response
1	1. Root CA, 2. Brand CA, 3. End-Entity CA	3
2	MAIN MENU	2 (Replace certificate requests with certificates)
3	key DB filename [eeca_in_key.db]	mca_in_key.db Note: this is the file from the Brand CA, containing the signed certificates

	setca_certmgr Prompt	Response
4	key DB password	abc123
5	key pair DB filename [eeca_keypair.db]	mca_keypair.db Note: this is the original Brand key pair database, containing the private keys and public key certificate request
6	key pair DB password	abc123
7	key DB filename [eeca_key.db]	mca_key.db Note: this will be the complete Merchant CA key database, containing the key pairs and certificates
8	key DB password	abc123
9	Batch Processing: check all the key pairs...	y
10	MAIN MENU	0 (Exit)

You now have a file called mca_key.db in the current directory. This is the final Merchant CA key database for the Pass credit card brand.

A.2 Using setbrandca to Create a Certificate Hierarchy

The setbrandca command generates a complete hierarchy of CA key databases for test purposes. Figure 193 on page 290 shows a typical setbrandca invocation.

```
# setbrandca

setbrandca is a utility to generate key pairs and certificates for the root, brand, and end-entity
Certificate Authorities (CAs).

This utility will create four key-ring files to store these key pairs and certificates.

Creating the RootCA key-ring file ....
Enter the file name [rootcakey.db]:
Enter the password:
Enter the country name [DNK]: USA
Enter the organization name [EuroPaysRoot]: PassCardInc

Creating the RootCA key record for signing certificates and CRLs....
Enter the key label, so later on you can retrieve the key record using this label [rootca_certandcrlsign]:

Creating the RootCA key-ring file done. see rootcakey.db

Creating the RootCA key-ring file without root CA's private key....
Enter the file name [rootcakey0.db]:
Enter the password:

Creating the RootCA key-ring file without root CA's private key done. see rootcakey0.db

Creating the BrandCA key-ring file ....
Enter the file name [brandcakey.db]:
Enter the password:
Enter the country name [DNK]: USA
Enter the organization name [EuroPay]: PassCard
Enter the common name [EuroPay: BrandId]: Pass

Creating the BrandCA key record for signing certificates and CRLs....
Enter the key label, so later on you can retrieve the key record using this label [brandca_certandcrlsign]:

Creating the BrandCA key-ring file done. see brandcakey.db

Creating the End-EntityCA key-ring file ....
Enter the file name [eecakey.db]:
Enter the password:
Enter the country name [DNK]: USA
Enter the organization name [PBS]: ITSO

Creating the CCA key record for signing certificates....
Enter the key label, so later on you can retrieve the key record using this label [cca_certsign]:

Creating the MCA key record for signing certificates....
Enter the key label, so later on you can retrieve the key record using this label [mca_certsign]:

Creating the PCA key record for signing certificates....
Enter the key label, so later on you can retrieve the key record using this label [pca_certsign]:

Creating the CMPCA (the full combo of CCA, MCA, and PCA) key record for signing certificates....
Enter the key label, so later on you can retrieve the key record using this label [cmpca_certsign]:

Creating the CCA key record for key encryption....
Enter the key label, so later on you can retrieve the key record using this label [cca_keyencrypt]:

Creating the MCA key record for key encryption....
Enter the key label, so later on you can retrieve the key record using this label [mca_keyencrypt]:
```

Figure 193 (Part 1 of 2). setbrandca Dialog Example

```
Creating the PCA key record for key encryption....
Enter the key label, so later on you can retrieve the key record using this label [pca_keyencrypt]:

Creating the CMPCA (the full combo of CCA, MCA, and PCA) key record for key encryption....
Enter the key label, so later on you can retrieve the key record using this label [cmpca_keyencrypt]:

Creating the CCA key record for digital signature....
Enter the key label, so later on you can retrieve the key record using this label [cca_digisign]:

Creating the MCA key record for digital signature....
Enter the key label, so later on you can retrieve the key record using this label [mca_digisign]:

Creating the PCA key record for digital signature....
Enter the key label, so later on you can retrieve the key record using this label [pca_digisign]:

Creating the CMPCA (the full combo of CCA, MCA, and PCA) key record for digital signature....
Enter the key label, so later on you can retrieve the key record using this label [cmpca_digisign]:
#
# ls -l
total 232
-rw-r--r--  1 root    system    10056 May 07 17:24 brandcakey.db
-rw-r--r--  1 root    system      56 May 07 17:24 brandcakey.db.bak
-rw-r--r--  1 root    system   70056 May 07 17:29 eecakey.db
-rw-r--r--  1 root    system     56 May 07 17:29 eecakey.db.bak
-rw-r--r--  1 root    system    5056 May 07 17:16 rootcakey.db
-rw-r--r--  1 root    system     56 May 07 17:16 rootcakey.db.bak
-rw-r--r--  1 root    system    5056 May 07 17:19 rootcakey0.db
-rw-r--r--  1 root    system     56 May 07 17:19 rootcakey0.db.bak
-rw-r--r--  1 root    system      0 May 07 17:12 setbrandca.out
```

Figure 193 (Part 2 of 2). setbrandca Dialog Example

A.3 Displaying the Contents of a CA Key Database

The cmsutil command displays the contents of a key database. Figure 194 on page 292 shows an example of a display of a combined end entity CA database (Merchant, Cardholder and Payment Gateway).

```

//-----
CertificateLabel: "rootca_certandcr1sign"
RecordNo: "1"
//-----
Version: "3"
SerialNumber: "34323439353839313200"
Issuer: "CN = , OU = , O = PassCardInc, C = USA"
Subject: "CN = , OU = , O = PassCardInc, C = USA"
ValidNotBefore: "1996 01 01 00:00:00"
ValidNotAfter: "1998 01 01 00:00:00"
Signature: VERIFIED
Self-Signed: TRUE
BrandID: ROOT CA
MerchantID: NULL
MerchantAcquirerBIN: 0
AcquirerBIN: 0
//-----
// # Extensions: 8
//-----
OID: "{2, 5, 29, 35}"
ExtensionName: "AsnAuthorityKeyIdentifier"
AsnAuthorityKeyIdentifier.Critical: "FALSE" // FALSE (Our default)
AsnAuthorityKeyIdentifier.CertIssuer: "CN = , OU = , O = PassCardInc, C = USA"
AsnAuthorityKeyIdentifier.SerialNumber: "34323439353839313200"
//-----
OID: "{2, 5, 29, 15}"
ExtensionName: "AsnKeyUsage"
AsnKeyUsage.Critical: "FALSE" // FALSE (Our default)
AsnKeyUsage.KeyCertificateSigning: "TRUE"
AsnKeyUsage.Cr1Signing: "TRUE"
//-----
OID: "{2, 5, 29, 19}"
ExtensionName: "AsnBasicConstraints"
AsnBasicConstraints.Critical: "FALSE" // FALSE (Our default)
AsnBasicConstraints.IsCA: "TRUE"
AsnBasicConstraints.PathLength: "2"
//-----
OID: "{2, 5, 29, 16}"
ExtensionName: "AsnPrivateKeyUsagePeriod"
AsnPrivateKeyUsagePeriod.Critical: "FALSE" // FALSE (Our default)
AsnPrivateKeyUsagePeriod.NotBefore: "1996 01 01 00:00:00"
AsnPrivateKeyUsagePeriod.NotAfter: "1997 12 31 00:00:00"
//-----
OID: "{2, 99999, 3}"
ExtensionName: "AsnCertificateType"
AsnCertificateType.Critical: "TRUE" // TRUE (Our default)
AsnCertificateType.IsRootCertificationAuthority: "TRUE"
//-----
OID: "{2, 99999, 7}"
ExtensionName: "AsnSetQualifier"
AsnSetQualifier.Critical: "TRUE" // TRUE (Our default)
AsnSetQualifier.PolicyURL: "http://setca.ibm.com"
AsnSetQualifier.PolicyEmail: "setca@vnet.ibm.com"
//-----
OID: "{2, 5, 29, 32}"
ExtensionName: "AsnCertificatePolicies"
AsnCertificatePolicies.Critical: "FALSE" // FALSE (Our default)
AsnCertificatePolicies.PolicyIdentifier: "{1, 2, 3, 4}"
//-----
OID: "{2, 99999, 2}"
ExtensionName: "AsnHashedRootKey"
AsnHashedRootKey.Critical: "TRUE" // TRUE (Our default)
AsnHashedRootKey.DAlgorithm: "{1, 3, 14, 3, 2, 26}"
//-----

```

Figure 194 (Part 1 of 7). cmsutil Command Output Example

```

//-----
CertificateLabel: "brandca_certandcr1sign" 2
RecordNo: "2"
//-----
Version: "3"
SerialNumber: "3232343431323035383000"
Issuer: "CN = , OU = , O = PassCardInc, C = USA"
Subject: "CN = Pass, OU = , O = PassCard, C = USA"
ValidNotBefore: "1996 01 01 00:00:00"
ValidNotAfter: "1998 01 01 00:00:00"
Signature: VERIFIED
Self-Signed: FALSE
BrandID: Pass
MerchantID: NULL
MerchantAcquirerBIN: 0
AcquirerBIN: 0
//-----
// # Extensions: 7
//-----
OID: "{2, 5, 29, 35}"
ExtensionName: "AsnAuthorityKeyIdentifier"
AsnAuthorityKeyIdentifier.Critical: "FALSE" // FALSE (Our default)
AsnAuthorityKeyIdentifier.CertIssuer: "CN = , OU = , O = PassCardInc, C = USA"
AsnAuthorityKeyIdentifier.SerialNumber: "34323439353839313200"
//-----
OID: "{2, 5, 29, 15}"
ExtensionName: "AsnKeyUsage"
AsnKeyUsage.Critical: "FALSE" // FALSE (Our default)
AsnKeyUsage.KeyCertificateSigning: "TRUE"
AsnKeyUsage.Cr1Signing: "TRUE"
//-----
OID: "{2, 5, 29, 19}"
ExtensionName: "AsnBasicConstraints"
AsnBasicConstraints.Critical: "FALSE" // FALSE (Our default)
AsnBasicConstraints.IsCA: "TRUE"
AsnBasicConstraints.PathLength: "1"
//-----
OID: "{2, 5, 29, 16}"
ExtensionName: "AsnPrivateKeyUsagePeriod"
AsnPrivateKeyUsagePeriod.Critical: "FALSE" // FALSE (Our default)
AsnPrivateKeyUsagePeriod.NotBefore: "1996 01 01 00:00:00"
AsnPrivateKeyUsagePeriod.NotAfter: "1997 12 31 00:00:00"
//-----
OID: "{2, 99999, 3}"
ExtensionName: "AsnCertificateType"
AsnCertificateType.Critical: "TRUE" // TRUE (Our default)
AsnCertificateType.IsBrandCertificationAuthority: "TRUE"
//-----
OID: "{2, 99999, 7}"
ExtensionName: "AsnSetQualifier"
AsnSetQualifier.Critical: "TRUE" // TRUE (Our default)
AsnSetQualifier.PolicyURL: "http://setca.ibm.com"
AsnSetQualifier.PolicyEmail: "setca@vnet.ibm.com"
//-----
OID: "{2, 5, 29, 32}"
ExtensionName: "AsnCertificatePolicies"
AsnCertificatePolicies.Critical: "FALSE" // FALSE (Our default)
AsnCertificatePolicies.PolicyIdentifier: "{1, 2, 3, 4}"
//-----
Chain:
Signed By: CN = , OU = , O = PassCardInc, C = USA
//-----

```

Figure 194 (Part 2 of 7). cmsutil Command Output Example

```

//-----
CertificateLabel: "cca_certsign" 3
RecordNo: "3"
//-----
Version: "3"
SerialNumber: "37383035393634323900"
Issuer: "CN = Pass, OU = , O = PassCard, C = USA"
Subject: "CN = Pass, OU = , O = ITS0, C = USA"
ValidNotBefore: "1996 01 01 00:00:00"
ValidNotAfter: "1998 01 01 00:00:00"
Signature: VERIFIED
Self-Signed: FALSE
BrandID: Pass
MerchantID: NULL
MerchantAcquirerBIN: 0
AcquirerBIN: 0
//-----
// # Extensions: 7
//-----
OID: "{2, 5, 29, 35}"
ExtensionName: "AsnAuthorityKeyIdentifier"
AsnAuthorityKeyIdentifier.Critical: "FALSE" // FALSE (Our default)
AsnAuthorityKeyIdentifier.CertIssuer: "CN = , OU = , O = PassCardInc, C = USA"
AsnAuthorityKeyIdentifier.SerialNumber: "3232343431323035383000"
//-----
OID: "{2, 5, 29, 15}"
ExtensionName: "AsnKeyUsage"
AsnKeyUsage.Critical: "FALSE" // FALSE (Our default)
AsnKeyUsage.KeyCertificateSigning: "TRUE"
//-----
OID: "{2, 5, 29, 19}"
ExtensionName: "AsnBasicConstraints"
AsnBasicConstraints.Critical: "FALSE" // FALSE (Our default)
AsnBasicConstraints.IsCA: "TRUE"
AsnBasicConstraints.PathLength: "0"
//-----
OID: "{2, 5, 29, 16}"
ExtensionName: "AsnPrivateKeyUsagePeriod"
AsnPrivateKeyUsagePeriod.Critical: "FALSE" // FALSE (Our default)
AsnPrivateKeyUsagePeriod.NotBefore: "1996 01 01 00:00:00"
AsnPrivateKeyUsagePeriod.NotAfter: "1997 12 31 00:00:00"
//-----
OID: "{2, 9999, 3}"
ExtensionName: "AsnCertificateType"
AsnCertificateType.Critical: "TRUE" // TRUE (Our default)
AsnCertificateType.IsCardholderCertificationAuthority: "TRUE"
//-----
OID: "{2, 9999, 7}"
ExtensionName: "AsnSetQualifier"
AsnSetQualifier.Critical: "TRUE" // TRUE (Our default)
AsnSetQualifier.PolicyURL: "http://setca.ibm.com"
AsnSetQualifier.PolicyEmail: "setca@vnet.ibm.com"
//-----
OID: "{2, 5, 29, 32}"
ExtensionName: "AsnCertificatePolicies"
AsnCertificatePolicies.Critical: "FALSE" // FALSE (Our default)
AsnCertificatePolicies.PolicyIdentifier: "{1, 2, 3, 4}"
//-----
Chain:
Signed By: CN = Pass, OU = , O = PassCard, C = USA 4
Signed By: CN = , OU = , O = PassCardInc, C = USA
//-----

```

Figure 194 (Part 3 of 7). cmsutil Command Output Example

```

//-----
CertificateLabel: "mca_certsign"
RecordNo: "4"
//-----

<< Lines deleted - similar to CCA section above >>

//-----

//-----
CertificateLabel: "pca_certsign"
RecordNo: "5"
//-----

<< Lines deleted - similar to CCA section above >>

//-----

//-----
CertificateLabel: "cmpca_certsign"
RecordNo: "6"
//-----

<< Lines deleted - similar to CCA section above >>

//-----

//-----
CertificateLabel: "cca_keyencrypt"
RecordNo: "7"
//-----
Version: "3"
SerialNumber: "3338303933303433343300"
Issuer: "CN = Pass, OU = , O = PassCard, C = USA"
Subject: "CN = Pass, OU = , O = ITS0, C = USA"
ValidNotBefore: "1996 01 01 00:00:00"
ValidNotAfter: "1998 01 01 00:00:00"
Signature: VERIFIED
Self-Signed: FALSE
BrandID: Pass
MerchantID: NULL
MerchantAcquirerBIN: 0
AcquirerBIN: 0
//-----
// # Extensions: 8
//-----
OID: "{2, 5, 29, 35}"
ExtensionName: "AsnAuthorityKeyIdentifier"
AsnAuthorityKeyIdentifier.Critical: "FALSE" // FALSE (Our default)
AsnAuthorityKeyIdentifier.CertIssuer: "CN = , OU = , O = PassCardInc, C = USA"
AsnAuthorityKeyIdentifier.SerialNumber: "3232343431323035383000"
//-----
OID: "{2, 5, 29, 15}"
ExtensionName: "AsnKeyUsage"
AsnKeyUsage.Critical: "FALSE" // FALSE (Our default)
AsnKeyUsage.KeyEncipherment: "TRUE"
AsnKeyUsage.DataEncipherment: "TRUE"
//-----
OID: "{2, 5, 29, 19}"
ExtensionName: "AsnBasicConstraints"
AsnBasicConstraints.Critical: "FALSE" // FALSE (Our default)
AsnBasicConstraints.IsCA: "TRUE"
AsnBasicConstraints.PathLength: "0"

```

Figure 194 (Part 4 of 7). cmsutil Command Output Example

```

//-----
      OID: "{2, 5, 29, 16}"
      ExtensionName: "AsnPrivateKeyUsagePeriod"
AsnPrivateKeyUsagePeriod.Critical: "FALSE" // FALSE (Our default)
AsnPrivateKeyUsagePeriod.NotBefore: "1996 01 01 00:00:00"
AsnPrivateKeyUsagePeriod.NotAfter: "1997 12 31 00:00:00"
//-----
      OID: "{2, 99999, 3}"
      ExtensionName: "AsnCertificateType"
AsnCertificateType.Critical: "TRUE" // TRUE (Our default)
AsnCertificateType.IsCardholderCertificationAuthority: "TRUE"
//-----
      OID: "{2, 99999, 7}"
      ExtensionName: "AsnSetQualifier"
AsnSetQualifier.Critical: "TRUE" // TRUE (Our default)
AsnSetQualifier.PolicyURL: "http://setca.ibm.com"
AsnSetQualifier.PolicyEmail: "setca@vnet.ibm.com"
//-----
      OID: "{2, 5, 29, 32}"
      ExtensionName: "AsnCertificatePolicies"
AsnCertificatePolicies.Critical: "FALSE" // FALSE (Our default)
AsnCertificatePolicies.PolicyIdentifier: "{1, 2, 3, 4}"
//-----
      OID: "{2, 99999, 6}"
      ExtensionName: "AsnTunneling"
AsnTunneling.Critical: "FALSE" // FALSE (Our default)
AsnTunneling.SupportTunnel: "FALSE"
AsnTunneling.TunnelID: "{1, 2, 3, 4}"
AsnTunneling.TunnelID: "{1, 3, 5, 7, 9}"
//-----
      Chain:
      Signed By: CN = Pass, OU = , O = PassCard, C = USA 4
      Signed By: CN = , OU = , O = PassCardInc, C = USA
//-----

//-----
      CertificateLabel: "mca_keyencrypt"
      RecordNo: "8"
//-----

<< Lines deleted - similar to CCA section above >>

//-----

//-----
      CertificateLabel: "pca_keyencrypt"
      RecordNo: "9"
//-----

<< Lines deleted - similar to CCA section above >>

//-----

//-----
      CertificateLabel: "cmpca_keyencrypt"
      RecordNo: "10"
//-----

<< Lines deleted - similar to CCA section above >>

//-----

```

Figure 194 (Part 5 of 7). cmsutil Command Output Example


```

//-----
CertificateLabel: "cca_digisign"
RecordNo: "11"
//-----
Version: "3"
SerialNumber: "3337333231343630313800"
Issuer: "CN = Pass, OU = , O = PassCard, C = USA"
Subject: "CN = Pass, OU = , O = ITSO, C = USA"
ValidNotBefore: "1996 01 01 00:00:00"
ValidNotAfter: "1998 01 01 00:00:00"
Signature: VERIFIED
Self-Signed: FALSE
BrandID: Pass
MerchantID: NULL
MerchantAcquirerBIN: 0
AcquirerBIN: 0
//-----
// # Extensions: 7
//-----
OID: "{2, 5, 29, 35}"
ExtensionName: "AsnAuthorityKeyIdentifier"
AsnAuthorityKeyIdentifier.Critical: "FALSE" // FALSE (Our default)
AsnAuthorityKeyIdentifier.CertIssuer: "CN = , OU = , O = PassCardInc, C = USA"
AsnAuthorityKeyIdentifier.SerialNumber: "3232343431323035383000"
//-----
OID: "{2, 5, 29, 15}"
ExtensionName: "AsnKeyUsage"
AsnKeyUsage.Critical: "FALSE" // FALSE (Our default)
AsnKeyUsage.DigitalSignature: "TRUE"
//-----
OID: "{2, 5, 29, 19}"
ExtensionName: "AsnBasicConstraints"
AsnBasicConstraints.Critical: "FALSE" // FALSE (Our default)
AsnBasicConstraints.IsCA: "TRUE"
AsnBasicConstraints.PathLength: "0"
//-----
OID: "{2, 5, 29, 16}"
ExtensionName: "AsnPrivateKeyUsagePeriod"
AsnPrivateKeyUsagePeriod.Critical: "FALSE" // FALSE (Our default)
AsnPrivateKeyUsagePeriod.NotBefore: "1996 01 01 00:00:00"
AsnPrivateKeyUsagePeriod.NotAfter: "1997 12 31 00:00:00"
//-----
OID: "{2, 9999, 3}"
ExtensionName: "AsnCertificateType"
AsnCertificateType.Critical: "TRUE" // TRUE (Our default)
AsnCertificateType.IsCardholderCertificationAuthority: "TRUE"
//-----
OID: "{2, 9999, 7}"
ExtensionName: "AsnSetQualifier"
AsnSetQualifier.Critical: "TRUE" // TRUE (Our default)
AsnSetQualifier.PolicyURL: "http://setca.ibm.com"
AsnSetQualifier.PolicyEmail: "setca@vnet.ibm.com"
//-----
OID: "{2, 5, 29, 32}"
ExtensionName: "AsnCertificatePolicies"
AsnCertificatePolicies.Critical: "FALSE" // FALSE (Our default)
AsnCertificatePolicies.PolicyIdentifier: "{1, 2, 3, 4}"
//-----
Chain:
Signed By: CN = Pass, OU = , O = PassCard, C = USA
Signed By: CN = , OU = , O = PassCardInc, C = USA
//-----

```

Figure 194 (Part 6 of 7). cmsutil Command Output Example

```

//-----
CertificateLabel: "mca_digisign"
RecordNo: "12"
//-----

<< Lines deleted - similar to CCA section above >>

//-----

//-----
CertificateLabel: "pca_digisign"
RecordNo: "13"
//-----

<< Lines deleted - similar to CCA section above >>

//-----

//-----
CertificateLabel: "cmpca_digisign"
RecordNo: "14"
//-----

<< Lines deleted - similar to CCA section above >>

//-----

```

Figure 194 (Part 7 of 7). cmsutil Command Output Example

Note:

- 1** The Root key certificate is in all key databases so that the database owner can validate a certificate by following the chain of signatures up to the top of the hierarchy. This point of shared trust is central to the use of certificates for authentication.
- 2** Each lower-level CA also needs the certificates of all CAs in the chain of trust between it and the root. In this case we have displayed an end-entity CA database and you can see that the Brand CA certificate is listed. If this CA presents a certificate to another SET entity, the Brand certificate allows it to establish a chain of trust back to the root, even though the other SET entity may not be under the same brand CA.
- 3** The end entity CA has three key pairs, for certificate signing, digital signatures and for key encryption. This is the certificate signing key certificate. This CA is multi-purpose; it provides services for cardholders (CCA), merchants (MCA), payment gateways (PCA), and a combination of all three (CMPCA). For simplicity, we have removed the contents of all certificate entries except the cardholder CA from the display.
- 4** Each certificate section includes the trust chain by which the signature can be traced back to the Root CA.
- 5** This is the certificate entry for the cardholder CA key encryption key.
- 6** This is the certificate entry for the cardholder CA digital signature key.

Appendix B. Cryptographic Processes

Modern secure protocols are based on a number of underlying cryptographic techniques. In this section we explain the commonly used techniques.

We discuss five subject areas:

1. Symmetric key (or bulk) encryption
2. Public key encryption
3. Secure hash (or digest) functions
4. Digital signatures and other combinations of the above techniques
5. Certification mechanisms

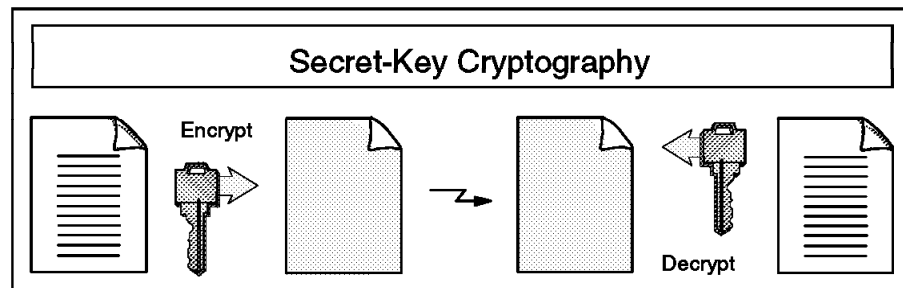
If you want to learn more about cryptography, we recommend the RSA Frequently Asked Questions document at <http://www.rsa.com/rsalabs/newfaq>.

B.1 Symmetric Key Encryption

This is a grown-up version of the kind of secret code that most of us played with at some time during childhood. Usually these use a simple character replacement algorithm; if you want to encrypt a message, you just replace each letter of the alphabet with another. For example:

Original letter: ABCDEFGHIJKLMNOPQRSTUVWXYZ
Replacement : GHIJKLMNOPQRSTUVWXYZABCDEF

In this case the letters in the alphabet have just been shifted seven places to the right, so HELLO WORLD would translate to NKRRU CUXRJ. The premise on which this code is based is that both the sender and the receiver know a common key, in this case the number of places to shift the letters. This *shared secret* allows the receiver of the message to reverse the encryption process and read the scrambled message. Symmetric encryption algorithms used by computers have the same elements as this simple example, namely a mechanism to scramble the message (also known as a cipher) and a shared secret (a key) that allows the receiver to unscramble the encrypted message.



4978497802

Figure 195. Symmetric Key Encryption

The strength of a symmetric key cipher of this kind is dictated by a number of factors. For example, it is important that it effectively randomizes the input, so that two related clear-text messages do not produce similar encrypted results. Our childish example falls down badly in this area, because each letter always

converts to the same encrypted result, and because it does not encrypt spaces. The kindergarten cryptanalyst can quite easily break the code by knowing that any one-letter word is likely to be an A. For full-strength symmetric ciphers, much of the work of the cryptanalyst involves trying to find patterns in the result of the algorithm, to use as a shortcut to breaking the code.

If the encryption algorithm has no flaws of this kind, the main factor governing its strength is the size of the *key space*; that is, the total number of possible shared secrets. Once again our simple example falls short, because it only has 25 possible keys. We could mount a *brute force* attack very easily by trying each key in turn until we found a message that makes sense. Real symmetric ciphers use numeric keys, usually of between 40 and 128 bits in size. Even for the smallest of these a brute force attack has to try, on average, 2 to the power 39 or about 550,000,000,000 possible keys. Each extra bit of key size doubles the key space.

B.1.1 Characteristics of Symmetric Key Algorithms

There are a number of symmetric key ciphers in use. We have listed the main ones below, together with a brief description of their capabilities. However, they also share several common characteristics:

- They are fast and cause a relatively low system overhead. For this reason symmetric key encryption is often referred to as bulk encryption, because it is effective on large data volumes.
- The algorithms are published openly and there are no commercial licensing issues to be considered in implementing them.
- They all fall under the control of the U.S. National Security Agency export restrictions. The precise operation of these restrictions is not a simple matter, but in essence this means that:
 1. Any software incorporating cryptographic technology that is exported by a U.S. company has to have a special export license.
 2. If the product includes symmetric encryption code that can be used for encrypting an arbitrary data stream, the license will *only* allow unrestricted export *if* the key size is smaller than a given, NSA-specified, value.

What this means is that to export full-strength cryptography, a company has to have a special license for each customer. Such licenses are only issued for customers that the U.S. government considers to be friendly, such as major banks and subsidiaries of U.S. companies. Until recently the threshold key size for a general export license was 40 bits. Several challenges have shown that a brute force attack can be mounted against a 40-bit key with relatively modest computing power. A government announcement in October 1996 opened the door to the use of larger keys, initially up to 56 bits, with the promise of unlimited key sizes when the computer industry develops effective key recovery technology. (Key recovery means that the key for a session can be discovered, given the knowledge of some other, master, key.) 56 bits may not sound a lot better than 40, but in fact it is 2 to the power 16, or 65,536 times more difficult to crack.

B.1.1.1 Common Symmetric Key Algorithms

DES The Data Encryption Standard is the most commonly used bulk cipher. It was originally developed by IBM in 1977 and it has resisted all attempts at cryptanalysis. DES breaks the data to be encrypted into a sequence of 64-bit blocks and then uses a 56-bit key to apply a sequence of mathematical transforms to it. There are a number of variations on the standard DES algorithm, such as cipher block chaining, in which each block of data is XOR'd with the previous block before encryption, and triple-DES in which DES is applied three times in succession.

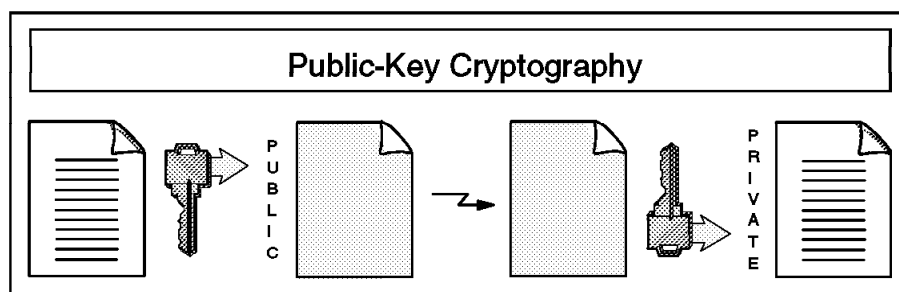
RC2/RC4 These related ciphers were developed by RSA Data Security, Inc. RC2 is a block cipher similar to DES, whereas RC4 operates on a stream of data. They both use a 128-bit key, but they support key masking. This means that part of the key may be set to a known value so that the effective key size is whatever remains from the 128-bit total. This has been used to good effect in producing 40-bit versions of software products for export.

IDEA The International Data Encryption Algorithm is another block cipher, in the mold of DES. It uses a 64-bit block size and a 128-bit key. IDEA is the bulk encryption algorithm used by Pretty Good Privacy (PGP).

B.2 Public Key Encryption

A non-mathematician can intuitively understand how a symmetric key cipher works by extrapolating from a familiar base. However, public key mechanisms are much less accessible to the lay person. In fact, it sometimes seems more like magic than technology. The fundamentals of public key encryption are:

1. Instead of a single encryption key, there are two related keys, a *key pair*.
2. Anything encrypted using one of the two keys can *only* be decrypted with the other key of the pair.



4978497801

Figure 196. Public Key Encryption

Let us say that two people, Bob and Alice, want to exchange messages using a public key algorithm. Alice generates a key pair and places one of the keys, the *private key* in a safe place. She sends the other half of the key pair (called, naturally, the public key) to Bob. Bob can now encrypt a message using the public key and only the owner of the matching private key, Alice, can decrypt it. Of course, if Alice wants to send a reply, Bob needs to create his own key pair and send the public key to Alice. The big advantage of this mechanism over the symmetric key mechanism is that there is no longer any secret to share. In fact,

it does not matter who has the public key, because it is useless without the matching private key.

There is one further advantage that public key gives us. In the above example, imagine that Alice uses her private key to encrypt a message and sends it to Bob. The message that is sent between them is still scrambled, but it is no longer private, because anyone with the public key can decrypt it (and we have said that we do not care who has the public key). So, what *can* we use this message from Alice for? The answer is: authentication. Because only Alice has access to the private key that created the message, it can *only* have come from her.

B.2.1 Characteristics of Public Key Encryption

Public key algorithms can trace their ancestry back to the Diffie-Hellman key exchange mechanism. Diffie-Hellman is not a general purpose encryption scheme, but rather a method of exchanging a secret key. There is only one widely used general purpose public key mechanism, the Rivest, Shamir and Adelman (RSA) algorithm, which is the property of RSA Data Security, Inc. Public key, like any encryption mechanism, relies on the fact that certain mathematical problems are very hard to solve. The problem that the RSA algorithm relies on is the factorization of large numbers. For a detailed description of the mathematics behind RSA public key encryption, refer to <http://www.rsa.com/rsalabs/newfaq/q8.html>.

Clearly, public key has a big practical advantage over symmetric key, in that there is no need to securely share a secret between the sender and receiver. However, the RSA algorithm is a much less efficient encryption technique than any commercial symmetric key algorithm, by a factor of 100 or more. It is therefore not a good choice for encryption of bulk data.

RSA is subject to the same US export restrictions as symmetric algorithms. However, the key in this case is actually a very large number. Very approximately, an RSA key size of 1024 bits corresponds to a full-strength symmetric key of 64 bits or more.

B.3 Secure Hash Functions

The third tool in our encryption armory is not actually an encryption mechanism at all. A secure hash is a way of creating a kind of “fingerprint” of a message. A secure hash function has three main attributes:

1. It takes a message of any size and generates a small, fixed size, block of data from it (called a *message digest*). Re-executing the hash function on the same source data will always yield the same resulting digest.
2. It is not predictable in operation. That is to say, a small change in the source message will have an unpredictably large effect on the final digest.
3. It is, for all intents and purposes, irreversible. In other words there is no way to derive the source data, given its digested form.

What use is a secure hash function? Its main function is to detect whether a piece of data has been modified or not. We see in B.4, “Combinations of Cryptographic Techniques” on page 303 how this is used in combination with RSA to generate a digital signature.

There are two secure hash algorithms in common use. The most widely implemented is MD5, which was developed by RSA Data Security, Inc. This generates a 128-bit digest from any length of input data. It is described in RFC1321. The other algorithm that is becoming increasingly common is the US government-developed Secure Hash Standard (SHS). This produces a 160-bit digest, slightly larger than MD5.

B.4 Combinations of Cryptographic Techniques

Two combinations of techniques are used very commonly, so we outline them here:

- Public key delivery of a symmetric key

It is most efficient to use a symmetric key algorithm for bulk data, but first you have to copy the key from sender to receiver. A common approach is to use a public key algorithm to protect the key transfer process.

- Digital Signatures

You do not always want to encrypt data in transit. Very often the contents of a message may not be secret, but you do want to be sure that it really came from the apparent sender. If Alice wants to prove to Bob that it was really she who sent him a message she will attach a signature to the message, exactly as she would if writing a real letter on paper. In the digital case she creates the signature by first generating a digest of the message and then encrypting the digest with her private key. When Bob receives the message he first decrypts the digest (with Alice's public key) and then generates his own digest from the received message. If the two digests match, Bob knows that:

1. The message is the same as when it was sent (because the digest is the same).
2. It really was sent by Alice, because only she has the private key.

B.5 Public Key Certificates

We have seen how public key cryptography overcomes the problem of having to pass a secret from sender to receiver. There is still a need to send a key, but now it is a public key, which anyone can see because it is only useful to an attacker if he also has the private key. However, this overlooks one crucial element of trust: how can you be sure that the public key really came from whom you think it came from?

One answer is to only pass public keys to someone you know. Bob and Alice have known each other for a long time (in fact people have started to talk), so they could share their public keys by exchanging diskettes. For normal cases, however, you need some way to be sure that a public key is authentic.

The mechanism for doing this is the *public key certificate*. This is a data structure containing a public key, plus details of the owner of the key, all digitally signed by some trusted third party. Now when Alice wants to send Bob her public key she actually sends a certificate. Bob receives the certificate and checks the signature. As long as it has been signed by a certifier that he trusts, he can accept that this really is Alice's key.

Appendix C. Secure Sockets Layer

The Secure Sockets Layer (SSL) protocol was originally created by Netscape Inc., but now it is implemented in World Wide Web browsers and servers from many vendors. SSL makes use of a number of cryptographic techniques, such as public key and symmetric key encryption, digital signatures and public key certificates.

SSL has two main objectives:

1. To ensure confidentiality, by encrypting the data that a client and server send.
2. To provide authentication of the session partners, using RSA public key methods. Most current implementations only require the server to be authenticated in this way, although the protocol does allow for client authentication.

Although SSL is normally used to provide secure encapsulation of HTTP, it can be applied to any TCP/IP application. A number of implementations for other protocols have been developed.

There are two parts to SSL:

- The *handshake*, in which the session partners introduce themselves and negotiate session characteristics.
- The *record protocol*, in which the session data is exchanged in an encrypted form.

C.1.1.1 The SSL Handshake

Figure 197 on page 306 shows a simplified version of the SSL handshake.

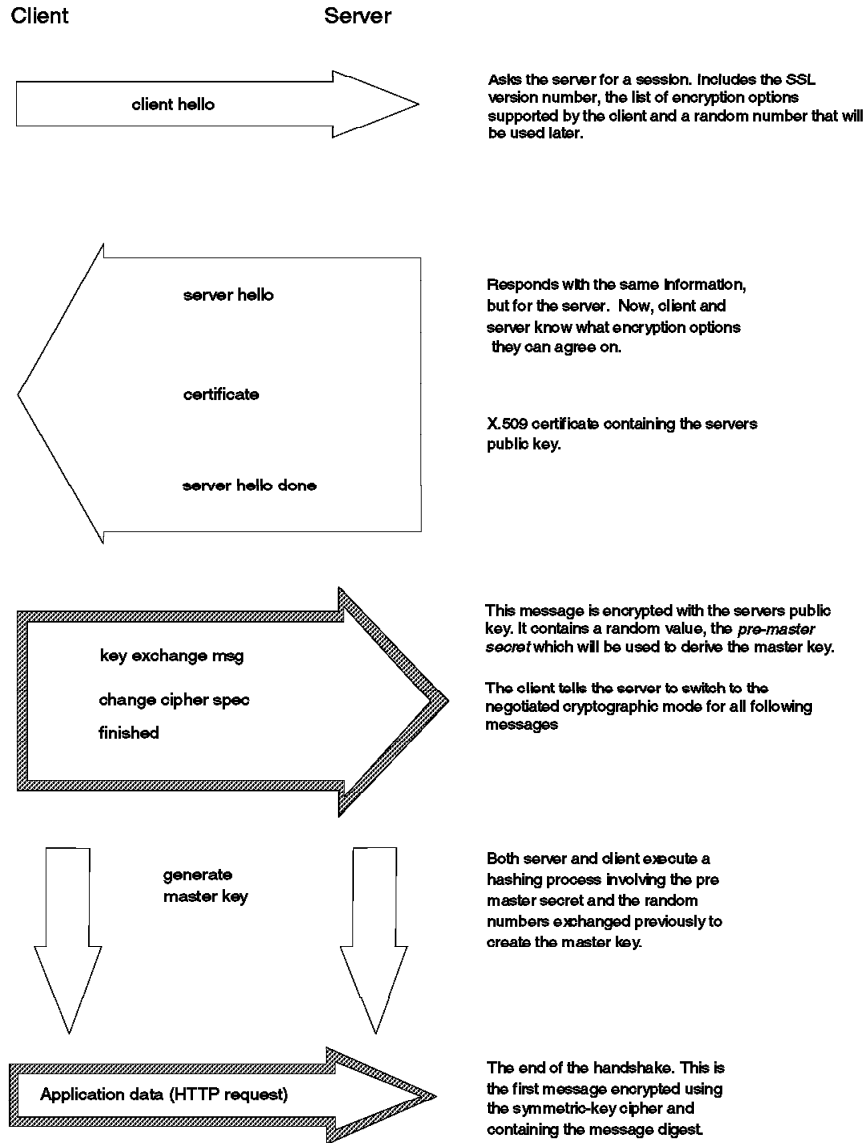


Figure 197. SSL Handshake Process

The two *hello* messages are used to exchange information about the capabilities of the client and server. This includes a list of *ciphersuites*, combinations of cryptographic algorithms and key sizes, that the client and server will accept for the session. Also the server provides a public key certificate. This is the method by which SSL checks identity and authenticity of the session partner. In this example we only show the steps for server authentication, but if client authentication was required there would be another message exchange using the client public key. Finally the session partners separately generate an encryption key, the *master key* from which they derive the keys to use in the encrypted session that follows.

You can see from this example that there is significant additional overhead in starting up an SSL session compared with a normal HTTP connection. The protocol avoids some of this overhead by allowing the client and server to retain

session key information and to resume that session without negotiating and authenticating a second time.

C.1.1.2 The SSL Record Protocol

Once the master key has been determined, the client and server can use it to encrypt application data. The SSL record protocol specifies a format for these messages. In general they include a message digest to ensure that they have not been altered and the whole message is encrypted using a symmetric cipher. Usually this uses the RC2 or RC4 algorithm, although DES, triple-DES and IDEA are also supported by the specification.

The U.S. National Security Agency (NSA), a department of the United States federal government imposes restrictions on the size of the encryption key that may be used in software exported outside the U.S. These rules have recently been relaxed, but the old regulations still affect current server and browser software. The effect is to limit the key to a size of 40 bits. The RC2 and RC4 algorithms achieve this by using a key in which all but 40 bits are set to a fixed value. International (export) versions of software products have this hobbled security built into them. SSL caters for mismatches between the export and nonexport versions in the negotiation phase of the handshake. For example, if a U.S. browser tries to connect with SSL to an export server, they will agree on export-strength encryption.

C.1.1.3 Using SSL in Practice

The negotiation and authentication process of the SSL handshake is rather complex, but fortunately it is transparent to the user. In fact, all that a user has to do to enter an SSL connection is to alter the URL prefix from http: to https:. This acts as a trigger to the browser software to start the SSL handshake. Once the SSL connection has been established the browser gives the user a visual indication. In the case of Netscape Navigator this is a key symbol at the lower left of the screen (see Figure 198).

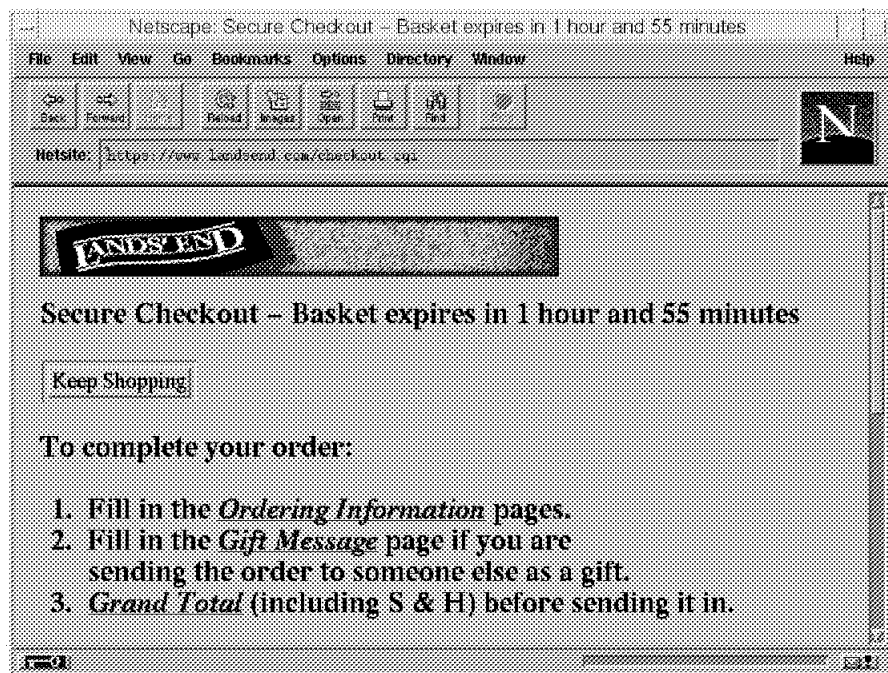


Figure 198. SSL Session Indicator in Netscape

From the point of view of the Webmaster, SSL is also quite simple. First the Webmaster needs to generate a key pair for the server, and obtain a certificate for it. Normally this involves providing documentation to a certifying authority and paying an annual fee, although it is also possible to generate your own certificates for testing and intranet use. We discuss how SSL uses certifying authorities below.

Once the server certificate has been installed, the Webmaster can create HTML links with an https: prefix to cause SSL to be invoked. For example:

```
<A HREF=https://my_server/secret.doc>Go into SSL</A>
```

C.1.1.4 SSL and Certifying Authorities

You can see from Figure 197 on page 306 that authentication in SSL depends on the client being able to trust the server's public key certificate. A certificate links the description of the owner of a key pair to the public part of the key. The validity of a certificate is guaranteed by the fact that it is signed by some trusted third party, the *certifying authority* (CA). But how does a certifying authority become trusted? In the case of an SSL-capable browser, the certificates of trusted authorities are kept in a key database, sometimes called a key ring file. The list of top-level authorities is pre-installed when you get the browser. Figure 199 shows part of the list of CA certificates provided by Netscape Navigator.

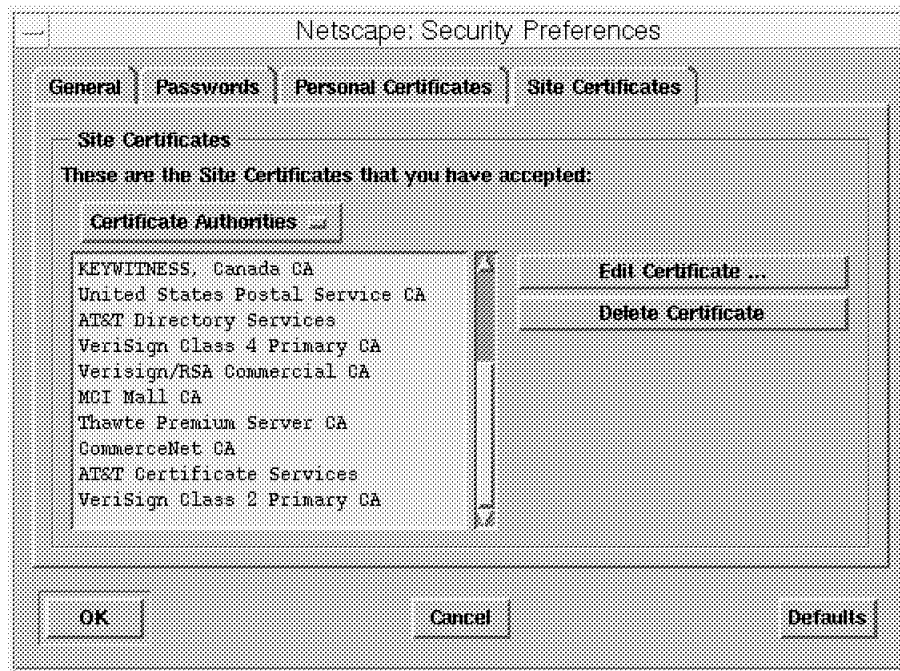


Figure 199. Certifying Authorities in Netscape Navigator

This approach has the benefit of being very simple to set up; a browser can authenticate any server that obtains a public key certificate from one of the CAs in the list, without any configuration or communication with the CA required. However, there are also some problems arising from this method. The first is that a new CA will not automatically be recognized until the browser (wherever it may be in the world) has been updated. The second problem is that there is no way for certificate revocations to be processed. (For example, if a CA determines that a public key owner is fraudulent after a certificate is issued, the

certificate will remain useable until it expires, without the end user being aware of any concern.)

The browser vendors have a two-part scheme to overcome the first problem (new CAs):

1. There is a special MIME format, `application/x-x509-ca-cert`, which allows a browser to receive a new CA certificate that has been signed by one of the known CAs. However, this method is not widely implemented yet. (See <http://home.netscape.com/eng/security/downloadcert.html> for details of download formats.)
2. The browsers will tell you that you are connecting to a secure server whose certificate is not from a known CA. You can then elect to trust just that server (that is *not* the CA that signed the server's certificate).

This latter approach is useful in an intranet environment, where you may want to create an enterprise-only CA. For Internet applications you should purchase a certificate from one of the known CAs, such as VeriSign. Figure 200 and Figure 201 on page 310 show how Netscape Navigator warns you about a site that has a certificate from an unknown CA, and then allows you to choose to trust the site anyway. The Microsoft Internet Explorer approach is similar.



Figure 200. Server Certificate Signed by Unknown CA

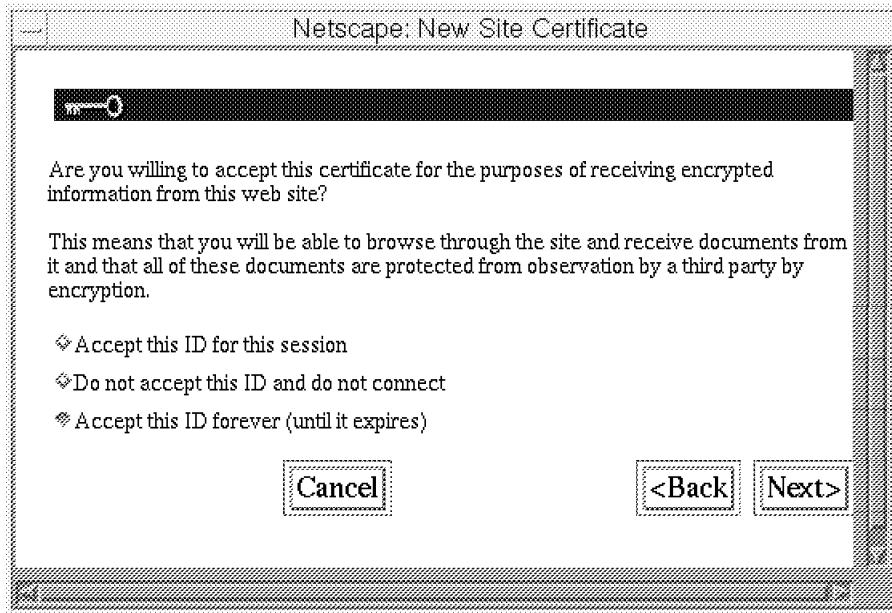


Figure 201. Option to Add Trusted Server. Note that this will not make the CA trusted, so you will see the same warnings if you access a second server with a certificate signed by the same CA.

Appendix D. Special Notices

This publication is intended to help administrators, programmers, consultants and services providers understand the Secure Electronic Transactions (SET) standard and the implementation of it by IBM products and services. The information in this publication is not intended as the specification of any programming interfaces that are provided by any IBM products. See the PUBLICATIONS section of the IBM Programming Announcement for the relevant products 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, 500 Columbus Avenue, Thornwood, NY 10594 USA.

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.

The following document contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples contain the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

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:

AIX	AS/400
DATABASE 2	DB2
IBM	RISC System/6000
RS/6000	System/390

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.

Java and HotJava are trademarks of Sun Microsystems, Incorporated.

Microsoft, Windows, Windows NT, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

Pentium, MMX, ProShare, LANDesk, and ActionMedia are trademarks or registered trademarks of Intel Corporation in the U.S. and other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Other company, product, and service names may be trademarks or service marks of others.

Appendix E. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

E.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see "How to Get ITSO Redbooks" on page 315.

- *Building a Firewall using Secured Network Gateway*, SG24-2577
- *Net.Commerce Examples*, SG24-4933

E.2 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

CD-ROM Title	Subscription Number	Collection Kit Number
System/390 Redbooks Collection	SBOF-7201	SK2T-2177
Networking and Systems Management Redbooks Collection	SBOF-7370	SK2T-6022
Transaction Processing and Data Management Redbook	SBOF-7240	SK2T-8038
AS/400 Redbooks Collection	SBOF-7270	SK2T-2849
RS/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
RS/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041
Application Development Redbooks Collection	SBOF-7290	SK2T-8037
Personal Systems Redbooks Collection	SBOF-7250	SK2T-8042

E.3 Other Publications

These publications are also relevant as further information sources:

- *Net.Commerce V1R1 Customization Guide and Reference*, SC09-2370
- *Net.Commerce V1R1 Open for Business*, SC09-2372
- *Net.Commerce V1R1.1 Installation and Operation*, GC09-2373

How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at URL <http://www.redbooks.ibm.com>.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **PUBORDER** — to order hardcopies in United States
- **GOPHER link to the Internet** - type GOPHER.WTSCPOK.ITSO.IBM.COM
- **Tools disks**

To get LIST3820s of redbooks, type one of the following commands:

```
TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
```

To get BookManager BOOKs of redbooks, type the following command:

```
TOOLCAT REDBOOKS
```

To get lists of redbooks, type one of the following commands:

```
TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET LISTSERV PACKAGE
```

To register for information on workshops, residencies, and redbooks, type the following command:

```
TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1996
```

For a list of product area specialists in the ITSO: type the following command:

```
TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ORGCARD PACKAGE
```

- **Redbooks Home Page on the World Wide Web**
<http://w3.itso.ibm.com/redbooks>
- **IBM Direct Publications Catalog on the World Wide Web**
<http://www.elink.ibm.link.ibm.com/pb1/pb1>

IBM employees may obtain LIST3820s of redbooks from this page.

- **REDBOOKS category on INEWS**
- **Online** — send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL
- **Internet Listserver**

With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an e-mail note to announce@webster.ibm.link.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank). A category form and detailed instructions will be sent to you.

Redpieces

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Home Page (<http://www.redbooks.ibm.com/redpieces.htm>). 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.

How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** — send orders to:

In United States:	IBMMAIL usib6fpl at ibmmail	Internet usib6fpl@ibmmail.com
In Canada:	caibmbkz at ibmmail	lmannix@vnet.ibm.com
Outside North America:	dkibmbsh at ibmmail	bookshop@dk.ibm.com

- **Telephone orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	(long distance charges apply)
(+45) 4810-1320 - Danish	(+45) 4810-1020 - German
(+45) 4810-1420 - Dutch	(+45) 4810-1620 - Italian
(+45) 4810-1540 - English	(+45) 4810-1270 - Norwegian
(+45) 4810-1670 - Finnish	(+45) 4810-1120 - Spanish
(+45) 4810-1220 - French	(+45) 4810-1170 - Swedish

- **Mail Orders** — send orders to:

IBM Publications Publications Customer Support P.O. Box 29570 Raleigh, NC 27626-0570 USA	IBM Publications 144-4th Avenue, S.W. Calgary, Alberta T2P 3N5 Canada	IBM Direct Services Sortemosevej 21 DK-3450 Allerød Denmark
--	--	--

- **Fax** — send orders to:

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	(+45) 48 14 2207 (long distance charge)

- **1-800-IBM-4FAX (United States) or (+1)001-408-256-5422 (Outside USA)** — ask for:

Index # 4421 Abstracts of new redbooks
Index # 4422 IBM redbooks
Index # 4420 Redbooks for last six months

- **Direct Services** - send note to softwareshop@vnet.ibm.com

- **On the World Wide Web**

Redbooks Home Page	http://www.redbooks.ibm.com
IBM Direct Publications Catalog	http://www.elink.ibm.com/pbl/pbl

- **Internet Listserver**

With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserv. To initiate the service, send an e-mail note to announce@webster.ibm.com with the keyword `subscribe` in the body of the note (leave the subject line blank).

Redpieces

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Home Page (<http://www.redbooks.ibm.com/redpieces.htm>). 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.

IBM Redbook 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.

Index

Special Characters

/etc/services 193
/tsm 177
.com 4
.edu. 4
.net 5
.tsmrc 182

A

Access Control 245, 258
Acquirer
 definition of 11
 relationship to merchant 13
Administrators for SETCA 160
ADSM 177
andRegister 186
anonymous payments 8
Authentication 17

B

bankcard association 11
BCA 53
bibliography 313
Brand 90, 122
 definition of 11
Brand Certificate Authority, see BCA
brand logo image file 148, 149
Bulk encryption 19

C

CA
 definition of 12
 in SET 51
CA certificate (installing) 153
CAPORT 165
capture 25, 35
 initiating from Net.Commerce SET Extension 115
CardCInitReq request 161
Cardholder
 certificate registration process 141
 definition of 11
 public key certificate 51
 relationship to card issuer 13
 requesting a public key certificate 58
 wakeup server
Cardholder Certificate Authority, see CCA
Carnegie Mellon University. 8
CCA 139
certificate chain 53, 155
Certificate request registration form 58

Certificates held by merchant 109
Certification Authority, see CA
Certification protocol in SET 51, 55, 153
CGI 106
 nph-msrvr 106
CGI-BIN 245
cheese 5
CommercePOINT Gateway 83
 adding a listener 193
 defining applications 188
 installation and customization 171
 pre-requisites 177
CommercePOINT Till 83, 107
CommercePOINT Wallet 83
 defining cards 88
 installation 85
Common Gateway Interface, see CGI
Confidentiality 17
Cryptography
 CPU utilization of 85
 use in SET 4, 7, 17

D

DB2 101, 179
 configuring for Net.Commerce 221
Defining cards in CommercePOINT Wallet 88
demography of Web users 5
DES 21
Digest functions, see Secure hash
DigiCash 8
Digital Cash 8
digital envelope 21
digital signature 21, 51, 58, 89
distinguished name 94
domain names 4
dual signature 22, 26

E

Eavesdropping 6
Electronic cash, see Digital cash
electronic wallet 9, 86

F

firewall 171
First Virtual 7

H

Hierarchy of trust 52, 93, 155
httpd 104, 226
https: prefix 96

I

IBM Internet Connection Secure Server, see ICSS
IBM Registry for SET 83, 139
 approver function 159
 components of 139
 configuration 148
 default user ID and password 157
 defining administrators 160
 files locations 147
 Java class files 140
 pre-requisites 143
 registration scenario 141
 starting the server 157
 Wakeup Server 161
ICSS 101
iKP 7
Impersonation 7
init_router 173
init_tsm 197
instance owner user ID 221, 243
Internet marketplace 3
Interoperability 17
IP ports 109, 119, 121, 140, 157, 165, 175, 193
Issuer 11
ITSO Corner 101

J

JEPI 9
Joint Electronic Payments Initiative, see JEPI

K

Key hierarchy for SET 153
key pairs 153
key ring file 111, 120, 226
key-exchange key 21, 27, 52, 75, 112
key.db 112, 118
KEYDB.PATH directive 153, 156

L

Lab Environment 83
LAD 184
legacy credit card processing systems 171
libcms.a 112
local application directory, see lad

M

MACRO_PATH 278
macros 107
 customizing for SET payments 123
mall definition in Net.Commerce 241
man-in-the-middle attack 17
marked for capture 117
MCA 53, 120, 139

Merchant

 definition of 11
 implementation in Net.Commerce 101
 key pairs required 112
 public key certificates 52, 120
 relationship to acquirer 13
 requesting public key certificates 69
Merchant Certificate Authority, see MCA
message digest 21
Micropayments 8
MIME types of SET requests 87
 authchk user exit 166
 set-payinquiry-initiation 87
 set-payment-initiation 87, 99
 set-reginquiry-initiation 88
 set-registration-initiation 87, 92, 162
Minipay 8
mpgcert 165
mserver.conf 111

N

narrowcasting 5
ncsetd 109
Net.Commerce 83, 101
 components of 101
Net.Commerce Administrator 251
Net.Commerce Server 106, 215
 configuring the store 262
 creating the mall 251
 customizing store macros 277
 customizing the online store 215, 241
 default user ID and password 227
 installing on Windows NT 223
 pre-requisites 215
 product categories 270
Net.Commerce Server Manager 106
Net.Commerce Server Manager 241
NetBill 8
NETCOMM.EXE 87
Netscape Navigator 87
nph-msrvr 114
nph-msrvr CGI program 106
nph-pay 109, 114
nrsetca group 147, 156

O

OI 26, 28, 44
ORD_LST_PEN 123
ORD_SETFAIL 114
Order Details 129
Order Failure 133
Order instruction, see OI
orderdsp.d2w 123

P

- paging space 217
- Pass brand (credit card) 148, 150
- payment authorization 35
- Payment authorization process 25
- Payment Capture Process 43
- Payment Gateway 171
 - adding a listener 193
 - authorization flow 175
 - definition of 12
 - interface definitions 188
 - role in SET payment protocol 25
 - translation exit 176, 198
- Payment Gateway Application 171
- Payment Gateway Certificate Authority, see PCA
- Payment Gateway Legacy Application 171
- Payment Gateway Listener 171
 - description of 174
- Payment Gateway Transaction Monitor 171
 - interfaces to 173
- Payment instruction, see PI
- Payment systems
 - digital cash 8
 - for credit cards 7
 - in general 3
- PCA 139
- pgconfig 192
- pgmaster 174
- PGTM Initialization 197
- PI 26, 28
- PINIT 26
- Public key
 - certification 12
 - generating key pairs 65, 305
- Public key certification 19, 51, 153
- Public key encryption 19
- Purchase process 25
- purchase request 32
- Purchase validation 114

R

- RCA 53
- Root certificate 93
- Root Certificate Authority, see RCA
- Root Key 53, 93
- Root public key 155

S

- s_ord.htm 115
- saf_daemon 173
- Sample Acquirer 118
- secret value 52
- Secure Electronic Transactions, see SET
- Secure hash 19, 21, 27
- self-signed certificate 54, 155, 226

SET

- CA hierarchy 12
- certification protocol details 51, 55
- limits of the protocol 15, 17
- message formats in general 19
- origins of 7
- payment protocol details 17, 25
- roles and responsibilities 11
- specification documents 17
- use of cryptography 4
- use of public keys 12
- set-registration-initiation 143, 162
- SETACQAUTHRESP table 113, 122
- SETACQUIRER table 113, 119, 121
- setadmin 140, 157
- SETBRAND table 114, 122
- setbrandca 156
- setca 157
- setca_certmgr 155
- SETMERCH table 113, 120, 165
- SETURESPPREA 137
- SHA-1 21
- shipping provider information 264
- shttpsrv 174
- signature key 21, 27, 52, 112
- Site Manager 107
- Smart Cards 9
- SNA Server/6000 177
- SSL 96, 104, 226, 252
- Store and forward facility (TSM) 187
- Store Manager 107
- SuperSET 9
- Symmetric key encryption, see Bulk encryption

T

- Template Designer 107
- TMVOLGRP environmental variable 179
- tran_table_config 183
- Transaction Service Manager, see TSM
- translation exit (payment gateway) 176, 308
- Translation exit (payment gateway) 187, 198
- Trojan Horse 7
- Trust 3
 - hierarchy of 52
- trust chain 28, 52, 63
- TSM 173
 - application ID number 186
 - store and forward facility 187

U

- URL for certification process 91
- User IDs for payment gateway 184

W

- Windows NT 222

ITSO Redbook Evaluation

Secure Electronic Transactions: Credit Card Payment on the Web in Theory and Practice
SG24-4978-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at <http://www.redbooks.com>
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redeval@vnet.ibm.com

**Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)**

Overall Satisfaction _____

Please answer the following questions:

Was this redbook published in time for your needs? Yes____ No____

If no, please explain:

What other redbooks would you like to see published?

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)



This soft copy for use by IBM employees only.

Printed in U.S.A.

SG24-4978-00

