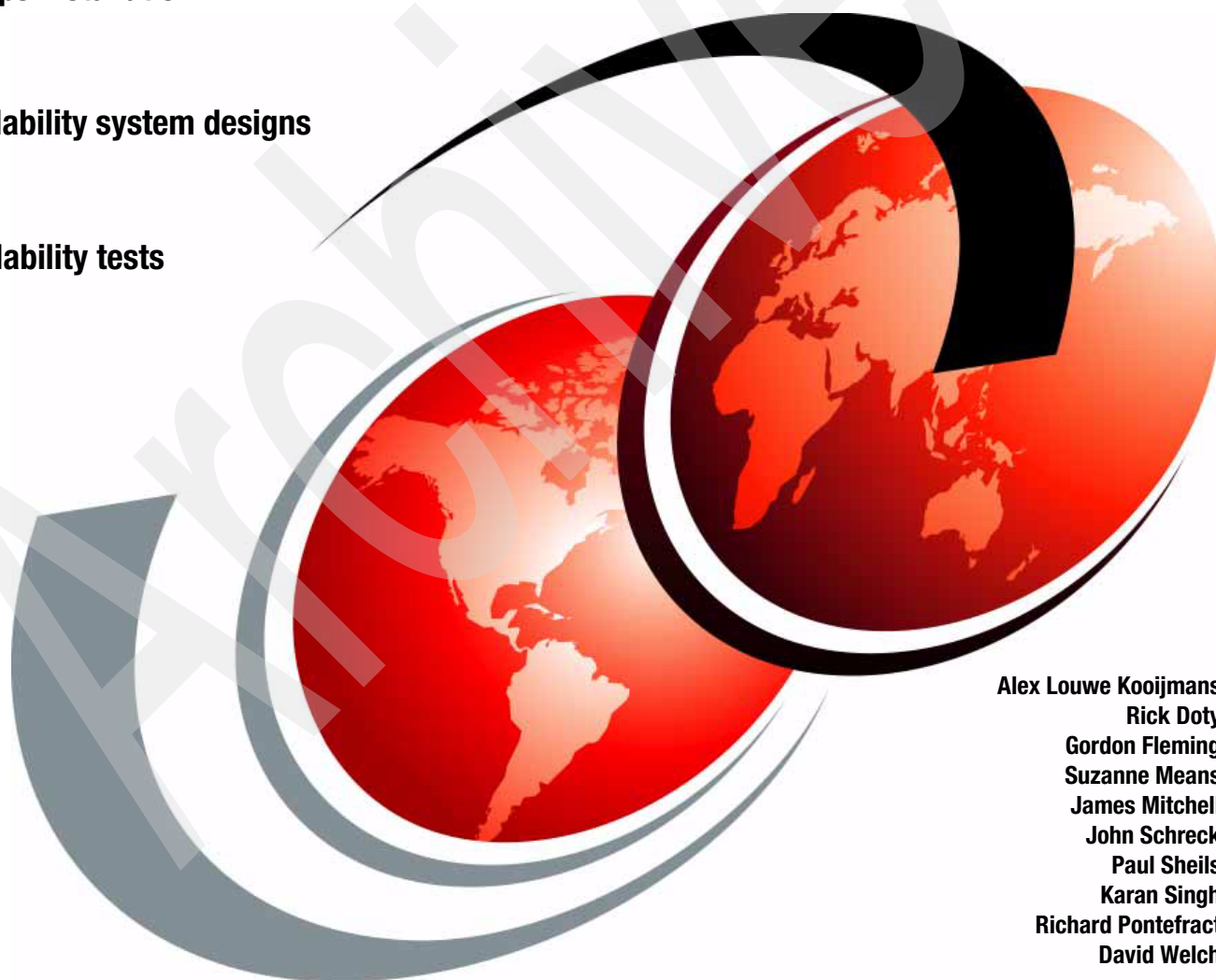


# A Guide to the ACI Worldwide BASE24-eps on z/OS

BASE24-eps installation

High-availability system designs

High-availability tests



Alex Louwe Kooijmans  
Rick Doty  
Gordon Fleming  
Suzanne Means  
James Mitchell  
John Schreck  
Paul Sheils  
Karan Singh  
Richard Pontefract  
David Welch

**Redbooks**





International Technical Support Organization

**A Guide to the ACI Worldwide BASE24-eps on z/OS**

June 2009

Archived

**Note:** Before using this information and the product it supports, read the information in “Notices” on page vii.

Archived

**First Edition (June 2009)**

This edition applies to Release 1, Version 08.2 of the ACI Worldwide BASE24-eps product on z/OS.

© Copyright International Business Machines Corporation 2009. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Contents

<b>Notices</b> .....	vii
Trademarks .....	viii
<b>Preface</b> .....	ix
The team that wrote this book .....	ix
Become a published author .....	xi
Comments welcome .....	xi
<b>Chapter 1. Running a payments system</b> .....	1
1.1 Industry growth .....	2
1.1.1 The requirement for reliability .....	2
1.1.2 System availability .....	2
1.1.3 Basic banking transaction categories .....	3
1.2 How ATM and POS processing is accomplished .....	4
1.2.1 Transaction acquisition and authorization .....	6
1.3 How System z addresses ATM, EFT, and POS processing requirements .....	6
1.3.1 Availability .....	6
1.3.2 Parallel Sysplex clustering .....	6
1.3.3 Manageability .....	7
1.3.4 Security .....	7
1.3.5 Scalability .....	8
1.3.6 Dynamic workload balancing .....	8
1.3.7 IBM Integrated Cryptographic Service Facility .....	8
1.3.8 Integration with external authorization systems .....	8
<b>Chapter 2. Introduction to BASE24-eps</b> .....	9
2.1 Consumer transaction support .....	10
2.2 Transaction switching and routing .....	10
2.3 Flexible authorization .....	10
2.4 Integrated consumer data .....	11
2.5 Traditional and emerging delivery channels .....	11
2.6 Reliable security infrastructure .....	12
2.7 Intuitive graphical user interface .....	12
2.8 Scriptable extracts and reports .....	12
2.9 National switch and international card scheme interfaces .....	12
2.10 Architecture .....	13
2.11 Operability .....	13
2.12 Scripting component .....	14
2.13 The ACI desktop .....	15
2.14 Rich functionality .....	15
2.15 Platform independence .....	15
2.16 Flexible architecture .....	15
2.17 Financial transaction flows .....	16
2.17.1 Authorization types .....	16
2.17.2 Authorization methods for on-us cards .....	17
2.17.3 Acquirer types .....	18
2.18 ACI Enterprise Payments Solutions .....	18
<b>Chapter 3. System z and z/OS</b> .....	19

3.1 An IBM tradition of System z value . . . . .	20
3.1.1 System z architecture . . . . .	20
3.1.2 Who uses mainframe computers . . . . .	20
3.2 System z in the finance industry . . . . .	21
3.2.1 High availability . . . . .	21
3.2.2 Centralized data storage . . . . .	22
3.2.3 Recovery . . . . .	23
3.2.4 Workload management . . . . .	23
3.2.5 Performance management . . . . .	23
3.2.6 Scalability . . . . .	23
3.2.7 Security . . . . .	24
3.2.8 Encryption . . . . .	25
3.3 Parallel Sysplex: A high-availability clustering configuration . . . . .	25
<b>Chapter 4. BASE24-eps architecture . . . . .</b>	<b>27</b>
4.1 BASE24-eps architecture . . . . .	28
4.1.1 Core platform services . . . . .	28
4.1.2 Core components and features . . . . .	30
4.2 BASE24-eps architecture on System z . . . . .	32
4.2.1 BASE24-eps core platform services on System z . . . . .	32
4.2.2 Core components and services . . . . .	35
4.3 BASE24-eps performance on System z . . . . .	44
4.3.1 Test results . . . . .	44
<b>Chapter 5. Designing the system layout . . . . .</b>	<b>47</b>
5.1 BASE24-eps high availability on System z . . . . .	48
5.1.1 Single-LPAR . . . . .	49
5.1.2 High availability: Dual LPAR . . . . .	51
5.1.3 High availability: Dual LPAR with Hyperswap . . . . .	52
5.1.4 High availability: Single site dual server with Hyperswap . . . . .	54
5.1.5 Disaster Recovery: Metro Mirror . . . . .	55
5.1.6 Disaster Recovery: Global Mirror . . . . .	57
5.1.7 Disaster Recovery: DB2 queue replication active/standby . . . . .	58
5.1.8 Disaster Recovery: Stretch Parallel Sysplex . . . . .	59
5.1.9 Disaster Recovery: DB2 Queue Replication Dual Active . . . . .	60
5.1.10 Dual active considerations . . . . .	62
5.2 Securing BASE24-eps on System z . . . . .	62
<b>Chapter 6. Installing BASE24-eps on z/OS . . . . .</b>	<b>63</b>
6.1 Target ITSO z/OS environment . . . . .	64
6.2 Our z/OS environment . . . . .	64
6.3 BASE24-eps configuration . . . . .	66
6.4 Pre-install checklist and worksheet . . . . .	67
6.5 ES_Install . . . . .	68
6.5.1 Executing the ES_Install program . . . . .	68
6.5.2 FTP files to target system . . . . .	71
6.6 Installing ICE-XS . . . . .	72
6.6.1 Extracting the ICE-XS files . . . . .	73
6.6.2 Creating the license files and ICE-XS parameter file . . . . .	73
6.6.3 Testing the execution of ICE-XS . . . . .	74
6.7 essetup . . . . .	75
6.8 Initial database load . . . . .	85
6.9 BASE24-eps directory structure . . . . .	85
6.10 Running the esinfo . . . . .	86

6.11	Installation verification . . . . .	87
6.12	Installing the user interface . . . . .	91
6.12.1	Basic installation . . . . .	91
6.12.2	Installing UI support on WebSphere Application Server . . . . .	92
6.12.3	Version Checker . . . . .	93
6.12.4	Now we can logon . . . . .	94
6.12.5	Client UI connectivity . . . . .	95
6.13	Operational considerations . . . . .	95
6.13.1	Defining ICE-XS processes to BASE24-eps . . . . .	95
6.13.2	JMX Rebuild . . . . .	97
6.13.3	OLTP Warmboot . . . . .	98
6.14	BASE24-eps operations . . . . .	98
6.14.1	Starting BASE24-eps . . . . .	98
6.14.2	Stopping BASE24-eps . . . . .	100
<b>Chapter 7.</b>	<b>BASE24-eps and System z cryptography . . . . .</b>	<b>101</b>
7.1	Basics of System z cryptography . . . . .	102
7.1.1	System z cryptographic facilities . . . . .	102
7.1.2	Cryptographic functions . . . . .	102
7.1.3	Clear key versus secure key . . . . .	102
7.1.4	Symmetric algorithms and keys . . . . .	102
7.1.5	Asymmetric algorithms and keys . . . . .	103
7.1.6	Cryptographic software . . . . .	103
7.1.7	Hardware connectivity . . . . .	103
7.2	Cryptographic hardware . . . . .	104
7.2.1	z10 Enterprise Class and z10 Business Class . . . . .	105
7.2.2	z9 Enterprise Class and z9 Business Class . . . . .	108
7.2.3	Trusted Key Entry Workstation . . . . .	109
7.3	BASE24-eps and IBM Crypto Express2 . . . . .	109
7.3.1	Connectivity . . . . .	109
7.3.2	Key Token data source . . . . .	110
7.3.3	Supported functions . . . . .	110
7.3.4	Key block format . . . . .	110
7.3.5	ICSF key labels and tokens . . . . .	110
7.3.6	How BASE24-eps derives ICSF key labels . . . . .	111
7.3.7	IBM key tokens . . . . .	113
7.4	Preparing data for the BASE24-eps database . . . . .	113
7.4.1	Key Generator Utility Program . . . . .	114
7.5	Configuring IBM Crypto in the BASE24-eps UI . . . . .	114
7.5.1	Configuration steps . . . . .	114
7.5.2	ICSF key definitions . . . . .	123
7.6	Other issues . . . . .	125
<b>Chapter 8.</b>	<b>Achieving high availability on z/OS . . . . .</b>	<b>127</b>
8.1	Overview . . . . .	128
8.2	BASE24-eps configuration for high availability . . . . .	129
8.3	z/OS considerations . . . . .	130
8.3.1	DB2 . . . . .	130
8.3.2	IBM WebSphere MQ . . . . .	130
8.3.3	Sysplex Distributor . . . . .	132
8.3.4	Shared filesystem . . . . .	133
8.4	BASE24-eps considerations . . . . .	133
8.4.1	Constructing the file system for the second system . . . . .	133

8.4.2	Configuring BASE24-eps parameters for the second system. . . . .	134
8.4.3	MQ changes for the second system . . . . .	135
8.4.4	Application Management Remote Agent configuration. . . . .	137
8.5	Starting BASE24-eps on SC67 . . . . .	147
8.6	Running workload on both systems . . . . .	150
<b>Chapter 9.</b>	<b>BASE24-eps high availability testing . . . . .</b>	<b>151</b>
9.1	Test scenarios. . . . .	152
9.2	IBM WebSphere MQ. . . . .	152
9.2.1	MQ manager loss . . . . .	152
9.2.2	IBM WebSphere MQ maintenance implementation . . . . .	153
9.2.3	Adding IBM WebSphere MQ instance in new LPAR. . . . .	153
9.3	Data environment . . . . .	154
9.3.1	DB2 catastrophic failure . . . . .	154
9.3.2	Data table unavailable at BASE24-eps initialization . . . . .	154
9.3.3	DB2 maintenance implementation . . . . .	155
9.4	BASE24-eps . . . . .	156
9.4.1	Integrated Server failure . . . . .	156
9.4.2	BASE24-eps maintenance: planned application upgrade. . . . .	156
9.4.3	ICE-XS server failure . . . . .	157
9.4.4	ICE-XS maintenance . . . . .	157
9.4.5	Application Management server failure. . . . .	158
9.5	Hardware and z/OS system software . . . . .	158
9.5.1	Coupling facility failure . . . . .	159
9.5.2	LPAR failure . . . . .	159
9.5.3	IP virtualization failure. . . . .	160
9.5.4	ICSF failure . . . . .	160
9.5.5	z/OS maintenance . . . . .	161
9.5.6	Coupling facility maintenance . . . . .	162
<b>Appendix A.</b>	<b>ACI Simulation Services for Enterprise Testing. . . . .</b>	<b>163</b>
	ASSET simulator . . . . .	164
<b>Glossary</b>	<b>. . . . .</b>	<b>171</b>
<b>Related publications</b>	<b>. . . . .</b>	<b>173</b>
	IBM Redbooks . . . . .	173
	ACI Worldwide, Inc. publications . . . . .	173
	Online resources . . . . .	173
	How to get Redbooks. . . . .	173
	Help from IBM . . . . .	173
<b>Index</b>	<b>. . . . .</b>	<b>175</b>



# Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described 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:

*IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.*

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include 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.

## **COPYRIGHT LICENSE:**

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.


## **ACI states the following:**

The extent to which BASE24-eps can be modified or customized, as described in this document, may be limited by the terms of the contract between ACI and the customer. While all the extensibility described herein is achievable, some aspects may require the acquisition of certain product modules and/or services from ACI or may be subject to ACI's written consent under the terms of the customer's license agreement.

# Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AIX®	Language Environment®	System z9®
CICS®	MQSeries®	System z®
DB2®	OS/390®	Tivoli Enterprise Console®
eServer™	Parallel Sysplex®	Tivoli®
GDPS®	RACF®	VTAM®
Geographically Dispersed Parallel	Redbooks®	WebSphere®
Sysplex™	Redbooks (logo)  ®	z/OS®
IBM®	S/390®	z9®
InfoSphere™	System z10™	zSeries®

The following terms are trademarks of other companies:

Novell, SUSE, the Novell logo, and the N logo are registered trademarks of Novell, Inc. in the United States and other countries.

ACI Worldwide, ACI, BASE24, BASE24-eps, BASE24-es, ICE-XS, XPNET and ASSET are trademarks or registered trademarks of ACI Worldwide, Inc., or one of its subsidiaries, in the United States, other countries or both. These and other ACI trademarks are marked on their first occurrence in this information with the appropriate symbol (® or ™) indicating US registered or common law trademarks owned by ACI at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries.

Java, JavaScript, JMX, JVM, Sun, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Excel, Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

# Preface

In this IBM® Redbooks® publication, we explain how to use the ACI™ BASE24-eps™ product on z/OS®. BASE24-eps is a payment engine that the financial payments industry uses. The combination of BASE24-eps and System z® is a competitive and attractive end-to-end retail payments solution for the finance sector.

Failure in a financial payments environment is a high-visibility customer service issue, and outages at any level have debilitating effects on customer loyalty. The entire payments cycle must be conducted in near real-time. In such an environment, high availability is a mission-critical requirement. In this guide, we demonstrate how you can achieve a high availability configuration for BASE24-eps on z/OS.

We begin by outlining the requirements of a payments system, and then we introduce the structure and functionality that the BASE24-eps product offers. We also describe the strengths and abilities of System z and z/OS and explain the technical and physical architecture of BASE24-eps on z/OS.

We guide you in designing a system layout and in installing BASE24-eps. Finally, we detail the numerous failure scenarios that we tested to verify the robustness of the solution.

BASE24-eps is an evolving product. The information that we provide in this book is specific to BASE24-eps release 08.2 and is subject to change in subsequent releases of the product.

## The team that wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

**Alex Louwe Kooijmans** is a Project Leader with the International Technical Support Organization (ITSO) in Poughkeepsie, NY. He specializes in WebSphere®, Java™ and SOA on System z with a focus on integration, security, high availability, and application development. Previously, he worked as a Client IT Architect in the Financial Services sector, with IBM in The Netherlands, advising financial services companies on IT issues, such as software and hardware strategy and on demand. Alex also worked at the Technical Marketing Competence Center for zSeries® and Linux® in Boeblingen, Germany, providing support to customers implementing Java and WebSphere on zSeries. From 1997 to 2002, he worked on a previous assignment with the ITSO, managing various IBM Redbooks projects and delivering workshops around the world.

**Rick Doty** is a Systems Engineer in the ACI Performance and Architecture group in Omaha, NE, USA. He has 20 years with ACI and 40 years of experience in Information Technology (IT). He has a Bachelors degree in Applied Mathematics from Illinois Institute of Technology and a Masters degree in Computer Science from the University of Iowa. His areas of expertise include MVS, z/OS, VTAM®, and CICS®. He was a Systems Programmer for 12-14 years and also made minor contributions to the first BASE24-es® Redbooks project.

**Gordon Fleming** is a Senior Software Architect located in Omaha, Nebraska, USA. He has 22 years of experience in online transaction processing, primarily with ACI BASE24®, XPNET™, and BASE24-eps software. He has worked at ACI Worldwide™ for over 20 years and is one of the original architects of BASE24-eps System Interface Services. Gordon's

areas of expertise include multi-platform infrastructure design and development, BASE24-eps System Interface Services on z/OS, CICS, AIX®, and other platforms, and online transaction processing (OLTP), high availability, and Disaster Recovery. He has degrees from the University of Chicago and the University of Nebraska at Omaha and was a co-author of the previous edition of *A Guide to Using ACI Worldwide's BASE24-es on z/OS*.

**Suzanne Means** is a z/OS systems programmer with the IBM Integrated Technology Delivery organization. She has provided system software installation, maintenance, and operational support for customer installations for more than twenty years. Before moving to a systems software role, she had extensive experience in application programming and data storage administration. Like many computer professionals, she has a Liberal Arts Bachelors degree.

**James Mitchell** is a Staff Software Engineer and IBM Certified System z Specialist with the IBM Worldwide Banking Center of Excellence in Poughkeepsie, NY. He has 22 years of experience in System z operating system development, test, service, and technical sales/marketing support. Over the past two years, he provided technical consultation and support for the ACI BASE24-eps re-engineering effort on System z. He has a degree in Computer Science from Jackson State University.

**John Schreck** is a Senior Engineer with ACI Worldwide, Inc. and is located in the Dallas, TX area. John has 16 years of experience in the online transaction processing field. His current area of focus is BASE24-eps implementation on UNIX® and z/OS platforms. John has a Bachelor of Science in Business Administration degree in Computer Information Systems from the University of Central Missouri. John's areas of expertise include multi-platform design and implementation of high-availability EFT solutions for the retail and banking market segments.

**Paul Sheils** is a System z Specialist with STG Lab Services in Europe. He has nearly 25 years of experience in IT and joined IBM in 2001 during an outsourcing deal where he moved to the STG Lab Services in 2006. His main areas of expertise are z/OS and Parallel Sysplex technology, security, and cryptography.

**Karan Singh** is a Project Leader with the International Technical Support Organization (ITSO) in Poughkeepsie, NY. His areas of expertise include core z/OS technologies.

**Richard Pontefract** is a Product Consultant in the ACI ASSET™ Development Center in Leeds, United Kingdom (UK). He has a Bachelor of Science (Honours) degree in Computer Science from the University of Portsmouth. Richard has 18 years of experience in Information Technology focusing on a broad range of payment systems solutions, devices, and interfaces. His areas of expertise include the architecture, design, and customer implementation of middleware and communications subsystems for online transaction processing, specializing in C++ on UNIX and Windows platforms.

**David Welch** is a zOS Technical Specialist for IBM in New Zealand. He has 30 years of experience as a Systems Programmer with the majority of this supporting infrastructure for a major New Zealand bank. He has a Bachelor of Science degree in Mathematics from the University of Canterbury.

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

Richard Conway, Robert Haimowitz  
International Technical Support Organization, Poughkeepsie Center

Nishith Bajpai, Charles Codo, Alan F. Hoss, Jim Jeter, Jean Keeney, David A. Keil, Dan Kinney, Frank J. Troia  
ACI Worldwide

Guillaume Arnould, Neil Ashworth, Stefano Delle Chiaie, Vesna Eibel, Stephane Faure, James Goethals, Fabrice Jarassat, Jean-Luc Lepesant, Stephane Loth, Kenneth Muckenhaupt  
IBM

Thanks to the authors of the previous editions of this book.

- Authors of the first edition, *A Guide to Using ACI Worldwide's BASE24-es on z/OS*, published in August 2006, were:  
Alex Louwe Kooijmans, Edward Addison, Alexei Alaev, Dan Archer, Gordon Fleming, Claus Koefoed, Ron Schmidt, Bob Spanke, Mingming Tao, Dore Teichman

## Become a published author

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:  
[ibm.com/redbooks/residencies.html](http://ibm.com/redbooks/residencies.html)

## Comments welcome

Your comments are important to us!

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

- Use the online **Contact us** review Redbooks form found at:

[ibm.com/redbooks](http://ibm.com/redbooks)

- Send your comments in an e-mail to:

[redbooks@us.ibm.com](mailto:redbooks@us.ibm.com)

- Mail your comments to:

IBM Corporation, International Technical Support Organization  
Dept. HYTD Mail Station P099  
2455 South Road  
Poughkeepsie, NY 12601-5400

Archived

# Running a payments system

Banks and financial institutions are highly motivated by competitive forces to broadly deploy automated teller machines (ATMs) and engage in electronic funds transfer (EFT) activities. “The size of the ATM network has a significant impact on the demand for bank deposits”.<sup>1</sup> Customers consider the number and location of ATMs in a bank’s network when choosing a bank or financial institution, and have demonstrated that they are willing to pay a surcharge for convenience. While increasing customer convenience, ATMs also save financial institutions money and increase profits by reducing resource costs through automation.

In this chapter, we provide an overview of bank payment systems and requirements, and then describe how System z offers the availability, reliability, manageability, security, scalability, and workload balancing to help you meet those requirements.

---

<sup>1</sup> Knittel, C., and V. Stango. 2004. “Compatibility and Pricing with Indirect Network Effects: Evidence from ATMs.” NBER working paper 10774

## 1.1 Industry growth

Bank card processing, such as ATM, point-of-sale (POS) and credit, more than any other activity, wields an immediate and lasting impact on the reputation and image of a financial institution. Banks and credit card companies encourage customers to use their cards, inviting them to essentially live “cashless”. As a result, the ATM and POS marketplace has undergone a rapid, expansive change in recent years.

“Installation of ATMs and the proliferation of Retail POS has been particularly rapid in recent years. ATM growth was 9.3 percent per year from 1983 to 1995 but accelerated to an annual pace of 15.5 percent from 1996 to 2002. Much of the acceleration is due to placing ATMs in locations other than bank offices. These off-premise ATMs accounted for only 26 percent of total U.S. ATMs in 1994, but now account for 60 percent. On the debit card side of the industry, growth has been extremely rapid in point-of-sale (POS) debit card transactions. With an annual growth rate of 32 percent from 1995 to 2002, POS debit is the fastest growing type of payment in the United States. Today it accounts for nearly 12 percent of all retail non-cash payments, a fivefold increase in just five years. Growth has been sharp in both online (PIN-based) and offline (signature-based) debit. From 1995 to 2002, annual growth of online debit was 29 percent, while offline debit grew at 36 percent.”<sup>2</sup>

### 1.1.1 The requirement for reliability

Few issues impact the reputation of a financial institution, retailer, or customer more than ATM/credit card processing reliability. Trying to explain that a card being declined is the fault of the bank is a difficult task, and failures at the point-of-sale embarrass customers, damage the reputation of the bank itself, and may cause loss of retailer sales. As the transition to a cashless society continues, customers will quickly revert to using cash, and—more damagingly—take their business elsewhere, over a single embarrassing event.

Moreover, the brand name of the ATM network must be considered. Visa, MasterCard, STAR, and others value their name brand. They have service level agreements (SLA) with the financial institutions and prefer to avoid being the target of poor performance complaints by banking customers.

### 1.1.2 System availability

In the event that a financial institution is unable to respond to a transaction within a specified time period (usually about 10 seconds), the ATM or POS network might have the right to “stand in” for the bank, and following specified rules, approve or decline the transaction. In some instances, such as overseas banks and gateway transactions, more time is allocated, which could potentially expose the bank to overdraft of the customer’s account and the possibility of approving a fraudulent transaction. The specific rules and charges for stand in are usually the result of a negotiated agreement between the financial institution and the shared network.

A financial institution or card issuer with unsatisfactory or poor reliability or poor response-time performance might accept the risk of a shared network “standing in” for the bank, an activity that is neither free nor without risk.

---

<sup>2</sup> Knittel, C., and V. Stango. 2004. “Compatibility and Pricing with Indirect Network Effects: Evidence from ATMs.” NBER working paper 10774.



## System reliability and data integrity

A financial transaction can pass through several switches before it is completed on the round-trip journey from acquisition, to authorization, and back to its starting point, ATM, or POS station. When everything goes according to plan, the transaction completes and is prepared for final settlement, which might take place real-time or in some form of batch processing scenario. When failures and errors occur, the host issuer authorizing system must keep track of all transactions, never losing a single one.

There are several methods that are available for meeting the response time, availability, and data integrity requirements imposed on financial processing systems. These techniques go beyond simple fault-tolerance. The system must be continuously available. Fault tolerance is not a requirement for high availability; instead, it is simply one of the tools that contributes to the high availability capability.

### 1.1.3 Basic banking transaction categories

ATM or POS processing can occur across multiple paths, either directly (through a bank's owned ATM, typically called "On Us"), or through one of many shared networks where the card holder is using a terminal that is owned and operated by another entity.

An example of the cooperative networks is STAR, which is owned by First Data Corporation. With more than 1.7 million locations across the United States, handling over 5,700 financial institutions with over 134 million cards, STAR is just one of the many networks that are available to banks and other financial institutions that can be used to project their financial services to their customer base.

Common shared networks do more than simply provide credit card and ATM processing. They can also provide other services to financial institutions, such as:

- ▶ Debit and ATM network
- ▶ Surcharge-free programs
- ▶ Deposits, deposit sharing program
- ▶ Gateway connections
- ▶ ATM driving
- ▶ POS driving
- ▶ PIN-secured debit, signature debit, and stored value card processing
- ▶ Card authorization, activation, and production
- ▶ Merchant acquirer and agent bank programs
- ▶ Bill payment services
- ▶ Risk management and fraud prevention services

Other industry competitors in this marketplace worldwide are VisaNet, MasterCard, Pulse, NYCE, Multibanco, Interac, LINK, JCB, and others. In total, there are more than 60 major networks worldwide available to both consumers and financial institutions.

Shared networks provide added availability to financial institution customers, although they can also introduce a certain degree of risk. Most networks require stand-in authorization, which enables a network to authorize transactions when a card issuer or processor cannot do so. However, this in turn can lead to an increased risk of fraud.

To cite a useful example of the levels of uptime that are typically maintained, a major bank ATM processing system in The Netherlands has had zero downtime over the past four years, including maintenance and upgrades. This type of reliability is not unusual for the banking industry. Even systems in developing countries maintain availability targets that are higher than 96%.

## 1.2 How ATM and POS processing is accomplished

In many ways, ATM and POS processing and POS card processing are similar and follow similar paths through the network and associated systems. Four categories of transaction can take place, as Table 1-1 shows.

Table 1-1 Transaction switching schemes

Category	Description
On Us	The transaction arrives through a bank-owned ATM or POS device and is never routed outside of the bank. Example: A customer of the "SampleA" Credit Union uses an ATM card to withdraw money from the ATM that is located in the local credit union office.
Network On Us	The transaction originates from a sharing network, such as STAR or Pulse, in which both the bank and the device-owning bank are members of the same network. Example: A local customer travels somewhere away from home and is a member of "SampleA" Credit Union. The customer must use a device at the "SampleB" Credit Union. Both "SampleA" and "SampleB" are members of the STAR network.
Reciprocal Transaction	The card holder initiates a transaction at a device that is owned by a bank that is a member of a different regional network. In this case, a gateway is used to switch the transaction. Example: A New York resident attempts to withdraw money from an ATM or POS in Omaha, Nebraska. An agreement between the network in Nebraska and the network in New York allows the transaction to be switched from one regional network to another.
National Bridge Transactions	The card holder uses a device at a bank that is not their own, and the two banks belong to different regional networks that do not have any agreement. Both banks <i>must</i> belong to the same national network. The transaction is handed from the ATM or POS regional network to the national network, and finally to the authorizing bank's regional network. In this case, there are three switches involved.

Transactions are routed through the originating ATM or POS, either the bank's own network or through some combination of regional or national network. Figure 1-1 on page 5 demonstrates, at high level, the possible paths that a transaction might follow to completion.

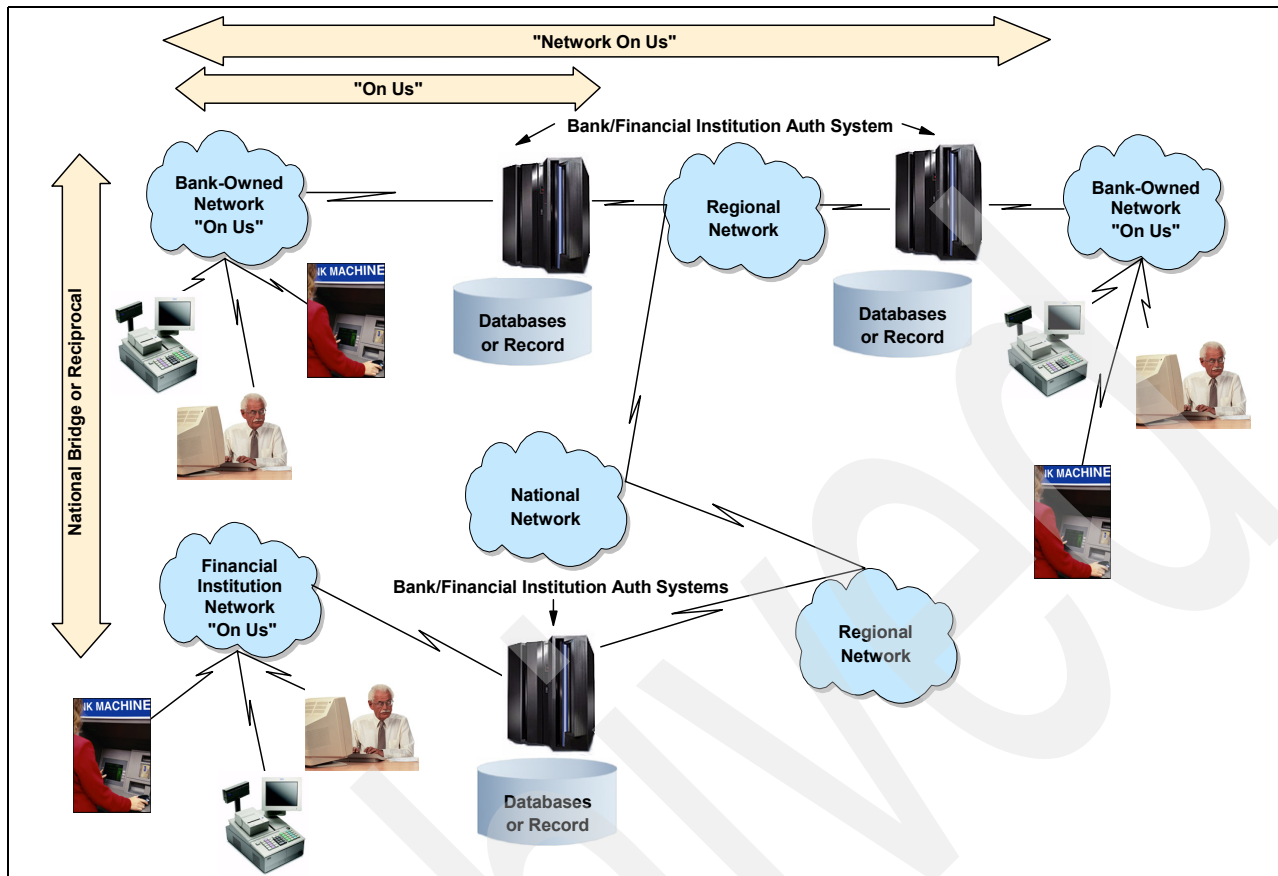


Figure 1-1 Transaction flows

Both the regional and national networks are switching systems that, to a fair extent, resemble the systems that are used within the financial institution. The switching systems drive transactions from initiation to destination. We use the word “switched” to describe the hand-off of a transaction from host-to network-to host:

- ▶ An ATM or POS transaction is accepted by an acquiring device that can accept cards from the issuing institution and either processed locally “On Us” or switched to one of the regional networks “Network On Us”.
- ▶ The networks either switch the transaction to another institution host for processing “Network On Us” or switch the transaction to a national network “National Bridge or Reciprocal”.
- ▶ Responses are switched from the owning host of the device, “On Us”, to the device or switched to a network “Network On Us”.
- ▶ The network either switches a Host Response to the transaction to the host of a member bank “Network On Us” or switches the response to a another network “National Bridge or Reciprocal”.
- ▶ The network switches the response to another network for later switching to the card-issuer host.

Although there are other variations on this theme, transactions are switched from host-to-host through network switches until the original transaction request is completed by a transaction response or it times out.

### 1.2.1 Transaction acquisition and authorization

ATM and POS transactions must traverse three basic steps through complete processing: Acquisition, Authentication, and Authorization. (Back end settlement and reconciliation activities are not a part of the actual real-time transaction).

## 1.3 How System z addresses ATM, EFT, and POS processing requirements

In the previous sections, we described at high level the processing flow and requirements for ATM and POS transactions, which are business critical transactions that require a platform (hardware and software) that provides robust levels of reliability, availability, and scalability. The IBM System z server and z/OS operating system provides an ideal environment to host payment systems. In the following sections, we list the processing requirements that can be met when running on a System z platform. For further details, refer to Chapter 3, “System z and z/OS” on page 19.

### 1.3.1 Availability

System z provides 24-hour a day, 7-days per week availability, which includes scheduled maintenance. Continuous availability goes beyond just hardware fault tolerance; it is achieved by a combination of hardware, application code, and good system management practices.

On a server basis, System z systems are equipped with features that provide for very high availability:

- ▶ Redundant I/O interconnect
- ▶ Concurrent Capacity Backup Downgrade (CBU Undo)
- ▶ Concurrent memory upgrade
- ▶ Enhanced driver maintenance
- ▶ Capacity backup upgrade
- ▶ On/Off capacity

### 1.3.2 Parallel Sysplex clustering

When configured properly, a Parallel Sysplex® cluster has no single point-of-failure and can provide customers with near continuous application availability over planned and unplanned outages. Events that otherwise seriously impact application availability (such as failures in hardware elements or critical operating system components) have no, or reduced, impact in a Parallel Sysplex environment.

With a Parallel Sysplex cluster, it is possible to construct a parallel processing environment with no single point-of-failure. Because all systems in the Parallel Sysplex can have concurrent access to all critical applications and data, the loss of a system due to either hardware or software failure does not necessitate loss of application availability. Peer instances of a failing subsystem executing on remaining healthy system nodes can take over recovery responsibility for resources that are held by the failing instance.

Alternatively, the failing subsystem can be automatically restarted on still-healthy systems using automatic restart capabilities to perform recovery for work in progress at the time of the failure. While the failing subsystem instance is unavailable, new work requests can be redirected to other data-sharing instances of the subsystem on other cluster nodes to provide

continuous application availability across the failure and subsequent recovery, which provides the ability to mask planned and unplanned outages from the end user.

A Parallel Sysplex cluster consists of up to 32 z/OS images coupled to one or more Coupling Facilities (CFs or ICFs) using high-speed specialized links for communication. The Coupling Facilities, at the heart of a Parallel Sysplex cluster, enable high speed, read/write data sharing and resource sharing among all of the z/OS images in a cluster. All images are also connected to a common time source to ensure that all events are properly sequenced in time.

The flexibility of System z, z/OS, and Parallel Sysplex allows customers to create many high availability system designs, from multiple LPARs in a Parallel Sysplex on multiple System z servers, to dual LPARs in a Parallel Sysplex on a single System z server. We outline several high-availability system designs in Chapter 5, “Designing the system layout” on page 47.

### 1.3.3 Manageability

A wide array of tools, which include the IBM Tivoli® product and other operational facilities, contribute to continuous availability. IBM Autonomic Computing facilities and tools provide for completely fault tolerant, manageable systems that can be upgraded and maintained without downtime.

Autonomic computing technologies that provide Self-Optimizing, Self-Configuring, and Self-Healing characteristics go beyond simple hardware fault tolerance. Additionally, the System z hardware environment provides:

- ▶ Fault Detection
- ▶ Automatic switching to backups where available
- ▶ (Chipkill memory, ECC cache, CP, Service Processor, system bus, Multipath I/O, and so on)
- ▶ Plug and Play and Hot swap I/O
- ▶ Capacity Upgrade on Demand

### 1.3.4 Security

On March 14, 2003, IBM eServer™ zSeries 900 was the first server to be awarded EAL5 security certification. The System z architecture is designed to prevent the flow of information among logical partitions on a system, thus helping to ensure that confidential or sensitive data remains within the boundaries of a single partition.

On February 15, 2005, IBM and Novell® announced that SUSE® Linux Enterprise Server 9 successfully completed a Common Criteria (CC) evaluation to achieve a new level of security certification (CAPP/EAL4+). IBM and Novell also achieved United States (US) Department of Defense (DoD) Common Operating Environment (COE) compliance, which is a Defense Information Systems Agency requirement for military computing products.

On March 2, 2006, z/OS V1.7 with the RACF® optional feature achieved EAL4+ for Controlled Access Protection Profile (CAPP) and Labeled Security Protection Profile (LSPP). This prestigious certification assures customers that z/OS V1.7 goes through an extensive and rigorous testing process and conforms to standards that the International Standards Organization sanctions.

These certification efforts highlight the IBM ongoing commitment to providing robust levels of security to assist customers in protecting their business critical data.

### 1.3.5 Scalability

The Capacity Upgrade on Demand (CUoD) capability allows you to non-disruptively add one or more Central Processors (CPs), Internal Coupling Facilities (ICFs), System z Application Assist Processor (zAAP), and Integrated Facility for Linux (IFLs) to increase server resources when they are needed, without incurring downtime. Capacity Upgrade on Demand can quickly add processors up to the maximum number of available inactive engines. Also, additional books (up to a maximum of four in total) can be installed concurrently, providing additional processing units and memory capacity to a z9® or z10® server.

In addition, the new Enhanced Book Availability function also enables a memory upgrade to an installed z9 or z10 book in a multi-book server. This feature provides customers with the capacity for much needed dynamic growth in an unpredictable ATM/EFT world.

The CUoD functions include:

- ▶ Non-disruptive CP, ICF, IFL, and zAAP upgrades
- ▶ Dynamic upgrade of all I/O cards in the I/O Cage
- ▶ Dynamic upgrade of memory

The Parallel Sysplex environment can scale near linearly from two to 32 systems. This environment can be a mix of any servers that support the Parallel Sysplex environment.

### 1.3.6 Dynamic workload balancing

To end users and business applications, the entire Parallel Sysplex cluster can be seen as a single logical resource. Just as work can be dynamically distributed across the individual processors within a single SMP server, so too can work be directed to any node in a Parallel Sysplex cluster that has the available capacity, which avoids the need to partition data or applications among individual nodes in the cluster or to replicate databases across multiple servers.

### 1.3.7 IBM Integrated Cryptographic Service Facility

In addition to the external security modules (HSMs) that are available on other platforms, BASE24-eps on System z can take full advantage of the IBM Crypto Express 2 card using the Integrated Cryptographic Service Facility (ICSF) for very high speed and highly available cryptographic services, such as Personal Identification Number (PIN) translation and verification and Message Authentication Code (MAC) generation and validation.

### 1.3.8 Integration with external authorization systems

On all platforms, BASE24-eps can use data communications to send requests and receive responses from external transaction authorization systems. On System z only, other means of communicating with external authorization systems are available, such as:

- ▶ IBM WebSphere MQ CICS Gateway for communicating synchronously or asynchronously with local CICS-based authorization systems. (Synchronous communications is recommended only for suitably reliable and low-latency systems.)
- ▶ The IBM External CICS Interface (EXCI) for communicating synchronously with suitably reliable and low-latency local CICS authorization systems with the lowest possible CPU cost.
- ▶ The IBM IMSConnect for communicating with local IMS-based authorization systems.

## Introduction to BASE24-eps

BASE24-eps is an integrated payment engine that is used to acquire, authenticate, route, switch, and authorize financial transactions across multiple channels. It provides a full range of functionality to support payment transactions, both the traditional transactions that institutions manage today (for example, debit and credit at the ATM and point-of-sale or telephone banking) and emerging transactions (such as, mobile commerce and Internet banking). The product supports all leading payment cards and offers standard interfaces to leading devices, switches, and back end host systems. The ACI fault-tolerant application software takes advantage of the best in systems software for reliability, availability, scalability and high-performance transaction throughput.

As the major retail banking payments engine of the ACI Enterprise Payments Solutions, BASE24-eps provides flexible integration points to other applications within enterprises and integration with other ACI Enterprise Payments Solutions products. By supporting XML and ISO-based interface standards and other industry-specific formats, BASE24-eps offers the utmost flexibility in a total, end-to-end solution.

The combination of BASE24-eps and System z is a competitive and attractive end-to-end retail payments solution for the finance sector. BASE24-eps on the System z demonstrates the value of the ACI and IBM partnership, which leverages the best-of-breed capabilities from each company.

## 2.1 Consumer transaction support

BASE24-eps provides comprehensive support for consumer e-payment transactions that are initiated by a variety of transaction instruments that include credit, debit, and chip cards.

As the payment industry evolves and new instruments emerge (for example, customer ID and mobile telephone numbers), the flexible nature of its architecture enables BASE24-eps to easily adapt to provide continued value.

Today, BASE24-eps supports a comprehensive cardholder and administrative transaction set that can be accessed from any appropriate delivery channel. Administrative transactions are also supported for settlement and reconciliation purposes.

BASE24-eps can be configured to maintain card and associated account information. Authorization logic can be scripted to perform a variety of tasks that include check current status of the card, compare cardholder use against limit profiles, determine whether the transaction is allowed based on a number of configurable options, and more. In addition, BASE24-eps can provide alternate routing or stand-in authorization if a configured primary external authorizer becomes unavailable.

## 2.2 Transaction switching and routing

BASE24-eps provides a highly flexible routing structure for transactions. This flexibility not only routes transactions to the appropriate network, card association, processor or internal system for authorization, but it also helps users gain lower interchange charges by factoring in the total path when determining the authorization destination.

Determining the card issuer using a Card prefix lookup is decoupled from choosing the destination. The Card prefix lookup determines the destination profile which is then used to determine the destination along with the:

- ▶ Source Profile
- ▶ Destination Profile
- ▶ Transaction Type
- ▶ Account Type 1 (FROM account type)
- ▶ Account Type 2 (TO account type)
- ▶ Method of consumer authentication (PIN present, chip card, and so on)

This flexibility in transaction routing accommodates different account types that might reside on different systems and different platforms. Users can customize their transaction processing at various stages in the transaction life cycle, which includes:

- ▶ Pre-screening before transactions are sent to an external authorizer
- ▶ Defining the processing steps for real time internal authorization
- ▶ Defining the processing steps for stand-in authorization
- ▶ Specifying the destination of advice messages following authorization
- ▶ Specifying how the database should be impacted during the post-authorization process

## 2.3 Flexible authorization

BASE24-eps supports consumer authentication and authorization processing using a powerful scripting engine. Using the scripting engine organizations can control and define application logic without modifying the product source code.



Using an interpreted scripting language with syntax that is similar to JavaScript™, using BASE24-eps users can create a variety of authorization scripts to tailor the authorization logic to meet specific business requirements or service agreements. Organizations can decide what data is used in the authorization process and when the data is used, regardless of whether the data is part of the transaction or from an alternative source.

If more complex authorization modifications or extensions are required, the scripting engine also eases the introduction of a new authorization service written in C++. The service can expose a scripted operator, referred to as an exported operator, to the scripting engines that the script can then invoke. These services can be developed independently without affecting the rest of the system.

ACI provides a set of sample scripts that cover the basic positive, negative, or usage-based authorization processes. This flexibility shortens the time that is needed to develop new products and services or accommodate changes that are requested by the business department of an organization. Separating the business logic (in scripts) from the product source code also facilitates script compatibility with future releases.

Users can choose what level of authorization should be performed on BASE24-eps, which can range from full authorization using a card, limit, account and balance information, to handling pre-screening before using a host for authorization, to just providing a stand-in capability using negative card data. All limits are user-defined to provide full flexibility in controlling the use of the cards and accounts.

## **2.4 Integrated consumer data**

The component architecture of BASE24-eps is designed for flexibility to leverage consumer data resident in external systems and databases. Users can develop components that expose consumer data to the scripting engine to allow more intuitive authorization functionality. This consumer data can be held in customer relationship management (CRM) systems, fraud management systems, customer information files, core banking systems, and other applications.

By exposing more consumer information to the authorization process, organizations can improve consumer relationships by approving transactions that are based on more comprehensive consumer information. They can also intelligently manage risk by denying transactions based on certain risk factors.

## **2.5 Traditional and emerging delivery channels**

In addition to support for traditional delivery channels, which includes ATM and point-of-sale (POS), BASE24-eps can process payments that are initiated through Internet shopping networks, personal digital assistants (PDAs), mobile telephones, Web ATMs, and home banking and branch systems. BASE24-eps offers a powerful, flexible foundation for delivering common services across multiple consumer access channels, computer systems, and databases.

Through XML and ISO-based interface standards and other industry-specific formats, transaction services can be exposed to any channel. Thus, BASE24-eps can provide a single point-of-access across an enterprise for the service of consumer payments, which eliminates the costs of maintaining multiple service points.

## 2.6 Reliable security infrastructure

Organizations' transaction security requirements can vary greatly depending on the environment. An organization typically requires an integrated system of software, industry-standard hardware, and procedures to properly implement financial transaction security.

BASE24-eps is designed to be flexible in its transaction security support and to provide a range of hardware options. The application addresses the diverse needs of large-scale transaction processing systems where the originator of a transaction might operate under an entirely different transaction security scheme than the authorizer. Regardless of the origin or destination of payments, BASE24-eps meets the current industry requirements for security, which includes Triple DES and EMV support.

BASE24-eps operates in a network environment where sensitive data, such as the personal identification number (PIN), is secured through encryption, and the system provides cryptographic functions, such as PIN encryption, PIN verification, message authentication, chip authentication, and card verification, using interfaces to external hardware security modules (HSMs) or System z Crypto Express2 cards.

## 2.7 Intuitive graphical user interface

The ACI user interface presents a task-oriented view of the application for multiple users ranging from business to technical to administrative, and it incorporates graphical elements, such as hyperlinks, buttons, and pull-down menus. Users can also choose to display text labels in their local language to accommodate adaptation into their environment. Integrated help at the window level and the field level minimizes the need for extensive user training, while streamlining business processes and providing greater flexibility.

Written in Java and C++ and using XML message formats, the ACI user interface provides a flexible operating environment that multiple ACI applications use. A security administrator configures access permissions through the ACI user interface where a user security and user audit environment are shared by all ACI applications.

## 2.8 Scriptable extracts and reports

The BASE24-eps scripting engine gives users added flexibility and control over the reporting and extracting of financial transaction data. In an area where customization is required to meet integration needs, the scripting feature allows real time definition of essential and ad hoc reports and user-defined file layouts to serve as input to existing batch processes or reporting tools.

## 2.9 National switch and international card scheme interfaces

BASE24-eps incorporates off-the-shelf support for a range of international card scheme interfaces that include Visa, MasterCard, American Express, and many national switch interfaces. These interfaces are built using a framework methodology covering ISO 8583 (1987), ISO 8583 (1993), and XML standards. This methodology makes use of "inheritance" to facilitate reusability of components and allows either ACI or the customer organization to quickly build new interfaces.

## 2.10 Architecture

ACI uses an object-oriented design and development style to implement the Enterprise Services architecture of BASE24-eps. This architecture helps to reduce impacts that are associated with extending the core product code. The use of object-oriented programming languages, such as C++ and Java, enhances the extensibility of BASE24-eps solutions and minimizes time-to-market for new products and services. By extending integration flexibility, BASE24-eps allows access to more customer information.

BASE24-eps software components use this architecture to create flexible business services that allow users to quickly develop and deploy new products and deliver enhanced customer service. The components are organized according to the function that they perform to support processing for the required business services. Each business component performs a specific type of processing (that is, authorization or routing) or controls a specific part of the file system (for example, account or customer).

The architecture of BASE24-eps is designed for multiple platform configurations. The platform is defined as the hardware, operating system, middleware, and file system. However, platform-specific processing is isolated into specific components to allow the rest of the application to be common across all platforms.

BASE24-eps software components are organized according to the function that they perform:

- ▶ Adaptors manage the information that is exchanged between the end user and the business components. Adaptors can be designed to communicate with any acquiring or issuing entity, which includes Internet portals, hardware devices, service providers or interchanges (Visa, MasterCard, and so on), and in-house systems.
- ▶ Business components perform the processing that is required for the business services that are offered. Each business component performs a specific type of processing (that is, authorization, routing) or controls a specific part of the file system (for example, prefix, perusal).
- ▶ Foundation components are the basic building blocks for any application. They provide services (information and processing) that business components require without regard to the specific business component that requires the service, for example, the FN time component obtains the current system time in the format that the application needs. When any business component needs the system time, it obtains it from this foundation component.
- ▶ Platform-specific components insulate the application from changes in the operating system, middleware, and data structure, for example, while the implementation of the foundation time component is the same on every platform, processing that the platform-specific time component performs differs across platforms. The foundation time component uses the platform-specific time component when it needs access to system services.

## 2.11 Operability

BASE24-eps supports high volume, high availability, and 24/7 operability through a scalable, high-available software architecture that runs on a variety of platforms:

- ▶ Flexible journal configuration and settlement cutover:
  - Allows for 24/7 cutover processing and uninterrupted processing across separate time frames.

- ▶ Implement new business logic without downtime:
  - Code and file system changes that affect configuration do not require a system restart.
- ▶ Hardware resilience:
  - The system and data access layers take advantage of each platform's failover processing capabilities, all with the same set of application code.
- ▶ Consistent processing cost:
  - The asynchronous messaging model of BASE24-eps provides a consistent per transaction processing cost regardless of the transaction volume, which allows the application to grow as needed with a predictable hardware requirement.

## 2.12 Scripting component

The BASE24-eps scripting facility gives organizations a powerful syntax that is similar to JavaScript to allow modification of application logic without having to modify the source code. The application uses these scripts to create journal perusal queries, define journal extract and reporting requirements, and as part of the authorization process.

Scripts are maintained and compiled through the user interface. Users can display a script repository that shows all scripts that are available for use by BASE24-eps. A script editor allows the user to add, edit, delete, and compile scripts through the user interface. During the compile process, scripts are checked for syntax errors and saved on the BASE24-eps system. Rather than compiling into machine code, these scripts are ordered into a list of serial instructions that the script engine can interpret in real time during transaction processing.

Compiled scripts are loaded into memory where they can be retrieved for execution during transaction processing. If a change must be made to script logic, then the script can be updated, recompiled, and placed back into use without ever taking the affected programs out of service.

Scripts have access to data from multiple sources. The primary source is from transaction data elements. Application files for authorization are also available to the script facility, which include card file, limits file, usage accumulation file, account with balance file, and pre-authorization file. For greater decision-making flexibility, access to additional customer information can be obtained through custom-written components that "expose" the proprietary structure of a customer's file to the script. These custom files can be core banking system files, ACI or other third-party card management systems, or a variety of other sources.

A single script can contain all of the tasks that BASE24-eps must perform to authorize a transaction. The tasks can also be split into multiple scripts that are organized in a hierarchical structure.

Because of the component-based design of the application and the scripting language, scripts that the user implements might not need modification when a new release of the application becomes available, which allows users to easily upgrade to new releases of the product. If a user plans to take advantage of new functionality from within the script, then changes are required.

## 2.13 The ACI desktop

The ACI desktop, BASE24-eps' user interface, employs Java, C++, and XML technologies that provide the user interfaces that are needed to manage all components of the application.

With the system's built-in user security feature, users are assigned roles that grant them permission to specific functions and tasks that are associated with various windows. Users are authenticated during the logon process, thereby minimizing the risk that is associated with unauthorized users gaining access to functions that they are not permitted to perform.

The user audit function is responsible for maintaining a secure audit database where all file maintenance transactions and modifications to the user security database are recorded. Before and after images of the affected record are logged wherever appropriate.

The user interface design incorporates the flexibility for users to alter the layout and wording on the desktop to meet individual organization needs. All text and positional information is maintained in configuration files, so adapting the user interface without altering the product code is particularly easy. This structure also incorporates multi-language capability.

## 2.14 Rich functionality

BASE24-eps provides full functionality to support payment transactions across multiple channels. The software is parameter-driven, which allows users to configure a system that meets their unique business requirements. The ACI product investment strategy accommodates periodic new releases of software that provides support for both regulatory changes and new trends in electronic delivery.

## 2.15 Platform independence

BASE24-eps supports a broad range of computing environments, which allows customers to operate the ACI best-of-breed software on their choice of industry-standard platforms. BASE24-eps operates on a variety of HP, IBM, and Sun™ servers. On each platform, ACI software takes advantage of the best in systems software for reliability, availability, and high-performance throughput.

## 2.16 Flexible architecture

Because the design of BASE24-eps includes support for scripting, there is little need for customer technical staff to have knowledge of the core languages of the application. A working knowledge of JavaScript programming methods prepares an experienced programmer to maintain and create the business logic that is necessary to meet the institution's authorization processing needs.

Because the BASE24-eps application is component-based, ACI customers have the freedom to develop components in-house, which extends product functionality. To accomplish this task, some basic skills concepts are required:

- ▶ UML (Unified Modeling Language)
- ▶ C++
- ▶ Java

## 2.17 Financial transaction flows

BASE24-eps card transactions might be broadly categorized as on-us (transactions that involve a card that is issued by the institution that is running the local copy of BASE24-eps) and not-on-us (transactions that involve a card that some other institution issues). On-us transactions are generally authorized by BASE24-eps or are routed by BASE24-eps to a back end host application for authorization. Not-on-us transactions are generally switched by BASE24-eps to a national or local card network for authorization.

Transactions can be further categorized by authorization type. Both on-us and not-on-us transactions can also be independently categorized by acquirer type as switch-acquired transactions, device-acquired transactions, or host-acquired transactions.

### 2.17.1 Authorization types

A BASE24-eps installation typically supports at least two authorization types: offline and online. Not-on-us transactions are typically switched transactions, while on-us transactions are processed with one of the other authorization types.

#### Switched transactions

BASE24-eps routes switched transactions to a local or national card network for authorization. The card networks commonly prefer that the BASE24-eps system connect to them as a TCP/IP client through existing protocols that are commonly still supported.

#### Offline authorization

BASE24-eps authorizes offline transactions. Authorization requests are not forwarded to a host application; therefore, offline authorization scripts perform more extensive authorization processing and data checking than the scripts that are used to perform prescreening in an online environment. In offline authorization, BASE24-eps maintains authorization data and records of financial transactions processed authorization data. Therefore the BASE24-eps database must be periodically synchronized with the database of record. Authorization data must be refreshed from the database of record, and the database of record must be updated with the financial transactions that BASE24-eps processes.

#### Online authorization

Online transactions are transactions that BASE24-eps routes to a back-end host application for authorization. If BASE24-eps cannot communicate with the host application, it declines all transactions that would otherwise get routed to the host application. Although BASE24-eps is not the authorizer, it can be configured to prescreen transactions. If a transaction does not satisfy the prescreen criteria, it is declined and a notification is sent to the host application. If a transaction does satisfy the prescreen criteria, it is forwarded to the host application for authorization. Prescreening scripts are designed to do limited authorization processing.

On other platforms, connectivity to the back end host application is generally accomplished by some variety of TCP/IP data communications. Although this is also possible in the z/OS implementation of BASE24-eps, it is generally preferable on System z with a System z-based host application to take advantage of the fact that BASE24-eps and the back end application are co-located. BASE24-eps supports several options natively. Connectivity is generally driven by the requirements of the back end host system, and is often customized to meet those requirements.

## Online/Offline authorization

In a combined Online/Offline environment, BASE24-eps can be configured to prescreen transactions before routing to a back end host application for authorization, as in Online authorization. However, in an Online/Offline environment, BASE24-eps also stands in for the host application and authorizes transactions when the host application is unavailable. Transactions that BASE24-eps authorizes are stored and forwarded to the host application when communication is restored.

Considerations for communications with the back end host application are the same as for online authorization that we described in “Online authorization” on page 16. Considerations for the authorization database and financial transaction journals are similar to the considerations for offline authorization, which we described in “Offline authorization” on page 16.

### 2.17.2 Authorization methods for on-us cards

Typically, three basic authorization methods are used to authorize on-us cards in a BASE24-eps system:

- ▶ Negative Authorization with Usage Accumulation Method
- ▶ Positive Authorization Method
- ▶ Positive Balance Authorization Method

Although other authorization methods can be scripted, ACI provides a sample set of the fundamental authorization scripts for each authorization method: Negative Authorization with Usage Accumulation, Positive Authorization, and Positive Balance Authorization.

#### Negative Authorization with Usage Accumulation method

With the Negative Authorization with Usage Accumulation method (also known as Negative Card with Usages or NEGU method), BASE24-eps assumes a card is good if it is not found in the BASE24-eps authorization database. Only bad cards exist in the database for this authorization method.

If the transaction is a cash disbursement, the usage accumulation totals in the BASE24-eps database are checked against transaction limits. Based on those totals and limits the BASE24-eps NEGU authorization script approves or declines the transaction request and updates the usage accumulation totals accordingly.

#### Positive Authorization method

The Positive Authorization method (also known as Positive Card with Usages or PCA method) uses BASE24-eps authorization scripts that authorize transactions using information that is maintained in the BASE24-eps database. For the transaction to be approved, a record must exist in the database for the card that initiated the transaction and the status of the account for the card must be good. If the transaction is a cash disbursement, the amount previously disbursed plus any holds plus the new transaction amount must not exceed the cardholder's transaction limits. Based on these totals and limits, the script approves or declines the transaction and updates the usage totals accordingly.

#### Positive Balance Authorization method

The Positive Balance Authorization method (also known as Positive Card with Usages and Balances or PCBA method) uses BASE24-eps authorization scripts that authorize transactions using information that is maintained in the BASE24-eps database. In addition to the authorization steps that are supported in the PCA method, a record with the account balance must exist in the BASE24-eps database and the amount of the transaction must not

exceed the balance in the account. Based on these constraints, the script approves or declines the transaction request and updates totals accordingly.

### 2.17.3 Acquirer types

The acquirer type generally has little direct impact on flow of the transaction other than determining the specific module within BASE24-eps to which the external message is routed for parsing.

#### Device-acquired transactions

Device-acquired transactions include transactions that originate from ATM and POS devices. They can include a mix of on-us and not-on-us transactions, and can be authorized by any of the authorization types.

Device-acquired transactions are further divided based on the actual type of the acquiring device and the message format that it supports. Separate parser components within the Integrated Server are required for each message format.

#### Switch-acquired transactions

Switch-acquired transactions are transactions that are acquired from local or national card networks and routed to BASE24-eps for authorization. They generally represent on-us transactions only and are authorized as one of the Online, Offline, or Online/Offline authorization types.

#### Host-acquired transactions

Host-acquired transactions are typically transactions devices or card networks (that are connected to a pre-existing host application) acquire and route to BASE24-eps for authorization or routing. Host-acquired transactions are most generally not-on-us transactions (because on-us transactions are often processed by the pre-existing host application itself), but can be a mix of switched and offline transactions.

Connectivity is generally driven by the requirements of the acquiring host system and is almost always customized to meet those requirements.

## 2.18 ACI Enterprise Payments Solutions

ACI offers mission-critical e-payment software solutions to manage transactions from the point-of-access to settlement. BASE24-eps is part of the ACI Enterprise Payments Solutions, which is a broad suite of products that ACI developed. The ACI Enterprise Payments Solutions include software to enable transaction processing through evolving Internet and wireless channels and traditional ATM and POS channels. ACI solutions processes transactions in real time and automates the back-office functions that are associated with settlement, dispute processing, fraud detection, and account service.



## System z and z/OS

In this chapter, we discuss the robust and mature architecture of z Series processors and the z/OS operating system. We also discuss how these capabilities support the availability, reliability, manageability, and data integrity requirements of an ACI BASE24-eps implementation.

The topics that we discuss in this chapter are:

- ▶ 3.1, “An IBM tradition of System z value” on page 20
- ▶ 3.2, “System z in the finance industry” on page 21
- ▶ 3.3, “Parallel Sysplex: A high-availability clustering configuration” on page 25

## 3.1 An IBM tradition of System z value

As the world's largest provider of financial services business solutions, IBM can help customers of all sizes transform their business and capitalize on modern applications, which are all built on an optimized technology infrastructure with exceptional levels of availability, performance, and cost control. Two thirds of the world's business transactions are processed on IBM System z mainframes and have been for four decades.

### 3.1.1 System z architecture

Starting with the first large machines, which arrived on the scene in the 1960s, each new generation of mainframe computers included improvements in the following areas of the architecture:

- ▶ More and faster processors
- ▶ More physical memory and greater memory addressing capability
- ▶ Enhanced devices for data storage
- ▶ More and faster channels between storage devices and processors
- ▶ Dynamic capabilities for upgrading both hardware and software
- ▶ Increased automation of hardware error checking and recovery
- ▶ A greater ability to divide the resources of one machine into multiple, logically independent and isolated systems that each run its own operating system
- ▶ Advanced clustering technologies, such as Parallel Sysplex, and the ability to share data among multiple systems.

Mainframe computers remain the most stable, secure, and compatible of all computing platforms. The latest models can handle the most advanced and demanding customer workloads, yet continue to run applications that were written decades ago.

With the expanded functions and added tiers of data processing capabilities, such as Web-serving, autonomics, disaster recovery, and grid computing, the mainframe computer is riding the next wave of growth in the IT industry.

### 3.1.2 Who uses mainframe computers

Just about *everyone* has used a mainframe computer at one point or another, for example, many automated teller machines are connected to mainframes. In banking, finance, healthcare, insurance, utilities, government, and a multitude of other public and private enterprises, the mainframe computer continues to be the foundation of modern business.

Until the mid-1990s, mainframes provided the *only* acceptable means of handling the data processing requirements of a large business. These requirements were then (and are often now) based on large and complex batch jobs, such as payroll and general ledger processing.

The mainframe owes much of its popularity and longevity to its inherent reliability and stability, a result of careful and steady technological advances. No other computer architecture can claim as much continuous, evolutionary improvement, while maintaining compatibility with previous releases.

Because of these design strengths, the mainframe is often used by IT organizations to host the most important, *mission-critical* applications. These applications typically include

customer order processing, financial transactions, production and inventory control, payroll, and many other types of work.

For more information about the value of the IBM System z platform, see *IBM System z Strengths and Values*, SG24-7333.<sup>1</sup>

## 3.2 System z in the finance industry

Financial institutions cite the following reasons for choosing the IBM System z. We discuss them in more detail in this chapter:

- ▶ High availability
- ▶ Centralized data storage
- ▶ Recovery
- ▶ Workload management
- ▶ Performance management
- ▶ Scalability
- ▶ Security
- ▶ Encryption

### 3.2.1 High availability

There are some financial applications that require near-continuous availability. Availability in this context means the ability of all or some of the users and applications to access the production databases. The question of what percentage of work can be processed on a continuous basis is an economic decision based on cost and the value of continuous availability to the organization.

Although there is no standard definition of near-continuous availability, many large companies adopted a goal of no more than five to 10 hours per year of planned and unplanned outages, which translates to an overall systems availability of 99.9%, which contrasts with the historical paradigm of a maintenance window of eight hours every weekend or 400 hours of outage per year. Other companies are adopting service-level objectives that are slightly less demanding but still challenging, for example, an objective of a quarterly four-hour maintenance window (16 hours per year of planned outage) requires many of the same design principles that the near-continuous availability objective requires.

Let us discuss this concept in more detail.

#### Continuous availability

One popular definition of continuous availability is *high availability plus continuous operations*. There are four essential building blocks for a continuously available system:

- ▶ A design with no single points-of-failure
- ▶ Highly reliable hardware and software components
- ▶ Ability to avoid planned outages
- ▶ Seamless mechanisms for relocating work loads

#### High availability

There is much confusion in the industry over the use of the terms *reliability* and *availability*. Reliability is the resilience of a system or component to unplanned outages, which is typically achieved through quality components, internal redundancy, and sophisticated recovery or

<sup>1</sup> <http://www.redbooks.ibm.com/abstracts/sg247333.html>

correction mechanisms that applies to software and hardware. z/OS has millions of lines of code that are devoted to correction and recovery. This reliability coupled with a design that eliminates single points-of-failure at the component level provides what is known in the industry as high availability. In fact, this is high reliability, or the ability to avoid unplanned incidents. To achieve continuous availability, we strive for a design that has 99.999% reliability.

### **Continuous operations**

The other component of continuous availability is continuous operations, which is the ability to minimize planned outages, such as administrative or maintenance work. The continuous operations include the ability to upgrade and maintain the central processors, the operating systems, DB2®, coupling facilities, and ACI application software without disrupting the cardholder's ability to complete a given transaction.

### **System z features**

The System z server architecture is designed for continuous availability. It includes self-healing capabilities to avoid downtime that is caused by system crashes.

The System z strategy is to provide reliability, availability, and serviceability (RAS) is a building-block approach that was developed to meet customers' stringent requirements for achieving continuous reliable operation (CRO). The building blocks are:

- ▶ Error prevention
- ▶ Error detection
- ▶ Recovery
- ▶ Problem determination
- ▶ Service structure
- ▶ Change management
- ▶ Measurement and analysis

A primary focus is on preventing failures from occurring in the first place, which is accomplished by using high-reliability components and employing screening, sorting, burn-in, and run-in techniques. Failures are also eliminated through rigorous design rules, design walkthroughs, peer reviews, simulations, and extensive engineering and manufacturing testing.

For more information about the latest System z hardware availability and scalability items, see the IBM System z10™ Enterprise Class Technical Guide, SG24-7516 at:

<http://www.redbooks.ibm.com/abstracts/sg247516.html>

## **3.2.2 Centralized data storage**

Centralized data storage provides customers that use IBM System z with a highly regarded database server technology, configuration options, support, services, and financial incentives for high-availability, security-rich database serving and business value.

System z provides the strategic, centralized access point for customer and corporate data that is used for marketing and analytics. Multiple applications running in individual logical partitions on System z access the centralized data repository. This approach provides very high systems availability with attractive total cost-of-ownership for customer solutions.

### 3.2.3 Recovery

With z/OS and DB2 integrated backup and recovery back office functions stay up and running.

#### Resource Recovery Services

With the increasing number of resource managers that are available on z/OS, there was a need for a general sync point manager that any resource manager could exploit. Resource Recovery Services (RRS) enables transactions to update protected resources managed by many resource managers.

#### Automatic Restart Manager

The Automatic Restart Manager (ARM) enables fast recovery of the subsystems that might hold critical resources at the time of failure. If other instances of the subsystem in a Parallel Sysplex need any of these critical resources, fast recovery makes these resources available more quickly. Even though automation packages are used today to restart the subsystem to resolve such deadlocks, ARM can be activated closer to the time of failure. It provides automatic restart after a started task or job failure and automatic redistribution of work to an appropriate system following a system failure.

### 3.2.4 Workload management

One of the strengths of the System z platform and the z/OS operating system is the ability to run multiple workloads simultaneously, either within one z/OS image or across multiple images. The function that makes this possible is dynamic workload management, which is implemented in the Workload Manager (WLM) component of z/OS. It prioritizes production jobs and user response time to ensure that the bank's business objectives are met.

Using WLM banks can balance their needs for line reporting, transaction management, and payments operations all on the same system to meet customer peak performance needs and balance the priorities.

Even more flexibility is provided by using Intelligent Resource Director (IRD) in conjunction with WLM. Whereas WLM can move workload to available resources, IRD can move processing power between Logical Partitions (LPs) to meet workload requirements. IRD can also be used to dynamically manage the Channel subsystem for I/O operations.

### 3.2.5 Performance management

Financial institutions need to provide fast response time to their customers. z/OS provides the ability to share resources and direct them dynamically and virtually, whenever and wherever they are needed, according to priorities and objectives that the bank sets. System z helps financial institutions to consolidate and tightly integrate multiple workloads on a single server with centralized systems management to reduce costs.

### 3.2.6 Scalability

Scalability tries to ensure that no performance limitation is due to hardware, software, or the size of the computing problem. IBM System z has excellent scalability. This scalability is significantly influenced by the hardware configuration, software configuration, and workload.

Scalable systems (that scale up and scale out) can scale on more than one application or measurement. Increasing performance by adding more processors is commonly referred to

as *scaling up*. Increasing performance by adding additional systems is referred to as *scaling out*. System z can do both.

The *Capacity on Demand* (CoD) capability allows you to non-disruptively add one or more Central Processors (CPs), Internal Coupling Facilities (ICFs), System z Application Assist Processor (zAAP), System z Integrated Information Processor (zIIP), and Integrated Facility for Linux (IFLs) to increase server resources when they are needed without incurring downtime. Capacity on Demand can quickly add processors up to the maximum number of available inactive engines. Also, additional books can be installed concurrently, providing additional processing units and memory capacity to a System z server.

This capability can provide customers with the capacity for much needed dynamic growth in an unpredictable payment systems world.

The CoD functions include:

- ▶ Non-disruptive CP, ICF, IFL, zAAP, and zIIP upgrades
- ▶ Dynamic upgrade of I/O capacity
- ▶ Dynamic upgrade of memory

### 3.2.7 Security

It is generally believed that about 70 percent of all large enterprise business data resides on mainframe server platforms that run IBM z/OS or its predecessor, OS/390®. From a security perspective, mainframe computer operating systems are the de facto intellectual standard to which newer operating systems are compared.

z/OS, from its beginnings as Multiple Virtual Storage (MVS) in the 1970s, was built on System/360 (S/360), and now System z is a strong hardware architecture that provides for the safe execution of multiple applications on one system.

Working storage for each application is separated from the storage of other applications and from the storage that the supervisor functions of the operating system use. In addition, applications run in a different hardware "state" than supervisor functions and, therefore, do not have direct access to privileged hardware instructions. On top of this is layered a common point of user authentication and access control, both as a way to reduce the need for each application to provide these services uniquely and as a way to reduce the administrative burden.

Twenty years ago, of course, this security structure needed only to support an environment of isolated batch processing systems or systems with terminal access through private corporate networks. Over time, and especially in the past decade, customer businesses changed dramatically, reaching out to their suppliers, distributors, and customers through a variety of networking techniques. Businesses are now truly worldwide operations and rife with merger activity. In short, the nature of business changed and so have the security requirements.

One change is that a single application, which can now start on the Internet, is expected to run across multiple platforms. As this has happened, OS/390 has further transformed into z/OS, and IBM had to marry the security infrastructure with many open security standards.

A second change is a rise in the importance of encryption: IBM enterprise servers have a long history with encryption, having introduced optional hardware-based encryption in 1991. Since 1997, hardware encryption has been standard in S/390® processors and (now) System z processors because IBM understands that encryption is the basis of security when using the Internet and that customers require the performance and security that the IBM tamper-resistant hardware encryption provides.

Some security functions are provided in the base z/OS product and others are packaged in an optional z/OS Security Server feature. Chief among the functions that are packaged within the z/OS Security Server is the Resource Access Control Facility (RACF), which incorporates various elements of security, such as user identification, authentication, and access control.

The functions of the z/OS Security Server can be classified as:

- ▶ User identification and authentication
- ▶ Basic authentication
- ▶ Trusted third-party identification and authentication
- ▶ System Authorization Facility
- ▶ Access control
- ▶ Auditing and logging
- ▶ Networking and communications security

### 3.2.8 Encryption

Encryption is a vital part of today's information systems. Transactions that are sent across networks must be protected from eavesdropping and alteration. Data files on Internet-connected servers must be protected from malicious hackers. Secure Sockets Layer (SSL) traffic must be encrypted at high speeds. The list of areas that benefit from encryption grows every year.

In addition to the Hardware Security Modules (HSMs) that are available with other platforms, BASE24-eps on System z can take full advantage of the IBM Crypto Express 2 card using the Integrated Cryptographic Service Facility (ICSF) for very high speed and highly available cryptographic services, such as Personal Identification Number (PIN) translation and verification and Message Authentication Code (MAC) generation and validation.

## 3.3 Parallel Sysplex: A high-availability clustering configuration

The z/OS operating system can be configured into a Parallel Sysplex, which is the clustering technology for the mainframes. A sysplex refers to a tightly coupled cluster of independent instances of the z/OS operating system. The main objective of a Parallel Sysplex is continuous availability without compromising perceived client performance.

A sysplex can be either basic or parallel. A basic sysplex can communicate using channel-to-channel (CTC) connections between LPARs. Parallel Sysplex uses a processor called a coupling facility (CF). Information, such as workload, status, and data transmission, occurs through the coupling facility. The information sharing is constant and continuous, which allows the independent z/OS images to know detailed information about the current status of all images within the sysplex.

Parallel Sysplex architecture is designed to integrate up to 32 systems in one cluster. Each of these systems can handle multiple different workloads and access databases using data-sharing technology. A properly configured Parallel Sysplex cluster is designed to remain available to its users and applications with minimal downtime. Here are some examples:

- ▶ Hardware and software components provide for concurrency to facilitate nondisruptive maintenance, for example, Capacity Upgrade on Demand allows processing or coupling capacity to be added, one engine at a time without disruption to running workloads.

- ▶ DASD subsystems employ disk mirroring or RAID technologies to help protect against data loss. They exploit technologies to enable point-in-time backup without the need to shut down applications.
- ▶ Networking technologies deliver functions, such as VTAM Generic Resources, Multi-Node Persistent Sessions, Virtual IP Addressing, and Sysplex Distributor, to provide fault-tolerant network connections.
- ▶ I/O subsystems support multiple paths and dynamic switching to prevent loss of data access and improved throughput.
- ▶ z/OS software components allow new software releases to coexist with previous levels of those software components to facilitate rolling maintenance.
- ▶ Business applications are *data sharing-enabled* and cloned across servers to allow workload balancing, which also maintains application availability in the event of an outage.
- ▶ Operational and recovery processes are fully automated and transparent to users. They reduce or eliminate the need for human intervention.

High availability requires at least two server instances that provide the same service to their clients, so they can allow for the recovery of service when failures occur. That is, servers perform a backup function for each other within the cluster. High availability minimizes unplanned outages.

Continuous availability is achieved with the System z and z/OS Parallel Sysplex clustering technology. This technology implements a data-sharing design that allows a database to be concurrently read and updated by application clones running on multiple z/OS images on one or more physical servers. Continuous availability avoids or minimizes unplanned outages (high availability) and reduces or eliminates planned outages.

The Workload Manager balances application workloads across the systems in the Parallel Sysplex. If there is a failure or a planned outage on one system, other systems within the Parallel Sysplex take over the full workload. By implementing Geographically Dispersed Parallel Sysplex™ (GDPS®), the servers can be as far as 40 km apart from each other, which avoids a total site-wide disaster.

Using the Parallel Sysplex clustering architecture, you can achieve a near-continuous availability of 99.999% or five minutes of downtime a year.



## BASE24-eps architecture

In this chapter, we provide additional details about the BASE24-eps architecture.

The topics that we discuss are:

- ▶ 4.2, “BASE24-eps architecture on System z” on page 32
- ▶ 4.3, “BASE24-eps performance on System z” on page 44

## 4.1 BASE24-eps architecture

BASE24-eps is designed to run on many hardware, database, and middleware configurations (platforms). The logical architecture of BASE24-eps is the same on every platform.

Figure 4-1 shows the BASE24-eps logical architecture, which we describe in more detail in this section.

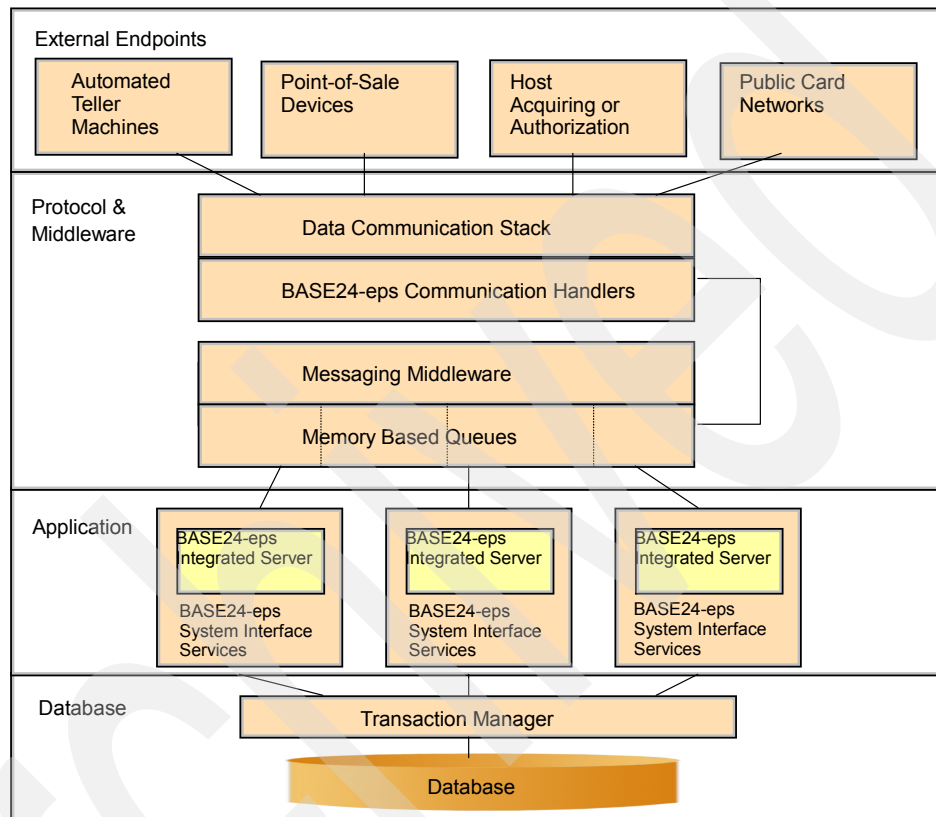


Figure 4-1 BASE24-eps logical architecture

### 4.1.1 Core platform services

Any platform on which BASE24-eps is implemented must provide certain core services. In this section, we describe the essential services that each platform must provide to be a candidate for a BASE24-eps implementation.

#### Messaging middleware

ACI employs an asynchronous model for all mission-critical online applications, of which BASE24-eps is one. BASE24-eps relies on the use of messaging queues that act as buffers between endpoints and the payment engine. Queuing of messages during transient spikes of transaction activity allow BASE24-eps to process transaction load with a consistent resource usage pattern.

In the event of a slowdown in the network or an unresponsive host system or storage problem, industry standard end-to-end protocols dictate time-out processing to occur. This model allows BASE24-eps to provide a consistent cost per transaction, which positions customers to better plan for growth and effectively handle current volumes.

While BASE24-eps can work with recoverable message queues, it does not require them. The possibility of message loss in the data communications network requires the use of application-level end-to-end protocols that provide recovery mechanisms for lost messages. Those end-to-end protocols also make it possible to use more efficient memory-based message queuing internally within BASE24-eps. In the unlikely event that messages on a memory-based queue are lost, end-to-end protocols assure financial integrity.

BASE24-eps performs best with a message queuing subsystem that allows blocking queue reads.

Figure 4-2 shows the asynchronous messaging model.

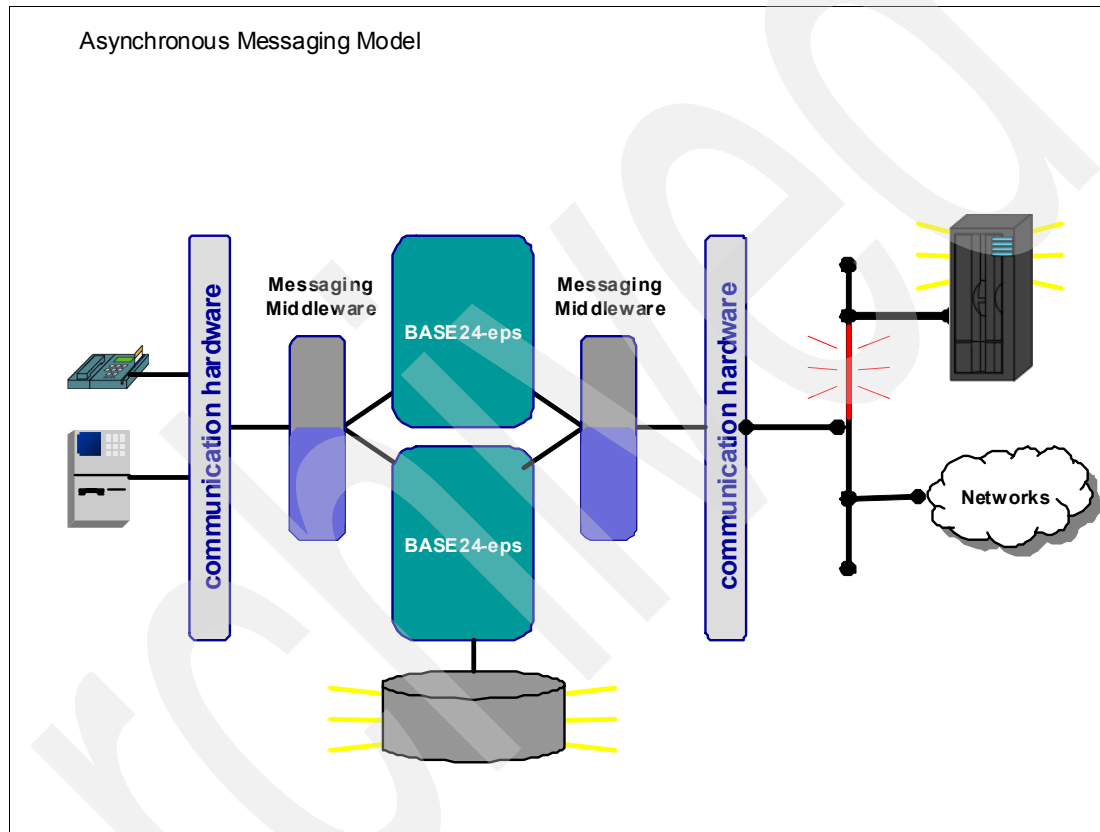


Figure 4-2 Asynchronous messaging model

## Database

BASE24-eps requires indexed access to structured data along with row locking and the other features that are commonly associated with a structured access method.

BASE24-eps data abstraction allows the use of a wide variety of databases. The BASE24-eps Data Access Layer (DAL) can make use of indexed-sequential, relational databases and memory tables, which are all accessed in a consistent fashion by the application logic.

In addition to indexed access to structured data BASE24-eps can, if the platform supports it, access certain tables as static memory tables for improved performance.

## Transaction management

BASE24-eps requires a transaction manager that can provide transactional integrity and rollback capability. BASE24-eps relies on the transaction manager to ensure that the

database integrity is maintained and that all interrelated files and tables are in a consistent state, even after failures occur.

### **Data communications**

At a minimum, BASE24-eps requires a sockets-based TCP/IP communications stack.

### **Other requirements**

In addition to the services that we define here, BASE24-eps has inherent non-functional requirements for a platform that meets requisite standards of scalability, manageability, and reliability.

## **4.1.2 Core components and features**

BASE24-eps, when deployed on any platform, has certain core components and features, and we discuss these core components and features in this section.

### **Communications handlers**

The communications handlers acquire transactions from remote systems and device or connection concentrators. The communications handler provides the bridge between the networking world and the application world, and as such, offers the application abstraction from communications details, application notification, and error handling.

### **Stateless application server processing**

BASE24-eps achieves scalability, availability, load balancing, and manageability through a population of context-free application server processes (Integrated Server) that provide all acquiring and issuing endpoint logic into a single executable. Any Integrated Server process can handle any transaction message (request, response, or external file update message).

### **User interface server**

The BASE24-eps ACI Desktop services are provided by a Java-based UI server known as ESWeb. ESWeb is hosted in a Web application server that does not require advanced Java Enterprise Edition features.

There are a few services that ESWeb does not provide. In those cases, ESWeb forwards the request to the XML Server. The XML Server is a C++ based server.

### **User interface client**

The BASE24-eps ACI Desktop Client is a thick-client GUI employing Java Swing technology. A Desktop Client accesses the UI Server over SSL-protected HTTP communications.

### **Application management**

The ACI Application management service includes an Application Management Agent component that is installed in any LPAR that executes ACI software. The Application Management Agent is the repository for application (and middleware) health status and statistics information, and it is responsible for making this data available to external enterprise management platforms, such as IBM Tivoli Monitoring. The Application Management Agent also acts as a point of command and control for many aspects of the BASE24-eps application.

## Diagnostic events

The ACI Event Service provides a common way for the application to generate operator alerts and notifications. The ACI Event Service provides decoupling of event generation and event distribution and configurable routing of operator events. The Event Service also provides the ability to supply rich event streams to standard console browsing and management products.

## Portable application architecture

Internally, each BASE24-eps address space is made up of three tiers. The bulk of the code is platform-independent.

Platform independent Business services (BU) components make up the highest tier. These Business services consist of a collection of C++ objects that are coordinated by a central scripting engine. Also located in the highest tier are platform-independent interfaces to various external endpoints and device interfaces to remote endpoints, such as Point Of Sale (POS) and Automated Teller Machine (ATM) devices, and switch interfaces to external networks, such as VISA and MasterCard. Switch interfaces are version-independent interfaces that are not tied to a specific release of BASE24-eps, which allows more efficient implementation of mandates exposed by the external networks across releases of BASE24-eps.

Platform-independent Foundation (FN) components make up the second tier. Foundation components extend and add functionality to the more basic APIs that are provided by the lowest tier, System Interface Services (SIS).

System Interface Services make up the lowest tier and provide a platform-independent API and a platform-dependent implementation that allow the code to perform its business functions on a defined list of platforms that provide the prerequisite services.

Figure 4-3 on page 32 shows the internal structure of a BASE24-eps address space.

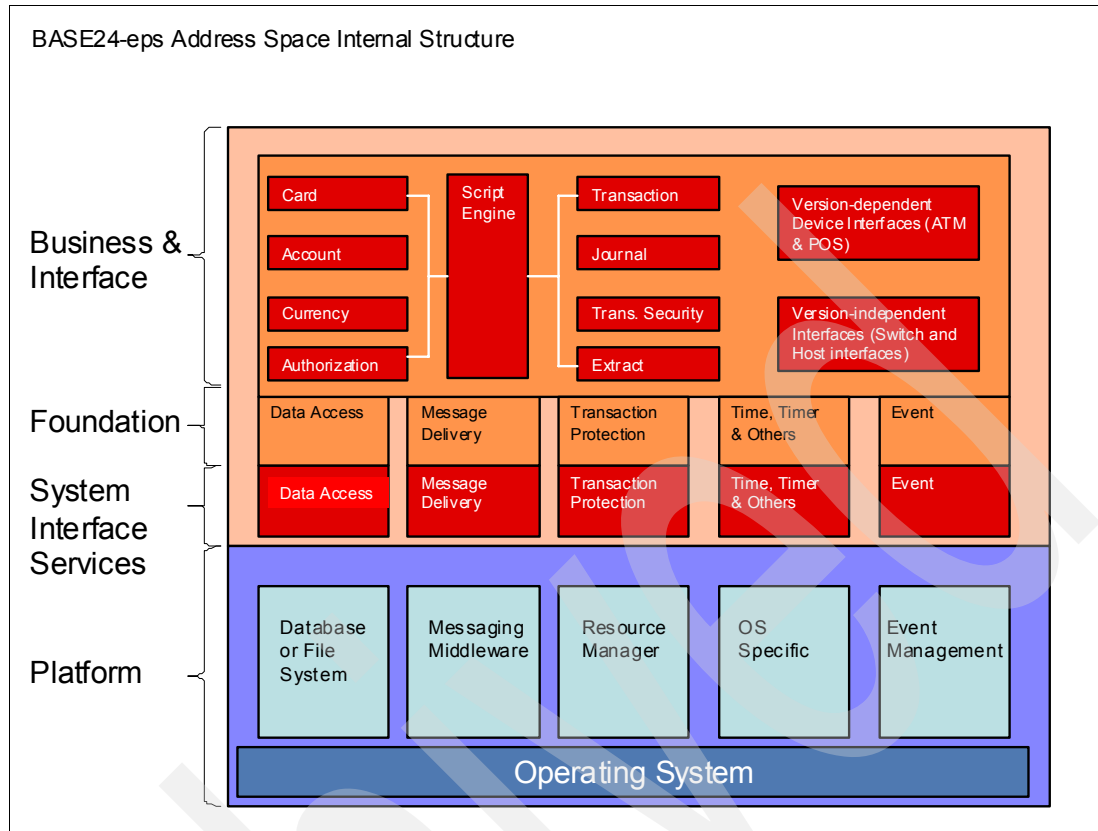


Figure 4-3 BASE24-eps address space internal structure

The core component of BASE24-eps is the BASE24-eps Integrated Server, so-called because it typically contains a C++ component that can manage any type of remote endpoint message. For this reason and because the Integrated Server is context-free (transaction context is saved on the system rather than in address space memory), it is never necessary to route any inbound financial message to any specific instance of the Integrated Server. The Integrated Server is implemented as a service class, wherein any member of the class can deal with any unit of work.

System Interface Services components rely on the underlying middleware and operating system to provide certain essential services. System Interface Services can be implemented and BASE24-eps can be deployed on any platform that provides those essential services.

## 4.2 BASE24-eps architecture on System z

Starting in the Version 08.2 release of BASE24-eps for System z, the redesigned System Interface Services (SIS) layer takes better advantage of the advanced z/OS and middleware that are available on System z. In this section, we describe how the platform requirements of the BASE24-eps architecture are met in the System z implementation.

### 4.2.1 BASE24-eps core platform services on System z

In this section, we describe how System z provides the core platform services that a BASE24-eps implementation requires.

## Messaging middleware

The Integrated Server consumes transaction messages from a common input queue that a number of running instances of the Integrated Server share. The Integrated Server input queue does not need to be configured for persistent message delivery because financial compensating transactions keep the database financially consistent in the case of an IBM WebSphere MQ Queue Manager failure.

IBM WebSphere MQ Shared Queues offer the best performance characteristics while also taking advantage of standard z/OS Workload Management capabilities.

IBM WebSphere MQ provides both persistent and non-persistent queues. Persistent queues store queued messages in an underlying data store where they can survive the failure of an IBM WebSphere MQ address space.

The end-to-end application-level protocols used by all remote endpoints make it generally unnecessary for BASE24-eps to make use of persistent queuing, which enables it to take advantage of the far better performance characteristic of memory-based queuing. With a few exceptions, using persistent IBM WebSphere MQ queues might result in significant performance degradation without a correspondingly large benefit, and is discouraged.

IBM WebSphere MQ also provides transactional queues such that queued messages survive the failure of a client address space. Once again, the trade-off between performance and any functionality gained is not good, and BASE24-eps does not generally make use of transactional IBM WebSphere MQ queues.

### **BASE24-eps queue definitions**

A basic set of IBM WebSphere MQ queue definitions are generated by the BASE24-eps installation procedures and placed in the MQSC file on the zFS. The queues can be defined automatically by the installation program or the queue definitions might be submitted to a system programmer.

There are at least two IBM WebSphere MQ queue definitions that are associated with each BASE24-eps process that uses the SIS Message Delivery Service to acquire work:

- ▶ The *service queue* is where the process gets its work. The service queue is shared by many instances of the server class for load balancing and availability. Messages on the service queue are messages that can be processed by *any* instance of a service class. These queues are by convention named <service class name>, for example, the service queue for the Integrated Server service class is 'IS'. The input queue for a process instance is defined in the runtime parameters that are passed to the process at startup.
- ▶ The *command queue* is associated with only one process instance. The command queue provides addressability too at the individual process level for command and control messages that must be processed by each instance of a service class. The queue name of the command queue is by convention <service class name><instance>, for example, the command queue for the first instance of the integrated server class is 'IS1'. The name of the command queue is a runtime parameter passed to the process at startup.

These queues are defined for each process that uses SIS message delivery to *acquire* work. Additional IBM WebSphere MQ queues must be defined for each thread in a SIS MDS-based application that uses the SIS Message Delivery Service to initiate synchronous IO operations. When a thread issues a synchronous request over IBM WebSphere MQ, it waits for a response on the dedicated response queue. These queues are statically defined because a process that issues a synchronous IO request usually does so in high-volume low-latency processing, where it is desirable to avoid the overhead of a dynamically created temporary queue. These queues are named <service class name><instance>.SYNC<logical thread

id>, for example, a dedicated response queue for logical thread 1 in process instance IS1 would be 'IS1.SYNC1'.

### SIS Message Delivery Service configuration files

Table 4-1 contains the definitions of files that System Interface Services (SIS) use for message routing on System z. An installation program creates these files and a text editor maintains them as necessary. The ADMF DB2 table contains dynamic information only and should not require user maintenance.

Table 4-1 System Interface Services (SIS) files

File	Description
Asynchronous Destination Map File (ADMF)	zFS file mapping of the BASE24-eps symbolic name to physical destination (typically an IBM WebSphere MQ queue) for statically-configured asynchronous destinations
Synchronous Destination Map File (SYDMF)	zFS file mapping of the BASE24-eps symbolic name to physical destination (typically an IBM WebSphere MQ queue) for statically-configured synchronous destinations
ADMF Table	DB2 table mapping of the BASE24-eps symbolic name to physical destination (typically an IBM WebSphere MQ queue) for dynamically-configured asynchronous destinations (for example, a symbolic name that is associated with a remote endpoint that might be associated with one of several MQs, depending on the LPAR to which the connection is routed). The ADMF table provides facilities that implement highly-available endpoint connection processing.

### Database

DB2 on z/OS is used as the database implementation. DB2 Data Sharing might be employed in a multiple-LPAR or Sysplex configuration.

### SIS Data Abstraction Layer configuration files

Table 4-2 shows the files that the System Interface Services (SIS) Data Abstraction Layer (DAL) uses. These files are not directly user maintainable; instead, they are generated during product installation and if necessary should be maintained only through the ACI Metadata Utility.

Table 4-2 System Interface Services (SIS) Data Abstraction Layer (DAL) files

File	Description
mdb.csv	zFS file describing schema, BASE24-eps data types that are associated with columns, indices, and so on. The format of this file is portable and is common to all platforms and thus contains information not used in the DB2 implementation such as column offset.
config.csv	zFS file maps the table assign names used by the portable BASE24-eps application code to platform specific table names, in this case DB2 tables, and associates the assign names with the appropriate schema from mdb.csv.

### Transaction management

DB2 transaction handling calls supply the transaction management for database integrity. The long-running BASE24-eps application processes manages the boundaries of a database



transaction explicitly using DB2 APIs. The BASE24-eps application is not itself a transaction manager.

## Data communications

z/OS sockets provide TCP/IP connectivity for ACI Communications Handlers. In addition, heritage protocols that might not be supported natively by the ACI Communications Handler are supported by means of High Performance Routing (HPR/IP) between the ACI Communications Handler and the IBM Virtual Telecommunications Access Method (VTAM).

Table 4-3 shows the configuration files that the ICE-XS™ Communications Handler use.

Table 4-3 ICE-XS communications handler configuration files

File	Description
icexs.param	A zFS text file is used as a template for the user to create new params files. The actual params file is specified on the run line with the <code>-params</code> option.
LICENCE_FILE	A parameter in the <i>params</i> file points to the zFS file system location where the license file is located. The license file is not user-modifiable and is used for execution and features authorization.
CONTROL_FILE	A parameter in the <i>params</i> file points to a configuration control file on the zFS file system. The configuration control file contains a disk-version (binary format) of the run-time configuration of ICE-XS. It is created and maintained by the ICE-XS subsystem and is not directly user-modifiable.
SECURITY_CONTROL_FILE	A parameter in the <i>params</i> file points to a <i>security control file</i> on the zFS file system. The security control file contains a disk-version (binary format) of the command security definitions, which are stored in a separate file so that they are unaffected by cold starts and also to enable the user to secure them more strictly.
NOF-XS Command files	NOF-XS is the command processor that is used to configure and manage the ICE-XS subsystem. NOF-XS command file(s) are optional text files on the zFS file system that contain NOF-XS commands that are used to configure the ICE-XS subsystem. These user-maintained files are optional only because the user could choose to enter their configuration interactively through NOF-XS every time they performed a cold start of the ICE-XS subsystem. In practice, though, users set up a configuration file or a set of configuration files that are logically broken up.

## 4.2.2 Core components and services

In this section, we describe how BASE24-eps core components and services are implemented on System z.

### Communications handlers

Communications handling is provided by a z/OS-resident application known as ICE-XS. ICE-XS interfaces to the IP stack on z/OS for a wide variety of communications handling options, which include:

- ▶ Standard TCP/IP Socket handling, which includes SSL authentication and encryption.
- ▶ Standard POS Device (VISAIL, SPDH) messaging support

- ▶ SNA using Data Link Switching (DLSw) and High Performance Routing over IP (HPR/IP). SNA support includes:
  - APPN Network Node (NN) and End Node (EN)
  - PU 2.0
  - PU 4/5 subset (support of downstream PU 2.0)
  - Independent LUs of any type (not just APPC)
  - Dependent LUs of any type
- ▶ X.25 over TCP/IP (Cisco XOT)
- ▶ HTTP Server and Client
- ▶ SOAP Server including Web Services Security (WSS)

ICE-XS also directly supports IBM WebSphere MQ external interfaces. Other connectivity protocols can be supported by means of integration with VTAM-based terminals.

ICE-XS consists of a set of components. Some are required for all intended uses, and some are needed only if particular features of ICE-XS are required. Figure 4-4 shows the relationships between the ICE-XS. A brief description of each of the components follows.

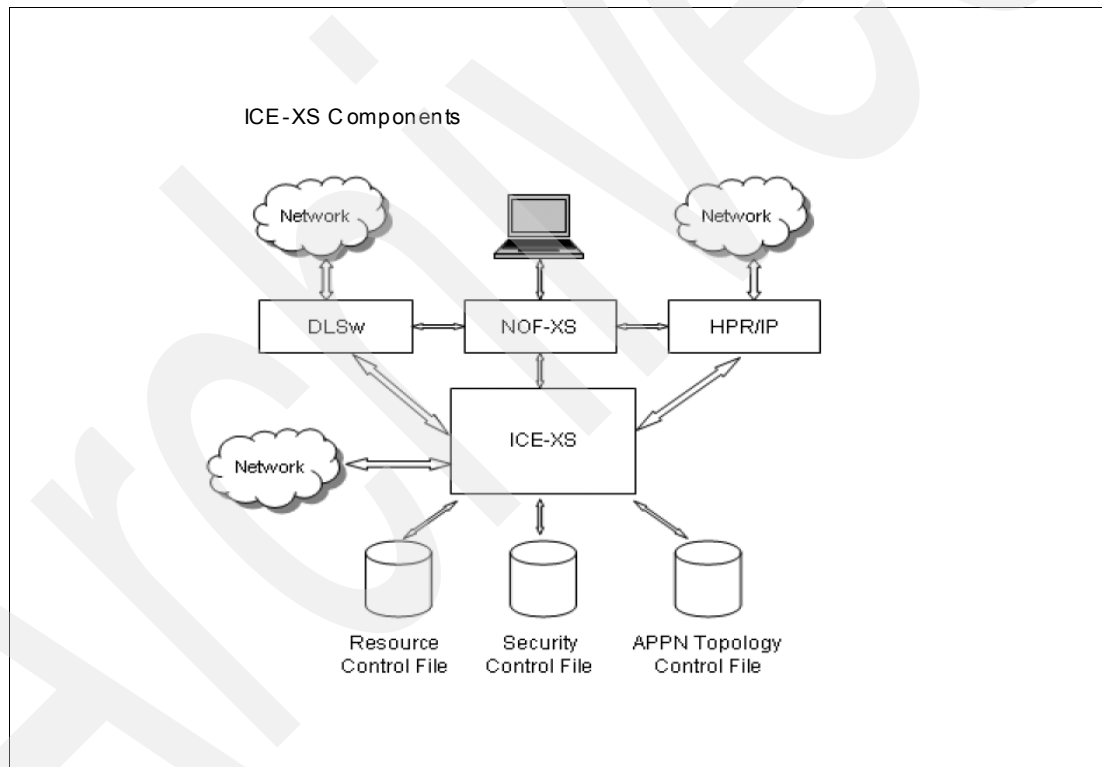


Figure 4-4 ICE-XS components

### **ICE-XS**

ICE-XS is the main process that provides interfaces to applications on the local system and coordinates the flow of messages to and from the various networks that it supports. ICE-XS contains the vast bulk of the logic for its supported network protocols.

### **NOF-XS**

The NOF-XS process is used to for real-time command and control of the ICE-XS, DLSw, and HPR/IP processes. It enables administrators to configure and monitor resources in each of those components.

### ***DLSw***

The DLSw process is optional and implements the Data Link Switching protocol when communicating with SNA networks over TCP/IP connections.

### ***HPR/IP***

The HPR/IP process is optional and implements the High Performance Routing (HPR) over Internet Protocol (IP) when communicating with SNA networks over UDP/IP datagrams.

### ***Resource Control File***

The RCF stores the configuration and state information that relates to resources that the ICE-XS process manages. DLSw and HPR/IP also maintain similar control files of their own (not shown).

### ***Security Control File***

The Security Control File stores authorization rules that determine which users can perform particular NOF-XS commands. DLSw and HPR/IP also maintain similar control files of their own (not shown).

### ***APPN Topology Control File***

The APPN Topology Control file is used by ICE-XS to maintain knowledge of an entire APPN network topology. It is only used if ICE-XS is configured to run as an APPN Network Node.

## **Stateless Application Server Processing**

BASE24-eps address spaces on System z are implemented as long-running batch processes that make use of POSIX services.

Although BASE24-eps address spaces can be started as MVS jobs with appropriate JCL or as UNIX System Services processes from a UNIX System Services prompt, they are typically started and managed by the ACI Application Management Service.

Address spaces started with appropriate JCL appear as MVS tasks in response to the **D J,L** MVS console command. Other address spaces appear as UNIX System Services processes in response to the **D OMVS,ASID=ALL** MVS console command or the shell **ps -ef** command.

### ***BASE24-eps Process Environment on System z***

BASE24-eps address spaces consist of multiple POSIX threads (pthreads). Configuring multiple pthreads per address space provides some significant performance gains over simply configuring more address spaces. While it is necessary to configure at least two address spaces for high availability, additional threads of execution are best configured by adding more logical threads per address space rather than more address spaces. In a typical deployment on the current System z architecture, this is true up to about six logical threads per address space. Beyond that, resource contention within the address space becomes an issue and configuring additional address spaces rather than additional logical threads per address space becomes more efficient.

The ACI threading model is a modified boss/worker model, where the boss thread dispatches a number of logical worker threads and then monitors their health. Each logical worker thread in an address space performs exactly the same processing as every other and exactly the same work as would an address space with a single logical worker thread. The number of logical threads is a runtime parameter passed to the address space at application startup. These logical threads are statically allocated, which keeps with the ACI asynchronous IO model.

Each logical worker thread can and usually does start additional POSIX pthreads to perform asynchronous IO. A pthread is dedicated to both the service and the command queue (“Messaging middleware” on page 28). These threads communicate with the logical worker threads using POSIX condition variables.

The boss thread also starts two additional utility threads to assist in handling POSIX signals.

Figure 4-5 shows the BASE24-eps threading model.

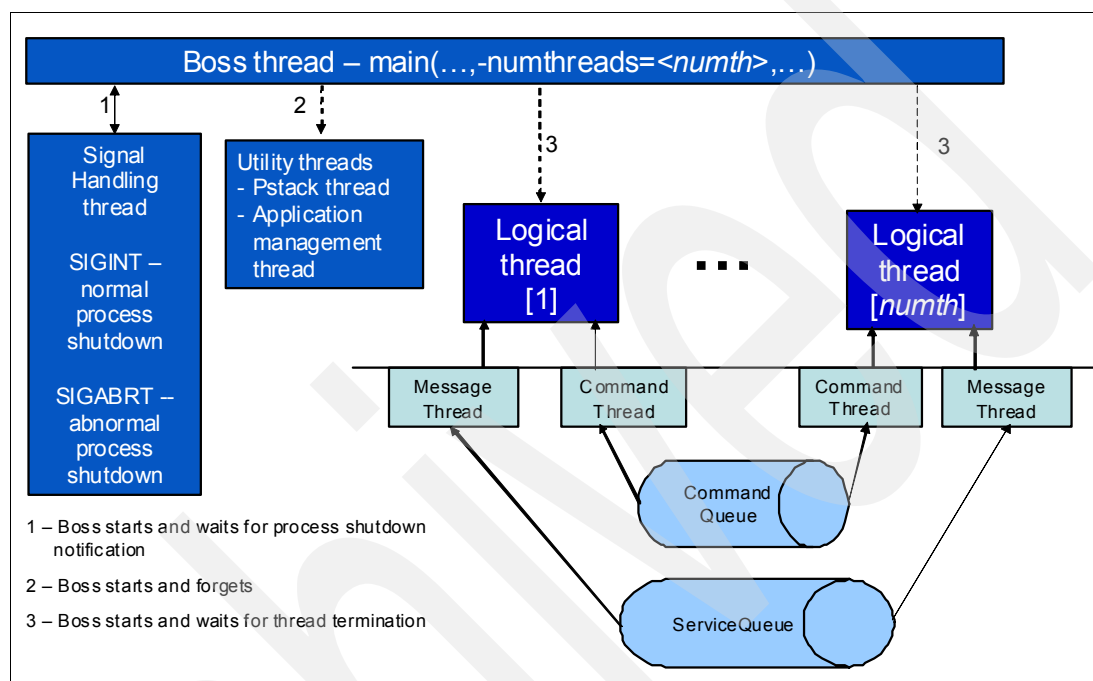


Figure 4-5 BASE24-eps threading model

### Long-running tasks

Most BASE24-eps address spaces are large C++ component-based applications. They consist of numerous statically-linked objects. They perform a substantial amount of work when they are initialized, initializing class member variables and registering with component factories.

While this architecture provides maximum flexibility (new components only need be statically linked into the executable to register with component factories and become available to the other components in the application) it also imposes a high startup cost. Accordingly, BASE24-eps applications are generally not triggered by the messaging middleware but are started by the management application and continue to run indefinitely.

### Context-free servers

BASE24-eps Integrated Servers are context-free in that they store request data on the system so that any financial message can be freely routed to any instance of the Integrated Server service class.

IBM WebSphere MQ on System z provides features that are unavailable on other platforms. One of these features is IBM WebSphere MQ Shared Queues. Using Shared Queues in a multi-LPAR environment, messages are queued in the coupling facility rather than in any individual LPAR, and are accessible to any queue manager in any LPAR.

Shared Queues facilitate the implementation of BASE24-eps context-free servers in two ways:

- ▶ Integrated Servers in one or more LPARs all pull their messages from the same Shared Queue. This pull-based model in conjunction with z/OS Workload Manager (WLM) provides better load balancing than the simple push-based model used on other platforms, where the MQ Manager decides on which Cluster Queue to place the message.
- ▶ A Shared Queue is a fast and convenient place to store transaction context. Relative to storing context in a database, MQ Shared Queues provide very fast and efficient access to context data from any LPAR in the Sysplex.

### **User Interface Server**

ESWeb can be executed in WebSphere Application Server on the System z or on a front-end system that is running in a Web application server. If configured on a front-end, access to the database is handled through remote database handling known as TDAL.

### **User Interface Client**

The BASE24-eps ACI Desktop Client is a thick-client GUI that employs Java Swing technologies. A Desktop Client accesses the UI Server over SSL-protected HTTP communications.

### **Application management**

BASE24-eps agents provide external interfaces for accessing the management data remotely, and it executes managed applications or acts as a proxy for applications that are external to it.

The Application Management functionality is implemented through two components:

- ▶ An agent that runs on the platform being monitored. The agent is a standalone Java application that is responsible for collection and distribution of all management data. The agent uses the Java Management Extensions (JMX™) technology.
- ▶ ACI Desktop screens that provide a graphical representation of the management data within the system.

The managed resources of the Application Management agents are defined as:

- ▶ Processes: Configuration and start/stop/restart for all styles of BASE24-eps application process.
- ▶ Queue Managers and Queues: Support for monitoring WebSphere MQ queues, configuration, and queue manager startup are still handled by the standard facilities within MQ.
- ▶ Data Communication facilities.

Figure 4-6 on page 40 shows a possible deployment on two Logical Partitions (LPARs) being managed by the ACI Management System.

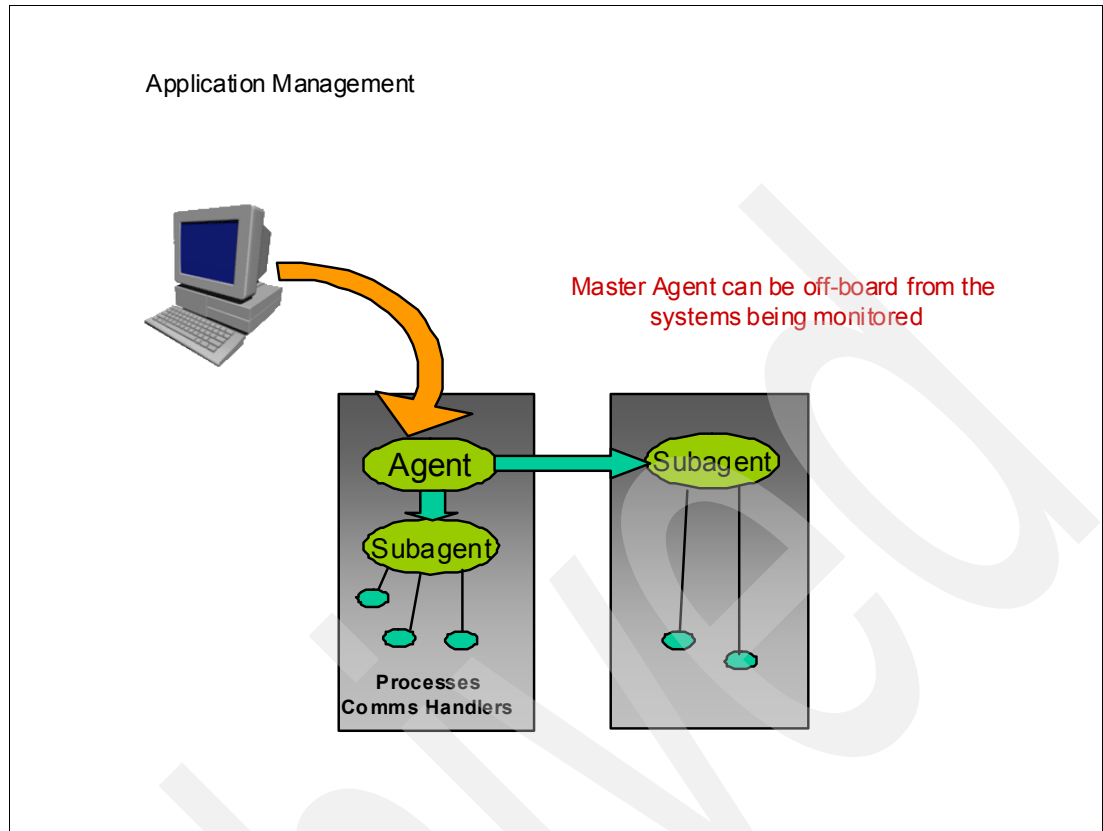


Figure 4-6 ACI management system

The ACI Application Management Server is a Java Management eXtension (JMX)-based address space that starts and manages the other ACI address spaces. The JMX address space contains a proxy JMX Bean (mBean) for each C or C++ address space. The actual management of the C/C++ address space is performed by the proxy mBean. The JMX API provides access for the ACI Application Management user interface and other management applications.

Figure 4-7 on page 41 shows the ACI Application Management Server with several JMX adapters managing two BASE24-eps Integrated Server processes.



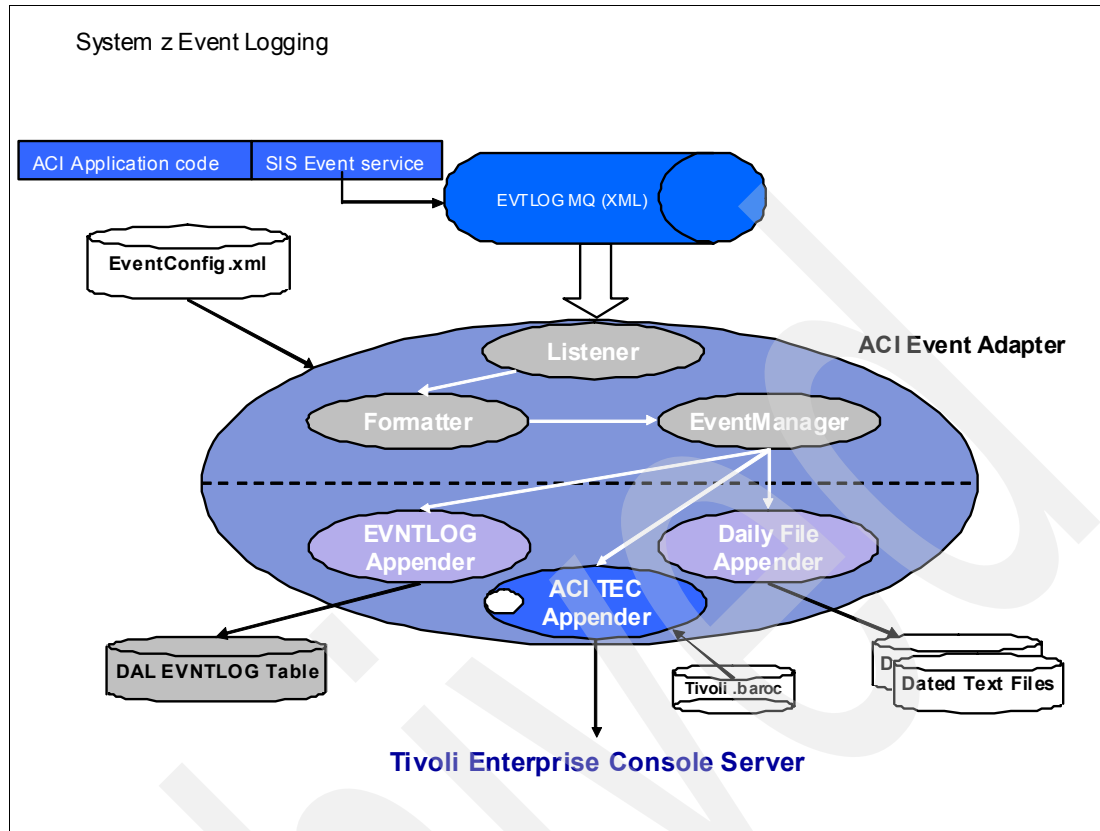


Figure 4-8 System z event logging

## Portable application architecture

In this section, we detail the portable application architecture of BASE24-eps.

### Message routing

BASE24-eps routing is configured by symbolic name. The platform-independent business logic and configuration files System Interface Services configuration files map the symbolic name exposed to the platform-independent business logic to platform-dependent constructs, such as IBM WebSphere MQ queues.

BASE24-eps applications, in most cases, send messages asynchronously, that is, they save context, send a message, and then proceed with other work and process the response to that message if and when it is received, rather than specifically waiting for a response. IBM WebSphere MQ is used for all BASE24-eps asynchronous messages that are destined for locations within the LPAR or Sysplex.

Some BASE24-eps messages are sent synchronously to low-latency destinations when the cost of asynchronous processing is prohibitive, such as messages to heritage Hardware Security Modules or HSMs. On System z, most synchronous messages are also routed over IBM WebSphere MQ Queues.

On System z only, BASE24-eps can optionally be configured to send certain financial request messages synchronously to a suitable low-latency and reliable CICS-based authorization system using either the CICS-MQ Bridging facility or the IBM External CICS Interface (EXCI). While requiring more threads of execution to support a specified transaction rate, this configuration can eliminate the overhead of asynchronous processing and managing transaction context.



## Transaction flows

Figure 4-9 illustrates the flow of a transaction through the system.

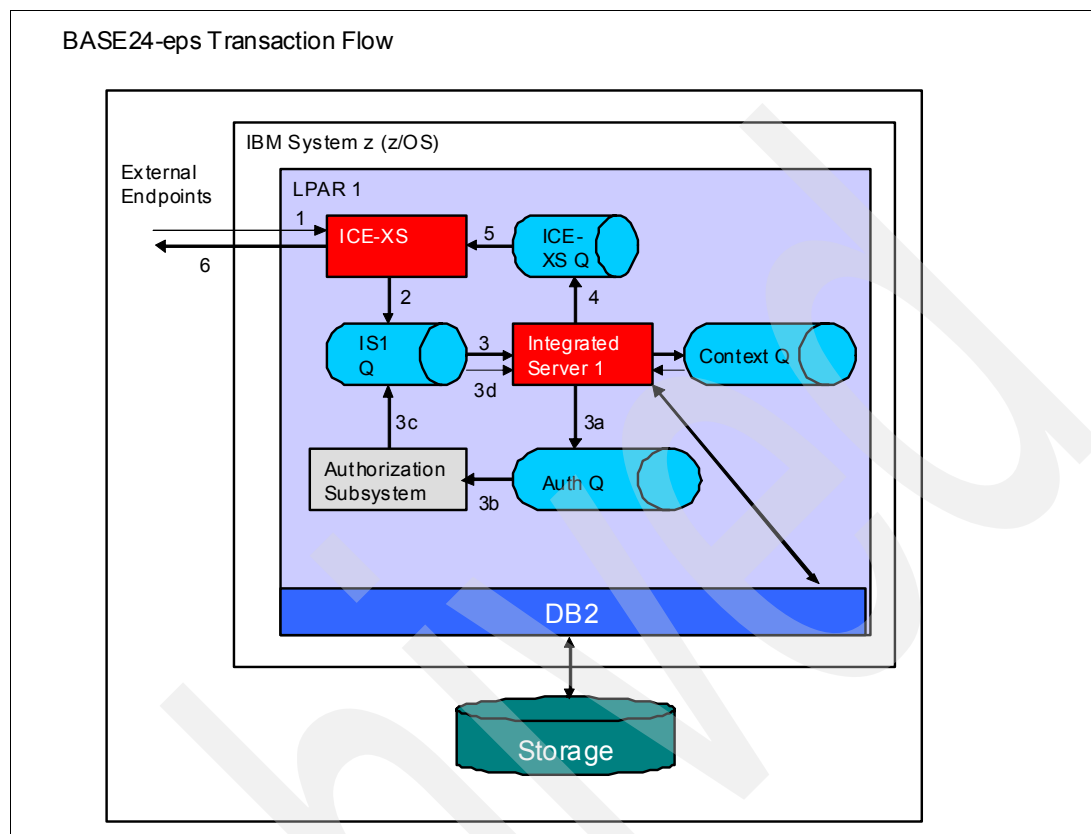


Figure 4-9 Transaction flow through the system

The following step explanations correspond with the numbers in Figure 4-9.

In step 1, an external endpoint sends a financial authorization request into the system.

In step 2, the ICE-XS communications handler receives the message and determines that it must be routed to the queue that is associated with the Integrated Server.

In step 3, the Integrated Server reads the message, translates it into a canonical internal format, and accesses the DB2 database to determine how it is to be processed. The IS might determine that the request is to be processed internally, in which case it proceeds directly to step 4. Alternatively, it might determine that the request is to be authorized externally, in this case by a CICS-based authorization subsystem.

In step 3a, the IS translates the request into the format that is supported by the authorization subsystem, saves the transaction context in the Context Queue, adds the header that is required by the IBM WebSphere MQ CICS Gateway, and places the request on the outbound MQ.

In step 3b, the CICS Gateway delivers the request to the authorization system.

In step 3c, the authorization processes the request and places the response on the Integrated Server input queue.

In step 3d, the Integrated Server reads the message from its input queue and determines that the message is a response. It retrieves context from the Context Queue to determine the originating endpoint.

In step 4, the Integrated Server translates the response into the format that is required by the originating endpoint and places the response on the ICE-XS queue that is associated with the originating endpoint.

In step 5, the ICE-XS process reads the response from its input queue and determines to which of the several remote endpoints that it manages, the response should be routed.

In step 6, the ICE-XS process routes the response to the originating endpoint.

## 4.3 BASE24-eps performance on System z

ACI Worldwide and IBM conducted a series of tests in May and June of 2008 at the IBM Poughkeepsie Benchmark Center to demonstrate the performance characteristics of BASE24-eps on the IBM System z platform. The objective was to demonstrate how BASE24-eps could exploit the IBM System z architecture to take advantage of the full range of its capabilities, such as load balancing and continuous availability, performance, and scalability.

### 4.3.1 Test results

The test results demonstrate that the combined solution of BASE24-eps and System z provide a highly scalable and efficient platform for payments processing.

#### Performance test

The latest version of BASE24-eps was tested on the IBM System z9® Enterprise Class server running under the z/OS V1R8 Operating System. Blended mixes of financial transactions were processed through both single and dual logical partition configurations. The benchmark specifically tested:

- ▶ IBM DB2 v8.1 for the database implementation with Data Sharing that is used to enable a single view of the database when dual LPARs were running
- ▶ IBM WebSphere MQ V6R0 for the messaging middleware implementation
- ▶ IBM InfoSphere™ Replication Server (previously known as WebSphere Replication Server) for data replication across dual data stores

Within BASE24-eps, the Integrated Servers (IS) were configured to run as multi-threaded processes (varying from 2 to 15 threads). The test environment simulated a typical card issuer with 3,200,000 cardholders. The transactions were acquired from an interchange simulator, pre-screened based upon cardholder data, and presented to a host application for authorization. The simulated test scenarios are consistent with previous benchmarks of BASE24-eps that have run on a variety of platforms.

#### Test objectives

The objective of the performance benchmark was to prove the aspects that are related to scalability, availability, and sizing:

- ▶ Achieve a consistent processor consumption rate that enables predictable sizing and linear scalability.

- ▶ Demonstrate efficient performance while handling peak payment transaction volumes.
- ▶ Demonstrate throughput and scalability when running various numbers of Integrated Server (IS) processes.
- ▶ Demonstrate availability by utilizing IBM InfoSphere Replication Server (Q Replication) with high transaction volumes.

### Performance test results

This joint ACI and IBM BASE24-eps performance benchmark achieved all of the targeted objectives:

- ▶ Performance and scalability were close to linear while the cost per transaction remained low and constant. The CPU cost per transaction (measured in MIPS<sup>1</sup>) remained virtually constant as volumes increased to a throughput of over 2,000 transactions per second.
- ▶ The test achieved more than 2,000 transactions per second on a System z9 EC with 14 CPUs. This volume represents twice the daily peak volume requirements of the world's largest transaction processors. MIPS per transaction remained nearly constant, so Total Cost of Ownership (TCO) per transaction becomes lower with higher transaction volumes (as support costs are fairly constant as volume grows).

Figure 4-10 on page 46 shows total MIPS cost for a blended mix of transactions (switch acquired, POS device acquired, and ATM acquired) at transaction rates up to two thousand transactions per second. As the graph shows, CPU utilization increased linearly as transaction rates increased.

<sup>1</sup> MIPS (Million Instructions Per Second) is a measure of CPU capacity commonly used on System z. While on other platforms it is common to speak of performance in terms of CPU percentage or CPU time - 100 Transactions Per Second (TPS) used 100% of one CPU, therefore each transaction used 1% or 10 milliseconds of CPU. MIPS provides a measure of CPU utilization that can be applied across different CPUs. If a CPU is rated at 100 MIPS and 100 TPS uses 100% of one CPU, the cost is said to be one MIPS/Tran. It can then easily be calculated that a CPU rated at 1000 MPS could support 1000 TPS; or, alternatively, that 100 TPS can be expected to use 10% of that CPU's cycles.

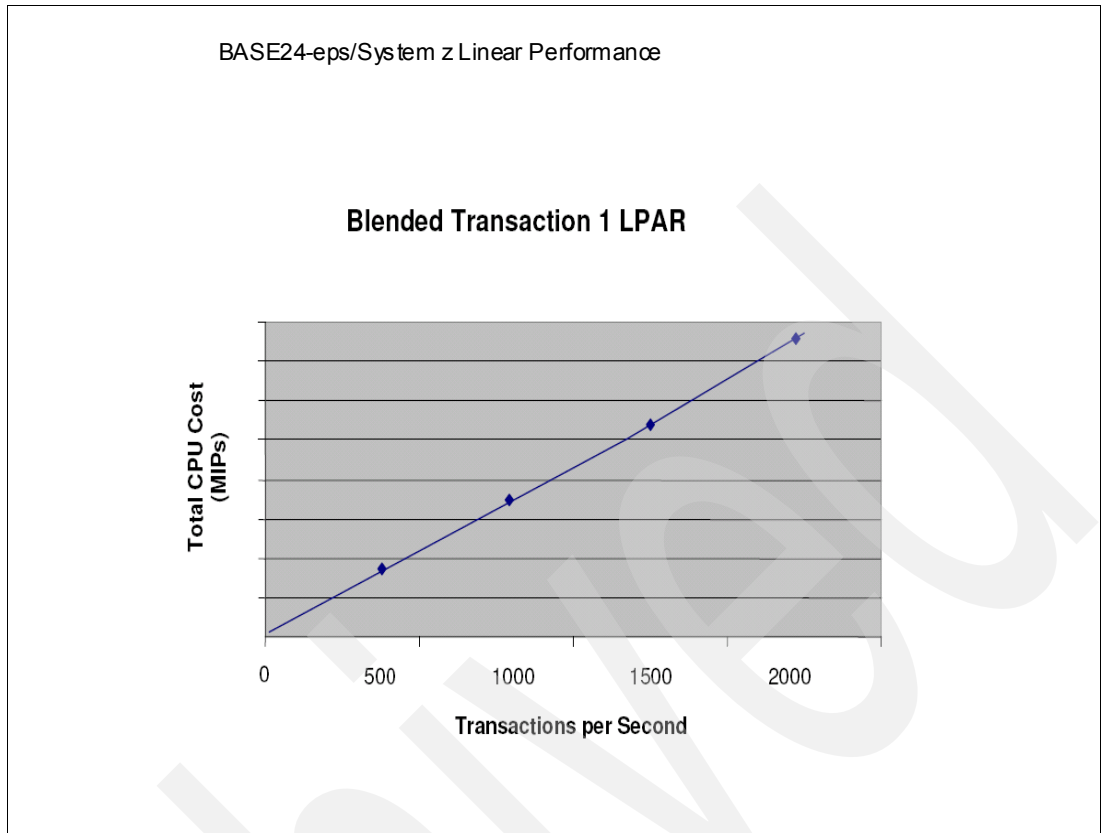


Figure 4-10 BASE24-eps System z Linear Performance

### Summary

The performance test of BASE24-eps exceeded both companies' objectives and expectations. The results demonstrate that the combination of BASE24-eps and System z is a very competitive and attractive end-to-end retail payments solution for the finance sector.



## Designing the system layout

In this chapter, we discuss the system design layout for deploying BASE24-eps on System z.

The topics that we discuss are:

- ▶ 5.1, “BASE24-eps high availability on System z” on page 48
- ▶ 5.2, “Securing BASE24-eps on System z” on page 62

## 5.1 BASE24-eps high availability on System z

BASE24-eps is designed for high availability and data integrity at a cost that is reasonable to the type of consumer-oriented transactions being processed. It is not designed to prevent any and all possibilities of lost transactions. Even with reliable communications protocols, messages can be lost in the network and financial end-to-end protocols are developed to account for and compensate for the possibility of lost messages. Because message loss is always possible, BASE24-eps acknowledges this fact and relies on the financial end-to-end protocols to compensate for lost messages. The goals of BASE24-eps are:

- ▶ **Architected high availability:** No single BASE24-eps component failure will bring Online Transaction Processing (OLTP) to a halt.
- ▶ **Data integrity:** The database is always in a consistent state, either all or none of the database updates that are associated with a message are applied to the database.
- ▶ **Financial integrity:** BASE24-eps architecture and financial end-to-end protocols can virtually eliminate any possibility of a persistent out-of-balance condition with any remote endpoint.
- ▶ **Low cost:** BASE24-eps is extensively tuned to minimize the overall cost and latency that is involved in processing. The System z implementation makes extensive use of high-performance WebSphere MQ Shared Queues and direct z/OS facilities, where appropriate, to meet this goal.

### High availability

Availability for a transaction processing system, such as BASE24-eps, is typically measured by the number of nines that a system can achieve. Referring to Table 5-1, the first row would be considered two nines and the last row is considered seven 9s of availability.

With each nine comes a higher degree of cost and complexity. Customers must weigh the cost of downtime against the cost of avoiding downtime to determine the high-availability model that best fits their business requirements.

Table 5-1 illustrates the relationship between the nine and the actual business service downtime. The table reflects mean downtime per year and not actual anticipated downtime in any given year.

*Table 5-1 Mean downtime per year*

Availability		Downtime per year
99%	two nines	Less than 88 hours
99.9%	three nines	Less than 9 hours
99.99%	four nines	Less than 53 minutes
99.999%	five nines	Less than 5.3 minutes
99.9999%	six nines	Less than 32 seconds
99.99999%	seven nines	Less than 3.2 seconds

When designing a system for high availability, you must consider both planned and unplanned events that can make the system unusable:

- ▶ Planned events include moving to new versions of the application software or operating system, introducing new LPARs or CICS regions into the system, maintaining the hardware or introducing new hardware components, and so on.
- ▶ Unplanned events include application, operating system, or hardware failures and the like. In a properly configured and maintained system, both planned and unplanned events can be managed to meet the customer's availability requirements.

## Disaster Recovery

A single site can be designed to provide very high availability, but it cannot withstand a catastrophic event that takes the site completely offline.

As previously described, the number of *nines* that a solution provides can be a useful metric for the degree of availability that a solution provides. Similarly, the concepts of Recovery Time Objective (RTO) and Recovery Point Objective (RPO) are useful in evaluating Disaster Recovery (DR) solutions:

- ▶ RTO represents the time necessary in the event of a primary system disaster to fail processing over to the backup system.
- ▶ RPO represents the age delta between the last consistent set of data at the secondary site as compared with the primary site's data at the moment disaster struck and reflects the amount of that is data lost in the event of a primary system failure.

Because there are many possible ways to deploy BASE24-eps at a single site to achieve varying numbers of nines, there are many ways to deploy BASE24-eps at multiple sites to achieve varying RPOs and RTOs.

The next sections show various System z deployment options to achieve varying degrees of availability and resilience in the event of disaster.

**Note:** The design examples in the next sections do not address the network topology. A high-availability system design must also provide redundancy within the network infrastructure.

### 5.1.1 Single-LPAR

BASE24-eps can be deployed in a single z/OS Logical Partition (LPAR). While this does permit BASE24-eps to benefit from the highly available System z hardware and Operating System (OS), single-LPAR deployment is still vulnerable to single points of failure including middleware and hardware failures, and is generally not recommended.

Figure 5-1 on page 50 shows a possible single-LPAR deployment.

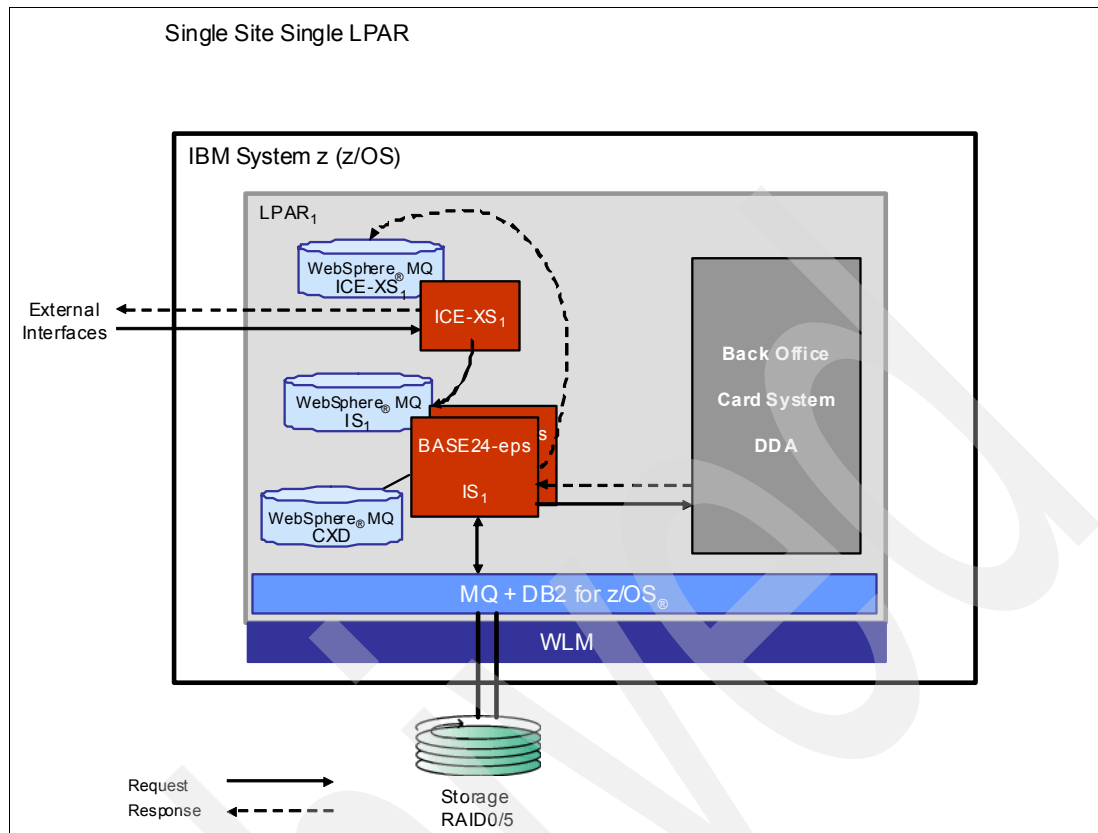


Figure 5-1 Single LPAR

### **Single site single LPAR availability characteristics**

The single site single LPAR availability characteristics are:

- ▶ Six nines availability for System z hardware only: unplanned
- ▶ Planned outages limited to two per year

### **Single site single LPAR functionality**

The single site single LPAR functionality is:

- ▶ Base configuration: single LPAR in a single site with a host application that is local to the machine:
  - Outage required for some hardware and system software maintenance.
  - Single Disk Storage Subsystem is single point-of-failure.
- ▶ Communication with external interfaces (devices, networks) is managed with an ACI component (ICE-XS) that is coupled with IBM WebSphere MQ:
  - In the event of a System z NIC failure, TCP/IP connections are transferred without breakage to another NIC, which is based on OSA QDIO Gratuitous ARP Fail-over where connections are not dropped.
- ▶ Messages are placed on a common inbound message queue to be processed by multiple instances of the BASE24-eps Integrated Server (IS). A failure of a single instance of the IS is masked as remaining instances carry the load while the failed instance is restarted.



- ▶ The context is stored in an IBM WebSphere MQ queue.
- ▶ Communication to host application can be synchronous (EXCI or IBM WebSphere MQ CICS Gateway) or asynchronous (TCP/IP, MQ, or other mechanism).
- ▶ Replies are deposited in an outbound queue and delivered by ICE-XS to the originator.

### 5.1.2 High availability: Dual LPAR

A dual LPAR deployment is generally preferable from the standpoint of scalability and availability. Figure 5-2 shows a deployment in two LPARs in a single System z server.

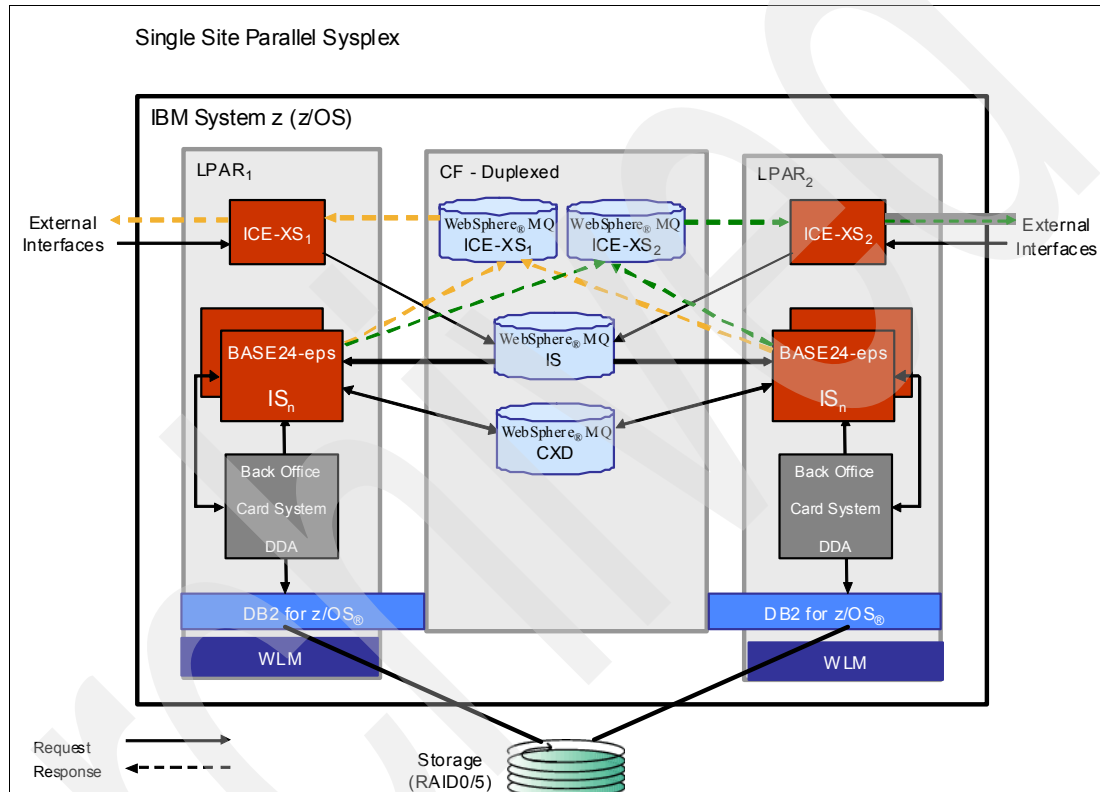


Figure 5-2 Dual LPAR

#### Single site Parallel Sysplex availability characteristics

The single site Parallel Sysplex availability characteristics are:

- ▶ Six nines availability for System z hardware and z/OS
- ▶ No planned outages necessary with Parallel Sysplex, except for some hardware upgrades
- ▶ Multiple application images protect from single application point-of-failure
- ▶ Coupling facilities are duplexed for additional availability

#### Single site Parallel Sysplex functionality

The single site Parallel Sysplex functionality is:

- ▶ Base configuration: Two LPARs at a single site on a single server with a host application that is local to the machine:
  - Outage required for some hardware maintenance.
  - Single Disk Storage Subsystem is single point-of-failure.

- ▶ Messages are placed on a common inbound IBM WebSphere MQ Shared Queue to be processed by multiple instances of the BASE24-eps Integrated Server (IS):
  - Failure of a single instance of the IS is masked as remaining instances carry the load while the failed instance is restarted.
  - Failure of a single LPAR is masked as remaining LPAR carries the load while the failed LPAR is restarted. The remaining LPAR can handle the full load because it can exploit the entire system's CPU capacity.
  - Work is balanced by the application's "pull model" into both LPARs under the control of System z's hypervisor: Processor Resource/Systems Manager (PR/SM).
- ▶ ICE-XS and Integrated Server (IS) data shared across systems with locks in the coupling facilities (CFs) maintained by DB2 Data Sharing and shared zFS.
- ▶ Communication with external interfaces (devices, networks) is managed with an ACI component (ICE-XS) that is coupled with IBM WebSphere MQ:
  - In the event of a System z NIC failure, TCP/IP connections are transferred without breakage to another NIC, which is based on OSA QDIO Gratuitous ARP Fail-over where connections are not dropped.
  - In the event of an LPAR failure, TCP/IP connections are broken and reestablished by System z Virtual IP Addressing (VIPA) on an available LPAR.
- ▶ The context is stored in the coupling facility as an IBM WebSphere MQ Shared Queue.
- ▶ Communication to host application can be synchronous (EXCI or IBM WebSphere MQ CICS Gateway) or asynchronous (TCP/IP, MQ, or other mechanism).
- ▶ Replies are deposited in an outbound IBM WebSphere MQ Shared Queue and delivered by ICE-XS to the originator.

### 5.1.3 High availability: Dual LPAR with Hyperswap

The Dual LPAR with Hyperswap deployment provides a very high degree of availability, probably approaching five 9s. However, even with a RAID array, the failure of an entire disk subsystem remains a single point-of-failure. Adding the IBM Hyperswap to the mix eliminates that point-of-failure, as shown in Figure 5-3 on page 53.

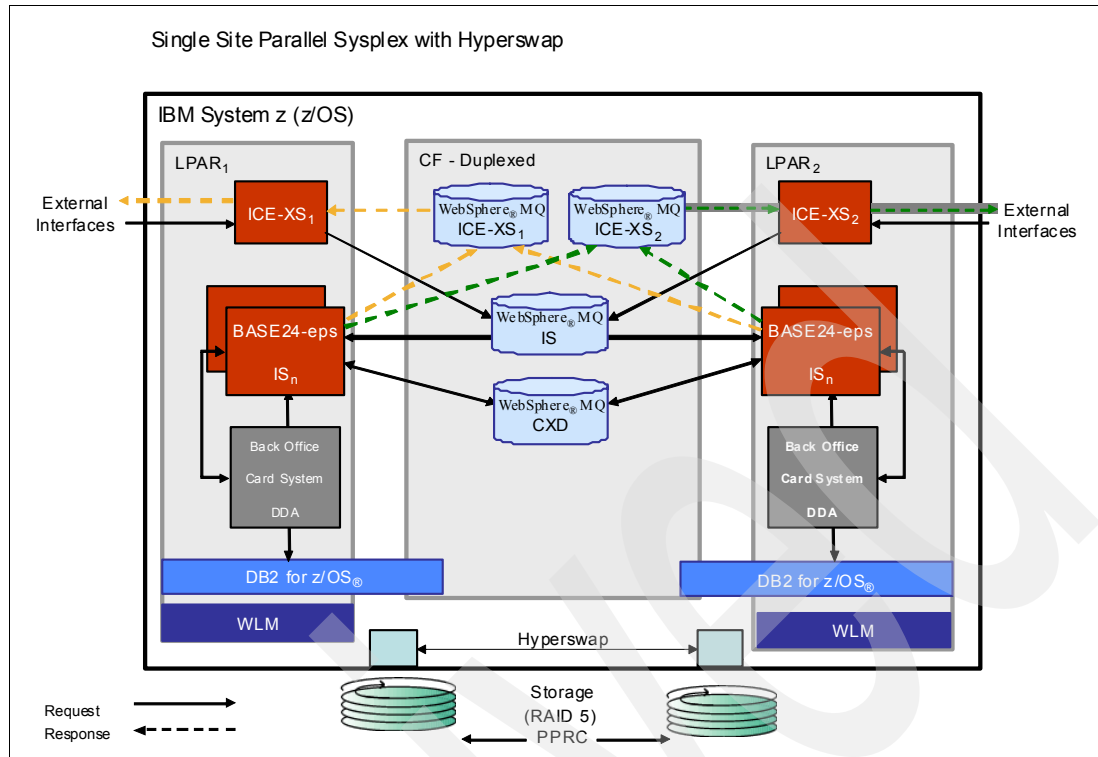


Figure 5-3 Single site Parallel Sysplex with Hyperswap

### Single site Parallel Sysplex with Hyperswap availability characteristics

The single site Parallel Sysplex with Hyperswap availability characteristics are:

- ▶ Six 9s availability for System z hardware and z/OS
- ▶ No planned outages necessary with Parallel Sysplex, except for some hardware upgrades
- ▶ Multiple application images protect from single application point of failure
- ▶ Coupling facilities are duplexed for additional availability
- ▶ Non-disruptive replacement/failure of SAN units

### Single Site Parallel Sysplex with Hyperswap functionality

The single site Parallel Sysplex with Hyperswap functionality is:

- ▶ Base configuration: Two LPARs at a single site on a single server with a host application that is local to the machine. Peer-to-Peer Remote Copy (PPRC) provides data copy to a separate storage subsystem. Hyperswap enables seamless switchover to secondary disks in the event of a primary storage subsystem failure:
  - Outage required for some hardware maintenance.
- ▶ Messages are placed on a common inbound IBM WebSphere MQ Shared Queue to be processed by multiple instances of the BASE24-eps Integrated Server (IS):
  - Failure of a single instance of the IS is masked as remaining instances carry the load while the failed instance is restarted.
  - Failure of a single LPAR is masked as remaining LPAR carries the load while the failed LPAR is restarted. The remaining LPAR can handle the full load because it can exploit the entire system's CPU capacity.
  - Work is balanced by the application's "pull model" into both LPARs under the control of System z's hypervisor: Processor Resource/Systems Manager (PR/SM).

- ▶ ICE-XS and Integrated Server (IS) data that is shared across systems with locks in the coupling facilities (CFs) maintained by DB2 Data Sharing and shared zFS.
- ▶ Communication with external interfaces (devices, networks) is managed with an ACI component (ICE-XS) that is coupled with IBM WebSphere MQ:
  - In the event of a System z NIC failure, TCP/IP connections are transferred without breakage to another NIC, which is based on OSA QDIO Gratuitous ARP Fail-over where connections are not dropped.
  - In the event of an LPAR failure TCP/IP connections are broken and reestablished by System z Virtual IP Addressing (VIPA) on an available LPAR.
- ▶ The context is stored in the coupling facility as an IBM WebSphere MQ Shared Queue.
- ▶ Communication to host application can be synchronous (EXCI or IBM WebSphere MQ CICS Gateway) or asynchronous (TCP/IP, MQ, or other mechanism).
- ▶ Replies are deposited in an outbound IBM WebSphere MQ Shared Queue and delivered by ICE-XS to the originator.

#### 5.1.4 High availability: Single site dual server with Hyperswap

Finally, to eliminate the remaining low odds of single point of failure (hardware maintenance), the previous configuration can be deployed on multiple System z servers, as shown in Figure 5-4. In theory, this deployment should totally eliminate all planned and unplanned outages.

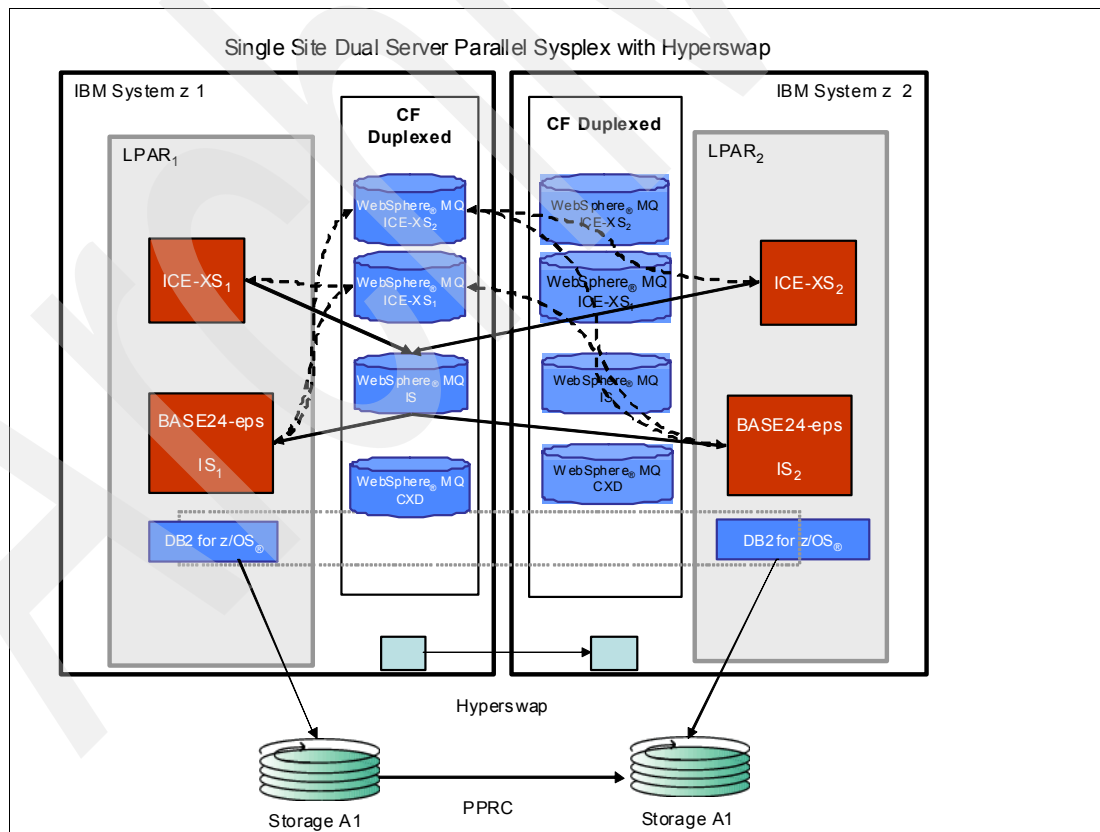


Figure 5-4 Single site dual server Parallel Sysplex with Hyperswap

### ***Single site dual server Parallel Sysplex with PPRC Hyperswap availability characteristics***

The single site dual server Parallel Sysplex with PPRC Hyperswap availability characteristics are:

- ▶ Six 9s availability for System z hardware and OS
- ▶ No planned outages required
- ▶ Multiple application images protect from single application point of failure
- ▶ Coupling facilities are duplexed for additional availability
- ▶ Non-disruptive replacement of all hardware components.

### ***Single site dual server Parallel Sysplex with PPRC Hyperswap functionality***

The single site dual server Parallel Sysplex with PPRC Hyperswap is:

- ▶ Base configuration: Two LPARs on two System z servers. PPRC provides data copy to a separate storage subsystem. Hyperswap enables seamless switchover to secondary disks in the event of a primary storage subsystem failure:
  - No planned or unplanned outages required.
- ▶ Messages are placed on a common inbound IBM WebSphere MQ Shared Queue to be processed by multiple instances of the BASE24-eps Integrated Server (IS):
  - Failure of a single instance of the IS is masked as remaining instances carry the load while the failed instance is restarted.
  - Failure of a single LPAR is masked as the remaining LPAR carries the load while the failed LPAR is restarted. The remaining LPAR can handle the full load because it can exploit the entire system's CPU capacity.
  - Work is balanced by the application's "pull model" into both LPARs under the control of System z's hypervisor: Processor Resource/Systems Manager (PR/SM).
- ▶ ICE-XS and integrated Server (IS) data is shared across systems with locks in the coupling facilities (CFs) maintained by DB2 Data Sharing and shared zFS.
- ▶ Communication with external interfaces (devices, networks) is managed with an ACI component (ICE-XS) that is coupled with IBM WebSphere MQ:
  - In the event of a z NIC failure, TCP/IP connections are transferred without breakage to another NIC, which is based on OSA QDIO Gratuitous ARP Fail-over where connections are not dropped.
  - In the event of an LPAR failure, TCP/IP connections are broken and reestablished by System z Virtual IP Addressing (VIPA) on an available LPAR.
- ▶ The context is stored in the coupling facility as an IBM WebSphere MQ Shared Queue.
- ▶ Communication to host application can be synchronous (EXCI or IBM WebSphere MQ CICS Gateway) or asynchronous (TCP/IP, MQ, or other mechanism).
- ▶ Replies are deposited in an outbound IBM WebSphere MQ Shared Queue and delivered by ICE-XS to the originator.

## **5.1.5 Disaster Recovery: Metro Mirror**

Although the single site dual server Parallel Sysplex with PPRC Hyperswap deployment that we described in 5.1.4, "High availability: Single site dual server with Hyperswap" on page 54 provides virtually continuous availability through planned and unplanned outages, it is still vulnerable to catastrophes, such as fire, flood, earthquake, terrorism, and even sustained power outages.

Figure 5-5 shows a typical configuration for a Disaster Recovery deployment at metro distances.

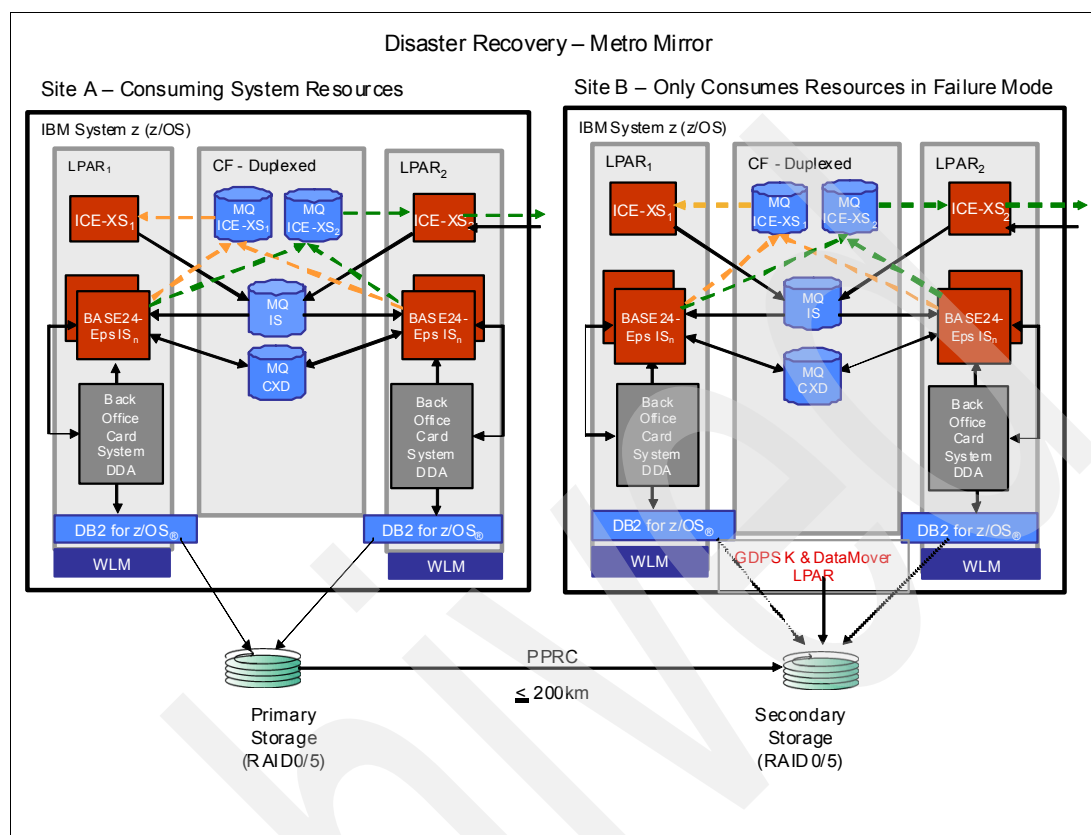


Figure 5-5 Disaster Recovery Metro Mirror

### Disaster Recover: Metro Mirror availability characteristics

The Metro Mirror availability characteristics are:

- ▶ Each site can be one of the deployments that we described in 5.1.1, “Single-LPAR” on page 49 through 5.1.4, “High availability: Single site dual server with Hyperswap” on page 54 and has the associated availability characteristics.
- ▶ Commonly, Site B might have one of the less available (and less expensive) deployments.
- ▶ Site B is inactive and not consuming resources.
- ▶ If PPRC is configured for synchronous operation the Recovery Point Objective might approach 0 data loss.
- ▶ Because Site B is inactive, the Recovery Time Objective is determined by the amount of time that is required to bring Site B active.
- ▶ Higher initial cost if GDPS and PPRC are not already in place; lower MIPS cost per transaction.

### Disaster Recover: Metro Mirror functionality

The Metro Mirror functionality is:

- ▶ PPRC provides remote data copy at the disk subsystem layer.
- ▶ Distance between sites is limited to 200 km.

- ▶ Controlling system in Site B.
- ▶ Site A failure invokes an operator prompt to take appropriate recovery action (select a recovery script).

### 5.1.6 Disaster Recovery: Global Mirror

When regulatory or other requirements demand that the primary and backup systems be greater than 200km apart, GDPS provides an Active/Standby option, as shown in Figure 5-6.

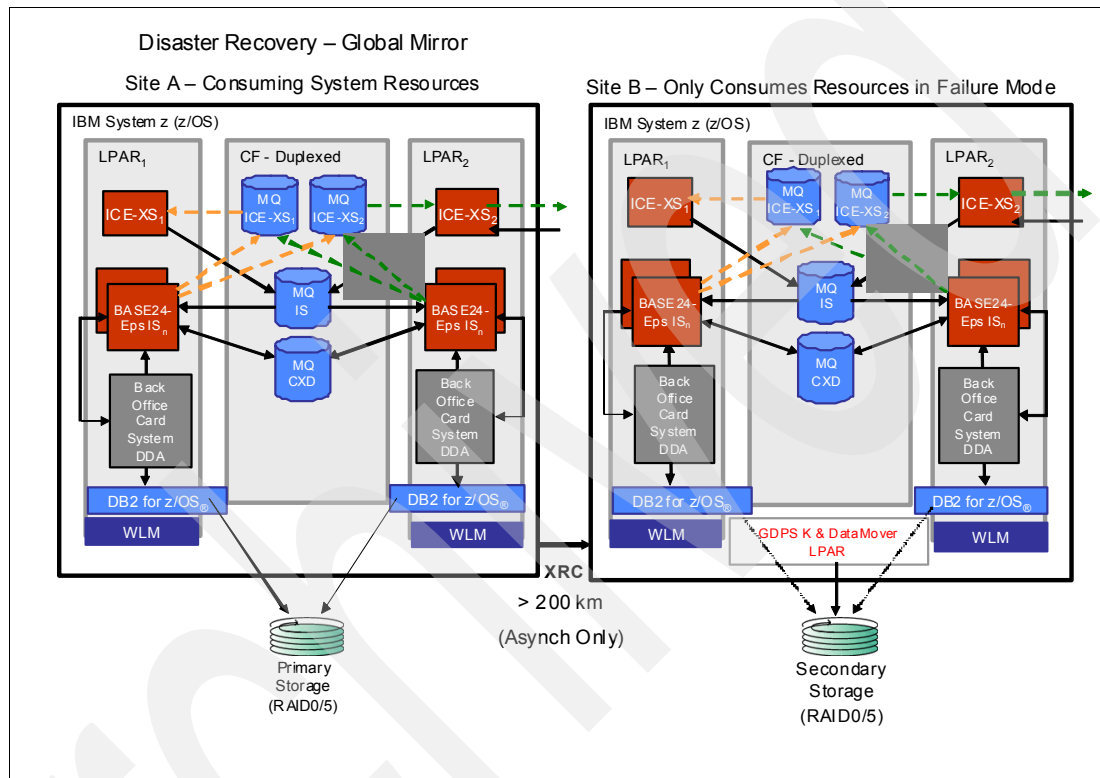


Figure 5-6 Disaster Recovery: Global Mirror

#### Disaster Recovery: Global Mirror availability characteristics

The Global Mirror availability characteristics are:

- ▶ Each site can be one of the deployments that we described in 5.1.1, “Single-LPAR” on page 49 through 5.1.4, “High availability: Single site dual server with Hyperswap” on page 54, and has the associated availability characteristics:
  - Commonly Site B might have one of the less available (and less expensive) deployments.
- ▶ Site B is inactive and not consuming resources.
- ▶ Extended Remote Copy (XRC) is always asynchronous so some data loss is likely.
- ▶ Because Site B is inactive, the Recovery Time Objective is determined by the amount of time that is required to bring Site B active.
- ▶ Higher initial cost if GDPS and XRC are not already in place; lower MIPS cost per transaction.

### **Disaster Recovery: Global Mirror functionality**

The Global Mirror functionality is:

- ▶ XRC provides remote data copy at the z/OS layer.
- ▶ Unlimited Distance between sites.
- ▶ Controlling system in Site B.
- ▶ Site A failure invokes an operator prompt to take appropriate recovery action (select a recovery script).

### **5.1.7 Disaster Recovery: DB2 queue replication active/standby**

An alternative to PPRC and XRC for an active/standby DR system is provided by InfoSphere Replication Server (formerly known as WebSphere Replication Server). If a GDPS is not already in place, Queue Replication requires a lower initial outlay.

Because Site B is already active, RTO is lower than with PPRC and XRC, but the cost in MIPS/Tran might be higher. Figure 5-7 shows active/standby queue replication.

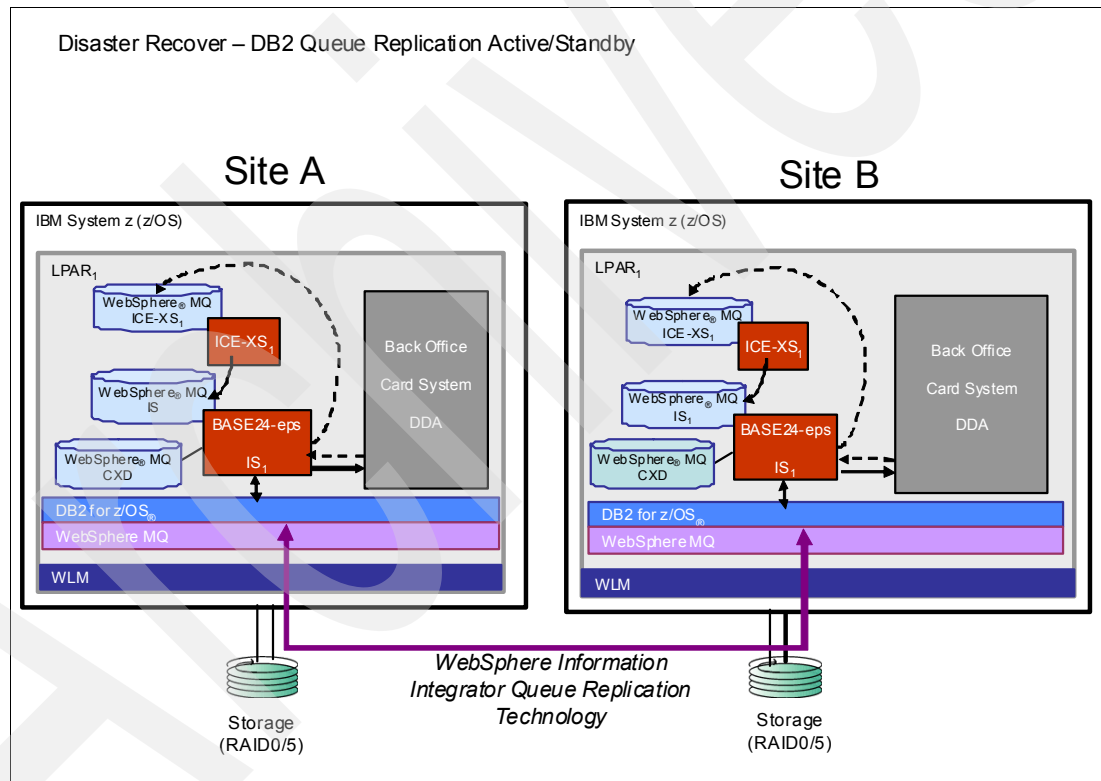


Figure 5-7 Disaster Recovery: DB2 queue replication active/standby

### **DB2 queue replication active/standby availability characteristics**

The DB2 queue replication active/standby availability characteristics are:

- ▶ Each site can be one of the deployments that we described in 5.1.1, “Single-LPAR” on page 49 through 5.1.4, “High availability: Single site dual server with Hyperswap” on page 54, and has the associated availability characteristics:
  - Commonly, Site B might have one of the less available (and less expensive) deployments.



- ▶ Site B is active and consuming resources. The DB2 subsystem must be active and all other subsystems except the BASE24-eps communications handlers can be active.
- ▶ Queue Replication is always asynchronous; therefore, some data loss is possible (RPO is non-zero).
- ▶ Queue replication maintains transactional boundaries so that data integrity is maintained.
- ▶ Because Site B is fairly active, the Recovery Time Objective is measured in minutes.
- ▶ Lower initial cost if GDPS and XRC are not already in place; higher MIPS cost per transaction.

#### ***DB2 queue replication active/standby functionality***

The DB2 queue replication active/standby functionality is:

- ▶ Queue Replication provides remote data copy at the DB2 layer
- ▶ Unlimited Distance between sites
- ▶ Operator alerts or Tivoli automation to start BASE24-eps communications handlers on Site B

### **5.1.8 Disaster Recovery: Stretch Parallel Sysplex**

The lowest RTO and RPO at metro distances (less than or equal to 10-20 km) can arguably be achieved by using a 'stretch' Parallel Sysplex to position the two servers that we described in 5.1.4, "High availability: Single site dual server with Hyperswap" on page 54, up to 10-20km apart. At metro distances, this deployment provides a Dual Active environment with both servers actively processing transactions, providing near-zero recovery time and with near-zero data loss in the event that one site fails. Figure 5-8 on page 60 shows this configuration.

Use care in this case to account for the increased latency that a distributed deployment can add to the transaction path. The latency of database and coupling facility operations can be increased and it might be necessary to configure additional threads of execution to compensate for the increased latency. Latency accessing serialized resources might require partitioning those resources or can in the worst case limit throughput. ACI and IBM stand ready to help with this type of issue.

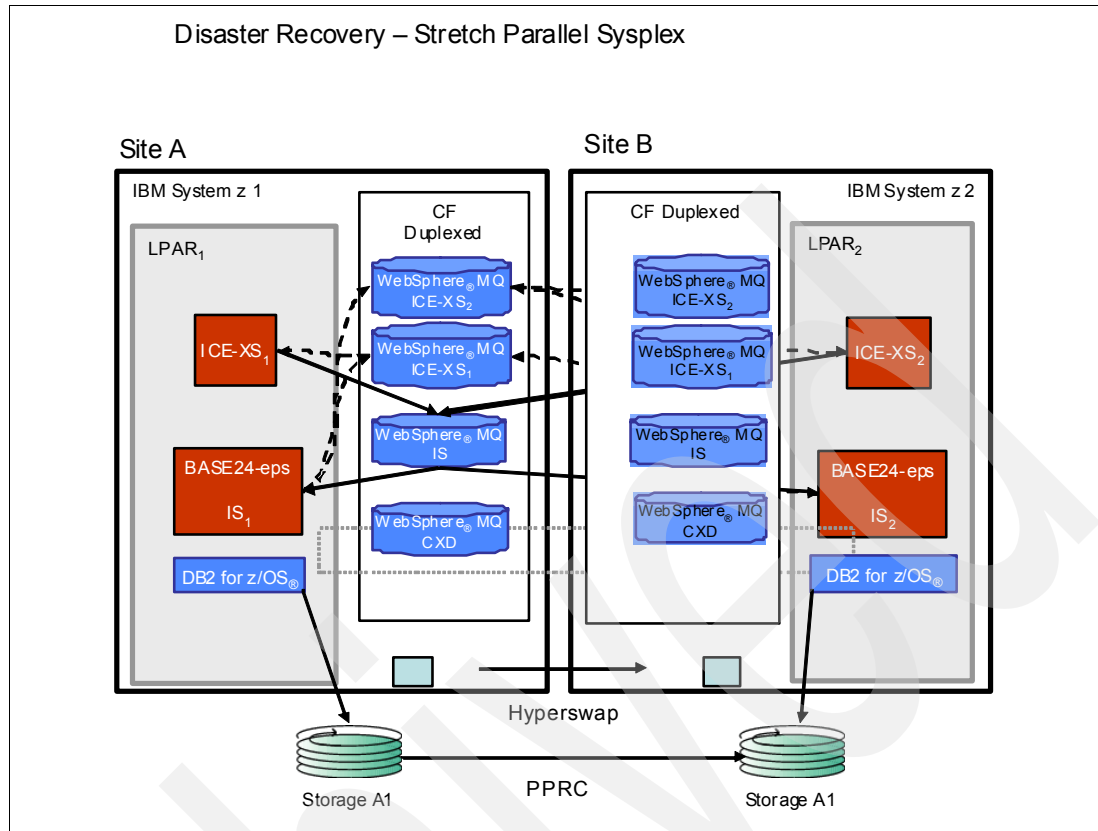


Figure 5-8 Disaster Recovery: stretch Parallel Sysplex

### Disaster Recovery: Stretch Parallel Sysplex availability characteristics

The stretch Parallel Sysplex availability characteristics are:

- ▶ The availability characteristics are essentially those that we described in 5.1.4, “High availability: Single site dual server with Hyperswap” on page 54 with the added safety provided by geographic distribution.
- ▶ Both sites are actively processing transactions and are proven ready.
- ▶ RTO and RPO should be near zero.
- ▶ Higher initial cost than queue replication options if PPRC is not already in place; lower MIPS cost per transaction.

### Disaster Recovery: Stretch Parallel Sysplex functionality

The stretch Parallel Sysplex functionality is:

- ▶ ‘Stretch’ parallel Sysplex provides data sharing/replication
- ▶ Metro distance between sites ( $\leq 10\text{-}20$  km)

## 5.1.9 Disaster Recovery: DB2 Queue Replication Dual Active

At distances greater than 10-20km the best RTO and RPO are provided by InfoSphere Replication Server (formerly known as WebSphere Replication Server) in conjunction with the ACI BASE24-eps conflict resolution subsystem. Figure 5-9 on page 61 shows a base Dual Active system.

## Disaster Recovery – DB2 Queue Replication Dual Active

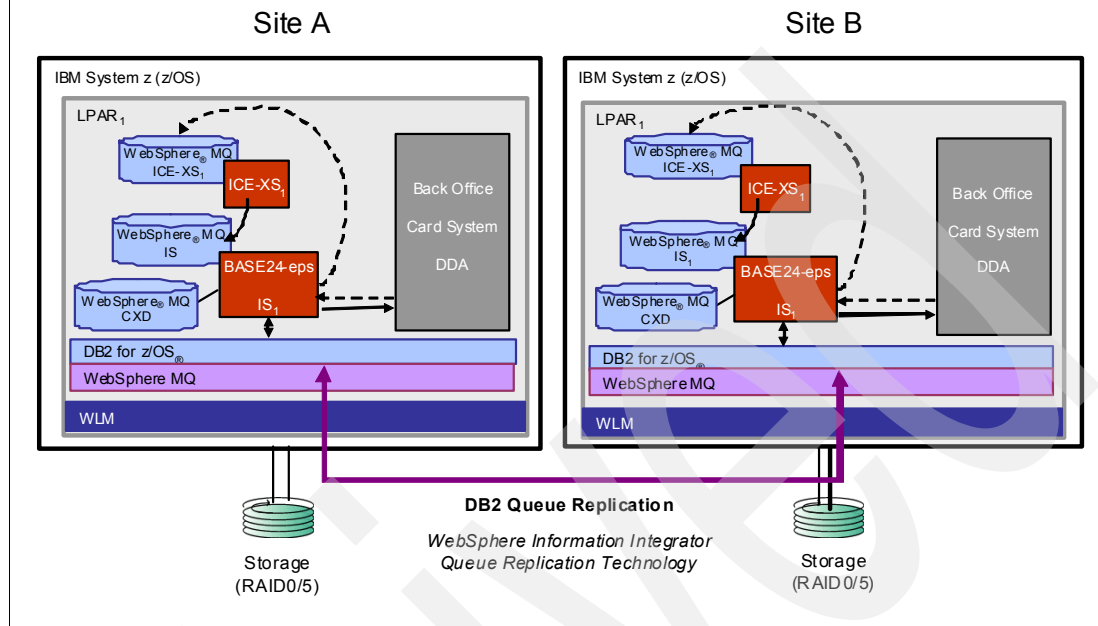


Figure 5-9 Disaster Recovery - DB2 Queue Replication Dual Active

### **DB2 Queue Replication Dual Active availability characteristics**

The DB2 Queue Replication Dual Active availability characteristics are:

- ▶ Each site can be one of the deployments described in 5.1.1, “Single-LPAR” on page 49 through 5.1.4, “High availability: Single site dual server with Hyperswap” on page 54 and has the associated availability characteristics:
  - Typically both sites would have the same availability characteristics.
- ▶ Both sites are actively processing transactions so that RTO can be near 0.
- ▶ Queue Replication is always asynchronous; therefore, some data loss is likely.
- ▶ Relatively low initial cost; highest MIPS cost per transaction.

### **DB2 Queue Replication Dual Active functionality**

The DB2 Queue Replication Dual Active functionality is:

- ▶ Queue Replication provides remote data copy at the DB2 layer.
- ▶ Unlimited distance between sites.
- ▶ Communications handlers are available on both sites, so it is only necessary to reestablish connections.
- ▶ Reestablishment of inbound connections can be managed by remote endpoints that connect to the IP address of an alternate site or by means of a highly-available network router.

### 5.1.10 Dual active considerations

In any asynchronous dual active system there is the possibility of a *collision*, which results when a database record is added or updated on one system. Then, while that operation is being replicated to the second system, that same record is added or updated on the second system.

When the replicated operations are applied on the respective systems, there is a conflict. Blindly applying the replicated operation on the first site results in overwriting an update on the first site, which would be lost. Blindly applying the replicated operation on the second site results in overwriting an operation, which would be lost.

Replication systems typically provide relatively limited options to deal with this situation. Configuration options might allow you to specify that the newest operation be applied on both systems. Alternatively one system might be configured as the 'winner' and have its data reflected on the target system while the other system is configured as the loser and has its data overwritten by the winner, or perhaps the system can be configured so that in the event of a collision both systems retain their own data, the collision is logged, and until it is resolved manually the two systems remain out of synch with each other.

Collisions in a dual-active Online Transaction Processing (OLTP) system are relatively common, and none of the above options are acceptable. Columns in an OLTP database might need to reflect more complex changes, such as columns where what is important is not the final value but an application-defined change in the value. BASE24-eps in conjunction with InfoSphere Replication Server provides near-real-time programmatic collision resolution.

## 5.2 Securing BASE24-eps on System z

While security considerations for a test system such as ours might not be stringent, a production system should be secured in accordance with the *ACI System Object Security Standard BASE24-eps 8.2 Requirements*.

## Installing BASE24-eps on z/OS

In this chapter, we describe the target installation environment and the steps that are necessary to install, configure, and start the BASE24-eps system.

The topics that we discuss are:

- ▶ 6.2, “Our z/OS environment” on page 64
- ▶ 6.3, “BASE24-eps configuration” on page 66
- ▶ 6.4, “Pre-install checklist and worksheet” on page 67
- ▶ 6.5, “ES\_Install” on page 68
- ▶ 6.6, “Installing ICE-XS” on page 72
- ▶ 6.7, “essetup” on page 75
- ▶ 6.8, “Initial database load” on page 85
- ▶ 6.9, “BASE24-eps directory structure” on page 85
- ▶ 6.10, “Running the esinfo” on page 86
- ▶ 6.11, “Installation verification” on page 87
- ▶ 6.12, “Installing the user interface” on page 91
- ▶ 6.13, “Operational considerations” on page 95
- ▶ 6.14, “BASE24-eps operations” on page 98

## 6.1 Target ITSO z/OS environment

In this chapter, we detail our installation of the BASE24-eps software. We used the documents provided as part of the software distribution during the installation and configuration process.

The authoritative reference for installation are the BASE24-eps documents:

- ▶ ACI BASE24-eps (v08.2) Hardware and Software Requirements

We used the *BASE24-eps (v08.2) Hardware and Software Requirements* document to verify that we had all of the software and hardware requirements in place before beginning the installation work.

- ▶ ACI BASE24-eps (v08.2) z/OS Install Guide

We used the *BASE24-eps (v08.2) z/OS Install Guide* as a reference guide to the actual installation process.

- ▶ ACI BASE24-eps z/OS Pre-Install Checklist and Worksheet

The *BASE24-eps z/OS Pre-Install Checklist and Worksheet* was essential in planning and design stage prior to the actual installation work.

ACI Worldwide electronically distributes the BASE24-eps software. Support and instructions on acquiring the distribution electronically is provided by ACI Worldwide. If you are interested in obtaining access to ACI documentation, contact an ACI representative. Contact information is at the ACI Web site:

<http://www.aciworldwide.com>

## 6.2 Our z/OS environment

The target z/OS environment consists of three z/OS systems SC48, SC52, and SC67, which are part of a multi-system Parallel Sysplex.

Systems SC48 and SC67 have an instance of a DB2 Datasharing group allowing access to the DB2 database from either system. They also have a MQ Queue Sharing group, which similarly allows access to shared queues from either system. Both of these facilities are provided by DB2 and MQ using Parallel Sysplex technology (and require use of coupling facility hardware). We also make use of System z Crypto Express2 hardware for BASE24-eps cryptographic processing.

System SC52 provides the Java Server component that provides access to the BASE24-eps application for the desktop client (UI).

Figure 6-1 on page 65 is a graphical depiction of the environment.

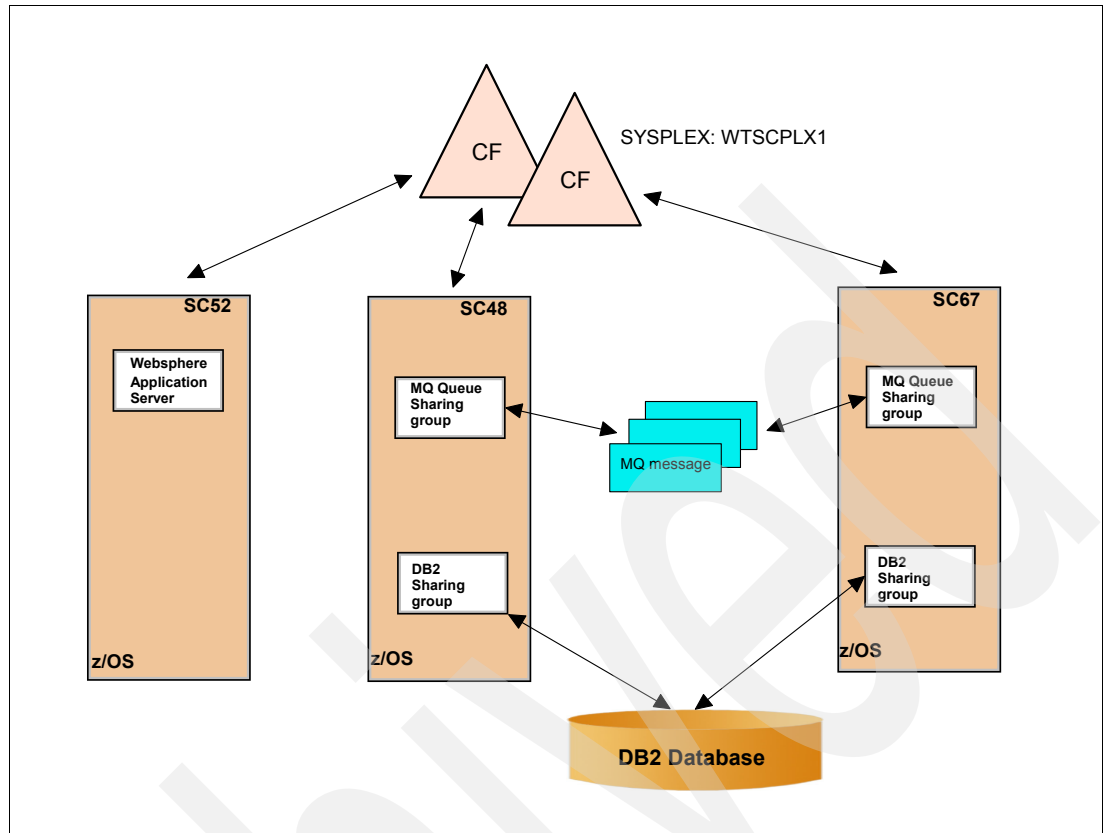


Figure 6-1 z/OS Configuration

The software environment is:

- ▶ z/OS 1.9
- ▶ DB2 Version 9.1
- ▶ IBM WebSphere MQ Version 6.0
- ▶ WebSphere Application Server Version 6.1

Other components, such as Communications Server, Language Environment® and DFSMS, are standard parts of the z/OS Operating System.

On our systems, the BASE24-eps software was installed into a shared zFS file system that was mounted at:

`/usr/aci`

It is assumed that the MQ and DB2 subsystems are configured with suitable run time options that are set to allow successful execution of the BASE24-eps application, which makes extensive use of DB2 and MQ services, which includes options, such as DB2 EDM pool size, buffer pools, and number of allowed MQ connections.

**Unable to instantiate memory table error:** The BASE24-eps emtbuild process might fail with the error message unable to instantiate memory table. This message typically appears because the MAXMMAPAREA OMVS parameter is set too low. MAXMMAPAREA is the maximum amount of data space storage in Pages that can be allocated for memory mapping of z/OS unix files. Each page is 4 KB in size, for example, a 1 MB size memory mapped file requires a minimum of 256 pages in MAXMMAPAREA. MAXMMAPAREA is calculated as (size of load modules/shared objects in pages + (total size of BASE24-eps OLTPs \* 2) + adjustment factor for growth). We used 40960.

**BASE24-eps processes:** BASE24-eps processes stay up indefinitely. If the OMVS parameter MAXCPU TIME is set, BASE24-eps processes might be cancelled prematurely. MAXCPU TIME should be unlimited.

As an example we initially had problems with the default setting of IDBACK, which is common to both MQ and DB2 and is the maximum number of background connections that are allowed into the subsystem (that is connected to MQ and DB2). For DB2, this value is set in the DSNZPARM module, and for MQ the value is set in the CSQZPARM module.

Other software requirements are:

- ▶ C/C++ Compiler
- ▶ ICSF (Cryptographic Services)
- ▶ IBM Java Virtual Machine
- ▶ gzip and tar
- ▶ Perl

**Compilers:** IBM provides both c89 and xlc C++ compilers. The scripts generated by the BASE24-eps Version 08.2 installation programs are currently incompatible with the xlc compiler. To assure successful completion of the installation, the path to the c89 C++ compiler must be defined first in the UNIX System Services PATH variable.

## 6.3 BASE24-eps configuration

Our first BASE24-eps configuration was to install a single instance of the BASE24-eps application on system SC48, with the Java Server component (WebSphere Application Server) on system SC52, as shown in Figure 6-2 on page 67.



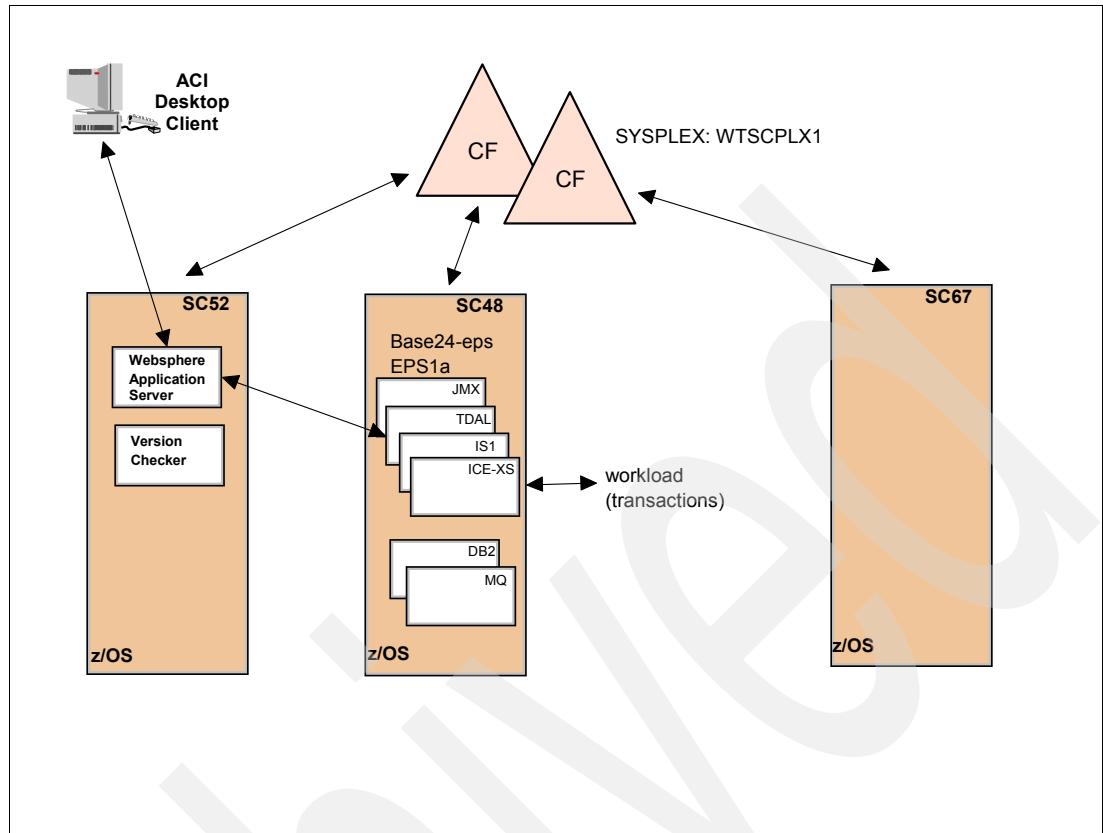


Figure 6-2 Single system BASE24-eps implementation

Incoming workload (transactions) are provided by PCs that are running the ASSET simulation tool. Further details on the ASSET simulation tool are in Appendix A, “ACI Simulation Services for Enterprise Testing” on page 163.

Because WebSphere Application Server was installed on a dedicated LPAR with no DB2 instance, it makes use of the ACI TDAL utility to provide DB2 connectivity. Alternatively, WebSphere Application Server could have been installed in SC48 and made use of the DB2 connectivity configured in SC48, eliminating the requirement for TDAL.

**Note:** ACI and IBM recommend that in a production system, particularly one where high availability is a consideration, IBM WebSphere Application Server be installed in the same LPARs as BASE24-eps, to provide more robust DB2 connectivity and IBM WebSphere Application Server redundancy.

## 6.4 Pre-install checklist and worksheet

Prior to any installation of the BASE24-eps software it is first necessary to review the *BASE24-eps (v08.2) Hardware and Software Requirements* and *BASE24-eps (v08.2) z/OS Install Guide* documents. Then go through the *ACI BASE24-eps z/OS Pre-Install Checklist and Worksheet* and collect the required site information.

For our installation of BASE24-eps, we created a spreadsheet from the information that we gathered. This spreadsheet was used in the ES\_Install process so that we could conveniently cut and paste configuration values, as shown in Figure 6-6 on page 70.

Collecting accurate information is key to a successful installation of BASE24-eps because this information is used to populate installation and configuration data that might be difficult to change after the BASE24-eps software is installed into the UNIX System Services file system.

## 6.5 ES\_Install

The next step in the install process is to execute the ES\_Install program. The *BASE24-eps (v08.2) z/OS Install Guide* provides an overview of the ES\_Install process. This program:

- ▶ Prompts you for the required site and installation details that you captured when working through the pre-install checklist and worksheet.
- ▶ Installs the desktop client (User Interface) on the PC.
- ▶ Transfers (FTP) software and customization configuration to the z/OS system.

### 6.5.1 Executing the ES\_Install program

To execute the ES\_Install program:

1. From the directory that contains the software distribution, locate and execute the installation program ES\_Install, as shown in Figure 6-3.

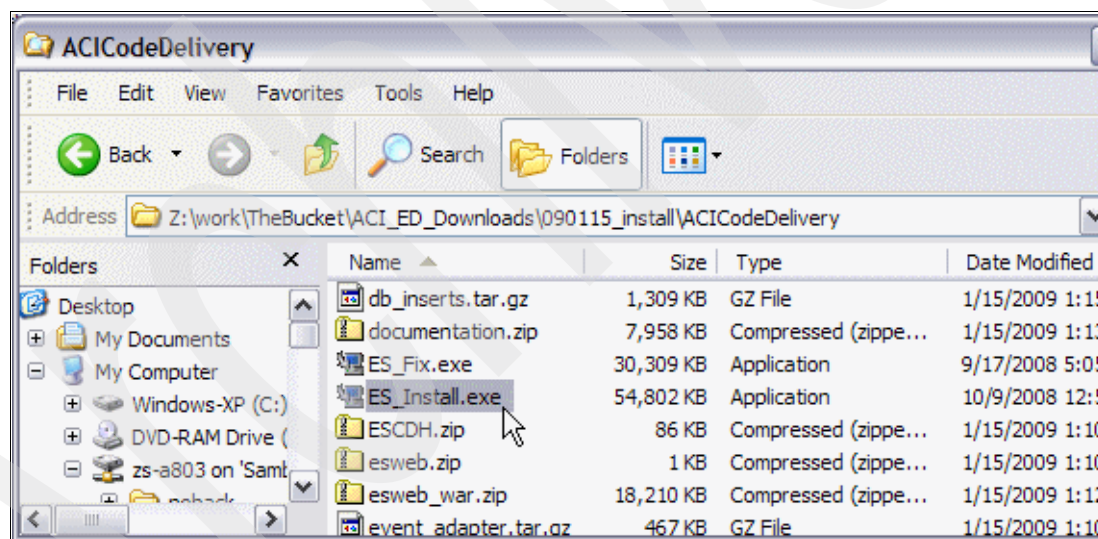


Figure 6-3 Execute ES\_Install

2. The installation process steps through several windows that require you to:
  - a. Select the target platform (in our case z/OS)
  - b. Enter site-specific values (such as host names) and installation values (such as the name of the UNIX System Services directory to use)
3. The initial install window, Figure 6-4 on page 69, provides some usage instructions for the install. Click **OK** to proceed.

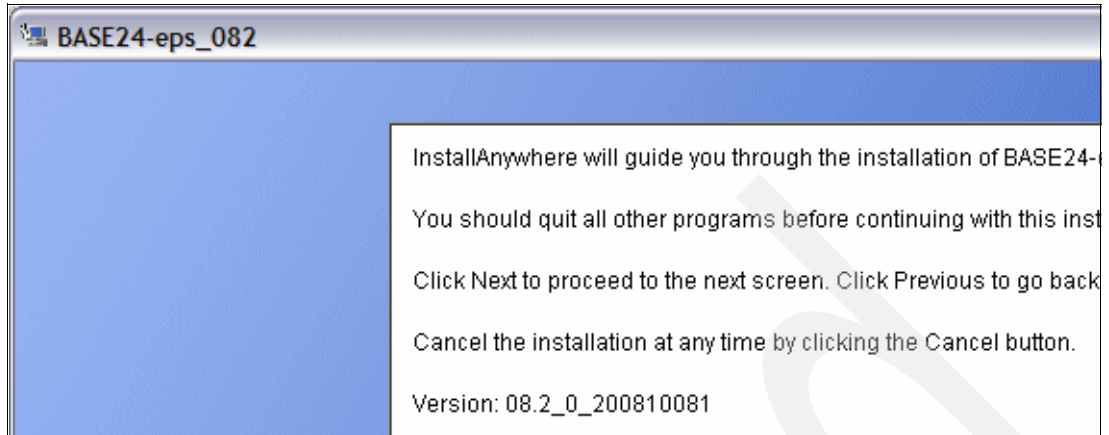


Figure 6-4 First install window

4. On the Choose Install Set window, select the **IBMZ** option (for installation on an IBM System z server), as shown in Figure 6-5.

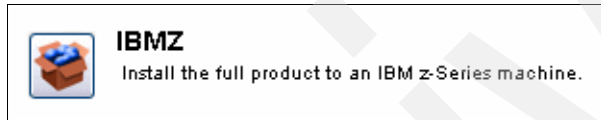


Figure 6-5 Select the install option for z-Series

Using our site survey spreadsheet we copy the required values into the input fields on the windows, a sample is shown in Figure 6-6 on page 70.

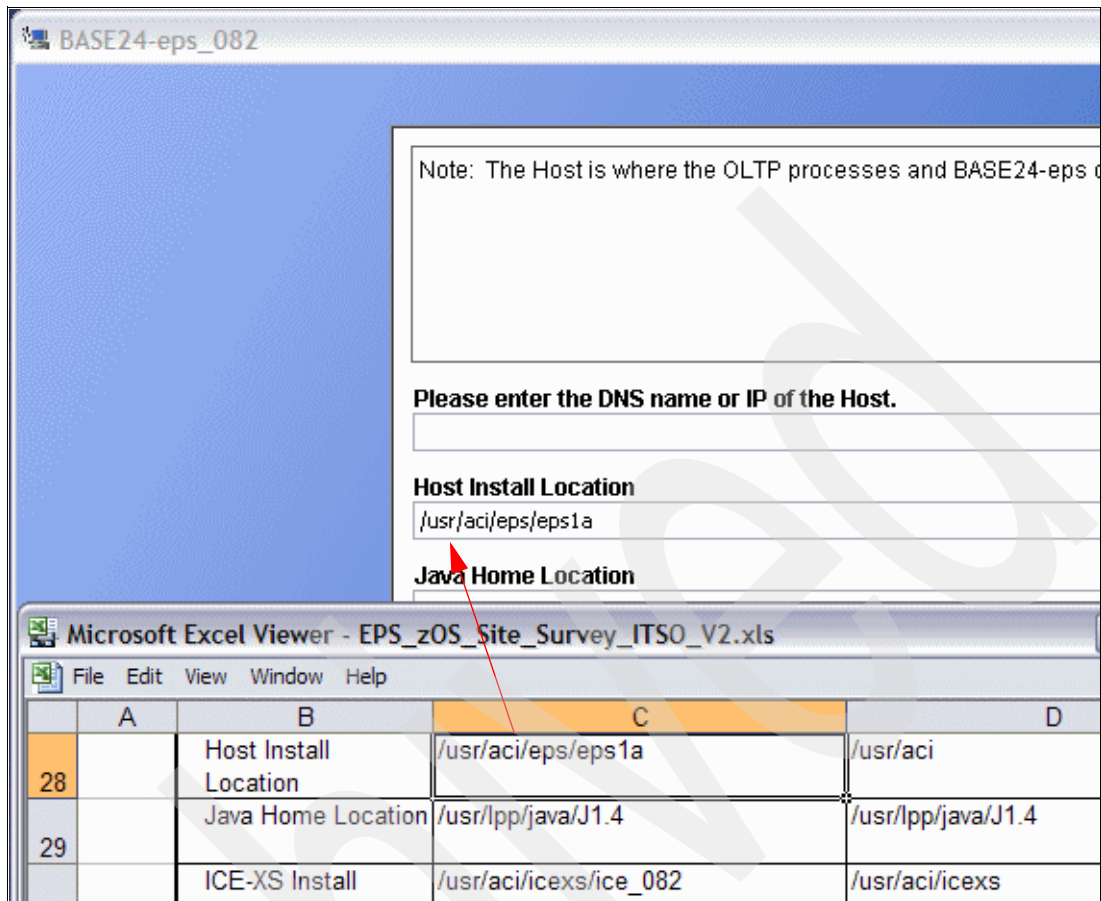


Figure 6-6 Enter the required input values

5. After entering all of the required information, review the collected data, and click **Next** to continue, as shown in Figure 6-7 on page 71.

**Note:** Because later steps use the entered values to set configuration values for the BASE24-eps system, any errors in the entered data might require you to restart the ES\_Install step.

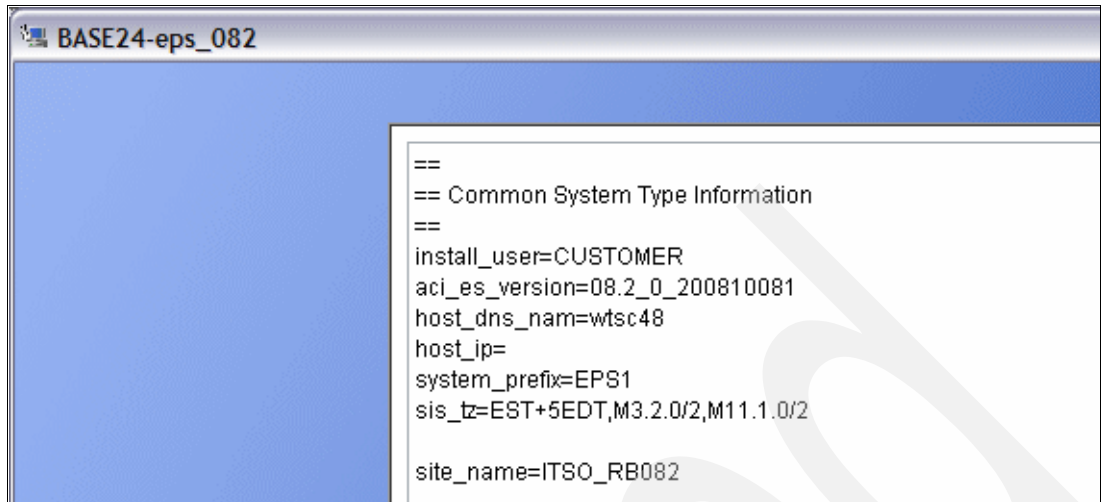


Figure 6-7 Review entered values before continuing

## 6.5.2 FTP files to target system

After accepting the values (by clicking next to continue) the install process will FTP the required installation material to the target z/OS system, as shown in Figure 6-8.

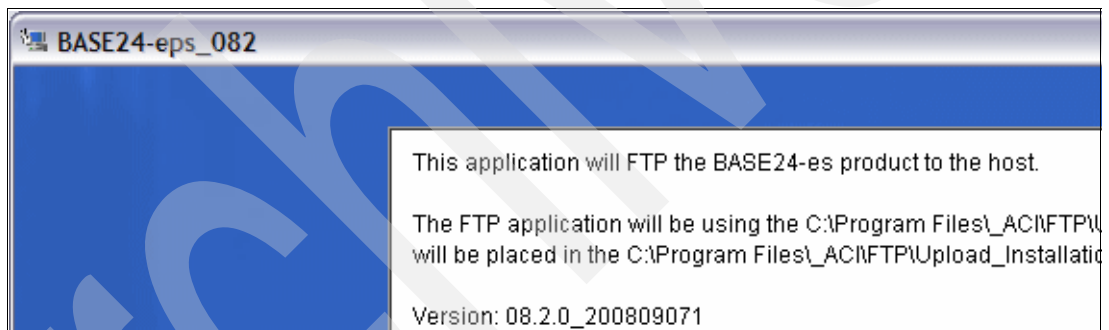


Figure 6-8 FTP the BASE24-eps product to the target z/OS system

The FTP process creates the initial directory structure (in the UNIX System Services file system) and transfers the files that are needed for the next two tasks:

1. Install ICE-XS.
2. Run **essetup**.

Figure 6-9 on page 72 is an example of the FTP transfer.

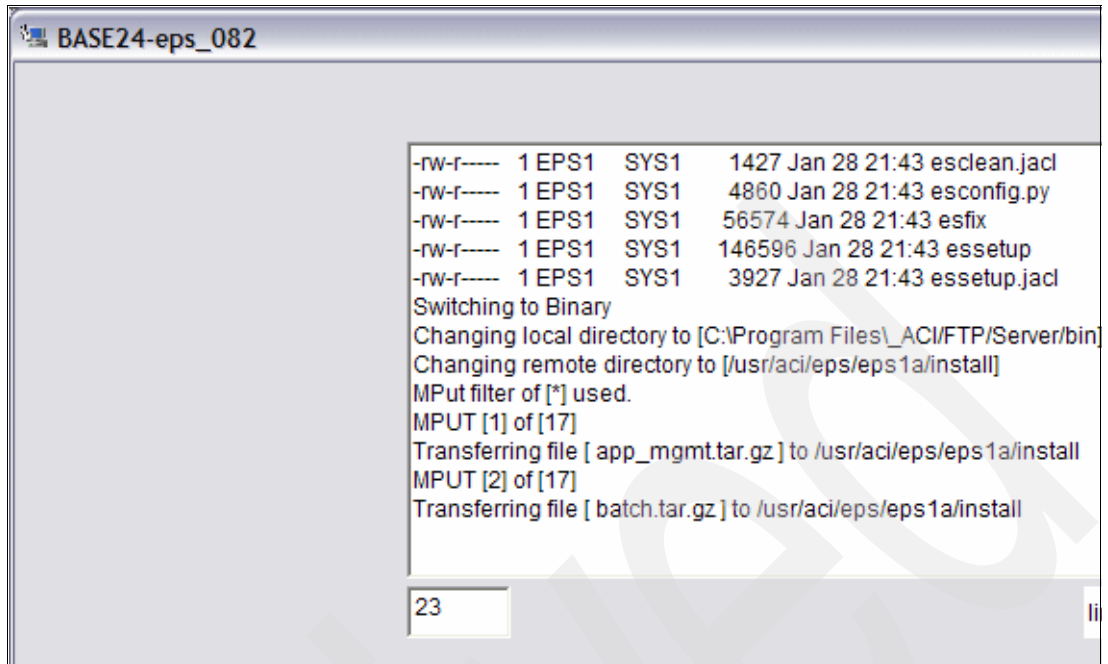


Figure 6-9 FTP progress window

The results of the FTP process are available in C:\Program Files\\_ACI\FTP, where \_ACI is the location we specified for the installation of the desktop client on the PC.

On completion of the file transfer process, the ES\_Install process is finished.

## 6.6 Installing ICE-XS

ICE-XS is a separate component that must be installed prior to running the **essetup** script.

ICE-XS is described in the *ACI Communication Services for Advanced Networking Administrator Guide* as

“ICE-XS stands for Internet Communications for the Enterprise - Cross System. ICE-XS acts as a communications access method, enabling applications or devices, such as 3270 terminals and printers, automatic teller machines, AS/400s, web service clients and IBM host applications (such as CICS and IMS) to communicate with applications on the subject platform and vice versa.”

Prior to starting the ICE-XS install, confirm that **gzip** is available on the system, as shown in Figure 6-10.

```

> whence gzip
/tools/gzip
  
```

Figure 6-10 Confirm that gzip is available on the system

The install file for ICE-XS copies into the BASE24-eps install directory: \$ES\_HOME/install

**Note:** For the install of BASE24-eps, on system SC48, the variable \$ES\_HOME has the value /usr/aci/eps/eps1a.

Installation instructions for ICE-XS are in the *ACI Administration Guide for ICE-XS*.

### 6.6.1 Extracting the ICE-XS files

To extract the ICE-XS files, copy the ICE-XS distribution `ice-xs_product.tar.gz` to the target directory `/usr/aci/icexs/`

`/usr/aci/eps/eps1a/ice-xs_product.tar.gz` → `/usr/aci/icexs/ice-xs_product.tar.gz`

In Figure 6-11, we show the command sequence used to unzip the ICE-XS software from the installation file. A symbolic link of `ice_082` is set to refer to this particular version of ICE-XS.

```
ulimit -SA unlimited

gzip -d ice-xs_product.tar.gz

tar -xf ice-xs_product.tar

cd V1R1080612HP3

uncompress ICEXS-V01R01080612HP03-z0S.tar.Z

tar -xvf ICEXS-V01R01080612HP03-z0S.tar

cd /usr/aci/icexs

ln -s V1R1080612HP3 ice_082
```

Figure 6-11 Extract the ICE-XS files

### 6.6.2 Creating the license files and ICE-XS parameter file

The license files for our systems are made available to us as files in a folder on a PC. Because ICE-XS expects these to be in ASCII format, we use FTP to copy these in binary mode to the target system.

**Note:** If the FTP was accomplished correctly, the license file should be viewable with readable characters using the `viascii` utility (`viascii <filename.xml>`). It should NOT be in readable format using `vi`.

For ease of maintenance:

1. Create a separate directory to hold the ICE-XS license files.
2. Create symbolic links for the license files.

Figure 6-12 on page 74 shows the syntax for transferring the license files.

Figure 6-13 on page 74 shows the syntax for creating the XML file.

```

220-FTPMVS1 IBM FTP CS V1R9 at wtsc48.itso.ibm.com, 15:01:06 on 2009-01-29.

230 EPS1 is logged on. Working directory is "EPS1.".
ftp> cd /usr/aci/icexs/
250 HFS directory /usr/aci/icexs/ is the current working directory
ftp> mkdir license_files
257 "/usr/aci/icexs/license_files" created.
ftp> binary
200 Representation type is Image
ftp> mput *xml
mput sc48_dsxl09011420540641.xml? y
200 Port request OK.
125 Storing data set /usr/aci/icexs/sc48_dsxl09011420540641.xml
250 Transfer completed successfully.

```

Figure 6-12 Transfer license files for ICE-XS

```
ln -s sc48_dsxl09011420540641.xml sc48_license.xml
```

Figure 6-13 Create sc48\_license.xml

Edit the sample parameter file `icexs.params`, as shown in Figure 6-14, to add the location of the license file for this machine. We use the symbolic link created in Figure 6-13. In preparation for our second installation we also add a comment as a reminder for system SC67.

```

== ICE-XS sample param file

== For multi-lpar installations, be sure to edit copies of this file
== to point to the lpar-specific license file:
==
== Lpar sc48:
param LICENCE_FILE /usr/aci/icexs/license_files/sc48_license.xml
==
== Lpar sc67:
== param LICENCE_FILE /usr/aci/icexs/license_files/sc67_license.xml
==
param MAX_POOL_SIZE 33554432

```

Figure 6-14 ICE-XS parameter file

### 6.6.3 Testing the execution of ICE-XS

To test the execution of ICE-XS:

1. Add the ICE-XS directory to your LIBPATH, as shown in Figure 6-15.

```
LIBPATH=$LIBPATH:/usr/aci/icexs/ice_082
export LIBPATH
```

Figure 6-15 Add ICE-XS to standard search path



2. Start ICE-XS. Figure 6-16 shows the test execution.

```
cd /usr/aci/icexs/ice_082

icexsd -param icexs.params -symname TESTICEXS -startup cold -pidfile .TESTICEX

Feb 07 14:28:05 ICE-XS.V01.R01.080612[67830184]: S00006: Process will be
started          with the following param values:
Feb 07 14:28:05 ICE-XS.V01.R01.080612[67830184]: S00007: CINIT_TIMER : 120

Feb 07 14:28:05 ICE-XS.V01.R01.080612[67830184]: S00011: Param processing
successful

kill -9 `cat .TESTICEXS.pid`
```

Figure 6-16 Test execution of ICE-XS

This completes the installation of ICE-XS. At this point we can clean up (delete) intermediate tar files and review setting of directory and file permissions.

## 6.7 essetup

For this step, we recommend that you start at least three telnet sessions to be used for:

- ▶ Running the essetup script
- ▶ Viewing the log file (essetup.log)
- ▶ Other investigation or user actions

**Note:** ACI strongly recommends that you execute the essetup script from a character-mode telnet prompt.

The DB2 and WebSphere MQ subsystems must also be active. In addition, the *ACI BASE24-eps (v08.2) z/OS Install Guide* provides other setup and tuning information that you must review prior to running essetup.

If the UNIX System Services file system is defined for sysplex sharing (that is SYSPLEX(YES) is specified in the BPXPRM parmlib member), path names can include the value of the \$VERSION symbol (Figure 6-17) at the start of the path name, which can cause the configuration build to use path names that might not work if the value of \$VERSION changes. You can fix this problem by adding **set -o logical** to the logon profile for the userID that is to be used for the installation. This means that **pwd** will return the expected logical path name (that is without the value of the \$VERSION symbol at the beginning).

```
DWELCH @ SC48:/Z19RC1/usr/aci/eps>pwd
/Z19RC1/usr/aci/eps
```

Figure 6-17 Path names with \$VERSION

Confirm that **gzip** is available (Figure 6-10 on page 72) and that the Perl scripting language is available.

```
> perl -v

This is perl, v5.8.7 built for os390-thread-multi
```

Figure 6-18 Perl is available

The essetup script will:

- ▶ Unzip the BASE24-eps software.
- ▶ Build command streams for definition of MQ and DB2 objects.
- ▶ Apply the customization and configuration values entered during the ES\_Install process.
- ▶ Copy the ICE-XS parameters into the required location for BASE24-eps.
- ▶ Compile and link C++ programs.
- ▶ Configure components, such as ICE-XS, metadata, application data, ESWeb, JMX application management, and Event Adapter.
- ▶ Create metadata that describes DB2 tables and table schemas.
- ▶ Load application data into the database.

The *BASE24-eps (v08.2) zOS Install Guide* provides an overview of the essetup steps.

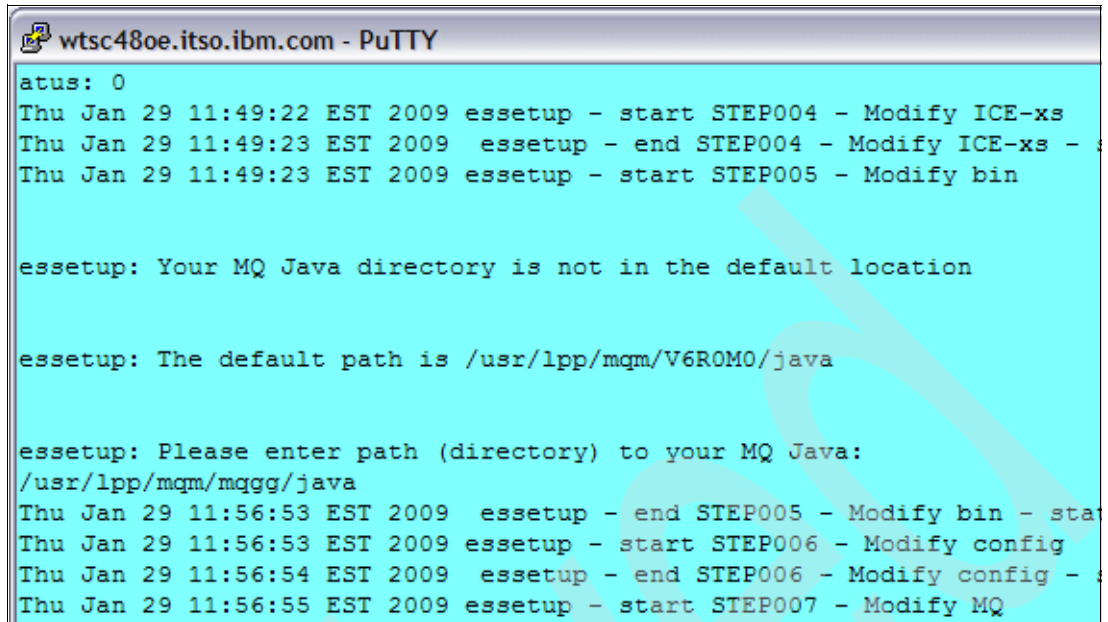
To execute the essetup script:

1. Navigate to the install folder under the BASE24-eps install location. Make the essetup script executable by issuing the **chmod u+x essetup** command.
2. Start the script by executing the essetup script, as shown in Figure 6-19.

```
wtsc48oe.itso.ibm.com - PuTTY
EPS1 @ SC48:/usr/aci/eps/eps1a/install>ls
JavaSF.tar.gz          db_inserts.tar.gz      ice-xs_product.tar.gz
Modules.lst            esclean                 lib.tar.gz
SETUPINI               esclean.jacl           makes.tar.gz
app_mgmt.tar.gz        esconfig.py            metadata.tar.gz
batch.tar.gz           esfix                  mq.tar.gz
bin.tar.gz             essetup                sisobj.tar.gz
cdeinstall.pl          essetup.jacl           source_servercsn.tar.gz
cdepatch.pl            event_adapter.tar.gz   vsp.tar.gz
config.tar.gz          ice-xs_config.tar.gz   xml4c.tar.gz
EPS1 @ SC48:/usr/aci/eps/eps1a/install>chmod u+x essetup
EPS1 @ SC48:/usr/aci/eps/eps1a/install>essetup
Thu Jan 29 11:45:25 EST 2009 essetup - start STEP001 - Cleanup ES_HOME
Thu Jan 29 11:45:26 EST 2009 essetup - end STEP001 - Cleanup ES_HOME
0
```

Figure 6-19 Execute the essetup installation script

Figure 6-20 on page 77 displays a sample of the message output during the execution of the essetup script.



```
wtsc48oe.itso.ibm.com - PuTTY
atus: 0
Thu Jan 29 11:49:22 EST 2009 essetup - start STEP004 - Modify ICE-xs
Thu Jan 29 11:49:23 EST 2009 essetup - end STEP004 - Modify ICE-xs -
Thu Jan 29 11:49:23 EST 2009 essetup - start STEP005 - Modify bin

essetup: Your MQ Java directory is not in the default location

essetup: The default path is /usr/lpp/mqm/V6R0M0/java

essetup: Please enter path (directory) to your MQ Java:
/usr/lpp/mqm/mqgg/java
Thu Jan 29 11:56:53 EST 2009 essetup - end STEP005 - Modify bin - stat
Thu Jan 29 11:56:53 EST 2009 essetup - start STEP006 - Modify config
Thu Jan 29 11:56:54 EST 2009 essetup - end STEP006 - Modify config -
Thu Jan 29 11:56:55 EST 2009 essetup - start STEP007 - Modify MQ
```

Figure 6-20 essetup progress

The execution of the essetup script will pause to allow the installation to run several required z/OS jobs.

The essetup script creates:

- ▶ A PDS that contains the DBRM for the BASE24-eps application
- ▶ A PDS that contains the batch jobs to bind DB2 plans and packages

The essetup script pauses, as shown in Figure 6-21 on page 78, so that these jobs can be executed on the target system. The members of the PDS that contain the JCL to be submitted are also indicated in the output message.

```
wtsc48oe.itso.ibm.com - PuTTY

essetup: Please enter path (directory) to your MQ Java:
/usr/lpp/mqm/mqgg/java
Thu Jan 29 11:56:53 EST 2009 essetup - end STEP005 - Modify bin - stat
Thu Jan 29 11:56:53 EST 2009 essetup - start STEP006 - Modify config
Thu Jan 29 11:56:54 EST 2009 essetup - end STEP006 - Modify config -
Thu Jan 29 11:56:55 EST 2009 essetup - start STEP007 - Modify MQ
Thu Jan 29 11:56:57 EST 2009 essetup - end STEP007 - Modify MQ - stat
Thu Jan 29 11:56:57 EST 2009 essetup - start STEP008 - Copy DAL Insert
Thu Jan 29 11:57:40 EST 2009 essetup - end STEP008 - Copy DAL Insert
0
Thu Jan 29 11:57:40 EST 2009 essetup - start STEP009 - Compile/Link C+

We pushed a DB2 Plan and JCL to the following PDS:

    DBRM => 'B24EPS.EPS1.DB2(LIBSIDB2) '
    JCL   => 'B24EPS.EPS1.JCL(RUNPLAN) '
          'B24EPS.EPS1.JCL(RUNPKG) '

Do not continue until you run the above 2 JCL. Press 'y' to continue.
█
```

Figure 6-21 essetup pauses to allow execution of DB2 jobs

After the JCL jobs run and a **y** is entered to allow the essetup to continue, there will be another prompt for user input. At this point, Figure 6-22 on page 79, we select the option to “push” the MQ object definitions to a z/OS dataset, and we used a batch CSQUTIL job to define the MQ objects.

```
wtsc48oe.itso.ibm.com - PuTTY
Thu Jan 29 11:57:40 EST 2009 essetup - start STEP009 - Compile/Link C++
We pushed a DB2 Plan and JCL to the following PDS:

    DBRM => 'B24EPS.EPS1.DB2(LIBSIDB2)'

    JCL  => 'B24EPS.EPS1.JCL(RUNPLAN)'

           'B24EPS.EPS1.JCL(RUNPKG)'

Do not continue until you run the above 2 JCL.  Press 'y' to continue.
y
Thu Jan 29 13:36:25 EST 2009 essetup - end STEP009 - Compile/Link C++
0
Thu Jan 29 13:36:25 EST 2009 essetup - start STEP010 - Build MQ Manager

We are about to create EPS queues in Manager MQG1

Here are your options:
→ 1. You want us to push the Queue Definitions to PDS, for MQ admin
   2. You have already created the queues, and are ready to continue
   3. You want this step to add EPS queues to your MQG1.

Select the option above: █
```

Figure 6-22 MQ build decision point

Figure 6-23 on page 80 shows that we defined the MQ objects. The essetup script pauses so that we can decide how we will create the DB2 objects. We enter a 1 to “push” the database definitions to a z/OS data set.

```
wtsc48oe.itso.ibm.com - PuTTY
Copying file ... B24EPS.EPS1.MQ(DEFQLOC)

We pushed the MQ configuration to:

    cfg => 'B24EPS.EPS1.MQ(DEFQLOC)'

Do not continue until you created the queues. Press 'y' to continue.
Y
Thu Jan 29 13:55:18 EST 2009 essetup - start STEP011 - Configure Metadata

Enter the password for DB2 user EPS1:
Thu Jan 29 13:56:28 EST 2009 essetup - end STEP011 - Configure Metadata
s: 0
Thu Jan 29 13:56:28 EST 2009 essetup - start STEP012 - Create DB2 Database

We are ready to create tablespaces/tables and load the Database

Here are your options:
    1. Push DDL to PDS for DBA to run.
    2. You have already run the DDL, and want us to load seed data now.
    3. You want us to run DDL and load seed data now.

Select the option above:
```

Figure 6-23 DB2 build decision point

Having chosen option 1, we (Figure 6-24 on page 81):

- ▶ Created the DB2 SQL statements
- ▶ Executed the SQL via SPUFI

We were now ready to proceed, but:

- ▶ After entering a **Y** to continue, the script terminated due to an error
- ▶ The essetup script is restarted at the failing step

The reason for the failure in this step was due to a DB2 setup problem that is not directly relevant to this discussion; however, this does allow us to show that the script can be restarted. Our DB2 configuration problem was fixed, allowing us to restart at the failing step. You can identify the step at which the essetup script fails by looking at the essetup message output. A message about the start and end of each step is logged to the terminal. Figure 6-24 on page 81 shows the restart.

**Note:** Before executing the EPSDDL (the generated table creation DDL), review the DDL to ensure that the correct (site-specific) values for DB2 buffer pools are used.

```
wtsc48oe.itso.ibm.com - PuTTY
Thu Jan 29 13:56:28 EST 2009  essetup - end STEP011 - Configure Metadata
s: 0
Thu Jan 29 13:56:28 EST 2009 essetup - start STEP012 - Create DB2 Data
d

We are ready to create tablespaces/tables and load the Database

Here are your options:
  1. Push DDL to PDS for DBA to run.
  2. You have already run the DDL, and want us to load seed data now.
  3. You want us to run DDL and load seed data now.

Select the option above:1

Copying file ... 'B24EPS.EPS1.DB2 (EPSDDL) '

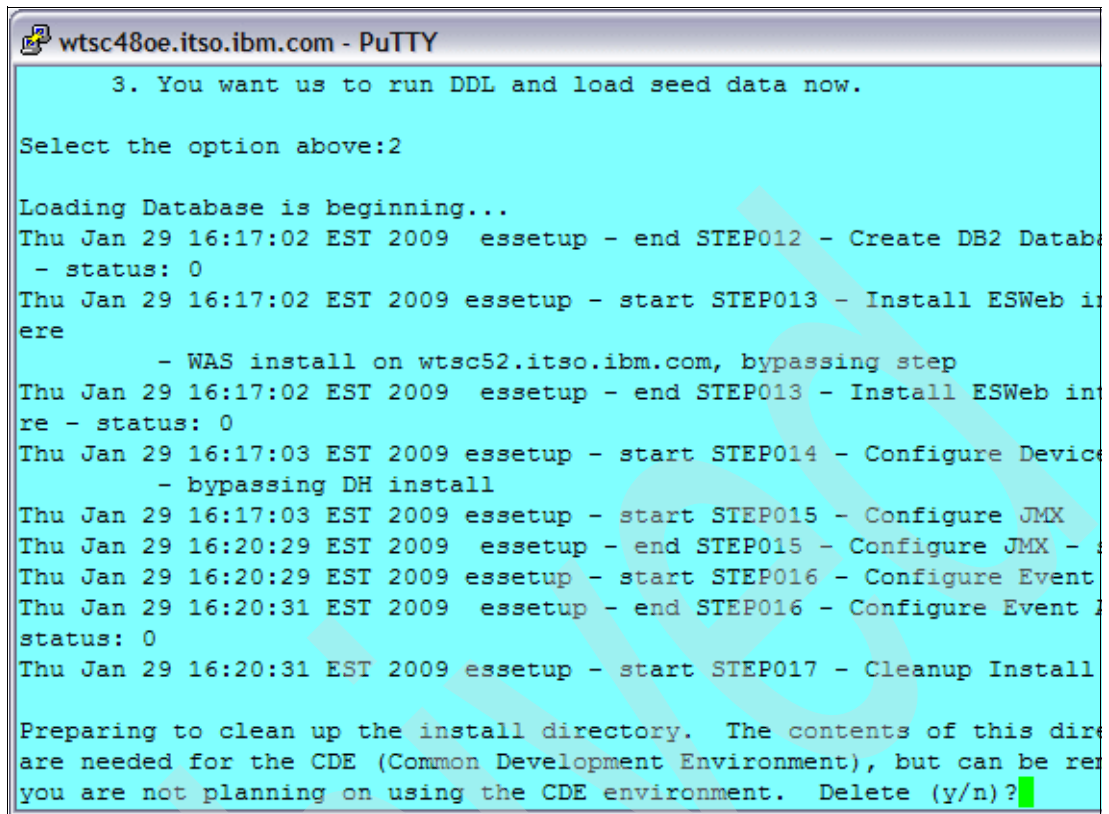
We pushed the the DLL to:

      DLL => 'B24EPS.EPS1.DB2 (EPSDDL) '

Do not continue until you ran the DLL.  Press 'y' to continue.
y
EPS1 @ SC48:/usr/aci/eps/eps1a/install>
EPS1 @ SC48:/usr/aci/eps/eps1a/install>essetup STEP012
```

Figure 6-24 Restart of essetup after an error

At Figure 6-25 on page 82 shows, the essetup script successfully restarted at our previously failing step and is now at the last step, as shown in Figure 6-25 on page 82. For our purposes, it does not matter which reply we make to this prompt. We reply **Y** and the script execution is complete.



```
wtsc48oe.itso.ibm.com - PuTTY

3. You want us to run DDL and load seed data now.

Select the option above:2

Loading Database is beginning...
Thu Jan 29 16:17:02 EST 2009 essetup - end STEP012 - Create DB2 Database
- status: 0
Thu Jan 29 16:17:02 EST 2009 essetup - start STEP013 - Install ESWeb in
ere
- WAS install on wtsc52.itso.ibm.com, bypassing step
Thu Jan 29 16:17:02 EST 2009 essetup - end STEP013 - Install ESWeb in
re - status: 0
Thu Jan 29 16:17:03 EST 2009 essetup - start STEP014 - Configure Device
- bypassing DH install
Thu Jan 29 16:17:03 EST 2009 essetup - start STEP015 - Configure JMX
Thu Jan 29 16:20:29 EST 2009 essetup - end STEP015 - Configure JMX -
Thu Jan 29 16:20:29 EST 2009 essetup - start STEP016 - Configure Event
Thu Jan 29 16:20:31 EST 2009 essetup - end STEP016 - Configure Event
status: 0
Thu Jan 29 16:20:31 EST 2009 essetup - start STEP017 - Cleanup Install

Preparing to clean up the install directory. The contents of this dire
are needed for the CDE (Common Development Environment), but can be rem
you are not planning on using the CDE environment. Delete (y/n)?
```

Figure 6-25 Execution of essetup complete

The essetup script also configured and started the ACIJMX process. We can check the status of the BASE24-eps application with **esstatus**, as shown in Figure 6-26 on page 83.



```
wtsc48oe.itso.ibm.com - PuTTY
ESJMX      JavaSF      config      make
ESWeb      Server      db          mwork
EvtAdapter batch      install    queuing
JCL        bin          lib        source_servercs
JWHAT      comm       logs       vsp

EPS1 @ SC48:/u/eps1/EPS1>cd bin
EPS1 @ SC48:/u/eps1/EPS1/bin>./env_vars
EPS1 @ SC48:/u/eps1/EPS1/bin>cd $ES_HOME
EPS1 @ SC48:/usr/aci/eps/eps1a>esstatus
*** MQ Series Manager ***
MQG1 is running (1)
EPS1.EVTLOG(110)

*** DB2 server ***
DB2 connection is established (2)

*** ES Processes for System: EPS1 ***
EPS: ACIJMX is running (PID: 67698826) (3)

*** ICE-XS Listeners ***
None

EPS1 @ SC48:/usr/aci/eps/eps1a>
```

Figure 6-26 BASE24-eps status

The esstatus results (in Figure 6-26) show that:

- ▶ There is connectivity to our MQ subsystem
- ▶ There is connectivity to our DB2 database
- ▶ Process ACIJMX is running

This completes the essetup process and the installation of BASE24-eps on this z/OS system.

We now have a:

- ▶ DB2 Database and tables
- ▶ Set of MQ queues
- ▶ File system that contains BASE24-eps

Use the DB2 Administration tool to view the tables that we created, as shown in Figure 6-27 on page 84.

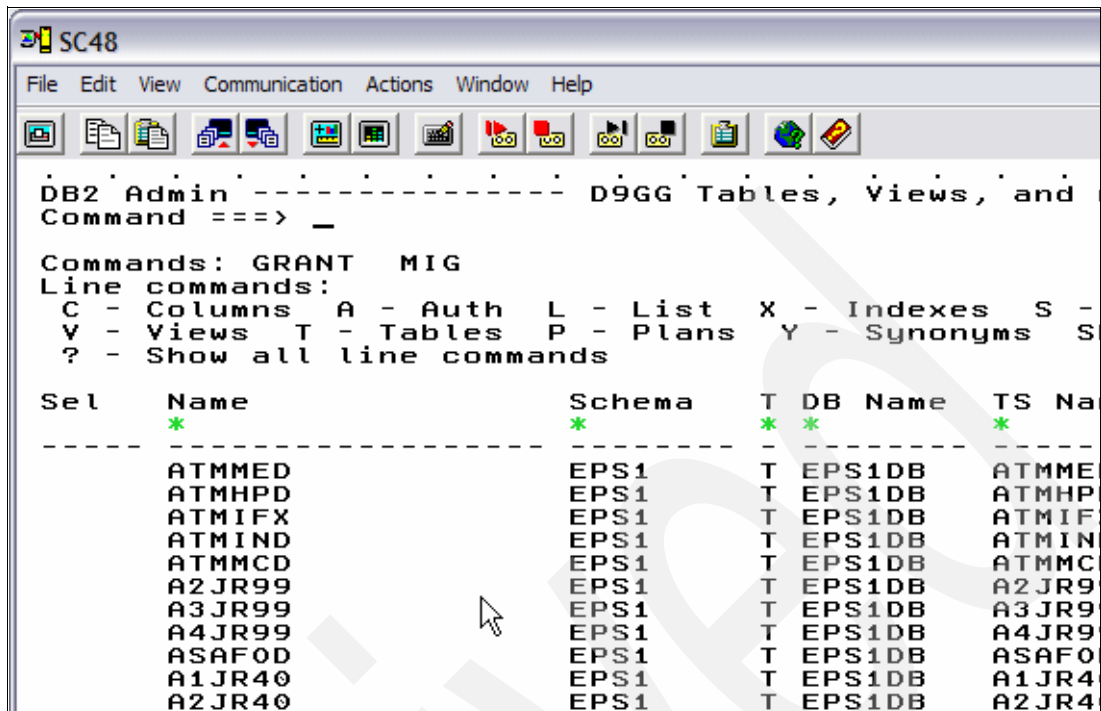


Figure 6-27 BASE24-eps DB2 objects

Use the MQ ISPF interface to view the MQ objects that we created, as shown in Figure 6-28.

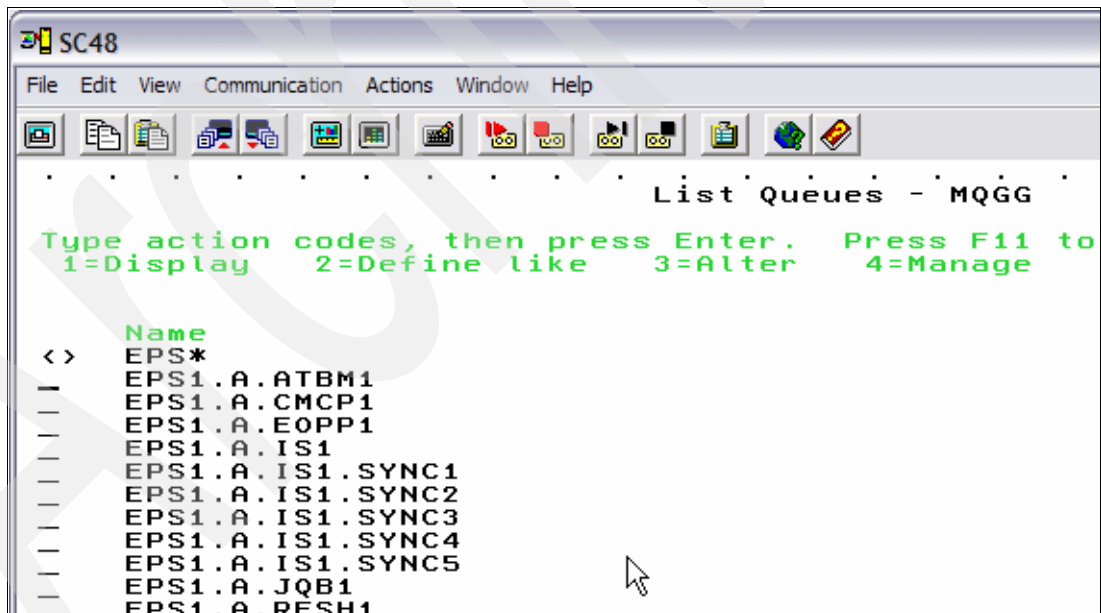


Figure 6-28 BASE24-eps MQ objects

As an alternative to using UNIX commands from a telnet session, zOS users can use ISHELL for an ISPF-like view of the BASE24-eps file system that was created, as shown in Figure 6-29 on page 85.

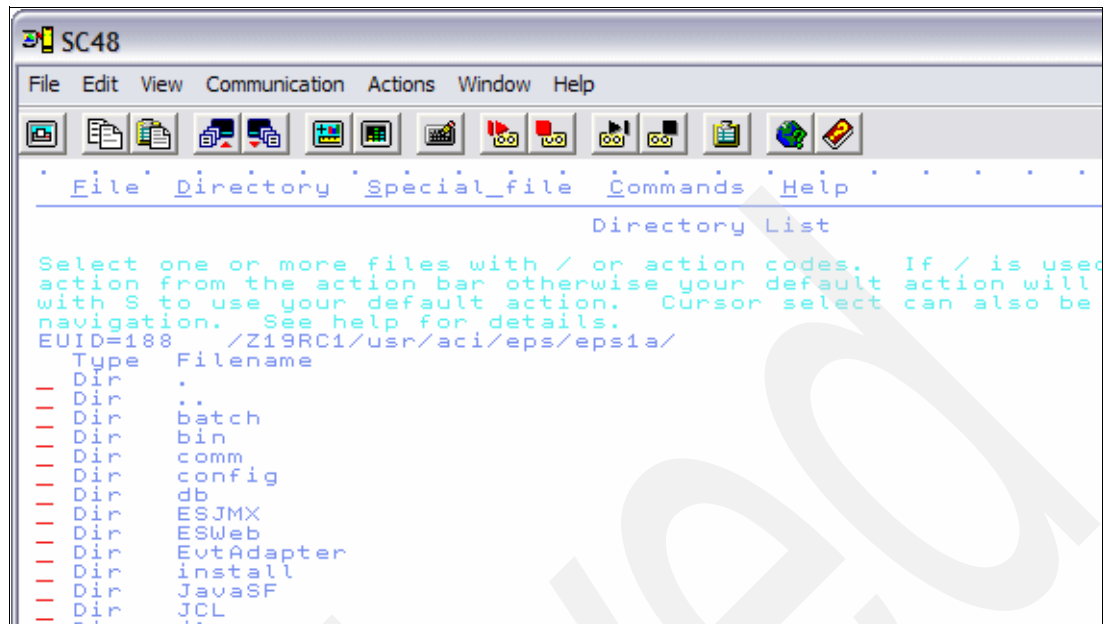


Figure 6-29 BASE24-eps file system

## 6.8 Initial database load

The essetup creates the application metadata, which is the internal application data that describes the format and layout of the data, as used by the BASE24-eps application. The essetup also creates the JCL to create a DB2 configuration with important seed data in the environment table, external connections, event tables, and others. Then you (after verifying Desktop connectivity) can load additional application data or overlay some of the default data as required. For our purposes, we also loaded additional data from an existing system that included such information as institutions, card numbers, scripts, and cryptographic processing.

The test data that we use is configured to work with the business data that we loaded into our BASE24-eps system.

Setting up a working environment from scratch is outside of the scope of this book.

## 6.9 BASE24-eps directory structure

In Figure 6-30 on page 86, the variable \$ES\_HOME has the value /usr/aci/eps/eps1a, which is the install path for system SC48.

<code>\$ES_HOME/bin</code>	contains scripts to start BASE24-eps processes
<code>\$ES_HOME/ESJMX/bin</code>	contains scripts to start and stop JMX
<code>\$ES_HOME/comm/ice-xs</code>	configuration files for ICE-XS
<code>\$ES_HOME/config</code>	configuration files for BASE24-eps
<code>\$ES_HOME/lib</code>	executable objects
<code>\$ES_HOME/logs</code>	output from executable scripts
<code>\$ES_HOME/ESJMX</code>	JMX directories

*Figure 6-30 Key BASE24-eps directories*

## 6.10 Running the esinfo

The **esinfo** script collects information about the system environment and the BASE24-eps installation and writes it to a report file called `esinfo.log`. This report contains useful information about how BASE24-eps is installed on this system.

The report that **esinfo** generates also contains information about port number assignments and utilization. For this effort, we found it useful to copy the `/tmp/EPS1-ports` file (generated by **esinfo**) into the `$ES_HOME/install` directory for use in maintaining our port assignments and for subsequent ICE-XS NOF file configuration efforts.

Figure 6-31 on page 87 shows a sample **esinfo** report.

```

Sat Feb  7 16:12:33 EST 2009 esinfo start of job

*****
*esinfo: System Information*
*****
*
machine class:          2094
machine node name:      SC48
operating system:       OS/390
release level:          19.00
version:                 03
Compiler information: V1.9

Who is on the box:
EPS1      ttyp0002      Feb  4 17:09 (welch.itso.ibm.com)
EPS1      ttyp0004      Feb  6 16:40 (9.12.5.68)
EPS1      ttyp0010      Feb  7 11:18 (9.12.5.67)

Who is running script: EPS1
uid=188(EPS1) gid=0(SYS1) groups=2500(WSCFG1)

Java information:
java version "1.4.2"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.2)
Classic VM (build 1.4.2, J2RE 1.4.2 IBM z/OS Persistent Reusable VM build
cm142-

ICE-xs information:
Ver 01 Rel 02 Mod 080612

DB2 information (from SETUPINI):
db2_creator=EPS1

```

Figure 6-31 Sample esinfo data

## 6.11 Installation verification

After the essetup process completes, we verify the installation by manually starting each of the BASE24-eps components, which is detailed in the *ACI BASE24-eps z/OS Installation Guide*.

**Note:** The BASE24-eps shell scripts rely on shell environment variables to distinguish between BASE24-eps instances. Before running any of the BASE24-eps scripts you must execute a shell script to load the environment variables, as demonstrated in Figure 6-32 on page 88.

```

EPS1 @ SC48:/u/eps1
> echo $ES_HOME

EPS1 @ SC48:/u/eps1
> cd /usr/aci/eps/eps1a/bin
EPS1 @ SC48:/usr/aci/eps/eps1a/bin
> . ./env_vars
EPS1 @ SC48:/usr/aci/eps/eps1a/bin
> echo $ES_HOME
/usr/aci/eps/eps1a

```

Figure 6-32 Establish environment variables

The processes that are started include:

- ▶ JMX (Application Manager)
- ▶ Event Queue Logger
- ▶ TDAL (DB2 connector)
- ▶ ICE-XS processes
- ▶ Version Checker
- ▶ XML Server
- ▶ IS Server

Each BASE24-eps process has a startup script under the \$ES\_HOME directory. The locations and names of the scripts are detailed in the *ACI BASE24-eps z/OS Installation Guide*, for example:

- ▶ RUNACIJMX
- ▶ RUNTDAL

Figure 6-33 shows the BASE24-eps status display.

```

*** MQ Series Manager ***
MQGG is running

*** DB2 server ***
DB2 connection is established

*** ES Processes for System: EPS1 ***
EPS: silisten is running (PID: 34144611)
EPS: tdal.exe is running (PID: 591121 591122 591504 591509 591522 591526)
EPS: ACIJMX is running (PID: 84476255)

*** ICE-XS Listeners ***
ICE-XS: EPS1_ANCR is running (PID: 590336)
ICE-XS: EPS1_APX is running (PID: 590344)
ICE-XS: EPS1_GUI is running (PID: 590565)
ICE-XS: EPS1_VISAI is running (PID: 590568)
ICE-XS: EPS1_VISA is running (PID: 590577)

```

Figure 6-33 BASE24-eps status display

Because on our system the BASE24-eps processes are started by user EPS1, all jobnames show as EPS1\* in a process display in SDSF (option PS). Scrolling right in the display, as shown in Figure 6-34, it is possible to identify which command was used to start the process, for example, you can see that the IS process is one of the EPS18 tasks. We recommend that you use the `_BPX_JOBNAME` variable to make it easier to see what function is being performed by each particular task.

```

SDSF PROCESS DISPLAY SC48 ALL LINE 1-
COMMAND INPUT ==>
NP  JOBNAME  Command
EPS1 sh -L
EPS19 /usr/aci/eps/eps1a/lib/safm.exe
EPS13 /usr/aci/eps/eps1a/lib/eopp.exe
EPS14 /usr/aci/eps/eps1a/lib/tbp.exe
EPS17 /usr/aci/eps/eps1a/lib/ttlm.exe
EPS11 /usr/aci/eps/eps1a/lib/atbm.exe
EPS12 /usr/aci/eps/eps1a/lib/jbtc.exe
EPS11 /usr/aci/icexs/ice_082/icexsd -symname
EPS16 /usr/aci/eps/eps1a/lib/cmcp.exe
EPS15 /usr/aci/eps/eps1a/lib/rfsh.exe
EPS18 /usr/aci/eps/eps1a/lib/is.exe
EPS12 /usr/aci/icexs/ice_082/icexsd -symname
EPS14 /usr/aci/eps/eps1a/lib/xml.exe
EPS19 /usr/aci/icexs/ice_082/icexsd -symname
EPS1 otelnetd -Y welch.Itso.ibm.com -p eps1
EPS17
EPS18 /usr/aci/eps/eps1a/lib/sislib/sitimrp
EPS13 /usr/aci/icexs/ice_082/icexsd -symname
EPS1 EXEC
EPS16 /usr/lpp/java/J1.4/bin/java
EPS15 /usr/aci/eps/eps1a/lib/sislib/silisten
  
```

Figure 6-34 SDSF display of BASE24-eps processes

Figure 6-35 on page 90 shows some of the DB2 threads that the BASE24-eps processes started, which are all background connections to DB2, and the maximum number of these is controlled by the `IDBACK` parameter that we mentioned in 6.1, “Target ITSO z/OS environment” on page 64. A similar display can be done for MQ background connections.

Sel	Name	St	A	Req	ID	Auth	ID	Pl
	RRSAF	T		6060	UNKNOWN#5936	EPS1	?	?
	RRSAF	T		3610	UNKNOWN#5092	EPS1	?	?
	RRSAF	T		18	safm.exe#509	EPS1	?	?
	RRSAF	T		41	safm.exe#509	EPS1	?	?
	RRSAF	T		18	xml.exe#8447	EPS1	?	?
	RRSAF	T		2697	xml.exe#8447	EPS1	?	?
	RRSAF	T		12066	UNKNOWN#5935	EPS1	?	?
	RRSAF	T		10931	UNKNOWN#5936	EPS1	?	?
	RRSAF	T		18	rfsh.exe#593	EPS1	?	?
	RRSAF	T		49	rfsh.exe#593	EPS1	?	?
	RRSAF	T		90	is.exe#59358	EPS1	?	?
	RRSAF	T		3291	is.exe#59358	EPS1	?	?

Figure 6-35 Display of BASE24-eps DB2 threads

Use the **runevtparser** script to view the application messages that are being logged to the Event Log, for example, Figure 6-36 depicts the Event Log messages that are associated with an ICE-XS connection failure.

```

SUBSYSTEM ID      : 300
EVENT NUMBER      : 10038
EVENT TEXT        : Incoming TCP/IP connection from STATION 9.12.4.42:
GROUTER EPS1_GUI established
SEVERITY          : Info
-----
DATE TIME         : 2009-02-09T13:56:46-05:00
TASK ID           : 723555
UNQ EVENT ID      : 45
GENERATOR         : EPS1_GUI_B
GENERATOR BLD VSN : V01.R01.080612
SUBSYSTEM ID      : 300
EVENT NUMBER      : 10084
EVENT TEXT        : Endpoint message delivery failure (SymSrc XMLI Sym
NTF), IXS0047: [CONN, Connection not found], APPL EPS1GUI, returning f
e 109
SEVERITY          : Warning
-----
DATE TIME         : 2009-02-09T13:56:53-05:00
TASK ID           : 590565
UNQ EVENT ID      : 49
GENERATOR         : EPS1_GUI
GENERATOR BLD VSN : V01.R01.080612
SUBSYSTEM ID      : 300
EVENT NUMBER      : 10038
EVENT TEXT        : Incoming TCP/IP connection from STATION 9.12.4.42:
GROUTER EPS1_GUI established

```

Figure 6-36 Event log



**Note:** BASE24-eps currently does not write any messages to the zOS system log.

After you verify that your BASE24-eps installation is working, normal operational practice is to manage the status of the application processes using the desktop client user interface and the ACI JMX Application Management interfaces.

## 6.12 Installing the user interface

The ACI desktop user interface (UI) is a desktop client that controls and manages the BASE24-eps application. The UI is installed as part of the ES\_Install process (6.5, “ES\_Install” on page 68); however, it will also need to be installed on other PCs.

### 6.12.1 Basic installation

To install:

1. From the directory that contains the software distribution, locate and execute the ES\_Install program, shown in Figure 6-3 on page 68.
2. Select the option to install the Desktop Client, as shown in Figure 6-37.



Figure 6-37 Desktop install

3. Enter the following information into the install dialog:
  - Host name of the Java Server, as shown in Figure 6-38 on page 92
  - Host name of the Version Checker
  - Base port number

In our configuration, the Java Server is an instance of WebSphere Application Server, which is running on system SC52. The Version Checker is also running on SC52.

The base port number is one of the BASE24-eps configuration values that is specified in the ES\_Install process.

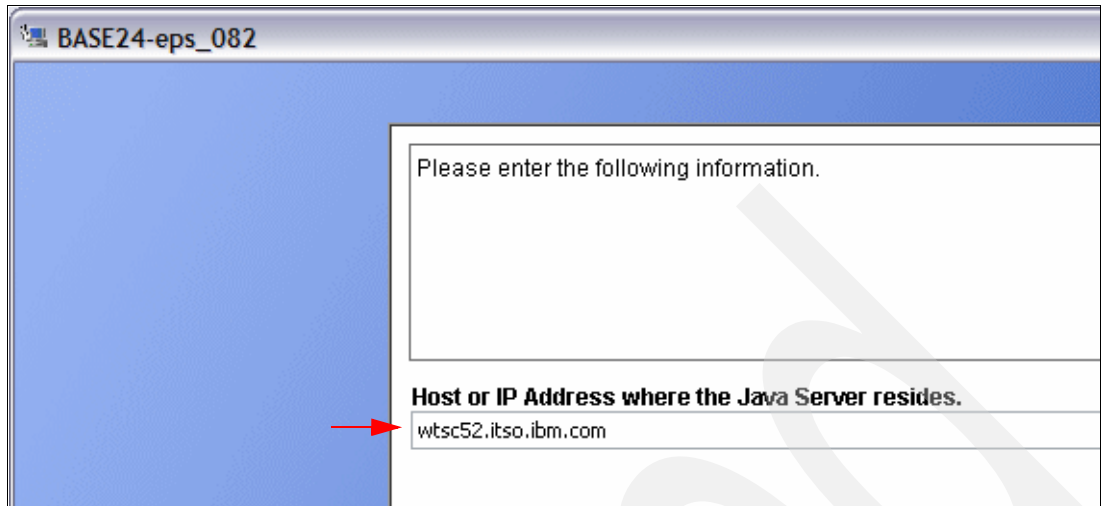


Figure 6-38 Java Server address

Continue through the windows until you complete the installation.

## 6.12.2 Installing UI support on WebSphere Application Server

In the ITSO BASE24-eps configuration, the Java Server component that provides support for the client UI is the instance of WebSphere Application Server that is running on z/OS system SC52.

The actual set up steps for installing the UI support are detailed in the ACI BASE24-eps z/OS Install Guide.

Figure 6-39 and Figure 6-40 on page 93 are sample WebSphere Application Server windows.

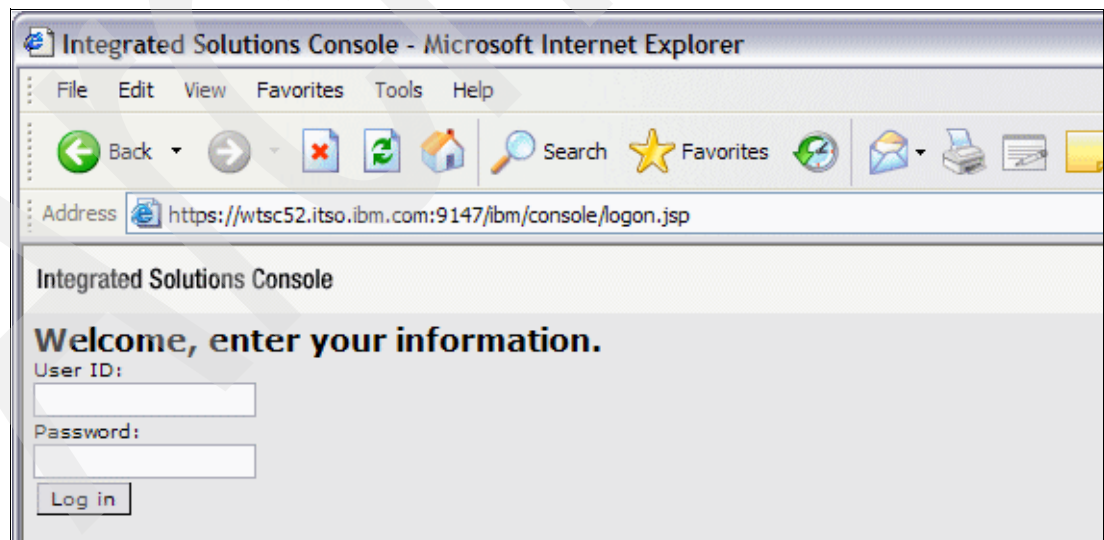


Figure 6-39 WebSphere Application Server logon

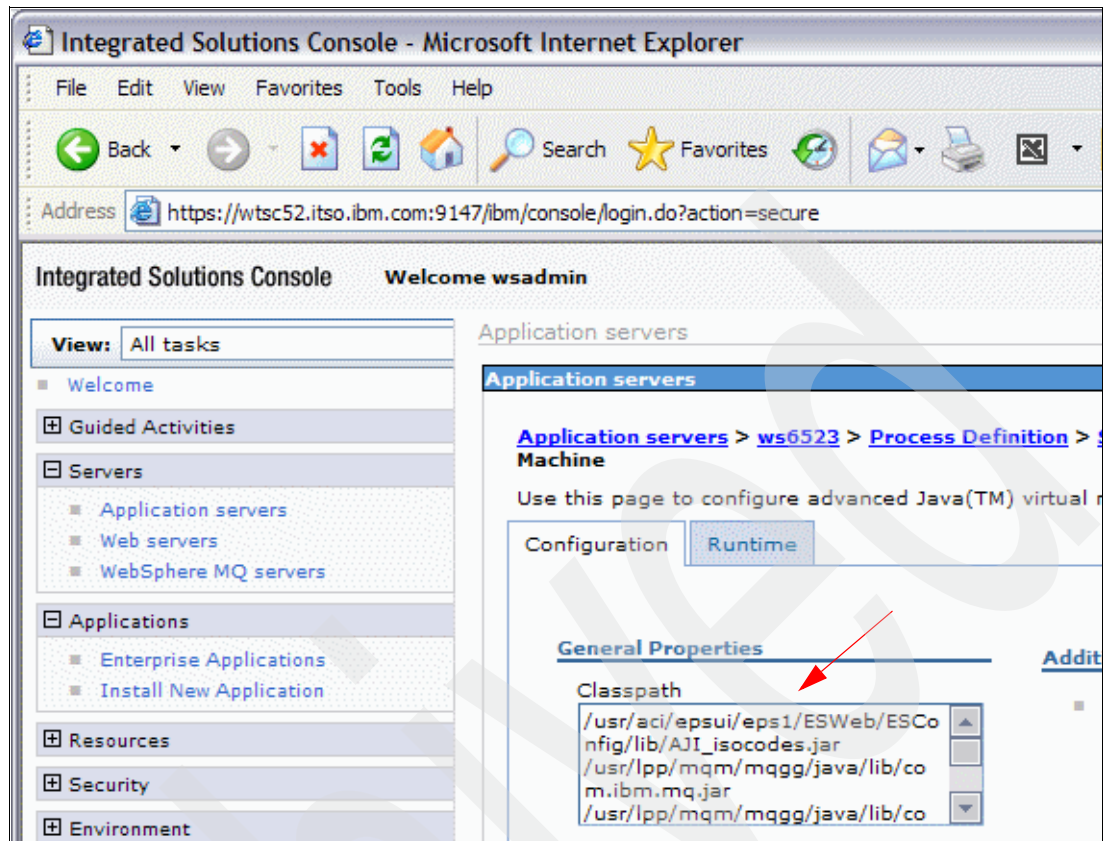


Figure 6-40 BASE24-eps configuration for WebSphere Application Server

### 6.12.3 Version Checker

The Version Checker is a component of the BASE24-eps product that provides the facility to check the level of code on the client and if necessary download new versions of the Java components.

In our environment, the Version Checker process is started on system SC52, as shown in Figure 6-41.

```
EPS1 @ SC52:/usr/aci/epsui/eps1/ESUI
> ./StartVersionChecker
Starting VersionChecker Server
```

Figure 6-41 Start the Version Checker process

When you launch the UI, it connects to the Version Checker and determines if there are any updates to the JAVA application to be downloaded from the current BASE24-eps software level on the zOS system. After this check completes and any JAR files are downloaded, you can logon to the UI.

Figure 6-42 on page 94 shows the Version Checker refreshing UI jar files.

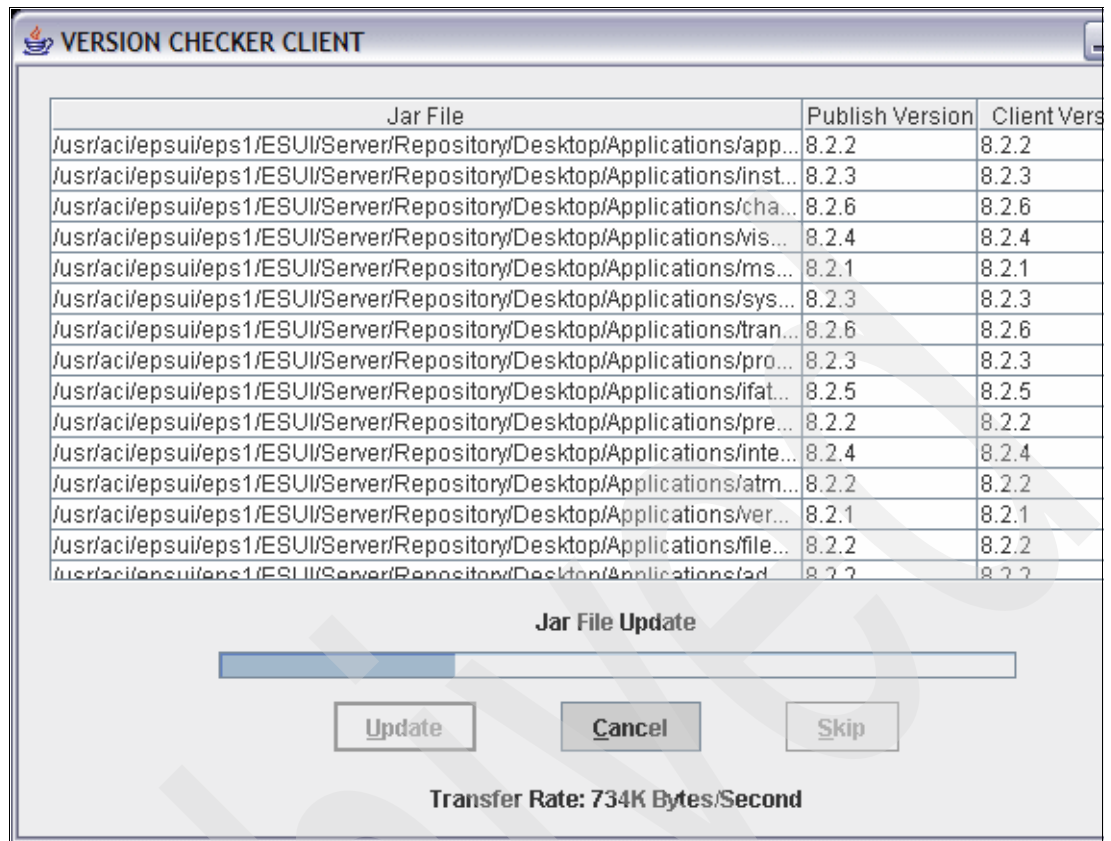


Figure 6-42 Version Checker refreshing UI jar files

#### 6.12.4 Now we can login

After the user interface is configured, you can log on, as shown in Figure 6-43.

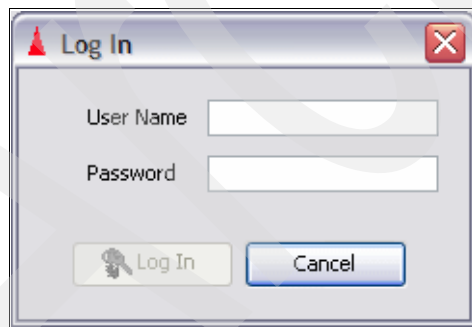


Figure 6-43 UI logon

After logging in you can access the user interface features using the drop-down menus, as shown in Figure 6-44 on page 95.

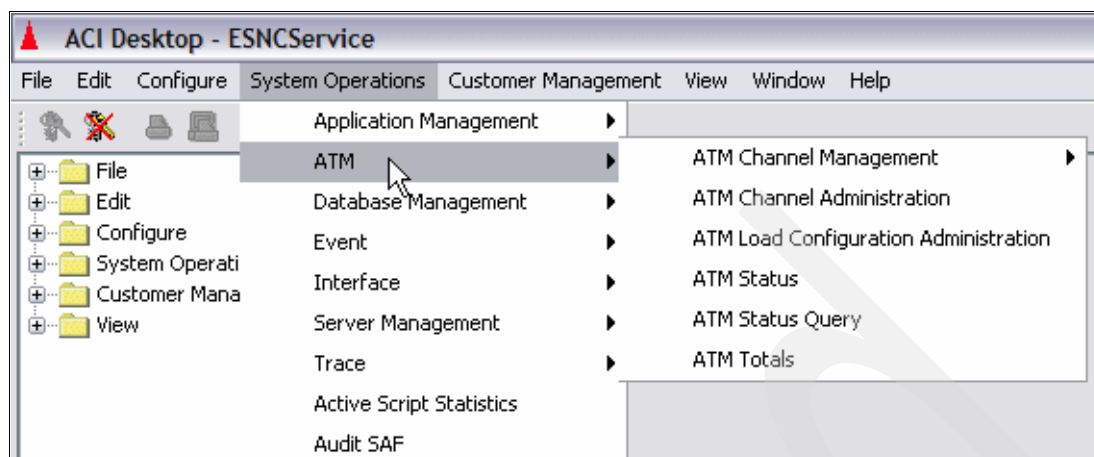


Figure 6-44 The BASE24-eps User Interface

## 6.12.5 Client UI connectivity

The NETSTAT display from SC52 in Figure 6-45 shows:

- ▶ The Version Checker process listening on port 8000
- ▶ The WebSphere Application Server listening on port 8002
- ▶ The WebSphere Application Server connected to BASE24-eps processes on system SC48

MVS User Id	TCP/IP Conn	NETSTAT Local Socket	CS V1R9	TCPIP Name: TCPIP	Foreign Socket	15:09:53 State
EPS18	00021F38	0.0.0.0..5744			0.0.0.0..0	Listen
EPS18	00021F37	0.0.0.0..8000		(1)	0.0.0.0..0	Listen
WS6523	0002208A	0.0.0.0..9144		(2)	0.0.0.0..0	Listen
WS6523	00022069	0.0.0.0..8002			0.0.0.0..0	Listen
WS6523	00022066	0.0.0.0..9146			0.0.0.0..0	Listen
WS6523D	00021FE7	0.0.0.0..9154			0.0.0.0..0	Listen
WS6523D	00021FE8	0.0.0.0..9155			0.0.0.0..0	Listen
WS6523S	0002466F	9.12.4.42..6178			9.12.4.38..8003	Establish
WS6523S	00024697	9.12.4.42..6182			9.12.4.38..8003	(3) Establish
WS6523S	00024698	9.12.4.42..6183			9.12.4.38..8003	Establish
WS6523S	00024699	9.12.4.42..6184			9.12.4.38..8003	Establish
WS6523S	0002466D	9.12.4.42..6177			9.12.4.38..8007	Establish

Figure 6-45 NETSTAT display

## 6.13 Operational considerations

In this section, we discuss operational considerations for BASE24-eps.

### 6.13.1 Defining ICE-XS processes to BASE24-eps

A key component of the BASE24-eps configuration are the ICE-XS processes that provide connectivity for external devices, such as ATM and POS devices.

Running ICE-XS processes can be seen in a esstatus display, as shown in Figure 6-46.

```
*** ICE-XS Listeners ***  
ICE-XS: EPS1_ANCR is running (PID: 590336)  
ICE-XS: EPS1_APX is running (PID: 590344)  
ICE-XS: EPS1_GUI is running (PID: 590565)  
ICE-XS: EPS1_VISAI is running (PID: 590568)  
ICE-XS: EPS1_VISA is running (PID: 590577)
```

Figure 6-46 ICE-XS processes

The parameter files for ICE-XS are the \*.nof files in \$ES\_HOME/comm/ice-xs/

An example definition for a connection to our device that is to send test VISA transactions is shown in Figure 6-47 on page 97.

Some details of interest to note include:

- ▶ MQ name
- ▶ MQ Queue names
- ▶ IP address of remote device (station)

RECVQMGR and RECVQ identify the MQ Manager and MQ Queue from which messages for BASE24-eps processes are received. Such messages include messages that are being sent back to the related device.

SENDQMGR and SENDQ identify the MQ Manager and MQ Queue that are used to send messages to the BASE24-eps processes. Such messages include those that originate at the related device.

Additional information related to the ICE-XS NOF file configuration is in the *Administrator Guide for ACI Communication Services for Advanced Networking*.

```

=====
==
== %name:          visa.nof %
== %version:       064_1 %
== %created_by:    jkreife %
== %date_created:  Mon Oct 09 17:26:55 2006 %
==
=====
(1)
OPEN EPS1_VISA
ADD APPL EPS1VISA, INTERFACE MQ, PROTOCOL B24, RECVQMGR MQGG, RECVQ EPS1VISA,
ENDQMGR MQGG, SENDQ EPS1.IS (2)
ADD MSGROUTER EPS1_VISA, APPL EPS1VISA, LPORT 8014, MAXCONNS 10
ADD DEVPROTOCOL DEVP_VISA, TYPE GENERIC2
== default:
== ADD CONNPROFILE EPS1_VISA, TYPE TCPIP, AUTHZADDR *
== james mitchell:
ADD CONNPROFILE EPS1_VISA, TYPE TCPIP, AUTHZADDR 9.12.5.67 (3)
ADD STATIONPOOL P1A-SIM-I-VISA, DEVPROTOCOL DEVP_VISA, MSGROUTER EPS1_VISA,
CONNPROFILE EPS1_VISA, CONTACT IN, ISTATUS ACTIVE, SYMSRC P1A-SIM-I-VISA,
TFVISA
== james mitchell laptop:
ADD CONNPROFILE JMLT_VISA, TYPE TCPIP, AUTHZADDR 9.57.138.180
"eps1_visa.nof" 22 lines, 1120 characters

```

Figure 6-47 Sample ICE-XS \*.nof parameter file

To put this configuration into service, any existing ICE-XS process (using this configuration) must be shutdown, followed by a cold start of the ICE-XS process. The shutdown can be done using the client UI; however, the cold restart must be performed using a telnet command line.

To restart this process (after a configuration change) requires a cold start of the ICE-XS process:

```
runicexs eps1_visa.nof cold
```

Following the successful cold start of the ICE-XS process, the process are not visible to UI Application Management (JMX). To make the process visible to the UI Application Management, it is necessary to stop and restart the ICE-XS process:

```
stopicexs eps1_visa.nof
```

```
runicexs eps1_visa.nof
```

## 6.13.2 JMX Rebuild

Changes to some configuration options require a rebuild of all or part of the JMX configuration. For configuration changes that involve BASE24-eps process definitions, their MQ queues, or existing ICE-XS \*.nof files, we used the `jmx_eps_reload` script. (The `jmx_eps_reload` script restores attributes to out-of-box values and should be disabled or removed in a production environment.).



Configuration changes that involve adding new ICE-XS \*.nof files use the jmx\_add\_nof script. The jmx\_add\_nof script does not support modifying existing ICE-XS configurations.

### 6.13.3 OLTP Warmboot

BASE24-eps makes extensive use of Online Transaction Processing tables or OLTPs. OLTPs are a high-performance memory mapped file that is used for the fastest possible access to read-only data. They are loaded from underlying DB2 tables.

Changes to the underlying DB2 tables require that the OLTP tables be reloaded and the BASE24-eps processes that use the OLTPs be warm booted so that the changes are picked up and put into effect.

Figure 6-48 shows the icon to start an OLTP rebuild from the client UI.

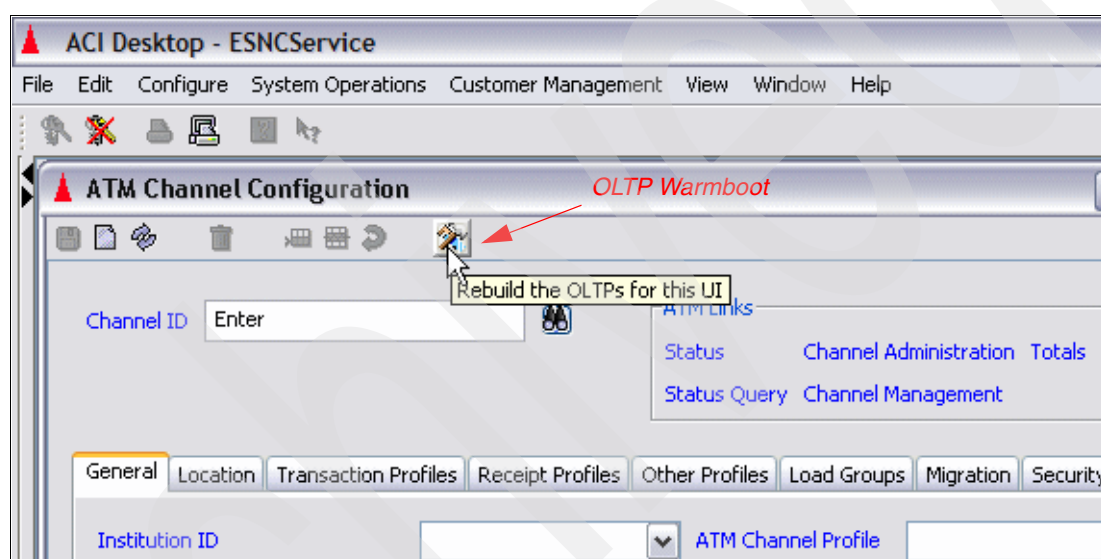


Figure 6-48 OLTP warmboot

## 6.14 BASE24-eps operations

In this section, we provide details about basic operations for BASE24-eps.

### 6.14.1 Starting BASE24-eps

To start BASE24-eps:

1. Start the base processes that are required for the client UI, as shown in Figure 6-49 on page 99:
  - Application Manager (JMX)
  - TDAL
2. Use the client UI to start the other BASE24-eps processes, as shown in Figure 6-50 on page 99.



```

EPS1 @ SC48:/usr/aci/eps/eps1a
> runacijmx
Application Management Server started

EPS1 @ SC48:/usr/aci/eps/eps1a
> runtdal
Process TDAL Started
Process 67702762

```

Figure 6-49 Start initial processes required for the client UI

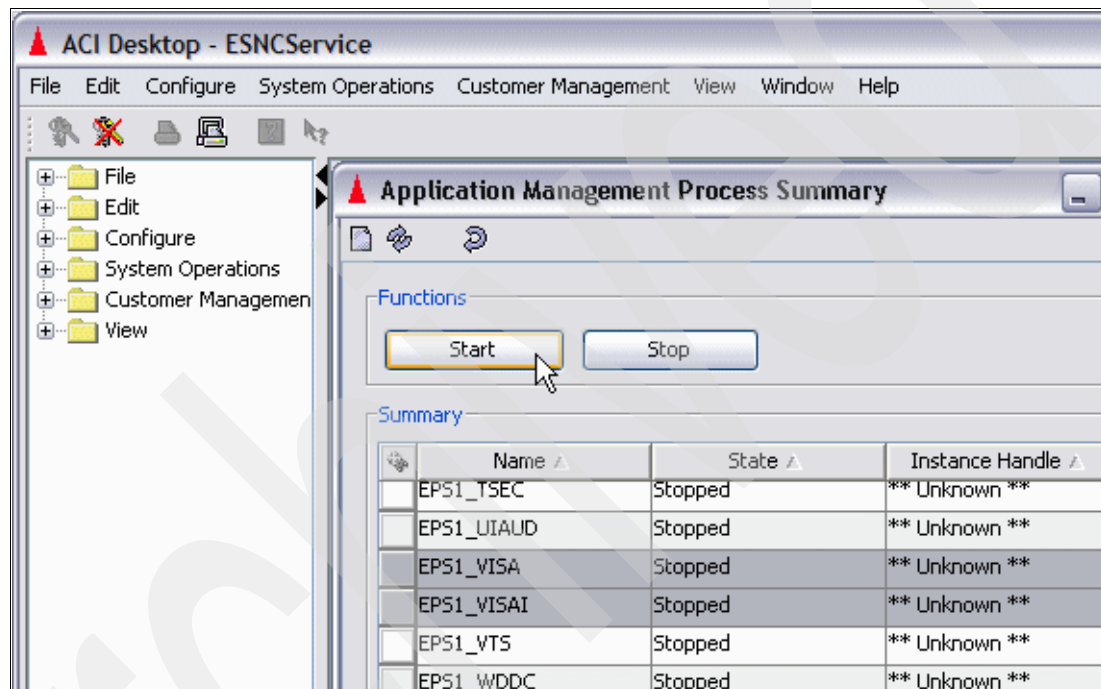


Figure 6-50 Start other BASE24-eps processes with the client UI

Figure 6-51 on page 100 shows a window for checking the BASE24-eps processes.

```

wtsc48oe.itso.ibm.com - PuTTY
EPS1@wtsc48:EPS1:ttyp0010:/usr/aci/eps/eps1a/bin
$ esstatus
*** MQ Series Manager ***
MQGG is running
EPS1.CTXD(25) EPS1.CXATMD(2)

*** DB2 server ***
DB2 connection is established

*** ES Processes for System: EPS1 ***
EPS: silisten is running (PID: 84479416)
EPS: ACIJMX is running (PID: 84479369)
EPS: RFSH is running (PID: 593263)
EPS: EVTLOG is running (PID: 593349)
EPS: SITIMR is running (PID: 593319)
EPS: TTLM is running (PID: 591089)
EPS: JQB is running (PID: 593324)

```

Figure 6-51 Check BASE24-eps status

One advantage in starting the BASE24-eps processes using the client UI is that JMX knows the desired status and restarts a process if it fails.

## 6.14.2 Stopping BASE24-eps

Use the client UI to stop the BASE24-eps processes. Do not stop TDAL.

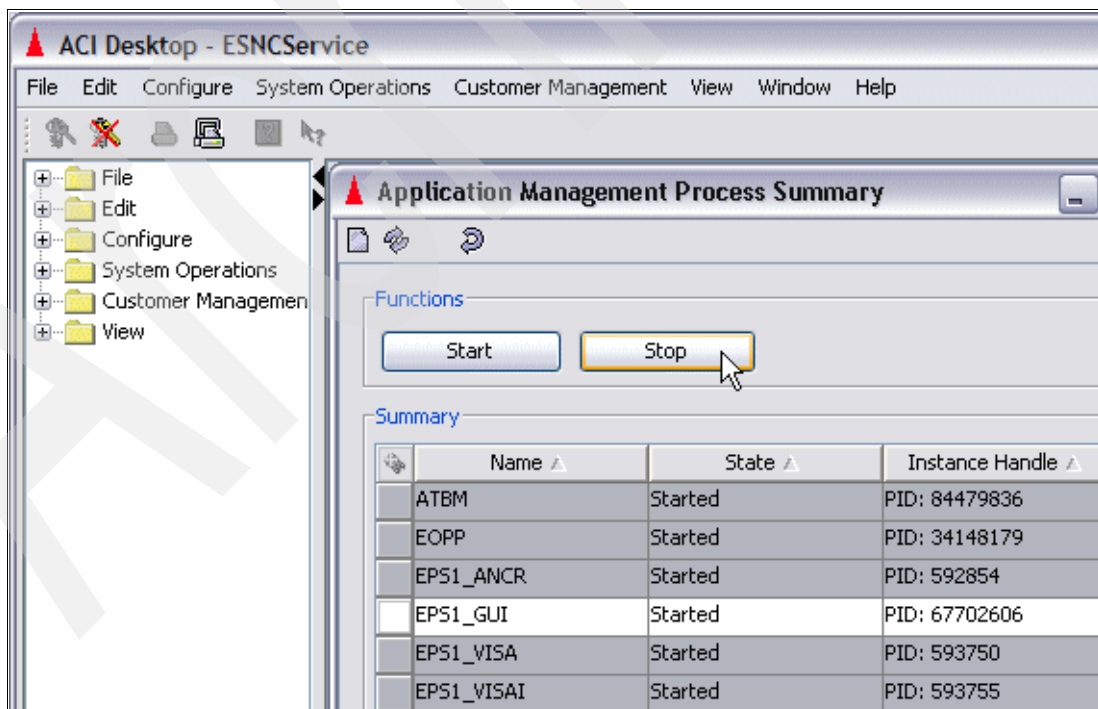


Figure 6-52 Use the UI to stop the BASE24-eps processes

Stop the remaining tasks from a telnet session: stopacijmx and stoptdal. Use **esstatus** and **SDSF** to confirm that all BASE24-eps processes were stopped.

## **BASE24-eps and System z cryptography**

In this chapter, we describe the System z cryptography requirements for BASE24-eps.

The topics that we discuss are:

- ▶ 7.1, “Basics of System z cryptography” on page 102
- ▶ 7.2, “Cryptographic hardware” on page 104
- ▶ 7.3, “BASE24-eps and IBM Crypto Express2” on page 109
- ▶ 7.4, “Preparing data for the BASE24-eps database” on page 113
- ▶ 7.5, “Configuring IBM Crypto in the BASE24-eps UI” on page 114
- ▶ 7.6, “Other issues” on page 125

## 7.1 Basics of System z cryptography

In this section, we discuss the basics of cryptography on System z and z/OS.

### 7.1.1 System z cryptographic facilities

The IBM System z cryptographic hardware provides a rich array of encryption capabilities. The functionality that is available depends on the specific platform and the hardware that was installed.

### 7.1.2 Cryptographic functions

System z cryptographic hardware supports several cryptographic capabilities, which include:

- ▶ Data confidentiality (encrypting/decrypting data)
- ▶ Message integrity (message authentication)
- ▶ Financial functions (protecting PINs associated with credit cards and financial transactions)
- ▶ Key management (security and integrity of keys)

### 7.1.3 Clear key versus secure key

IBM cryptographic hardware uses either *clear key* or *secure key* mechanisms. Regardless of the type of key used, there is no difference in the cryptographic algorithms, so the resulting encrypted data is the same if the key is the same. The difference is the additional protection provided for secure keys within the tamper-resistant hardware. The hardware has tamper-detection technology to protect against various attacks, such as power analysis. If tampering is detected, the card is zeroised, which wipes out the keys so they cannot be compromised.

When a secure key is created, it is encrypted under a master key, and the actual key value is never exposed outside of the secure hardware. If that key needs to leave the secure hardware (for example to be stored in the CKDS), an encrypted version of the key is used. The encrypted key must be decrypted from under the master key before it can be used to encrypt or decrypt data. If the key is to be sent elsewhere, for example to be exchanged with a partner, it can be encrypted under a key encrypting key instead of the master key, but the key will not exist in the clear outside of the secure hardware.

In contrast, a clear key might be seen anywhere it is used or stored, for example, during the key entry process or in memory when in use by an application. Clear keys are useful in some circumstances, such as System SSL, where a key is only used for a short time, and where performance costs must be traded off against the cost of the secure hardware.

### 7.1.4 Symmetric algorithms and keys

Symmetric keys are used with symmetric algorithms, for example, the Data Encryption Standard or DES. Symmetric means that both sides of the process, encryption and decryption, must share the key that is used in the algorithm, which must be the shared secret between the two parties that are involved because anyone who gains knowledge of the key can decrypt the encrypted data.

### 7.1.5 Asymmetric algorithms and keys

Asymmetric or public key algorithms use a pair of keys to protect data. In this case, two different but mathematically related keys are created. One is called the public key and can be made available to anyone who wants to send encrypted data to the owner of the private key. Data that is encrypted using a public key can only be decrypted using the corresponding private key. Anyone who has a copy of the public key can encrypt data, but the key cannot be used to decrypt data that was encrypted using that key. Because the private key must be used to decrypt the data that is encrypted by the public key, the private key must be protected and only accessible by the owner of the key pair. Equally, the owner of the private key can encrypt data, which can only be decrypted by using the public key, for example, this technique can be used to authenticate that encrypted messages originated from the owner of that key because only the corresponding public key can be used to decrypt.

### 7.1.6 Cryptographic software

The Integrated Cryptographic Service Facility or ICSF is the system component that provides the interface to the hardware. As new functions are implemented in hardware, new versions of ICSF are made available to exploit those functions. ICSF is available as a component of z/OS. The most current versions are available at:

<http://www.ibm.com/servers/eserver/zseries/zos/downloads/>

A number of crypto instructions are now available in the CP and can be coded directly in an application. However most of the crypto functions can only be accessed by using the ICSF Application Programming Interfaces (APIs). The API passes the cryptographic request to ICSF, which determines what hardware is available and which device can best service the request. The *ICSF Application Programmer's Guide* provides a table that describes the hardware that is required to support the APIs.

### 7.1.7 Hardware connectivity

In the z890/z990, the crypto architecture, Figure 7-1 on page 104, was changed to move much of the functionality out of the processor to use the PCI (Peripheral Interconnect) bus. The reasons for this change are:

- ▶ *Availability:* Adding functionality to a crypto processor required an outage of the entire machine. PCI cards are hot-pluggable, and so new function can be added by simply plugging the card with the new functionality into the I/O cage.
- ▶ *Scalability:* As workload increases, additional cards can be ordered and installed, without an outage to the LPAR, up to a maximum depending on the type of processor and the mix of cards that are installed. To take advantage of the PCI feature without an outage requires that the LPARs be configured in advance. As part of LPAR configuration, crypto features are defined to be available to the LPAR in the Activation profile. The candidate list says that the device is a candidate to be brought online and the online list controls which devices are brought online at LPAR activation, for example, if crypto slot 1 is in the online list for an LPAR and if a card resides in Slot1, that card is online to the operating system when the LPAR is activated. If crypto slot 2 is included in the candidate list, but no coprocessor is installed, there is no card to be made available to the LPAR. If a card is subsequently installed in that slot after the LPAR is activated, it can be brought online to the LPAR and made available to the operating system without an outage.
- ▶ *Cost:* By moving the secure hardware to the PCI cards the cost could be better managed. The cards use the Self-Timed Interface (STI), so the crypto work routed to them is asynchronous. After ICSF routes the work to the card, the general purpose CPs can

continue to handle other work (crypto and non-crypto) in the system. The PCI cards perform the crypto function and queue the results back to ICSF, which then provides the results back to the calling application.

The PCI implementation also supports User Defined Extensions (UDX), which provides the ability to load customer-specific code into the secure hardware giving the ability to manipulate data securely within the hardware protection of the card. UDXs are custom written and are usually developed in conjunction with IBM.

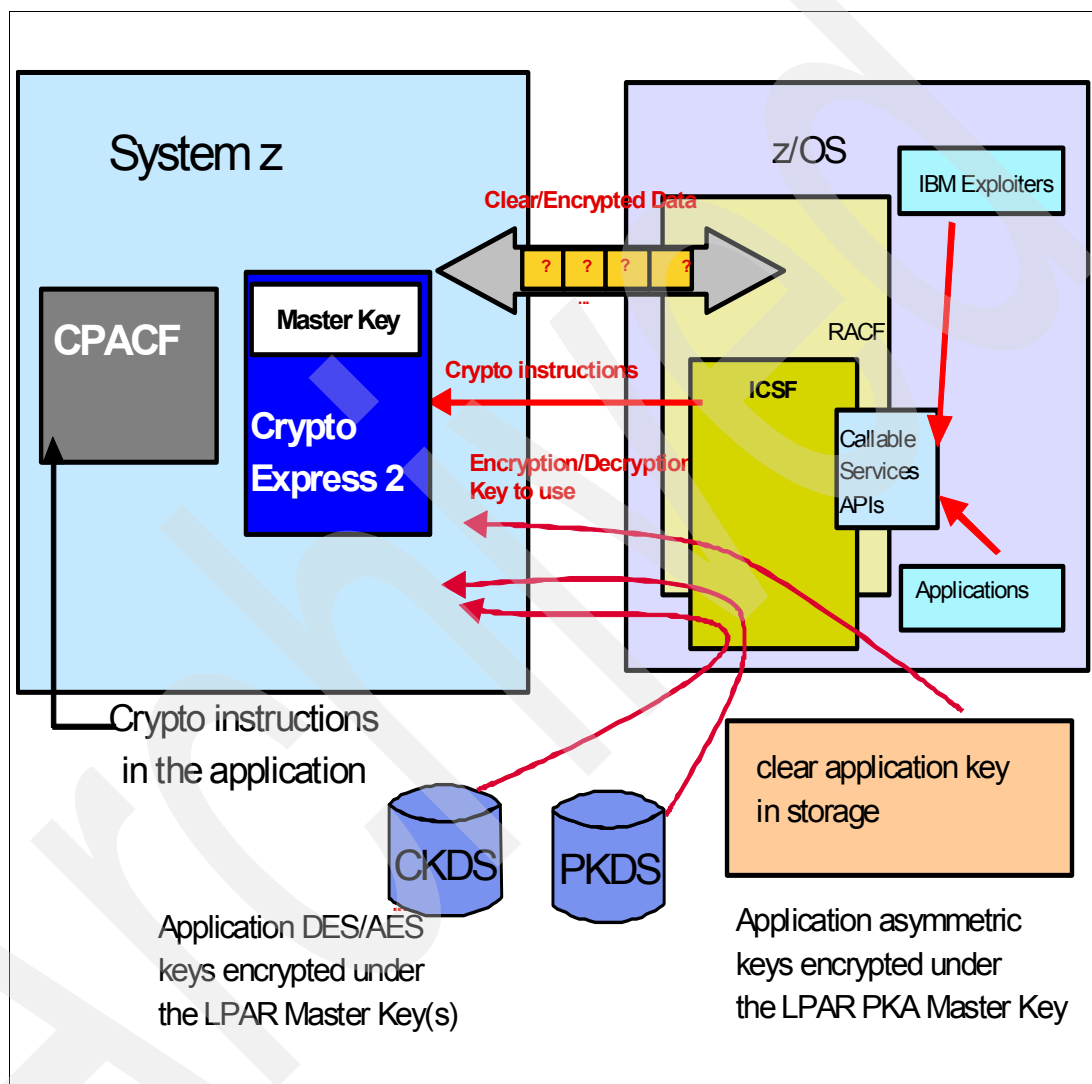


Figure 7-1 Overall view of z/OS crypto

## 7.2 Cryptographic hardware

System z cryptographic hardware has evolved over time since the introduction of the Cryptographic Coprocessor Facility, as shown in Figure 7-2 on page 105.

## System z Crypto evolution

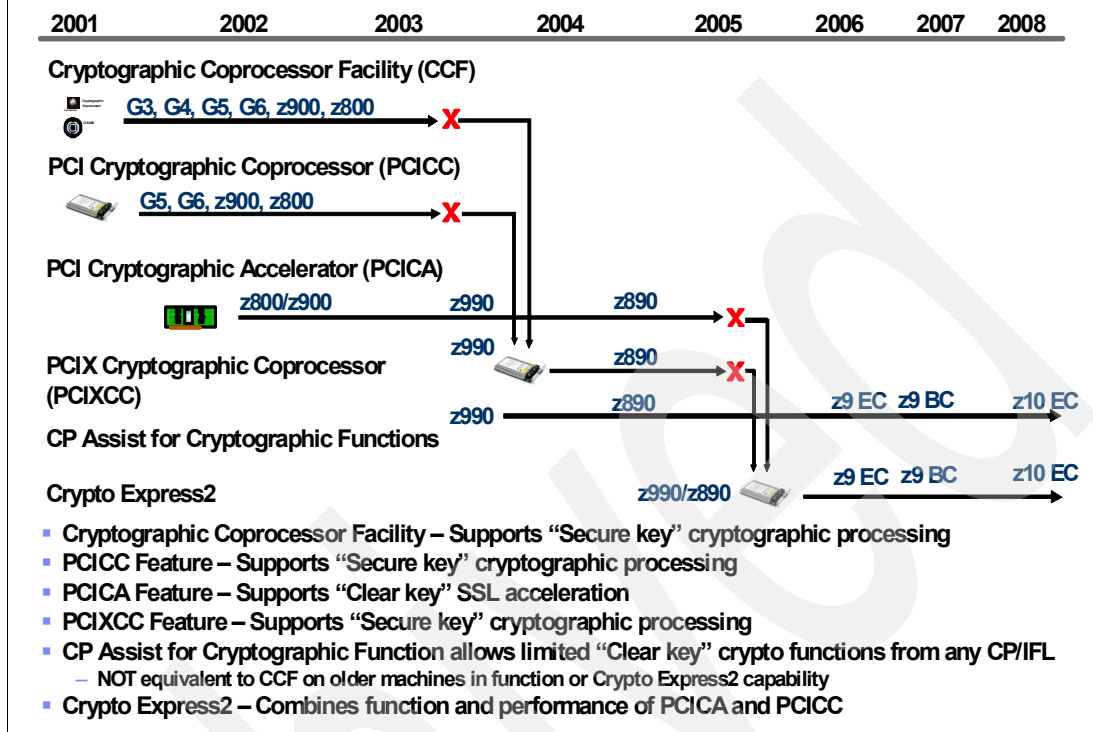


Figure 7-2 System z Crypto evolution

In the next section, we describe some of the most recent cryptographic hardware devices that are available on System z.

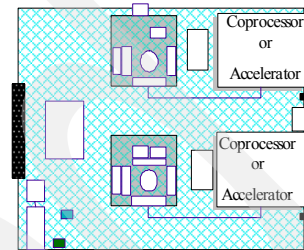
### 7.2.1 z10 Enterprise Class and z10 Business Class

The z10 Enterprise Class machine (z10 EC) was announced on February 26, 2008. Additional cryptographic capabilities were announced for the z10 EC with the announcement of the z10 Business Class (z10 BC) machine on October 21, 2008. The z10 crypto hardware is similar to the z9, but the z10 supports several new crypto functions that makes it easier to manage the crypto devices.

Figure 7-3 on page 106 shows the z10 cryptographic support.

## z10 Cryptographic Support

- **CP Assist for Cryptographic Function (CPACF)**
  - Standard on every CP and IFL
  - **Supports the following algorithms:**
    - DES, TDES, AES-128, AES-192, AES-256
    - SHA-1, SHA-224, SHA-256, SHA 384 & SHA 512
    - Pseudo random Number Generation (PRNG)
    - SHA-1, SHA-256, and SHA-512 are shipped enabled
  - UP to 4096-bit RSA keys
  - Random Number Generation Long (8 bytes to 8096 bits)
- **Crypto Express2**
  - Two features – 1 (z10 BC only) and 2 Coprocessor option (minimum of 2 features)
  - Two configuration modes
    - Coprocessor (default)
      - Federal Information Processing Standard (FIPS) 140-2 Level 4 certified
      - Accelerator (configured from the HMC)
        - Three configuration options (Two for 1 Coprocessor option)
          - Default set to Coprocessor
    - Concurrent Patch
    - Secure Key AES (128, 192 and 256 bit) support
    - Support for 13 through 19 Personal Account Numbers
  - **Dynamic Add Crypto to LPAR**
    - No recycling of LPAR
    - No POR required



Crypto Express2

Figure 7-3 z10 cryptographic support

There are two crypto hardware devices that are available on the z10: the CP Assist for Cryptographic Function (CPACF) is integrated into the PU in the host, and the Crypto Express2 is a PCI device that is installed in the I/O cage.

### CP assist for cryptographic function

The CP assist for cryptographic function (CPACF) provides clear key cryptography and hashing functions on the z10. This crypto engine provides support for clear key, symmetric algorithms DES, TDES, and AES. DES uses single length (8-byte) keys, while TDES can use single, double (16-byte), or triple (24-byte) length keys. The z10 CPACF supports AES keys of 128-bit, 192-bit, or 256-bit lengths. The z10 also supports SHA-1, SHA-256, and SHA-512 hashing algorithms in the CPACF hardware. ICSF extends that hashing support to the full suite of SHA-2 algorithms (adding SHA-224 and SHA-384). Support for AES-192 and AES-256 bit keys and SHA-512 are new on the z10 (not available on the z9 or earlier) and are referred to as *Enhancements to CP Assist for Cryptographic Function*. The SHA-1 algorithm is still widely used and required for certain protocols and applications. The CPACF also provides the ability to generate a random number using a pseudo-random number generator.

The CPACF is accessible through native assembler instructions or through the clear key APIs that are available through ICSF. See the *Principles of Operations* for the specific machine for the assembler instructions and the *ICSF Application Programmer's Guide* for the APIs that are available.

On the z10, there is one CPACF on each core and the CPACF is shared by two Processor Units (or PUs). Each PU that is configured as a CP, IFL, zIIP, or zAAP has access to the CPACF. The CPACF provides synchronous cryptographic operations, so when the CPACF is



processing a cryptographic request, the PU that passes that work to the CPACF is busy and unavailable for other work.

## **Crypto Express2 and Crypto Express2-1P**

The Crypto Express2 (CEX2) on the z10 is a PCI feature that is physically identical to the Crypto Express2 on the z9, z990, and z890; however, the microcode on the z10 provides several new capabilities.

The CEX2 feature includes two cryptographic processors, each of which can be configured independently as either a cryptographic coprocessor (the default) or as a cryptographic accelerator. The Crypto Express2-1P (CEX2-1P) is identical to the Crypto Express2 in functionality; however, it only has a single cryptographic processor and is only supported on the BC model. For availability reasons, if a CEX2 feature will be installed, at least two features must be ordered and a maximum of eight can be installed (depending on what other devices are in the I/O cage).

On the BC model, those two features can be two CEX2-1Ps or two CEX2s or one of each, but two features are required to provide redundancy. The cryptographic functions on the CEX2 are only available through the ICSF APIs. When the CEX2 is configured as an accelerator (CEX2A), it is a clear key device that only supports three cryptographic APIs. The three APIs are public key APIs that rely on very large prime numbers and are very expensive in terms of CPU utilization when implemented in software.

The CEX2A provides only limited cryptographic function, but it supports very high volume PKA workloads. When the CEX2 is configured as a coprocessor (CEX2C) it is a secure key device that requires a master key be loaded. As a coprocessor it supports all of the crypto functions that we previously identified: data confidentiality, message integrity, financial functions, and key management. The CEX2 supports secure key DES and TDES encryption, PKA encryption, and with the October 21, 2008 announcement it supports secure key AES. While the CEX2C supports the same three APIs as the accelerator, it can not drive the APIs as fast.

The CEX2C also supports PIN processing for financial APIs, which is a random number generator and provides the ability to create, delete, update, and store DES, TDES, AES, and PKA keys. The other new hardware support that was announced in October 2008 is the ability to use 13-digit through 19-digit PANs when calculating the VISA Card Verification Value (CVV) or Mastercard Card Verification Code (CVC). Previously the hardware supported the common industry standards of 13-digit, 16-digit, and 19-digit Personal Account Numbers.

The CEX2C on the z10 includes support for several new functions that were made available on the z9 using new microcode. It now supports ISO Format 3 PIN blocks, as defined in the ISO 9564-1 standard, which provides added security by padding the PIN block with random data before it is encrypted rather than padding with predictable values as used in other formats. The CEX2C includes support for RSA keys up to 4096-bits for key management operations, digital signatures, and query services. Retained keys for key management functions are not supported on the CEX2C on the z10. The CEX2C also provides the ability to generate random numbers up to 8192 bytes in length. On the z10 (and the z9), a crypto engine can be dynamically changed (from accelerator to coprocessor or coprocessor to accelerator) using the Hardware Management Console (HMC) without taking an outage of ICSF, the operating system, or the LPAR, which means that as workload changes you can reconfigure the crypto devices to take advantage of the performance and throughput characteristics of each without incurring an outage to implement the change.

## 7.2.2 z9 Enterprise Class and z9 Business Class

The crypto hardware architecture on the z9 is similar to the z10, as shown in Figure 7-4.

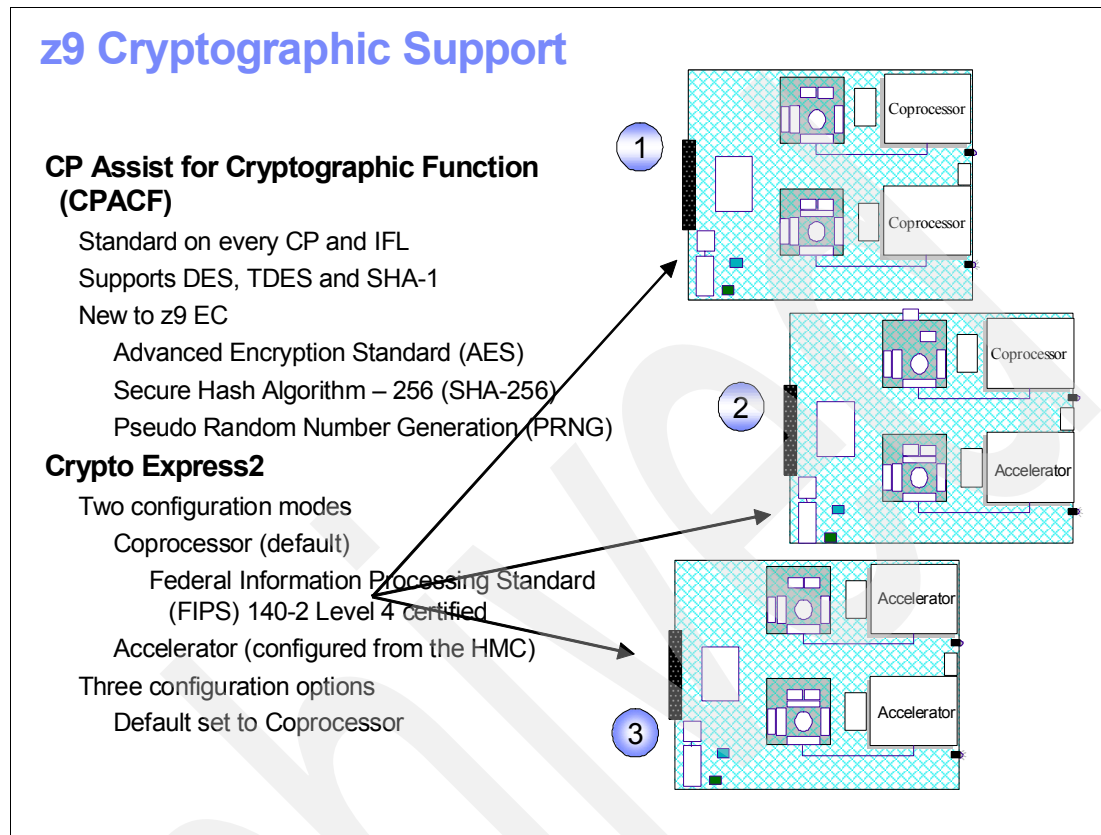


Figure 7-4 Cryptographic support

### CP Assist for Cryptographic Function

The CP Assist for Cryptographic Function provides clear key and SHA functions on the z9 Business Class (BC) and z9 Enterprise Class (EC). The CPACF on the z9 provides additional functionality over the CPACF on the z890/z990, but not all of the functions that are available on the z10. On the z9, the CPACF is part of every PU or processor unit (CP, IFL, zIIP, zAAP). On a 10-way machine, there are 10 CPACFs available. It provides synchronous cryptographic support, so when the CPACF is processing a cryptographic request, the corresponding general purpose PU is busy and cannot be doing other work.

On the z9 machines, the CPACF provides DES/TDES and AES 128-bit clear key symmetric encryption along with supporting the SHA-1 and SHA-256 hashing algorithms, and it also provides a pseudo-random number generator. The functions are available either using native assembler instructions or they can be invoked using the clear key APIs that are available through ICSF. Support for AES-192 and AES-256 clear key is not available in the CPACF on the z9, but is provided by the ICSF software.

### Crypto Express2 and Crypto Express2-1P

The Crypto Express2 feature is installed in the I/O cage on the z9 just like the z10. You can configure the same availability requirements on the z9 as on the z10. The CEX2C supports secure key DES/TDES, PIN processing for financial APIs, a pseudo-random number generator, and it will support the generation and management of keys, including PKA keys up to 2048-bits in length. In November, 2007, the CEX2 was enhanced to provide support for

4096-bit RSA keys using new microcode. With this latest microcode and the latest version of ICSF, these longer keys are supported for key management, digital signatures, and query services. With new microcode on the z9, the CEX2C also supports ISO Format 3 PIN blocks and provides a long random number generator. When the CEX2 is configured as an accelerator (CEX2A), it is a clear key device that supports the same three cryptographic PKA APIs as it supports on the z10, and at the same throughput as the z10.

### 7.2.3 Trusted Key Entry Workstation

The Trusted Key Entry Workstation (TKE) is used for the secure loading and manipulation of master and operational keys into the System z cryptographic hardware. The TKE is a workstation with a secure cryptographic card that attaches to the host using an Ethernet (TCP/IP) connection. It runs an operating system that supports a single application, so there is no external interface to the TKE and no other applications or products can be installed on the TKE. Like the Hardware Management Console (HMC), consider the TKE an appliance whose sole purpose is to protect the key entry process. On the TKE, the customer configures a stand-alone security environment, restricting who can use the key entry application on the workstation and controlling the authority to load both operational and master keys and to manipulate the secure hardware. With the proper authorities a key officer or team of security officers can create and change master keys and operational keys. These keys can then be loaded into the host cryptographic hardware using a secure connection (relying on the Diffie-Hellman key exchange protocol) so that the key values never exist in the clear in an address space or on the network.

The TKE actually incorporates two features, the hardware and the Licensed Internal Code (LIC), which includes the operating system and application software. The TKE version is closely associated with the host platforms that it will support. TKE V5.3 is required to support the z10 BC and can also connect to the z10 EC, z9 EC, z9BC and z890/z990. TKE V5.2 can be used with the z10 EC, z9 EC and z9 BC, z990 and z890. TKE V5.0 was the first to support the z9, but will also support the z800, z900, z890 and z990, as will TKE V5.1.

Beginning with TKE V5 and the z9, up to three TKEs can be ordered per CEC, providing the ability to order additional TKEs for redundancy, disaster recovery, or testing purposes. Optionally, the TKE can store keys, key parts, and provide access authorization using smart cards.

## 7.3 BASE24-eps and IBM Crypto Express2

In this section, we discuss how BASE24-eps utilizes the IBM Crypto Express2 features.

### 7.3.1 Connectivity

The BASE24-eps Transaction Security component uses ICSF to exploit the IBM Crypto Express2 cards. The connection is made through ICSF function calls that are bound into the Integrated Server process executable rather than through interprocess (IPC) request and response messages. Thus, no WebSphere MQ queues or ACI Communication Services for Advanced Networking (ICE-XS) channel protocols need to be configured, as are required for some other outboard cryptographic HSM devices.

### 7.3.2 Key Token data source

The Cryptographic Key Data Set (CKDS) is not mandatory for storing dynamic keys. For implementations with key populations that are greater than approximately 70,000 (for example, for large POS networks), ACI recommends that you store dynamic keys in the Key Token data source (KEY\_TOKEN) instead for performance reasons. KEY\_TOKEN is a DB2 table in our configuration. The value of the KTD\_USAGE Environment attribute indicates whether the KEY\_TOKEN is used. We did not have a large network for our tests and were not generating dynamic keys, so we did not explore the effect of this variable. We set it to FALSE and kept all of our keys in the CKDS.

### 7.3.3 Supported functions

The Crypto Express2 that is used with the Transaction Security component currently supports PIN encryption, PIN translation, PIN verification (for IBM DES and Visa PVV), PVV generation (for IBM DES and Visa PVV), CVV generation and verification, message authentication, EMV functions, dynamic key exchange (working key generation/export and import), and the Automated Key Distribution System. The Master File Key Conversion utility is supported if you are using the Key Token data source (KEY\_TOKEN). In our test scenarios, we were only able to generate workload that used PIN verification because of the limited test card base available.

**Note:** The *ACI BASE24-eps Transaction Security manual* notes that the Crypto Express2 only supports 13, 16, or 19-digit PANs for CVD generation and verification. However, ICSF FMID HCR7751 announcement in October 2008 provides support for 14, 15, 17, and 18-digit PANs. Some other crypto functions, for example, Diebold or NCR PIN functions, are currently not supported, and APACS, AS2805, and Dutch security functions are not currently supported because they are region or country-specific.

### 7.3.4 Key block format

ICSF does not use different key block formats, as some other products do. Therefore, the External Key Block Format field on the Channel Security Configuration window and the Interface Security Configuration window should always be set to *Standard IBM* when using ICSF, which we show in the next section when we describe the setup process in the BASE24-eps desktop UI.

### 7.3.5 ICSF key labels and tokens

ICSF uses a key label for referencing cryptographic keys. Keys are stored in ICSF's key datasets or in the KEY\_TOKEN table in the BASE24-eps database. In either case, a record in these data sources is called a key entry and has a label that is associated with it. Each key entry contains a key token, which is composed of the key value, encrypted under the master key, and control information. Initially, all key tokens for static keys must be loaded into the CKDS. The key tokens for dynamic keys (PIN, MAC, and data keys) are loaded as key exchanges occur. On initial calls to ICSF using a particular dynamic key, the Transaction Security component sets the key identifier to the key label. If the KTD\_USAGE Environment attribute is set to TRUE, the Transaction Security component loads any key tokens returned in the output from the callable service function from ICSF into the KEY\_TOKEN table. On subsequent calls to ICSF using a particular dynamic key, the Transaction Security component retrieves the key token from the KEY\_TOKEN table and sets the key identifier to

the key token in the call. The key identifier is always set to the ICSF key label in calls that use a static key (for example, interface key exchange keys, PIN verification keys, and so on).

### 7.3.6 How BASE24-eps derives ICSF key labels

ICSF key labels are 64-byte character strings, left-justified, and padded on the right with blanks. Special characters and embedded spaces are not allowed. ACI defined unique key labels for each key that can be used in the BASE24-eps database. To ensure uniqueness, BASE24-eps key labels are automatically derived based on the data source's primary key combined with an ACI-defined key type. The Transaction Security component uses an external format of the key label that allows for special characters (for example, underscores) and embedded spaces within the label. The CKDS and PKDS require key labels in the ICSF format, which does not allow for special characters or embedded spaces. Thus, when the Transaction Security component formats a key label as the key identifier in a callable service function, it replaces any special characters or embedded spaces in the BASE24-eps key label with the period character, as shown in Figure 7-5. Multiple embedded spaces are replaced with a single period character while each occurrence of a special character is replaced with a period.

Figure 7-5 is for a channel PIN master key for channel ID PS\_ATM1. The ACI key tag for this type of key is TMK\_PIN. The label is derived from the key tag as a prefix, appended by the particular identifier for the entity associated with the key. Any special characters or blanks are replaced by periods; therefore, the ICSF format of the key label in the function call is as shown in Figure 7-5.

TMK.PIN.PS.ATM1
-----------------

Figure 7-5 ICSF format of key label

Figure 7-6 is for a VISA PVV key, which is formed from the key type prefix of PVK\_VISA, followed by the entity name, in this case the BASE24-eps PIN verification profile that is associated with the card institution, the valid start and end dates, and the key index.

Profile: PINVER_PRFL1
Begin Date: 200902
End Date: 201510
Index: 01

Figure 7-6 VISA PVV data

Figure 7-7 shows the ICSF format of the key label in the function call.

PVK.VISA.PINVER.PRFL1.20090220151001
--------------------------------------

Figure 7-7 ICSF format of key label

When the Transaction Security component calls some ICSF callable services, it specifies one or more key identifiers as parameters to identify the keys to the callable service. The key identifiers can be the ICSF key label or the key token. No actual key values are stored in the BASE24-eps database except for the Host Public Key Security data source (HOST\_PUBLIC\_KEY) and Channel Public Key Security data source (CHAN\_PKEY\_SEC).

Key labels are automatically derived from primary keys and key tokens. The processing steps are:

1. The Transaction Security component retrieves a record from the BASE24-eps database.
2. If the KTD\_USAGE Environment attribute is set to true and the key type is a dynamic key type (PIN, MAC or Data working key), the Transaction Security component attempts to retrieve a key token from the KEY\_TOKEN table using the ACI BASE24-eps key label. (static keys are always stored in the CKDS.) If a valid record is not found, an error is returned and processing ends. Otherwise, processing continues with the next step.
3. If the Transaction Security component was able to retrieve a key token in step 2, it sets the key identifier in the call to ICSF to the key token. Otherwise, it sets the key identifier to the ICSF format of the key label.
4. If the call contains a key label, ICSF retrieves the key token from the CKDS or PKDS. Otherwise, processing continues with the next step.
5. After processing the call, ICSF might return the key token used for the service in the function call.
6. If the KTD\_USAGE Environment attribute is set to true, the Transaction Security component stores the key token from the callable service response in the KEY\_TOKEN; otherwise, it skips this step.

Table 7-1 shows the structure of the key labels used by BASE24-eps for authorization type keys (PIN verification, Card verification). For other key types see the *BASE24-eps Transaction Security manual*.

*Table 7-1 Structure of key labels BASE24-eps uses for authorization type keys*

Name	Position	Length	Description/Values
Key Tag	0	16	A descriptive tag for the type of key (TMK_PIN for a PIN master key or TMK_MAC for a MAC master key). In this case, TMK indicates terminal master key. If the key tag is less than 16 bytes, it is padded to the right with blanks.
Key Locator	17	16	The name of the profile, channel ID, host, interface, or transport key locator that is associated with the key. The key locator is all or part of the primary key to the BASE24-eps data source record for which the key is used. If the key locator is less than 16 bytes, it is padded to the right with blanks.
Begin Date/ Device Type	33	6 or 2	The begin date (in YYYYMM format) for a profile or the two-digit device type for a host public key in the BASE24-eps database. The begin date or device type is part of the primary key to the BASE24-eps data source record for which the key is used.
End Date/ Effective Date	39 or 35	6 or 8	The end date (in YYYYMM format) for a profile or the effective date (in YYYYMMDD format) for a host public key in the BASE24-eps database. The end date or effective date is part of the primary key to the BASE24-eps data source record for which the key is used.
Index	45	2	An index for BASE24-eps data source records that contain two or more keys of the same type.

### 7.3.7 IBM key tokens

An ICSF key token is 64 bytes, composed of the key value and control information. The control information is assigned to the key when ICSF creates the key. The key token can be either an internal key token, an external key token, or a null key token. For a detailed description of the format of a key token, refer to the *z/OS Cryptographic Services ICSF Application Programmers Guide*.

Through the use of key tokens, ICSF can:

- ▶ Support continuous operation across a master key change
- ▶ Control use of keys in cryptographic services

An internal key token is a token that can be used only on the ICSF system that created it (or another ICSF system with the same host master key). It contains a key that is encrypted under the master key. Using an external key token, keys can be exchanged between systems. It contains a key that is encrypted under a key-encrypting key. An external key token contains an encrypted key and control information to allow compatible cryptographic systems to:

- ▶ Have a standard method of exchanging keys
- ▶ Control the use of keys through the control vector
- ▶ Merge the key with other information needed to use the key

An application obtains an external key token by using one of the callable services, such as:

- ▶ Key generate
- ▶ Key export
- ▶ Data key export

A null key token can be used to import a key from a system that cannot produce external key tokens, for example, if you have an 8 or 16-byte key that was encrypted under an importer key, but is not embedded within a token, the encrypted key can be placed in a null key token and then imported into ICSF in operational form.

## 7.4 Preparing data for the BASE24-eps database

A secure key loading facility, such as the Trusted Key Entry (TKE) workstation, is usually used to enter master keys and static cryptographic keys into the Cryptographic Key Data Set (CKDS). If dynamic key management is used for PIN, MAC, and data keys, those keys are initially loaded during the first key exchange. All keys that are stored in the PKDS are loaded dynamically during AKDS processing. The CKDS can also be updated using ICSF callable services.

Because we are running on test systems, we do not use TKE for loading keys.

**Key types:** Some key types, specifically EMV Master Keys and DUKPT Base Derivation Keys (BDKs), can only be loaded using a TKE or similar interface even in a test system, such as the Key Generation Utility Program (KGUP), does not provide support for these key types.

**Key length:** Although you do not enter any keys into the BASE24-eps database when using ICSF, in some cases you must still enter the key length (single, double, or triple) in the BASE24-eps database. Key length is required for determining appropriate key length for dynamic key generation and for ensuring valid MAC type configuration. Key lengths are not required for authorization type keys (PIN Verification, Card Verification, and so on).

With ICSF, you can use either the *key generator utility program (KGUP)* or the key generate callable service to generate DES keys. It is recommended that the KGUP only be used in a test environment because it does not enforce encrypted key values or provide for dual control. The TKE workstation should be used in a production environment because it provides a secure method of adding keys. With KGUP, you can generate key-encrypting keys, PIN keys, data-encrypting keys, data-translation keys, and MAC keys. A master key variant enciphers each type of key that KGUP creates (except for CLRDES and CLRAES keys). After this program generates a key, it stores it in the CKDS where it can be used for online processing.

The key generate callable service creates all types of DES keys. It generates a single key or a pair of keys. Unlike KGUP, however, the key generate service does not store DES keys in the CKDS but returns them to the application program that called it.

### 7.4.1 Key Generator Utility Program

Each key that KGUP generates (except clear key value data-encrypting keys and clear AES keys) exist in the CKDS enciphered under the system's master key. Before the master key enciphers a key, the cryptographic facility XORs the master key with a control vector. A master key combined with a control vector in this way is called a master key variant. A unique control vector exists for each type of key that the master key enciphers. The control vector ensures that a key is only used in the cryptographic functions for which the key is intended, for example, the control vector for an input PIN encryption key ensures that such a key can be used only in PIN translate and PIN verification functions. When a command is issued to KGUP to generate a key, a key label is specified in the command statement. The key label is the method used by the application program to reference a specific key (the key label is stored in the BASE24-eps database instead of the actual key value). In addition to the generation of random key values, the KGUP utility also provides a means of encrypting known values (clear keys) and storing the results in the CKDS.

## 7.5 Configuring IBM Crypto in the BASE24-eps UI

The BASE24-eps desktop UI defines the IBM crypto hardware to BASE24-eps. Individual cards are not defined, only the type of device and some installation parameters.

### 7.5.1 Configuration steps

To configure the IBM Crypto in the BASE24-eps UI:

1. Set up the CEX2C on System z and ICSF.

In our setup, we had two LPARs on the same z9 sharing access to two CEX2 cards, one configured with two coprocessors and the other with one coprocessor and one accelerator, as shown in Figure 7-8 on page 115. Two cards must be ordered and installed for availability.



COPROCESSOR	SERIAL NUMBER	STATUS
-----	-----	-----
E01	94000264	ACTIVE
E02	95001434	ACTIVE
E03	95001437	ACTIVE
F00		ACTIVE

Figure 7-8 CEX2 coprocessors and accelerator

ICSF on each system had its own CKDS and PKDS and ran in non-compatibility mode.

- Using the ACI desktop, select **Configure** → **Transaction Security** → **Security Device**. Figure 7-9 shows the configuration.

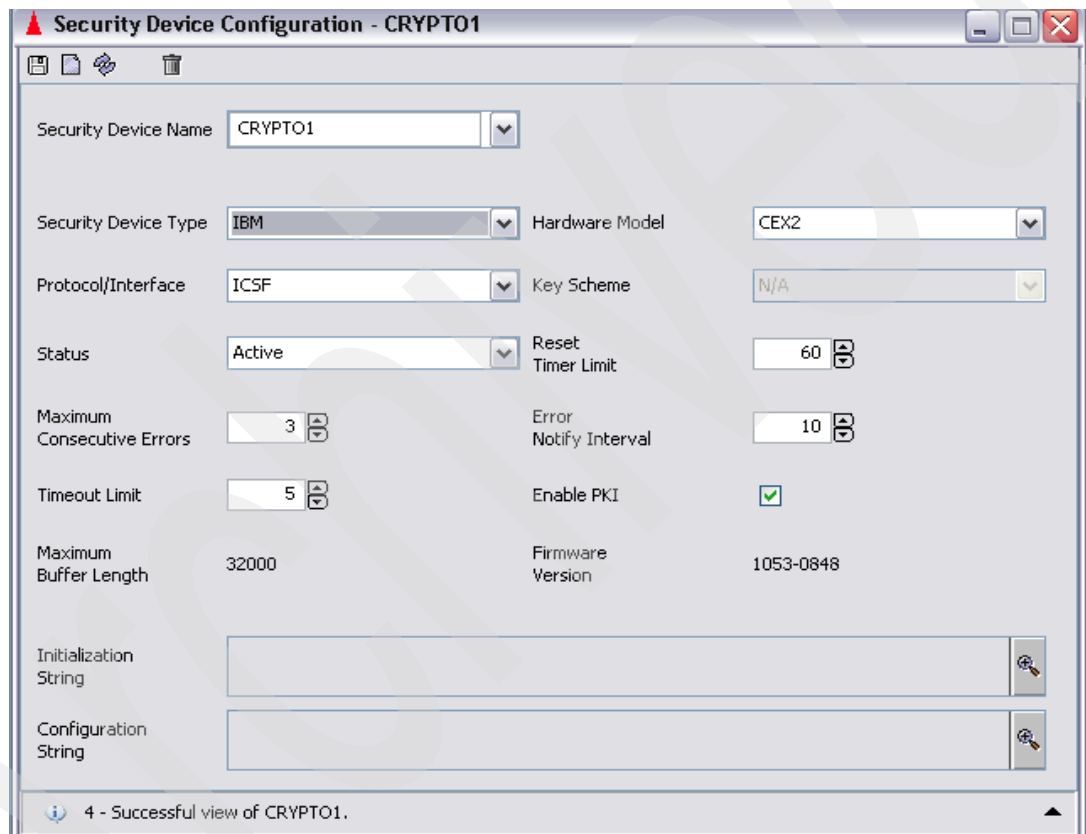


Figure 7-9 Security Device Configuration

Configure the parameters for one IBM ICSF entry on the Security Device Configuration window. Only one type of device can be used in any one environment. Figure 7-9 shows that we named the device CRYPTO1 and selected the parameters as shown. ICSF is the only Protocol/Interface value that is available for IBM crypto cards. Hardware models CEX2 and PCIXCC are available. The various numeric values refer to BASE24-eps error and timeout values, and we left these unchanged. IBM crypto devices also have no Initialization or Configuration string.

- Using the ACI desktop, access **System Operations** → **Environment Configuration**.
- Configure the following optional Environment attributes to override their default values on the Environment Configuration window, if desired. We left these to default:
  - KTD\_USAGE

- PIN\_LGTH\_ERR\_RETRY\_IND
- PIN\_VRFY\_FAIL\_RETRY\_IND
- PIN\_XLATE\_DISK\_READ
- TSEC\_STARTUP\_MSGS

5. Configure the BASE24-eps Interfaces to define their Security Configuration. Access **Configure** → **Transaction Security** → **Interface**.

Figure 7-10 through Figure 7-12 on page 118 show how we configured the interface simulating our VISA acquirer interface.

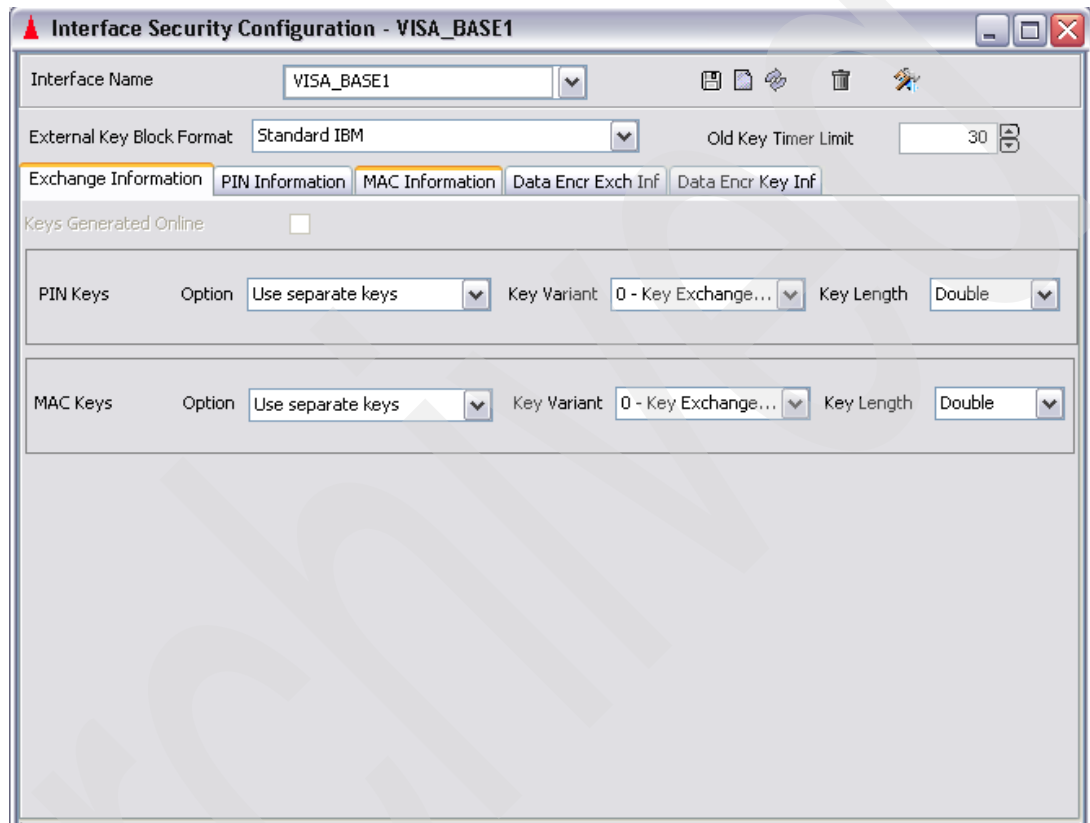


Figure 7-10 Visa acquirer interface definition

Figure 7-10 shows that the External Key Block Format is set to *Standard IBM*, which is the only value that is available for ICSF.

Figure 7-11 on page 117 shows that for this interface we selected ISO Format 0 PIN blocks, also known as VISA format 1, double length keys, and a key index of 1. The PIN PAD character is always set to F for this format.

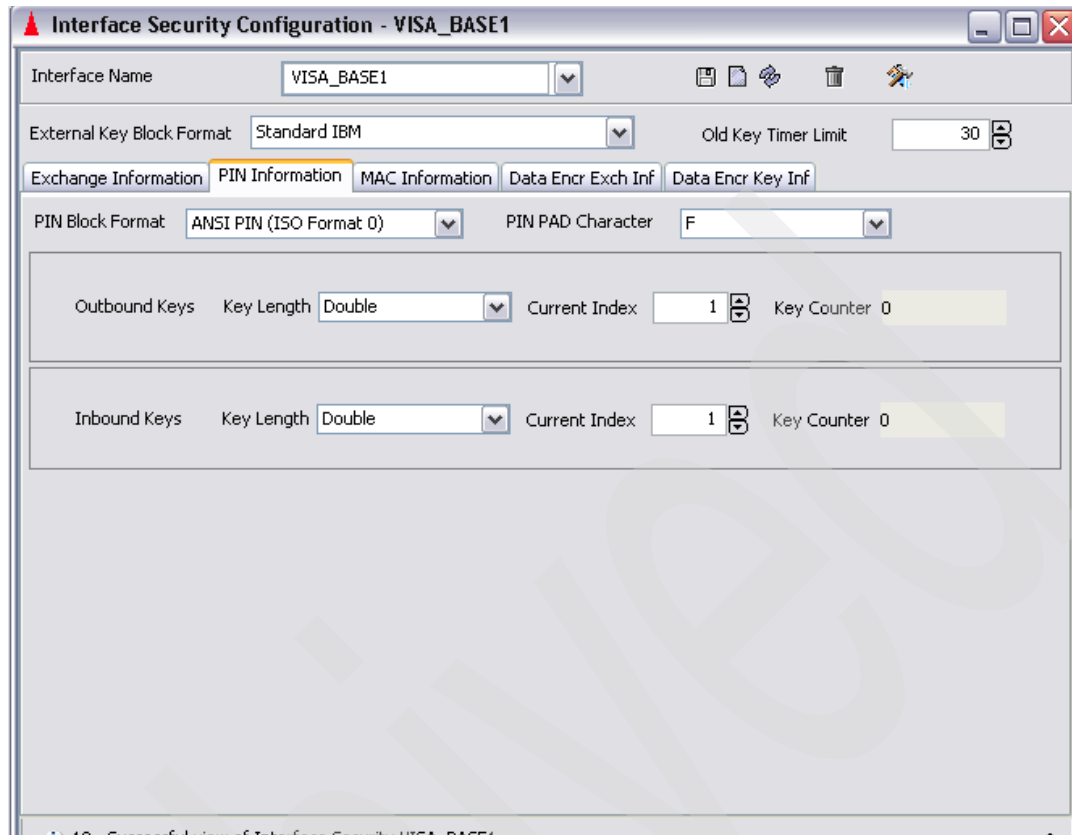


Figure 7-11 Pin Block settings

Figure 7-12 on page 118 through Figure 7-14 on page 120 show the settings for the other security tabs. In our scenarios, we are not using these features, but they are shown for completeness and to show the different choices that must be made.

**Interface Security Configuration - VISA\_BASE1**

Interface Name:

External Key Block Format:  Old Key Timer Limit:

Exchange Information | PIN Information | **MAC Information** | Data Encr Exch Inf | Data Encr Key Inf

MAC Option:  MAC Length:

Outbound Keys: Key Length:  Current Index:  Key Counter:

Inbound Keys: Key Length:  Current Index:  Key Counter:

Figure 7-12 Mac information

**Interface Security Configuration - VISA\_BASE1**

Interface Name:

External Key Block Format:  Old Key Timer Limit:

Exchange Information | PIN Information | MAC Information | **Data Encr Exch Inf** | Data Encr Key Inf

Exchange Keys Option:  Key Variant:  Key Length:

Initialization Vector

	Key
1	
2	

Figure 7-13 Data Encr Exch Inf settings

**Interface Security Configuration - VISA\_BASE1**

Interface Name:

External Key Block Format:  Old Key Timer Limit:

Exchange Information | PIN Information | MAC Information | Data Encr Exch Inf | **Data Encr Key Inf**

DES Mode:  Data Type:  Encryption Layers:

Inbound Keys Key Length:  Current Index:

Outbound Keys Key Length:  Current Index:

Figure 7-14 Data Encr Key Inf settings

In Figure 7-15 on page 121 we begin to set up an Authentication Profile to be used by the card institutions.

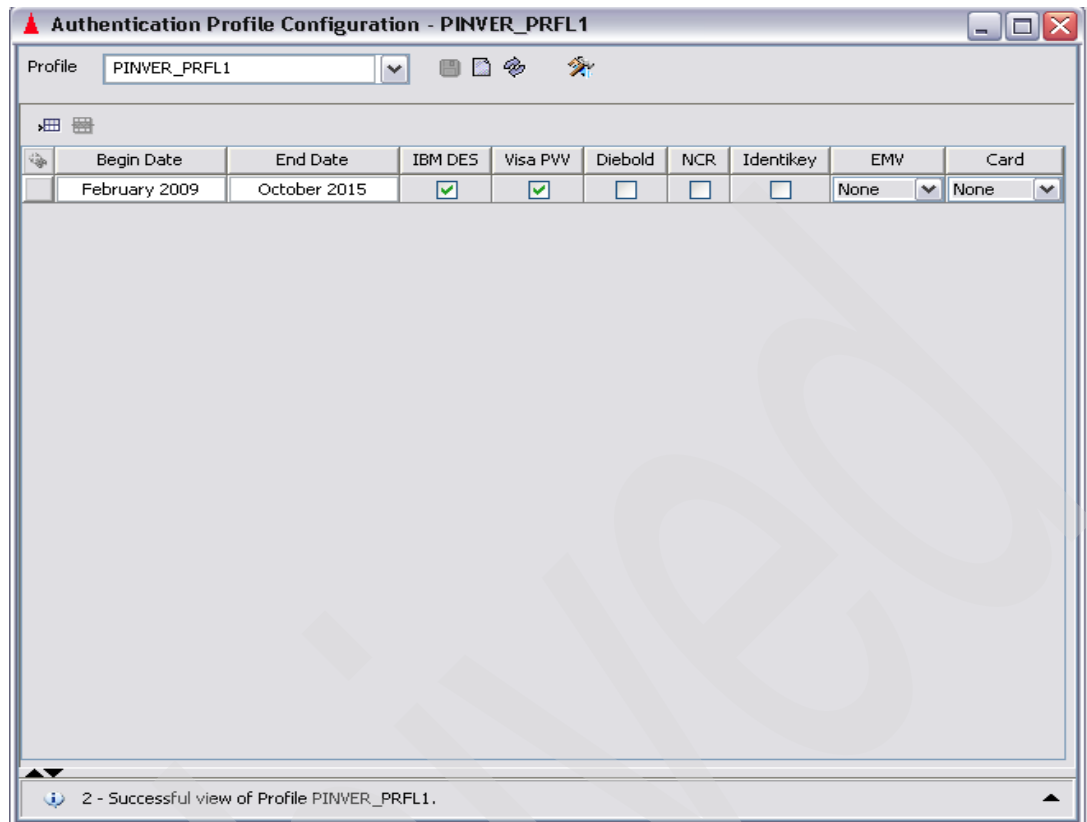
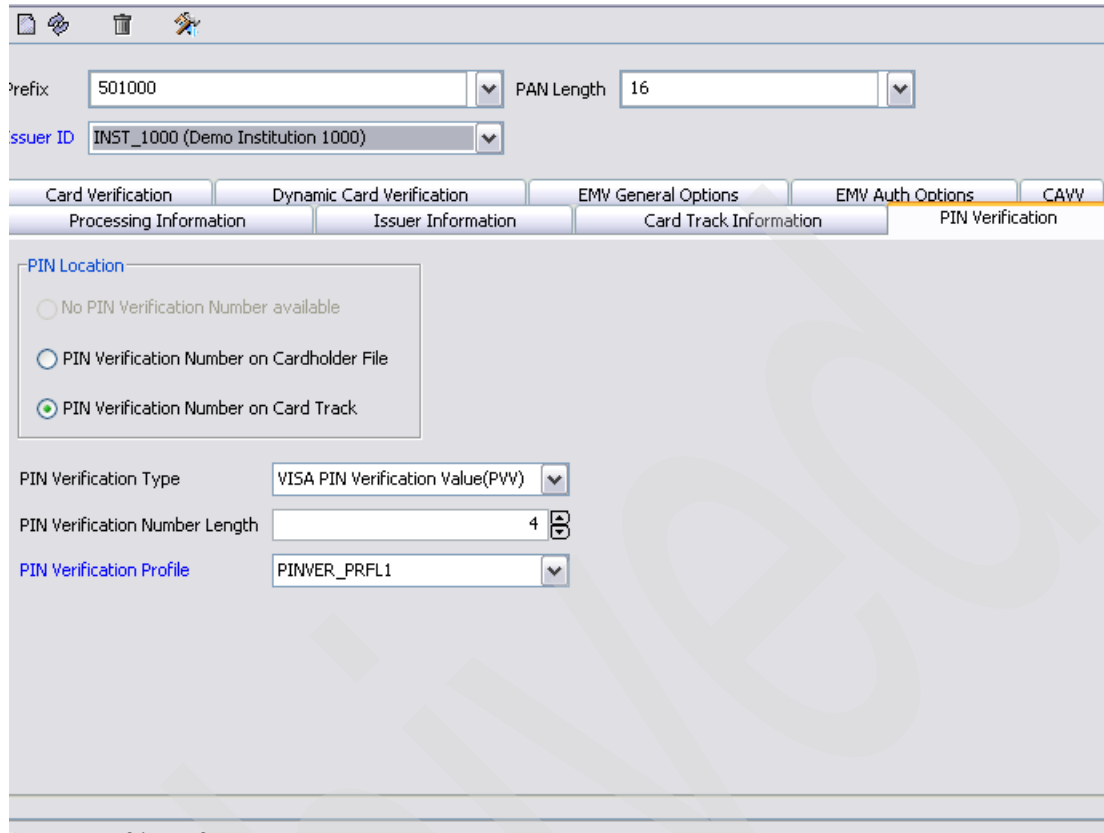


Figure 7-15 PIN Verification Profile screen

Figure 7-16 on page 122 we defined the PIN verification and Card Track information for each of the institutions for On-Us transactions.



Prefix: 501000 PAN Length: 16

Issuer ID: INST\_1000 (Demo Institution 1000)

Card Verification | Dynamic Card Verification | EMV General Options | EMV Auth Options | CAVV

Processing Information | Issuer Information | Card Track Information | PIN Verification

**PIN Location**

☐ No PIN Verification Number available

☐ PIN Verification Number on Cardholder File

☒ PIN Verification Number on Card Track

PIN Verification Type: VISA PIN Verification Value(PVV)

PIN Verification Number Length: 4

PIN Verification Profile: PINVER\_PRFL1

Figure 7-16 settings

Figure 7-17 on page 123 shows that for this card institution we specified that the PIN verification value was on the card TRACK2 data, had a length of 4, and also specified the associated PIN verification profile.



Figure 7-17 TRACK2 data settings

Figure 7-17 shows the values that we specified for the TRACK2 data, denoting the locations for the various data fields.

## 7.5.2 ICSF key definitions

After we defined the hardware, the interfaces, the authentication profile, and the ATM channels, we loaded the static keys that are related to these definitions. Because this was a test environment and we had control over the keys we were using in our ASSET test simulator, we loaded the key values using clear key statements in the ICSF KGUP utility.

One PIN verification key was needed for the Authentication Profile, as shown in Figure 7-18.

```
ADD TYPE(PINVER),
  CLEAR KEY(0123456789ABCDEF,FEDCBA9876543210),
  LAB(PVK.VISA.PINVER.PRFL1.20090220151001)
```

Figure 7-18 Add PIN verification key

Input and output PIN encryption keys were added for the VISA issuer and acquirer interfaces. The VISA issuer interface name is VISA\_ISS and the acquirer is VISA\_BASE1. The underscores are translated into periods in the ICSF key label. See Figure 7-19 on page 124 through Figure 7-21 on page 124.

```

ADD TYPE(IPINENC),
  CLEAR KEY(0123456789ABCDEF,FEDCBA9876543210),
  LAB(KPE.IN1.VISA.ISS)
ADD TYPE(IPINENC),
  CLEAR KEY(0123456789ABCDEF,FEDCBA9876543210),
  LAB(KPE.IN2.VISA.ISS)
ADD TYPE(OPINENC),
  CLEAR KEY(0123456789ABCDEF,FEDCBA9876543210),
  LAB(KPE.OUT1.VISA.ISS)
DELETE LAB(KPE.OUT2.VISA.ISS), TYPE(IPINENC)
ADD TYPE(OPINENC),
  CLEAR KEY(0123456789ABCDEF,FEDCBA9876543210),
  LAB(KPE.OUT2.VISA.ISS)

```

*Figure 7-19 Issuer input and output PIN encryption keys*

```

ADD TYPE(IPINENC),
  CLEAR KEY(0123456789ABCDEF,FEDCBA9876543210),
  LAB(KPE.IN1.VISA.BASE1)
ADD TYPE(IPINENC),
  CLEAR KEY(0123456789ABCDEF,FEDCBA9876543210),
  LAB(KPE.IN2.VISA.BASE1)

```

*Figure 7-20 Acquirer input and output PIN encryption keys*

Finally, keys were added for all of the ATM channels. Only one set is given here for one ATM (PS\_ATM1). Similar definitions were created for all of the ATMs we were simulating. Again, note that undercores are translated into periods in the key label.

```

ADD TYPE(EXPORTER), NOCV,
  CLEAR KEY(0123456789ABCDEF,FEDCBA9876543210),
  LAB(TMK.PIN.PS.ATM1)
ADD TYPE(EXPORTER), NOCV,
  CLEAR KEY(0123456789ABCDEF,FEDCBA9876543210),
  LAB(TMK.DATA.PS.ATM1)
ADD TYPE(EXPORTER), NOCV,
  CLEAR KEY(0123456789ABCDEF,FEDCBA9876543210),
  LAB(TMK.MAC.PS.ATM1)
ADD TYPE(IPINENC),
  CLEAR KEY(0123456789ABCDEF,FEDCBA9876543210),
  LAB(TPK.IN.PS.ATM1)
ADD TYPE(MAC),
  CLEAR KEY(0123456789ABCDEF,FEDCBA9876543210),
  LAB(TAK.IN.PS.ATM1)
ADD TYPE(DATA),
  CLEAR KEY(0123456789ABCDEF,FEDCBA9876543210),
  LAB(TEK.BOTH.PS.ATM1)

```

*Figure 7-21 ATM keys*

## 7.6 Other issues

Setting up ICSF is straightforward, and there are no particular environmental requirements for BASE24-eps. The format of the key labels that are used by BASE24-eps are fully documented in the *BASE24-eps Transaction Security Manual, R1.0v08.2, Jun-2008* in section 15 (Security Devices), subsection IBM Crypto Express2 HSM. This subsection describes the format of key labels for each supported key type as well as examples of how the dynamic portion of the key label is formed.

We used the BASE24-eps trace facility to show which key BASE24-eps was attempting to retrieve, as it constructed the appropriate label at various points in the PIN verification process.

One issue we encountered was when we initially set the KTD\_USAGE environment variable (see above) to TRUE because we wanted to store the key labels in BASE24-eps, but this did not work for our static keys, so we reverted to the default (FALSE) and kept all of the keys in the CKDS.

**Note:** The Key Token Data Source is used only for dynamic key storage. All static keys (Terminal Master Keys (non-AKDS), Key Exchange Keys, Authorization Keys (PIN Verification, EMV, Card Verification, and so on) are only stored in the CKDS. The KTD, when activated, is used to store PIN, MAC, and Data working keys that are exchanged on-line with an external entity.

Archived

## Achieving high availability on z/OS

In this chapter, we show the setup that is required to implement BASE24-eps in a multi-system configuration that provides high availability.

The topics that we discuss in this chapter are:

- ▶ 8.1, “Overview” on page 128
- ▶ 8.2, “BASE24-eps configuration for high availability” on page 129
- ▶ 8.3, “z/OS considerations” on page 130
- ▶ 8.4, “BASE24-eps considerations” on page 133
- ▶ 8.5, “Starting BASE24-eps on SC67” on page 147
- ▶ 8.6, “Running workload on both systems” on page 150

## 8.1 Overview

High availability is the ability of an appropriately configured system (or application) to continue to function or provide its services even when some components are not available, whether that be through failure or planned outage.

For BASE24-eps we achieve high availability through the use of System z technology (hardware and software) and multiple instances of the application.

We make use of:

- ▶ Multiple z/OS systems in a Parallel Sysplex
- ▶ Multiple instances of the application
- ▶ DB2 data sharing
- ▶ IBM WebSphere MQ queue sharing
- ▶ Sysplex distributor

A z/OS Parallel Sysplex is a collection of multiple z/OS systems that are coupled together by shared infrastructure, providing the capability to share workload across multiple systems and multiple processors. A key component of this is the coupling facility which is a special LPAR on System z processors that allows the systems in the sysplex to share data. All systems in the sysplex are also sharing a common time source (clock). For further information, see *z/OS Parallel Sysplex Overview: An Introduction to Data Sharing & Parallelism*, SA22-7661.

DB2 data sharing enables multiple instances of an application (running on separate z/OS systems) to read from and write to the same DB2 data concurrently. The DB2 subsystems that access shared data are collectively referred to as a DB2 data sharing group. A DB2 subsystem that belongs to a data sharing group is a member of that group. For further information about DB2 data sharing, see *DB2 for z/OS: Data Sharing in a Nutshell*, SG24-7322.

Similarly, a MQ queue sharing group consists of multiple MQ queue managers that share MQ object definitions and enable multiple instances of an application (running on separate z/OS systems) to read (get) and write (put) messages from shared local queues. A MQ queue sharing group also makes use of DB2 data sharing to share the definition of MQ objects across the members of the queue sharing group.

Sysplex Distributor is a function in z/OS Communications Server that allows an IP workload to be distributed to multiple server instances within the sysplex. A device on the IP network sees the sysplex cluster (DVIPA) as one IP address, regardless of the number of systems in the cluster. Sysplex Distributor is the internal decision-maker for application workload balancing (where the workload is TCP/IP connections). Workload can be distributed to multiple server application instances without requiring changes to network configuration. For further information, see *Communications Server for z/OS V1R9 TCP/IP Implementation Volume 3*, SG24-7534-00.

While Sysplex Distributor provides distribution of workload consisting of TCP/IP connections, distribution of actual financial transaction messages is provided by MQ Shared Queues. No matter which TCP/IP connection a message arrives on, it is placed on a common shared queue and can be processed by an available BASE24-eps Integrated Server process in any LPAR. So even if TCP/IP connection balancing results in a disproportionate number of messages being received by one LPAR, the distribution of messages over the shared queue results in most of the associated work being done in the 'best' available LPAR.

For our purposes, the target z/OS environment was already set up with these facilities in place for us to use.

## 8.2 BASE24-eps configuration for high availability

In the BASE24-eps configuration in Figure 8-1, both instances of the application are processing work. Both systems are sharing access to the DB2 database and both have access to the shared MQ queues. Incoming TCP/IP connections are distributed across the ICE-XS processes on both SC48 and SC67.

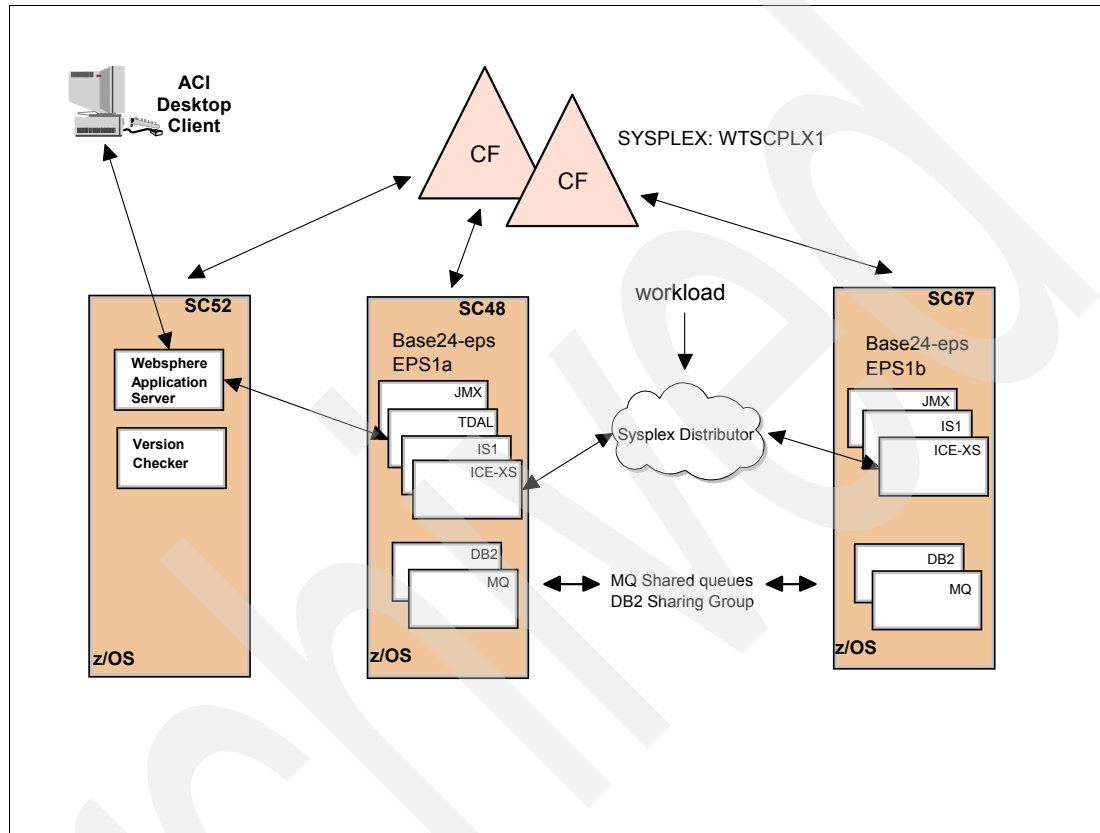


Figure 8-1 Multi-system (high availability) BASE24-eps configuration

A failure in a component (such as DB2 or MQ) on system SC48 (or the failure of the whole system) results in continued service as the BASE24-eps instance on SC67 continues to process work. A failure in a network component sees TCP/IP connections (when re-driven) being directed to the ICE-XS processes on the surviving system.

System SC67 could be on a different System z processor and could even be in a different physical location (subject to the current distance limitations for a sysplex). The high availability configuration could be further enhanced by using System z technology, such as GDPS Hyperswap capability. For further information about GDPS, see *GDPS Family - An Introduction to Concepts and Capabilities*, SG24-6374.

Figure 8-1 shows WebSphere Application Server in a dedicated LPAR, SC52. While some customers might prefer such a deployment, ACI and IBM recommend that WebSphere be deployed in two or more LPARs, each of which contains an instance of the DB2 sharing group.

## 8.3 z/OS considerations

In this section, we discuss the configuration of software that we used in our BASE24-eps test environment.

### 8.3.1 DB2

Our BASE24-eps DB2 objects were originally defined to the DB2 subsystem on system SC48, which is a member of a DB2 Sharing Group D9GG, shown in Figure 8-2. This group is comprised of DB2 subsystems on each of SC48 and SC67 with shared access to the DB2 tables using Parallel Sysplex facilities.

```
DSN7100I  -D9G1 DSN7GCMD
*** BEGIN DISPLAY OF GROUP(DB9GU  ) GROUP LEVEL(910) MODE(N )
          PROTOCOL LEVEL(2)  GROUP ATTACH NAME(D9GG)

-----
DB2      DB2 SYSTEM  IRLM
MEMBER  ID  SUBSYS  CMDPREF  STATUS  LVL NAME  SUBSYS  IRLMPROC
-----
D9G1     1  D9G1   -D9G1   ACTIVE  910 SC48   I9G1    D9G1IRLM
D9G2     2  D9G2   -D9G2   ACTIVE  910 SC67   I9G2    D9G2IRLM
-----

SCA  STRUCTURE SIZE:      8192 KB, STATUS= AC,   SCA IN USE:      4 %
LOCK1 STRUCTURE SIZE:      8192 KB
NUMBER LOCK ENTRIES:      2097152
NUMBER LIST ENTRIES:      9364, LIST ENTRIES IN USE:      57
*** END DISPLAY OF GROUP(DB9GU  )
DSN9022I  -D9G1 DSN7GCMD 'DISPLAY GROUP ' NORMAL COMPLETION
```

Figure 8-2 DB2 Datasharing group

Nothing further is required for access to the DB2 tables from our second system SC67.

### 8.3.2 IBM WebSphere MQ

The MQ Queue Managers, MQG1 on SC48 and MQG2 on SC67 are defined in a queue sharing group MQGG, as shown in Figure 8-3.

```
-MQG1 DISPLAY GROUP
CSQ5100I -MQG1 DISPLAY GROUP report ...
CSQ5102I -MQG1 Queue managers in group MQGG 300

-----
Name  Num  Prefix  Status  Ver  DB2  Connection
-----
MQG1   1  -MQG1  ACTIVE   600  D9G1  ACTIVE
MQG2   2  -MQG2  ACTIVE   600  D9G2  ACTIVE
-----

End of queue managers report
CSQ9022I -MQG1 CSQ5DQSG ' DISPLAY GROUP' NORMAL COMPLETION
```

Figure 8-3 MQ Queue Sharing group



Unlike DB2, MQ objects, such as a QLOCAL queue definition, can either be local to a particular queue manager (and therefore only visible on that system) or they can be defined as shared in which case they are visible to all systems in the sharing group. In our particular case, a message written to a shared queue is available to be read by BASE24-eps on either SC48 or SC67.

Our first install of BASE24-eps into a single system SC48 resulted in all of the MQ definitions being made as local (non-shared). Our high-availability configuration requires shared queues to be defined.

The definition for a shared queue requires specifying CFSTRUCT and QSGDISP(SHARED). You must delete and re-define any existing queue definitions to be able to specify these attributes.

CFSTRUCT is used to specify the name of the coupling facility structure, which is used to store messages. We used APPLICATION1 as the name, as shown in Figure 8-4.

```

SC48
File Edit View Communication Actions Window Help
Menu Utilities Compilers Help

BROWSE      B24EPS.MQ.CHANGES(DEFshr) - 01.02
Command ==>

***** Top of Data *****
//EPS1# JOB (6230), 'MQ DUAL', CLASS=C,
// MSGCLASS=X, REGION=1024K, MSGLEVEL=(1,1)
//*JOBPARM SYSAFF=(*)
//*
//CSQ EXEC PGM=CSQUTIL, PARM='MQGG'
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
  COMMAND DDNAME(CMD)
/*
//CMD DD *

DEFINE REPLACE -
  QLOCAL('EPS1.AGUI') -
  QSGDISP(SHARED) -
  STGCLASS('DEFAULT') -
  CFSTRUCT('APPLICATION1') -
  CLUSTER(' ') -
  CLUSNL(' ') -
  DESCR('Command Queue for ATM GUI connection') -
  PUT(ENABLED) -
  DEFPRTY(5) -
  
```

Figure 8-4 Changes for shared queues

The structure is defined in the sysplex CFRM policy and results in the creation of the structure inside of the coupling facility.

**Messages:** When messages are put onto a shared queue, they are placed directly onto the input queue of a local recipient that has an MQGET posted on that shared queue. If there is no local recipient with an MQGET posted on that shared queue, messages are stored in a coupling facility structure until they are retrieved by an MQGET (read).

### 8.3.3 Sysplex Distributor

To enable Sysplex Distributor for BASE24-eps, we defined the TCP/IP stack on SC48 as the primary Sysplex Distributor and the TCP/IP stack on SC67 as the backup. Figures 8-5 and 8-6 highlight the key statements added to the TCP/IP configuration for SC48 and SC67.

```
GLOBALCONFIG
  NOSEGMENTATIONOFFLOAD

IPCONFIG
  DATAGRamfwd
  DYNAMICXCF 10.1.100.48 255.255.255.0 1
  SYSPLEXRouting
  IGNORERedirect
  SOURCEVIPA

VIPADYNAMIC
  VIPADefine MOVE IMMEDIATE 255.255.252.0 9.12.4.108
  VIPADistribute DEFINE DISTMETHOD BASEWLM 9.12.4.108
    PORT 8014 8016 8017 8020 8100 8130
    DESTIP 10.1.100.48 10.1.100.67
  ENDVIPADYNAMIC
```

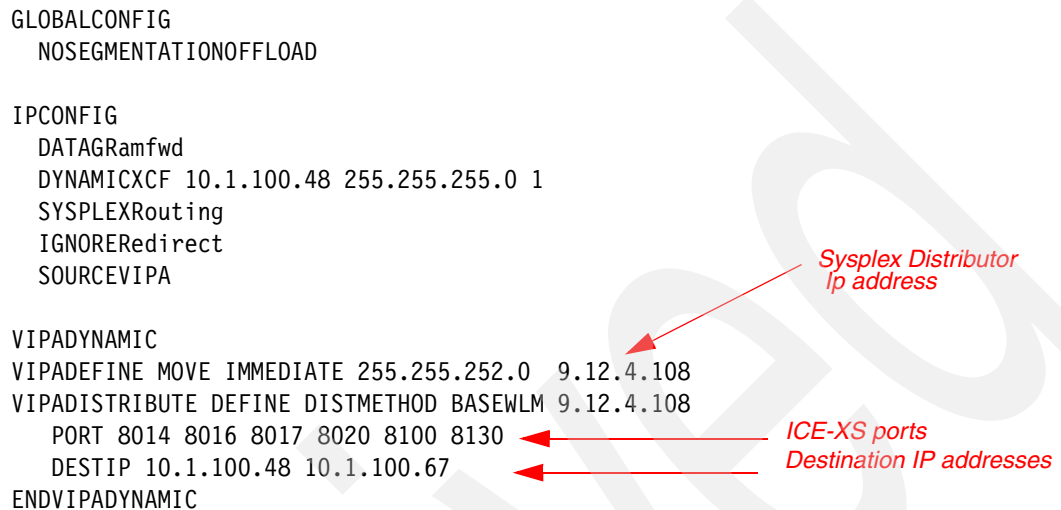


Figure 8-5 Primary Sysplex Distributor TCP/IP definitions for SC48

```
IPCONFIG
  DATAGRamfwd
  DYNAMICXCF 10.1.100.67 255.255.255.0 1
  SYSPLEXRouting
  IGNORERedirect

VIPADYNAMIC
  VIPAbackup MOVEABLE IMMEDIATE 255.255.252.0 9.12.4.108
  ENDVIPADYNAMIC
```

Figure 8-6 Backup Sysplex Distributor TCP/IP definitions for SC67

Incoming connections to the Sysplex Distributor address (9.12.4.108) on any of the defined ICE-XS ports is directed to one of our destination systems (10.1.100.48 or 10.1.100.67) using XCF.

We used the BASEWLM distribution, which is the default for Sysplex Distributor. BASEWLM provides capacity recommendations in the form of weights for each system. WLM assigns a relative weight to each system in the sysplex, with the highest weight going to the system with the most available CPU capacity.

In the event of an LPAR or TCP/IP outage on SC48, the Sysplex Distributor moves to the backup stack on SC67. This takeover by the SC67 stack is nondisruptive for all existing connections.

### 8.3.4 Shared filesystem

Our z/OS sysplex is setup with a shared file system (file system here means Unix System Services file system), which means that all file systems that are mounted by a system participating in the shared file system are available to all participating systems.

In our configuration, the zFS, which is mounted at /usr/aci, is accessible from both SC48, SC67, and SC52, and ownership of the filesystem is moved to another system if the owning system leaves the sysplex.

## 8.4 BASE24-eps considerations

In this section, we discuss the implementation of BASE24-eps for our test environment.

### 8.4.1 Constructing the file system for the second system

The process for deploying BASE24-eps in multi-LPAR configuration is documented in *ACI BASE24-eps Release 1.0, Version 08.2 IBM z/OS Multiple LPAR Migration Guide*.

A second instance of our BASE24-eps application is required to be set up on a second system called SC67. Because the software was already installed once, we make use of this to build a second UNIX System Services file system.

Our second system will use /usr/aci/eps/eps1b as the home directory. Table 8-1 shows the directory structure.

Table 8-1 Directory structure

System	\$ES_HOME
SC48	/usr/aci/eps/eps1a
SC67	/usr/aci/eps/eps1b

The existing file system for SC48 will be cloned for SC67. Some of the directories are copied, and some will be defined as symbolic links so that we can share common objects, such as the executable files.

Figure 8-7 on page 134 shows the directory structure for eps1b.

```

drwxr-xr-x 10 SYS1 512 Feb 14 15:11 ESJMX/
drwxr-xr-x 7 SYS1 416 Feb 14 15:11 EvtAdapter/
lrwxrwxrwx 1 SYS1 24 Feb 14 15:11 JWHAT@ -> /usr/aci/eps/eps1a/JWHAT
lrwxrwxrwx 1 SYS1 25 Feb 14 15:11 JavaSF@ -> /usr/aci/eps/eps1a/JavaSF
lrwxrwxrwx 1 SYS1 25 Feb 14 15:11 Server@ -> /usr/aci/eps/eps1a/Server
lrwxrwxrwx 1 SYS1 24 Feb 14 15:11 batch@ -> /usr/aci/eps/eps1a/batch
drwxr-xr-x 4 SYS1 3584 Feb 14 15:49 bin/
drwxr-xr-x 3 SYS1 288 Feb 14 15:11 comm/
lrwxrwxrwx 1 SYS1 25 Feb 14 15:11 config@ -> /usr/aci/eps/eps1a/config
lrwxrwxrwx 1 SYS1 21 Feb 14 15:11 db@ -> /usr/aci/eps/eps1a/db
lrwxrwxrwx 1 SYS1 22 Feb 14 15:11 lib@ -> /usr/aci/eps/eps1a/lib
lrwxrwxrwx 1 SYS1 23 Feb 14 15:11 logs@ -> /usr/aci/eps/eps1a/logs
lrwxrwxrwx 1 SYS1 23 Feb 14 15:11 make@ -> /usr/aci/eps/eps1a/make
lrwxrwxrwx 1 SYS1 24 Feb 14 15:11 mwork@ -> /usr/aci/eps/eps1a/mwork
drwxr-xr-x 4 SYS1 320 Feb 14 15:11 queuing/
lrwxrwxrwx 1 SYS1 35 Feb 14 15:11 source_servercs@ ->
/usr/aci/eps/eps1a/source_servercs
drwxr-xr-x 2 SYS1 288 Feb 14 15:11 vsp/

```

Figure 8-7 Directory structure for eps1b

You can see in Figure 8-7 that for SC67 some directories are symbolic links, for example \$ES\_HOME/lib on SC67 points to /usr/aci/eps/eps1a/lib, which means that both systems (SC48 and SC67) are sharing the same executable files.

In contrast, SC67 has its own copy of the /bin directory, that is, \$ES\_HOME/bin on SC67 is /usr/aci/eps/eps1b/bin, which is necessary because the /bin directory for SC67 contains env\_vars, which sets environment variables for the instance of BASE24-eps on that system. Table 8-2 shows an example of the directory structure.

Table 8-2 Example of directory structure

Directory	SC48	SC67
\$ES_HOME/lib	/usr/aci/eps/eps1a/lib	/usr/aci/eps/eps1a/lib
\$ES_HOME/bin	/usr/aci/eps/eps1a/bin	/usr/aci/eps/eps1b/bin
\$ES_HOME/comm	/usr/aci/eps/eps1a/comm	/usr/aci/eps/eps1b/comm

Use standard UNIX **copy** commands to copy the required directories from eps1a to eps1b. Define symbolic links for those directories that do not need to be copied.

## 8.4.2 Configuring BASE24-eps parameters for the second system

We now have a clone of the file system for BASE24-eps to use on SC67. We now make the necessary configuration changes to define our new BASE24-eps system.

For those directories (and files) that are copied from SC48 to make the SC67 version, we must update the system-specific SC48 values with those for SC67, for example, each system has its own copy of env\_vars, which sets the values of environment variables for that system. We must update this file with the values for SC67.

Table 8-3 on page 135 contains examples of system-specific configuration values.

Table 8-3 Examples of system-specific configuration values

SC48 value	SC67 value
/usr/aci/eps/eps1a	/usr/aci/eps/eps1b
\$PRFX_QMNGR.A	\$PRFX_QMNGR.B
wtsc48.itso.ibm.com	wtsc67.itso.ibm.com

In summary:

1. Edit configuration files to replace SC48 values with those for SC67.
2. Build ICE-XS parameter file for SC67.
3. Create the \*.nof files for the BASE24-eps network.
4. Cold start the ICE-XS processes.
5. Build the JMX configuration (run jmx\_eps\_reload).
6. Load the BASE24-eps cryptographic keys for SC67.

### 8.4.3 MQ changes for the second system

In addition to defining MQ queues for the second system, we also must change the definition of the existing queues to make them shared. If you were planning to implement a multi-system configuration from the very beginning, you might choose to define them as shared at the start.

#### Defining queues for the second system

The naming convention in use here is EPS1.A.\* for those queues that are specific to SC48 and EPS1.B.\* for those queues that are specific to SC67.

All queues are defined as shared with QSGDISP(SHARED) and CFSTRUCT('APPLICATION1'). Example 8-1 shows the MQ queues for the second system.

Example 8-1 MQ queues for second system

---

```
EPS1.B.ISS.VISA
EPS1.B.IS1
EPS1.B.IS1.SYNC1
EPS1.B.IS1.SYNC2
EPS1.B.IS1.SYNC3
EPS1.B.IS1.SYNC4
EPS1.B.IS1.SYNC5
EPS1.B.NCRDH
EPS1.B.VISA
EPS1.B.XML.SYNC1
EPS1.B.XML1
```

---

#### Manager processes

BASE24-eps manager processes, such as End of Period Processing (EOPP), might have only one instance concurrently executing in a single BASE24-eps system. In our configuration, we have these manager processes executing on system SC48.

The MQ queues for these manager processes are altered to change the SHARE attribute to NOSHARE so that only one instance of the application can open and get messages from the queue. Note that this parameter has nothing to do with MQ queue sharing group shared queues.

**Note:** We changed the names of these queues to better reflect the fact that they are not specific to one instance of a BASE24-eps. While the processes can only run on one system, it can be either system, and the processes might be moved if there is a planned outage on the system that is currently running those services. These manager processes do not prevent BASE24-eps from processing transactions as normal if they are not running.

All queues are defined as shared with QSGDISP(SHARED) CFSTRUCT('APPLICATION1') and NOSHARE. Example 8-2 shows the queue names for the BASE24-eps manager processes.

*Example 8-2 Queue names for the BASE24-eps manager processes*

---

```
EPS1.ATBM1
EPS1.CMCP1
EPS1.EOPP1
EPS1.JQB1
EPS1.RFSH1
EPS1.SAFMGR1
EPS1.TBP1
EPS1.TMR1
EPS1.TTLM1
```

---

### **Redefining all other MQ queues as shared**

All other BASE24-eps queues are changed to a shared queue definition.

Specify QSGDISP(SHARED) and CFSTRUCT('APPLICATION1').

### **Adding new queue names to the BASE24-eps MQ NAMELIST**

MQ NAMELIST are objects that are local to an MQ Manager. The NAMELIST definitions are updated in each participating MQ Manager. Example 8-3 shows the addition of new B queues to NAMELIST.

*Example 8-3 Add new B queues to NAMELIST*

---

```
DEFINE REPLACE -
  NAMELIST(EPS1WBLST) -
  DESCR('COMMAND QUEUE LIST FOR ALL EPS1 NODES') -
  NAMES( EPS1.ATBM1, -
    , EPS1.CMCP1 -
    , EPS1.EOPP1 -
    , EPS1.A.IS1 -
    , EPS1.B.IS1 -
    , EPS1.JQB1 -
    , EPS1.RFSH1 -
    , EPS1.SAFMGR1 -
    , EPS1.TBP1 -
    , EPS1.TMR1 -
    , EPS1.TTLM1 -
    , EPS1.A.XML1 -
    , EPS1.B.XML1 -
  )
```

---

## 8.4.4 Application Management Remote Agent configuration

In this section, we define the configuration steps to enable the Application Management Remote Agent communications between multiple LPARs. The configuration files reside in ESJMX/config on each LPAR. Configuration activities include:

- ▶ Defining Remote Agent Port in connector.xml
- ▶ Defining connector.xml in config.properties
- ▶ Defining UI Remote Agent Configuration
- ▶ Using External Connection to connect to another LPAR

### Defining Remote Agent Port in connector.xml

The connector.xml, located in \$ES\_HOME/ESJMX/config/, is modified for each LPAR. Remote Agent uses the defined port number to communicate with Application Management.

In Example 8-4, the PORTNUM placeholder is replaced with a port number, such as 8099 (an available port number for the example environment).

*Example 8-4 connector.xml sample*

---

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <startup>
    <create classname="mx4j.tools.naming.NamingService"
      objectname="naming:type=rmiregistry" >
      <arg type="int">PORTNUM</arg>
    </create>
    <call objectname="naming:type=rmiregistry" operation="start"/>
    <object objectid="rmi">
      <call classname="javax.management.remote.JMXConnectorServerFactory"
        method="newJMXConnectorServer">
        <arg type="javax.management.remote.JMXServiceURL">
          <new classname="javax.management.remote.JMXServiceURL">
            <arg type="string">
              service:jmx:rmi://localhost/jndi/rmi://localhost:PORTNUM/appmgmt</arg>
            </new>
          </arg>
          <arg type="java.util.Map" />
          <arg type="javax.management.MBeanServer" />
        </call>
      </object>
      <register objectname="connectors:type=rmi,protocol=jrmp">
        <arg type="object" refobjectid="rmi" />
      </register>
      <call method="start" refobjectid="rmi" />
    </startup>
    <shutdown>
      <call method="stop" refobjectid="rmi" />
      <call operation="stop" objectname="naming:type=rmiregistry" />
      <unregister objectname="connectors:type=rmi,protocol=jrmp" />
      <unregister objectname="naming:type=rmiregistry" />
    </shutdown>
  </configuration>
```

---

## Defining connector.xml in config.properties

The config.properties, located in \$ES\_HOME/ESJMX/config/, is modified for each LPAR.

In Example 8-5, the acijmx.jmx.connector.config.url is defined with the path and file name of the connector.xml file.

### Example 8-5 config.properties example

---

```
process.id=EPS1_JMX
event.log4j.config.url=file:/usr/aci/eps/eps1a/ESJMX/config/EventConfig.xml
trace.log4j.config.url=file:/usr/aci/eps/eps1a/ESJMX/config/TraceConfig.xml
config.use.applcnfg=false
trace.enabled=false
acijmx.jmx.server.domain=EPS1_JMX
acijmx.jmx.server.port=8011
acijmx.jmx.management.port=8012
acijmx.jmx.server.bootstrap=file:/usr/aci/eps/eps1a/ESJMX/config/acijmx.mlet
acijmx.jmx.server.host=wtsc48.itso.ibm.com
acijmx.jmx.install.path=/usr/aci/eps/eps1a/ESJMX
jmx.install.path=/usr/aci/eps/eps1a/ESJMX/ThirdParty/lib
acijmx.jmx.store.path=/usr/aci/eps/eps1a/ESJMX/store
acijmx.jmx.adaptor.config.url=file:/usr/aci/eps/eps1a/ESJMX/config/adaptor.xml
queuemanager.list=com.aciworldwide.es.mgmt.messaging.mq.WMQQueueManager
commshandler.list=com.aciworldwide.es.mgmt.comm.ice.ICECommsHandler
process.list=com.aciworldwide.es.mgmt.process.zos.ZProcess:com.aciworldwide.es.mgmt.process.zos.SISZProcess:com.aciworldwide.es.mgmt.process.zos.JSFZProcess
acijmx.jmx.connector.config.url=file:/usr/aci/eps/eps1a/ESJMX/config/connector.xml
```

---

## Restarting JMX

After configuring the connector.xml and config.properties files, JMX must be restarted. The following commands are used to restart the JMX application:

```
cd $ES_HOME/ESJMX/bin
```

```
stopacijmx
```

```
runacijmx
```

## Defining UI Remote Agent configuration

After restarting JMX, the ACI Desktop UI is used to configure the Remote Agent. To complete this configuration:

1. Start and logon to the ACI Desktop Interface.
2. Navigate to the Application Management panel. Click **System Operations**, and then click **Application Management**.
3. Click **TopologyManager**, and then click **AddRemoteAgent**, as shown in Figure 8-8 on page 139.



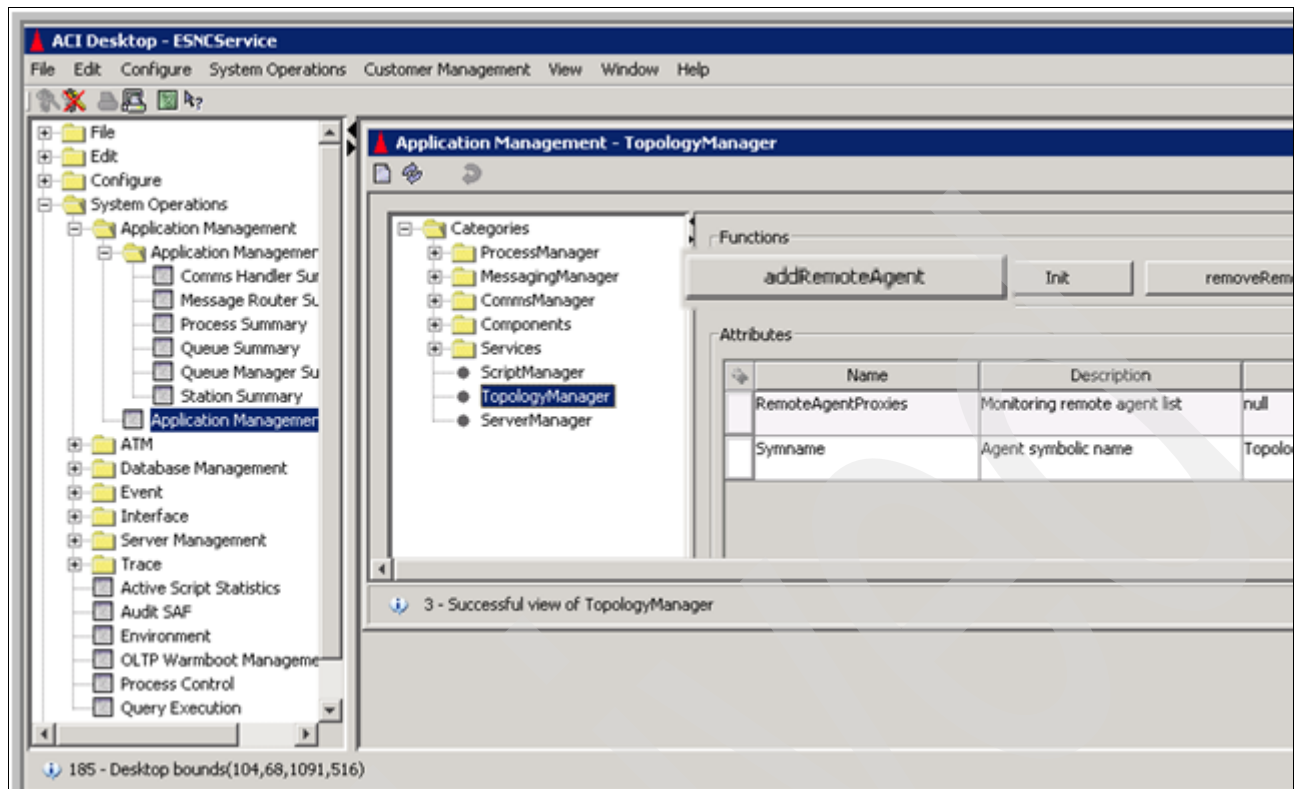


Figure 8-8 TopologyManager

4. Complete the Agent symbolic name. In this example, we use SC67 as the Agent symbolic name, as shown in Figure 8-9 on page 140.

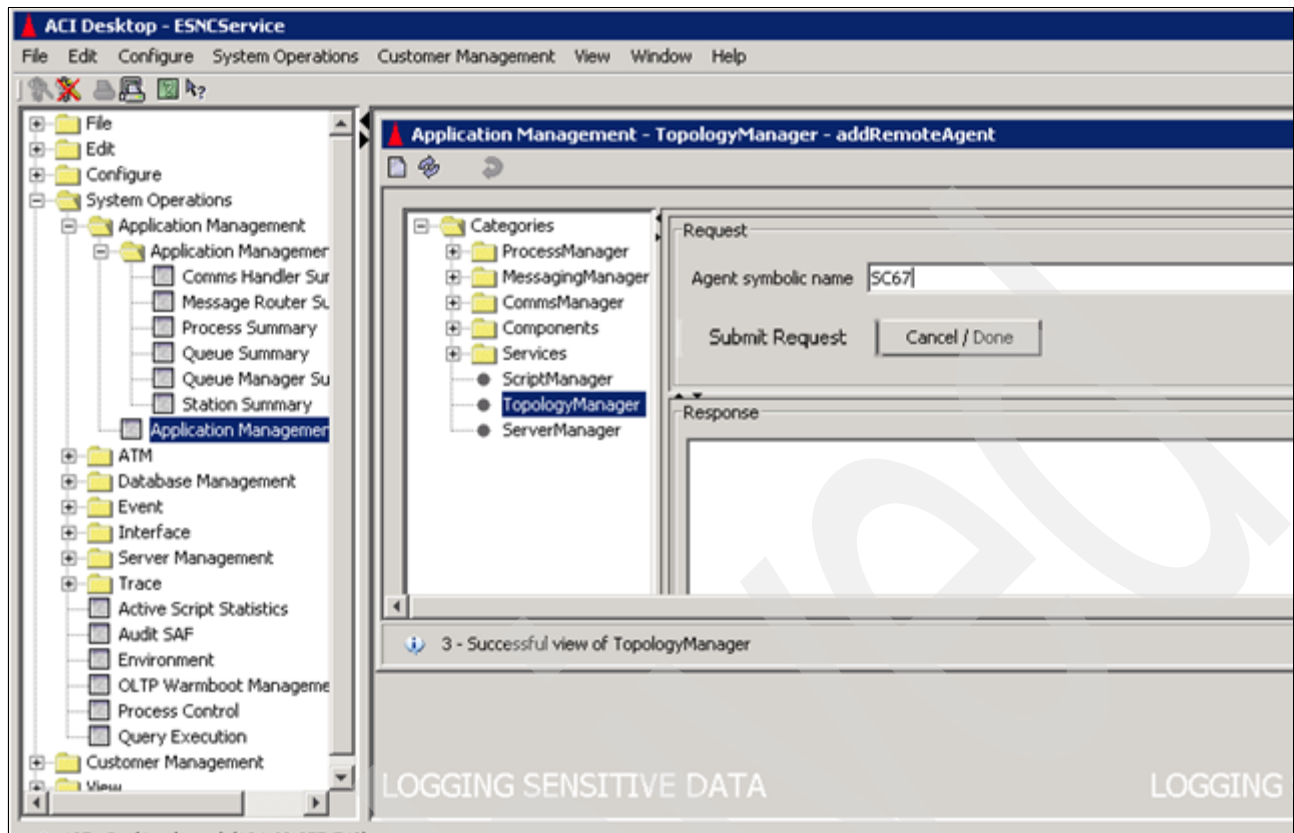


Figure 8-9 Agent symbolic name

5. Click **Submit Request** to add the remote agent.
6. Click **Cancel/Done** to exit the addRemoteAgent window.
7. Refresh the window, select the name just submitted, and click **Edit**, as shown in Figure 8-10 on page 141.

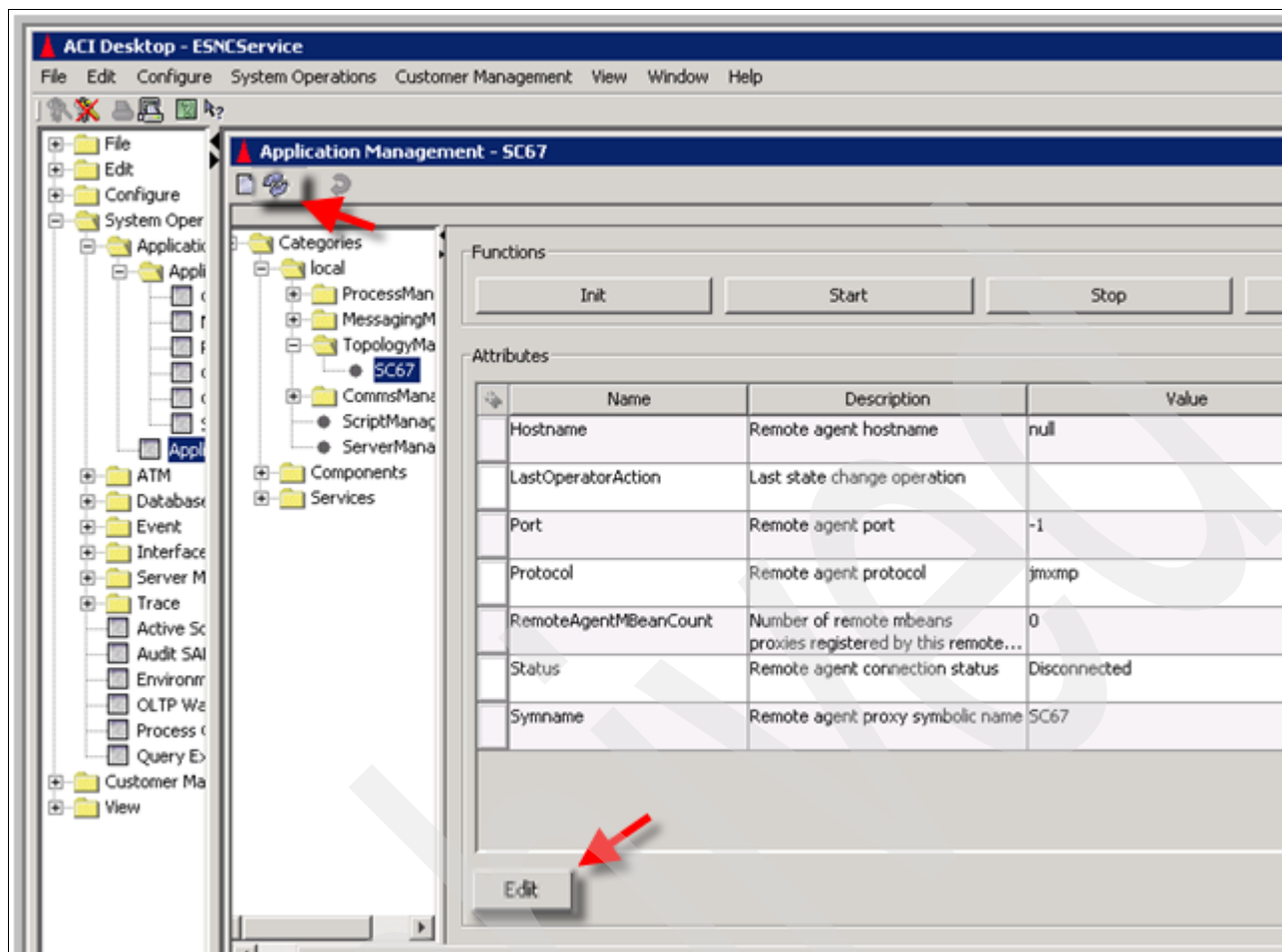


Figure 8-10 Refresh and Edit

8. Define Remote Agent connection parameters:

Specify the following information on this window, as shown in Figure 8-11 on page 142.

Hostname: The host name of the 'remote machine'. In the example system, our primary installation was to SC48. The ACI Desktop Application Management UI is, by default, pointed to SC48. In the next section, we set the UI to use SC67.

The Hostname parameter defined is that of the remote host when the UI is using the current host. Table 8-4 shows the values for the example system.

Table 8-4 current and remote host

If current host	Define remote host
SC48	SC67
SC67	SC48

Port: Define the same port number that was defined in the connector.xml file. The example system uses port number 8099 for all LPARs.

Protocol: The correct protocol value is: rmi

9. After you make the changes, click **Save**, and then click **Cancel/Done**.

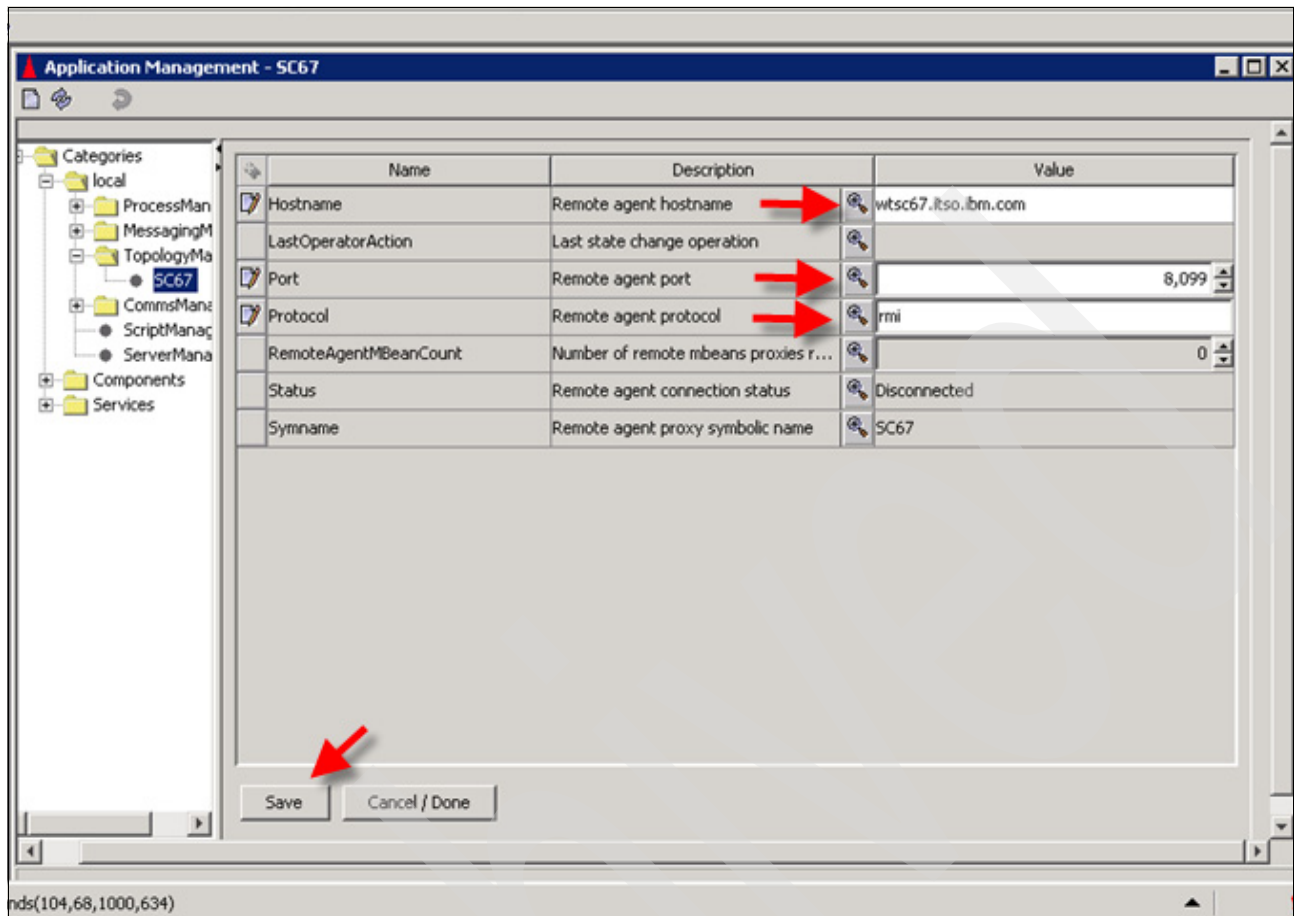


Figure 8-11 Define remote agent connection parameters

10. Start the Remote Agent by clicking **Start**, as shown in Figure 8-12 on page 143.

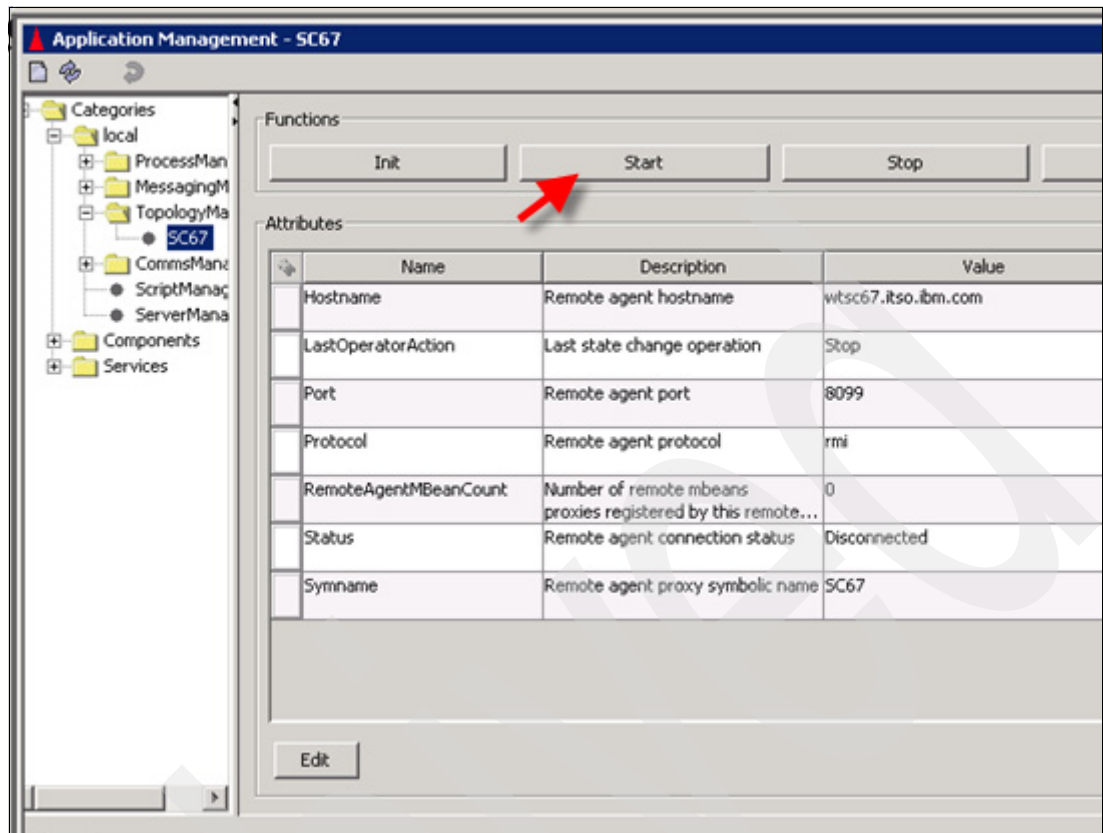


Figure 8-12 Start the Remote Agent

11. Click **Refresh**. The remote system can now be accessed from the ACI Desktop Application Management UI, as shown in Figure 8-13 on page 144.

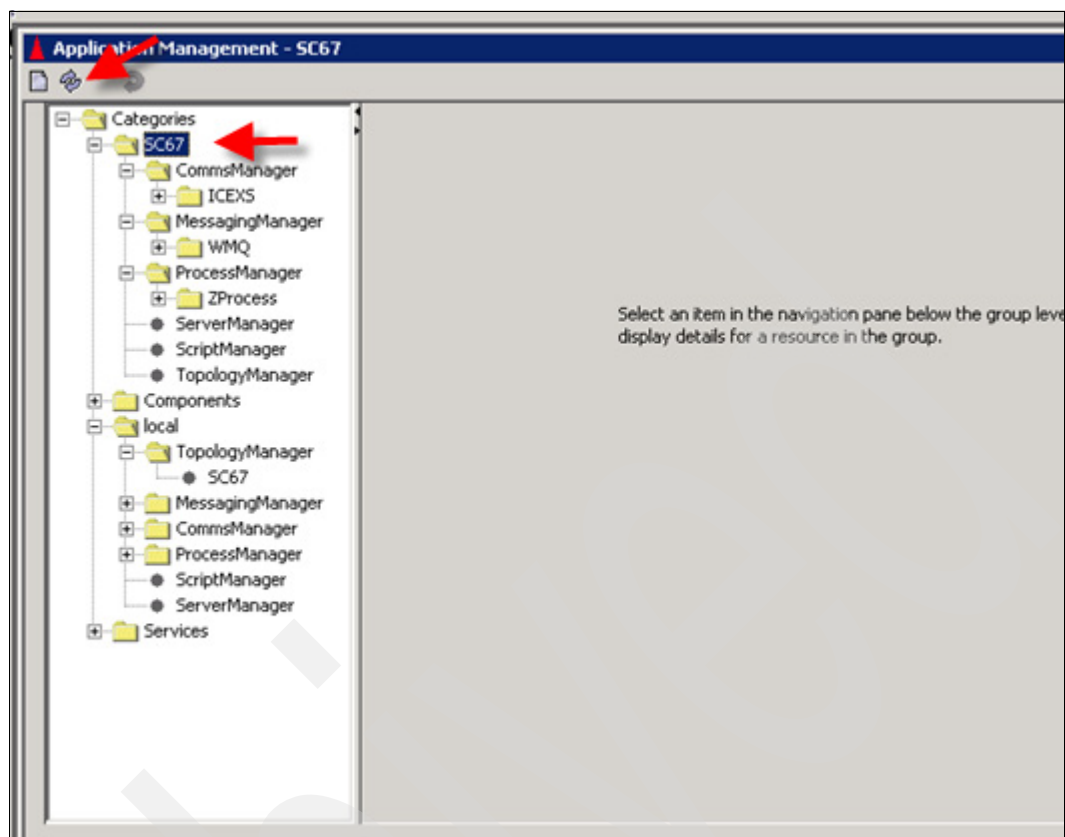


Figure 8-13 Refresh

### Using External Connection to connect to another LPAR

The ACI Desktop UI is used to connect to ACIJMX Agents on another LPAR. The connection is made using the External Connection Configuration window.

To alter the connection configuration:

1. Start and logon to the ACI Desktop Interface.
2. Navigate to the External Connection window (System Operations 'Server Management' External Connection), as shown in Figure 8-14 on page 145.

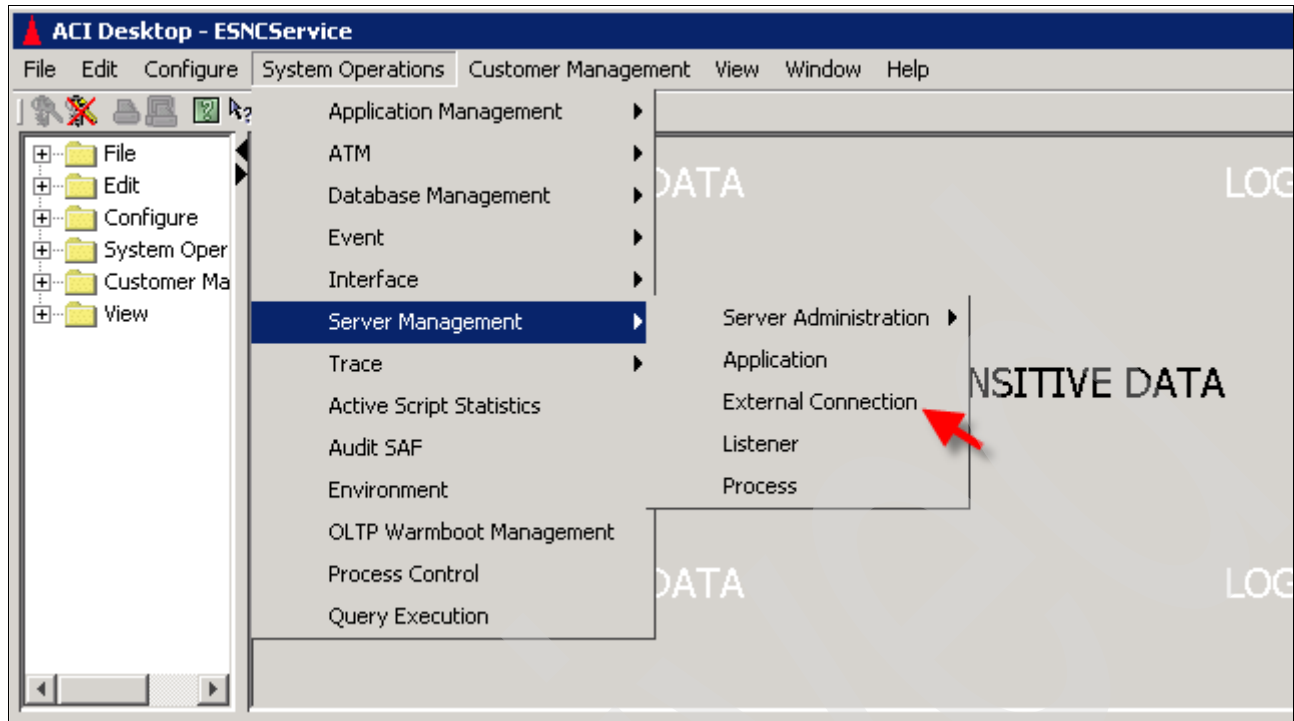


Figure 8-14 External Connection selection

3. Select the **Protocol Details** tab, as shown in Figure 8-15.

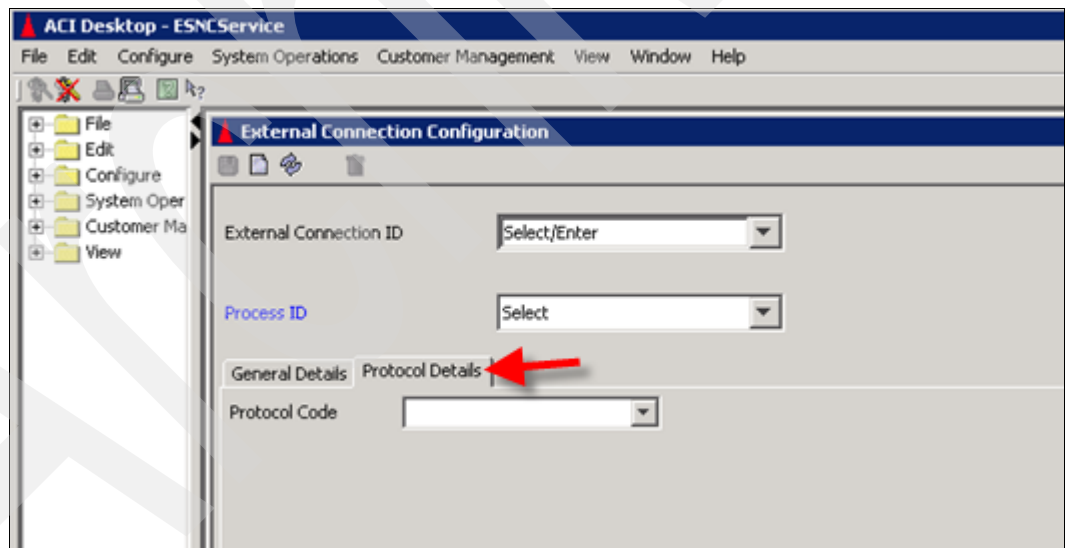


Figure 8-15 Protocol Details tab

4. Define External Connection Parameters.

The information in Table 8-5 on page 146 is specified on the window in Figure 8-16 on page 146. After the appropriate data is entered, click **Save** to save the External Connection and close the window, as shown in Figure 8-16 on page 146.

Table 8-5 External Connection Parameters specified under the Protocol Details tab

Field	Selection or value
External Connect ID	APPLICATION-MGMT
Process ID	Specify BASE24-eps UI
Protocol Code	HTTP
Protocol Options	Specify a period, '.'
Destination IP Address	For Application Management connections, an Internet-style address is defined. The address string is defined as: <b>http://&lt;hostname&gt;:&lt;port number&gt;</b> Enter the values for the ACIJMX agent to connect to a particular LPAR. The values for the LPARs in the example system are: <ul style="list-style-type: none"> <li>▶ SC48 http://wtsc48.itso.ibm.com</li> <li>▶ SC67 http://wtsc67.itso.ibm.com</li> </ul>
Port Number	For Application Management connections, the actual port number is defined in the Destination IP Address field and the Port Number field is set to "application/xml".

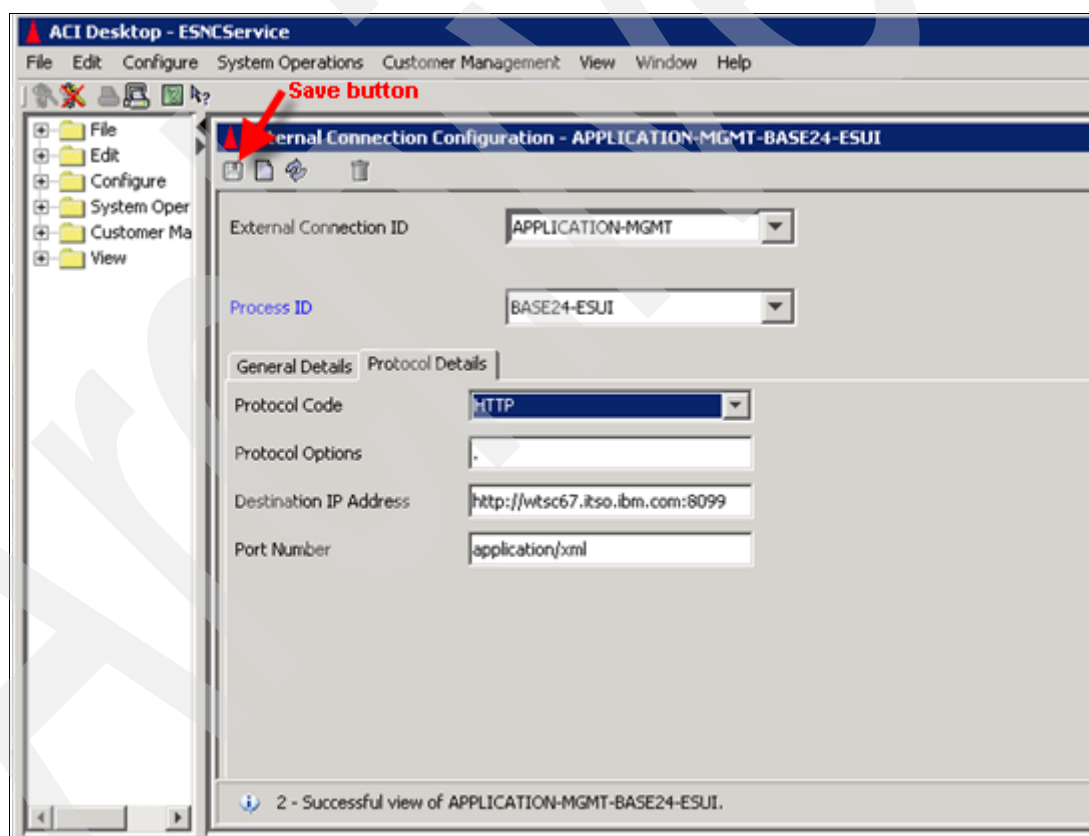


Figure 8-16 External Configuration Panel

## Verifying external connection

Use the ACI Desktop Application Management UI panels to verify the external connection (System Operations Application Management).



Figure 8-17 shows SC67 processes that are available on the Application Management Process Summary window.

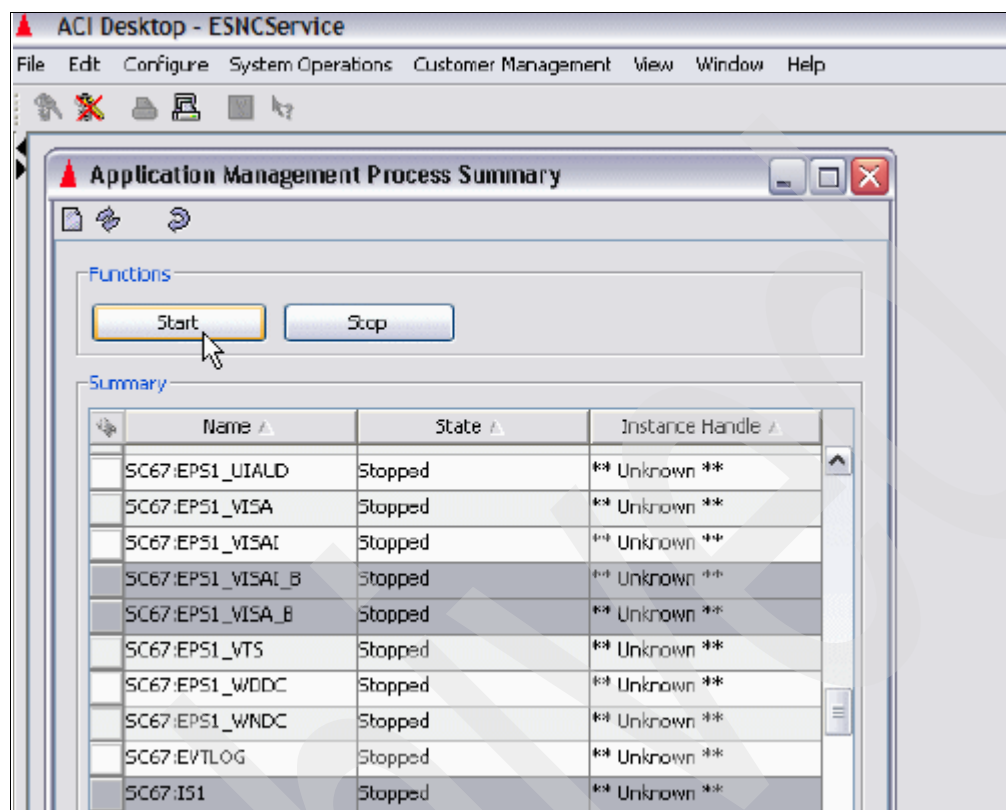


Figure 8-17 Process summary

### Defining additional remote agents

Additional remote agents can be defined by repeating the steps in “Defining UI Remote Agent configuration” on page 138.

## 8.5 Starting BASE24-eps on SC67

If you have not already done so, you must cold start the ICE-XS processes for SC67.

To start BASE24-eps on SC67:

1. From a telnet session, start JMX and TDAL on SC67, as shown in Figure 8-18 on page 148.



```
wtsc67oe.itso.ibm.com - PuTTY
EPS1@wtsc67:EPS1:ttyp0000:/usr/aci/eps/eps1b
$ runacijmx
Application Management Server started
EPS1@wtsc67:EPS1:ttyp0000:/usr/aci/eps/eps1b
$ runtdal
Process TDAL Started
Process 67831644
EPS1@wtsc67:EPS1:ttyp0000:/usr/aci/eps/eps1b
$ esstatus
*** MQ Series Manager ***
MQGG is running
EPS1.CXATMD(1)

*** DB2 server ***
DB2 connection is established

*** ES Processes for System: EPS1 ***
EPS: silisten is running (PID: 67831644)
EPS: ACIJMX is running (PID: 84608836)

*** ICE-XS Listeners ***
None

EPS1@wtsc67:EPS1:ttyp0000:/usr/aci/eps/eps1b
$
```

Figure 8-18 Start JMX and TDAL on SC67

2. Use the client UI to start the other BASE24-eps processes on SC67, as shown in Figure 8-19 on page 149. Note that the UI process summary panel in Figure 8-20 on page 149 has visibility of SC67.

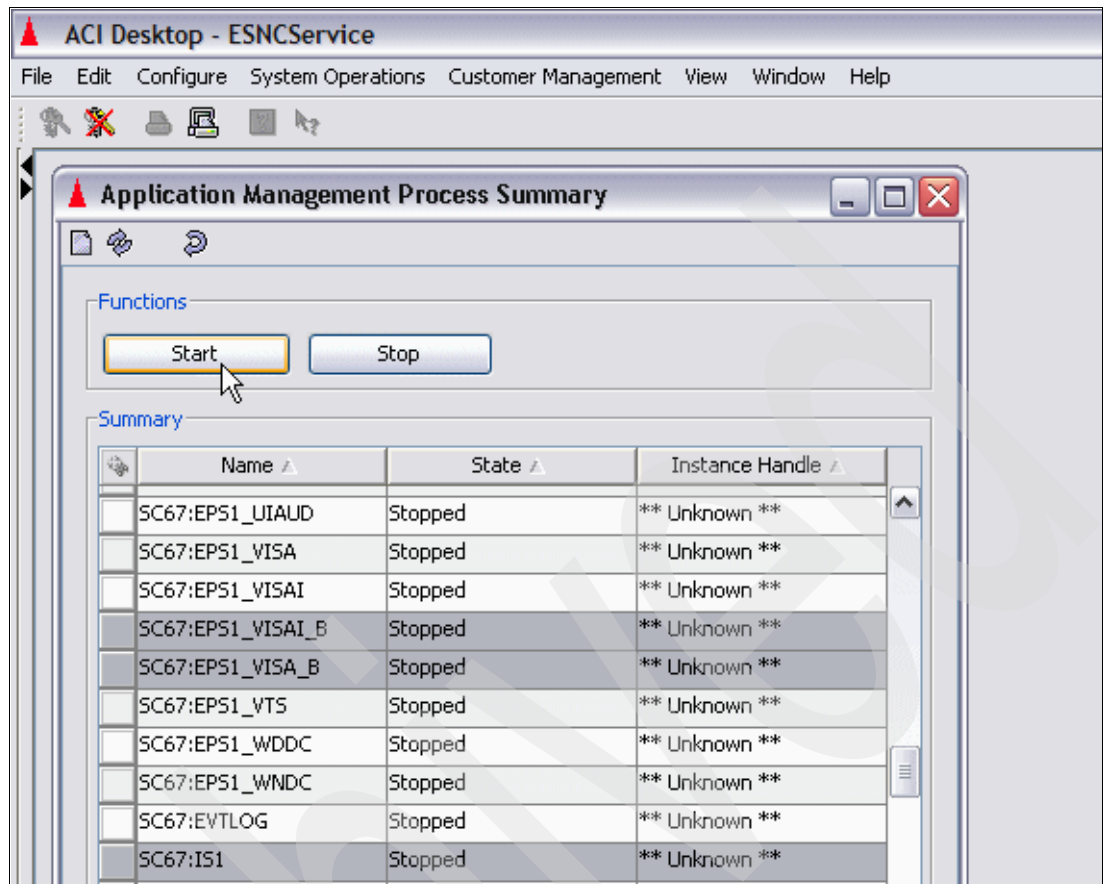


Figure 8-19 Start BASE24-eps processes on SC67



Figure 8-20 esstatus for SC67

The manager processes, such as JQB, EOPP, and SAFM are all running on SC48 and can be seen in the esstatus display on that system.

## 8.6 Running workload on both systems

BASE24-eps is active on both systems SC48 and SC67. We start up our simulators and direct the connection to the sysplex distributor VIPA, which results in some connections being made on SC48 and some on SC67.

For monitoring purposes, we use a script that we wrote called `ice_traffic_cnt`, which does a regular `netstat` display of the ICE-XS ports that are accepting connections. We start one of these displays for each system and position the telnet panels side-by-side, which allows us to watch connections being made on each system and to see the network traffic counts increase as the transactions come in to the system and the responses are returned.

Figure 8-21 shows the NETSTAT display for ICE-XS ports.

Wed Mar 4 16:56:59 EST 2009						
=====						
User Id	B Out	B In	L Port	Foreign Socket	State	
EPS11	0000000000	0000000000	08014	0.0.0.0..0	Listen	
EPS17	0000000000	0000000000	08010	0.0.0.0..0	Listen	
EPS18	0000000000	0000000000	08017	0.0.0.0..0	Listen	
EPS19	0000000000	0000000000	08020	0.0.0.0..0	Listen	
Wed Mar 4 16:58:32 EST 2009						
=====						
User Id	B Out	B In	L Port	Foreign Socket	State	
EPS11	0000000880	0000001122	08014	10.1.200.32..4751	Establish	
EPS11	0000000000	0000000000	08014	0.0.0.0..0	Listen	
EPS11	0000000880	0000001122	08014	10.1.200.36..4745	Establish	
EPS11	0000000763	0000001122	08014	10.1.200.39..4744	Establish	
EPS11	0000000880	0000001122	08014	10.1.200.35..4749	Establish	
EPS11	0000000763	0000000936	08014	10.1.200.41..4742	Establish	
EPS11	0000000763	0000000936	08014	10.1.200.40..4743	Establish	
EPS17	0000000000	0000000000	08010	0.0.0.0..0	Listen	
EPS18	0000000000	0000000000	08017	0.0.0.0..0	Listen	
EPS19	0000000000	0000000000	08020	0.0.0.0..0	Listen	

Figure 8-21 NETSTAT display for ICE-XS ports

In the first display in Figure 8-21, you can see the starting position with ICE-XS listening on various ports. The second display shows a number of connections to port 8014 with message traffic in and out. In our case, port 8014 was associated with VISA transactions.

Messages that ICE-XS receives are placed on the shared Integrated Server service queue where they can be processed by an IS process on either LPAR. One of the IS processes on either SC48 or SC67 reads the transaction off the queue and performs whatever processing is required. The transaction response is placed on the input queue for the ICE-XS process that manages the connection to the originating endpoint.

## BASE24-eps high availability testing

In this chapter, we document our test scenarios. First we describe the objective, the injection, and the expected behavior for each scenario. Then we describe the test we performed against each scenario and detail the actual behavior. In cases where the expected behavior differs from the actual test behavior, we provide a solution. We also describe recovery actions that are required to return to the normal operating environment after each test.

In this chapter, we cover:

- ▶ 9.1, “Test scenarios” on page 152
- ▶ 9.2, “IBM WebSphere MQ” on page 152
- ▶ 9.3, “Data environment” on page 154
- ▶ 9.4, “BASE24-eps” on page 156
- ▶ 9.5, “Hardware and z/OS system software” on page 158

## 9.1 Test scenarios

High availability is of prime importance to modern online transaction processing systems. Improvements in modern hardware made fault tolerance of less relative importance—the loss of data in the communications networks far exceeds the loss of information in the OLTP processing system itself, and end-to-end protocols designed to recover from loss of data in the communications networks also assure financial integrity during infrequent hardware failures. However, contractual obligations and end-user requirements continue to make high availability vitally important.

Our test configuration was designed to represent a high level of availability attainable at a single production site. All components were replicated to eliminate any single point of failure. Scenarios were developed to assure that the OLTP processing system remained available through a wide cross-section of potential hardware and software failures.

## 9.2 IBM WebSphere MQ

In this section, we discuss high availability tests that involve IBM WebSphere MQ.

### 9.2.1 MQ manager loss

This test details the results of an MQ manager loss.

#### Objective and injection

The objective of this test case is to verify that the workload keeps running even when an IBM WebSphere MQ queue manager address space fails. To simulate this scenario, we issue the MVS command **CANCEL MQG2MSTR** from SDSF on SC67.

#### Expected behavior

The ICE-XS process on SC67 will close existing connections and not accept new ones. The Integrated Server address spaces on SC67 will terminate abnormally. Requests in flight will be lost and redriven due to end-to-end protocols. Responses in flight will be reversed so that financial integrity is retained. Endpoints will reconnect, and TCP/IP virtualization will route reconnects to the surviving ICE-XS instance(s). Processing will continue on SC48.

#### Actual behavior

When the SC67 IBM WebSphere MQ address space is cancelled, all transaction processing moved to SC48, as expected. The SC67 ICE-XS processes close their sockets and are no longer listening on their ports. UI displays show that IS and XMLS processes ended abnormally on SC67. We see in the syslog that DB2 in-flight threads are failing. We also see the IS and XMLS failures in the syslog. Because the BASE24-eps event log subsystem relies on IBM WebSphere MQ®, we do not see any indication of failures in the BASE24-eps event log.

#### Recovery actions

We stopped all remaining BASE24-eps processes on SC67. We then restart MQ queue manager using command MVS command - **MQG2 START QMGR** from SDSF on SC67. When IBM WebSphere MQ is up, we restart all BASE24-eps processes.

## 9.2.2 IBM WebSphere MQ maintenance implementation

This test details the results of applying maintenance to an MQ instance.

### Objective and injection

The objective of this test is to verify that BASE24-eps processing will not be adversely affected by stopping IBM WebSphere MQ on one system in a sysplex for the purpose of implementing IBM WebSphere MQ maintenance.

We create this situation by bringing down BASE24-eps application processes, then IBM WebSphere MQ. We then bring them up in a controlled fashion. We use the UI for the application processes and MVS commands for IBM WebSphere MQ.

Our process was to:

1. Shut down the ICE-XS processes on SC48.
2. Shut down all other BASE24-eps processes on SC48.
3. Stop IBM WebSphere MQ on SC48.
4. Implement MQ maintenance.
5. Restart IBM WebSphere MQ on SC48.
6. Restart IS processes on SC48.
7. Restart ICE-XS processes on SC48.

### Expected behavior

Transaction traffic will move to the remaining system (SC67) and process normally.

### Actual behavior

Using the UI, we stop all ICE-XS processes on SC48, and then verify that transaction traffic is going to SC67 and processing as expected. We then stop the remaining BASE24-eps processes on SC48. Next, we shut down IBM WebSphere MQ, using the MVS command -MQGQ STOP QMGR.

Transaction processing continues normally on SC67.

### Recovery actions

Restart IBM WebSphere MQ with the MVS command -MQGQ START QMGR. When IBM WebSphere MQ is up, use the UI to manually restart all BASE24-eps processes on SC48.

## 9.2.3 Adding IBM WebSphere MQ instance in new LPAR

This test details the results of adding a new MQ instance in a new LPAR.

### Objective and injection

The objective of this scenario is to show that adding instances of BASE24-eps infrastructure components is non-disruptive to a running BASE24-eps instance. Configure a new IBM WebSphere MQ queue-sharing group instance and queues in new LPAR. Create Integrated Server definitions and ICE-XS definitions in new LPAR. Start IBM WebSphere MQ, Integrated Servers, and ICE-XS processes.

### Expected behavior

Workload is seamlessly redistributed across Integrated Servers in old and new LPARS. Connections are distributed across ICE-XS processes in old and new LPARs as endpoints reconnect.

### **Actual behavior**

This scenario was demonstrated when we added the BASE24-eps instance on the second LPAR in our test environment. The application was cloned, and IBM WebSphere MQ and DB2 configurations were added for SC67. When BASE24-eps processes were started on SC67, they were able to start taking over workload as new connections were established, while normal processing continued on SC48.

### **Recovery actions**

No recovery actions are applicable in this scenario.

## **9.3 Data environment**

In this section, we detail high availability tests that involve DB2.

### **9.3.1 DB2 catastrophic failure**

This test details the results of a DB2 catastrophic failure.

#### **Objective and injection**

The objective of this test is to verify that BASE24-eps transaction processing can continue on one system in a sysplex, if the DB2 is lost on another system.

We create this situation by cancelling the DB2 address space on one system using MVS command C D9G2MSTR on system SC67.

#### **Expected behavior**

ISs will try DB access and terminate abnormally when they are unable to access DB2 services. ICE-XS processes will continue on both SC48 and SC67. The ICE-XS on SC67 will continue to write to shared queues, where the IS processes on SC48 will continue to process work as normal.

#### **Actual behavior**

Incoming and outgoing transaction traffic continues through the ICE-XS processes on both SC48 and SC67. IBM WebSphere MQ recognizes that DB2 is down. An esstatus display indicates that DB2 is down, as expected. The event log indicates multiple read errors and SQL errors due to database unavailability on SC67, which is expected. The event log also shows us that normal transaction processing continues on SC48, as predicted.

#### **Recovery actions**

Restart DB2 with MVS command -D9G2 START DB2. After DB2 is up, manually restart all BASE24-eps processes on SC67.

### **9.3.2 Data table unavailable at BASE24-eps initialization**

This test details the results when a data table is unavailable at BASE24-eps initialization.

#### **Objective and injection**

The objective of this scenario is to ensure that BASE24-eps initialization gracefully handles the data table being unavailable.



The DB2 table ADMF was renamed and then BASE24-eps was initialized.

### **Expected behavior**

Application will not successfully start. Particular messages expected depends on which table(s) are missing.

### **Actual behavior**

BASE24-eps did not successfully initialize. A message was written to the event log stating that ADMF was not defined.

### **Recovery actions**

Restore the ADMF table, and restart BASE24-eps.

## **9.3.3 DB2 maintenance implementation**

This test details the results of applying DB2 maintenance.

### **Objective and injection**

The objective of this test is to verify that BASE24-eps processing continues normally when the application and DB2 are stopped in a controlled fashion, on one system in a sysplex. This scenario would be used for the purpose of implementing DB2 maintenance.

We create this situation by bringing down BASE24-eps application processes, and then bringing down DB2. After DB2 maintenance implementation, we restart DB2 and all BASE24-eps processes. We use the UI for the application processes and MVS commands for DB2.

Our process is:

1. Shut down all BASE24-eps processes on SC48.
2. Stop DB2 on SC48.
3. Implement DB2 maintenance.
4. Restart DB2 on SC48.
5. Restart BASE24-eps processes on SC48.

### **Expected behavior**

Transactions will continue to be processed by BASE24-eps on SC67. After DB2 and BASE24-eps are restarted on SC48, transactions will be processed by both SC48 and SC67, although all network traffic will still be going to SC67.

### **Actual behavior**

Stop ICE-XS, and then stop all BASE24-eps processes on SC48. Issue STOP DB2 for D9G1 on SC48. Sysplex Distributor remains on SC48, but all connections are being established with SC67 because ICE-XS is down on SC48.

### **Recovery actions**

Restart DB2 with MVS command -D9G1 START DB2. After DB2 is up, manually restart all BASE24-eps processes on SC48.

## 9.4 BASE24-eps

In this section, we discuss high availability tests that involve BASE24-eps processes.

### 9.4.1 Integrated Server failure

This test details the results of an Integrated Server failure.

#### Objective and injection

The objective of this test is to verify that BASE24-eps workload continues in the event of an IS failure. We simulate by cancelling one of the Integrated Server address spaces.

#### Expected behavior

Transactions actually being processed in the server at the time of the failure will be lost and redriven by financial end-to-end protocols. Processing continues across the remaining servers until the failed server instance is restarted by ACI Application Management, and then processing continues across all servers.

#### Actual behavior

Active tasks that display through SDSF shows that the cancelled IS process is gone. We then see a new process active and processing work, after JMX restarts an IS process automatically. There was no impact on transaction processing.

#### Recovery actions

No manual recovery actions are needed.

### 9.4.2 BASE24-eps maintenance: planned application upgrade

This test details the results of a planned upgrade of BASE24-eps.

#### Objective and injection

The objective of this test is to verify that we can implement application maintenance for BASE24-eps on one system without adversely affecting processing on the other system in the sysplex.

We simulate this situation by using the UI to do the following:

1. Shut down the ICE-XS processes on SC67.
2. Shut down all other BASE24-eps processes on SC6.7
3. Implement BASE24-eps maintenance, according to vendor recommendations.
4. Restart BASE24-eps processes on SC67.
5. Restart ICE-XS processes on SC67.

#### Expected behavior

When the application processes are stopped on SC67, transaction traffic is routed to SC48, and continue processing as normal.

#### Actual behavior

Using the UI, we stop all ICE-XS processes on SC67. We then verify that transaction traffic is going to SC48 and processing as expected. We then stop the remaining BASE24-eps processes on SC67.

Transaction processing continues normally on SC48.

### **Recovery actions**

Use the UI to manually restart all BASE24-eps processes on SC67.

## **9.4.3 ICE-XS server failure**

This test details the results of an ICE-XS server failure.

### **Objective and injection**

The objective of this test is to verify that BASE24-eps transaction processing continues in the event of an ICE-XS failure. To simulate this situation, we cancel an ICE-XS process that is running on one system (SC67 in this test case).

### **Expected behavior**

When the ICE-XS process is cancelled, devices that were connected to SC67 retry their failed connections. Sysplex distributor then directs the transactions to the other system, where an ICE-XS is still active. The failed ICE-XS process will be restarted by ACI Application Management (JMX).

### **Actual behavior**

After the ICE-XS process is cancelled on SC67, we observe transaction traffic moving to the other system. SC48 and processing continues normally. JMX starts a new ICE-XS process on SC67, which then accepts new connections for transaction traffic.

### **Recovery actions**

Existing connections will not move back to the restarted ICE-XS process, although it will accept new connections. Manual intervention to re-balance the network load might be appropriate.

## **9.4.4 ICE-XS maintenance**

This test details the results of applying maintenance to ICE-XS.

### **Objective and injection**

The objective of this test is to verify that we can implement application maintenance for ICE-XS on one system without adversely affecting processing on the other system in the sysplex.

We simulate this situation by using the UI to do the following:

1. Shut down the ICE-XS processes on SC67.
2. Shut down all other BASE24-eps processes on SC67.
3. Implement ICE-XS maintenance, according to vendor recommendations.
4. Restart BASE24-eps processes on SC67.
5. Restart ICE-XS processes on SC67.

### **Expected behavior**

After the ICE-XS processes are stopped on SC67, we expect to see transaction traffic routed to SC48, where normal BASE24-eps processing will continue.

### **Actual behavior**

Using the UI, we stop all ICE-XS processes on SC67. We then verify that transaction traffic is going to SC48 and processing as expected. We then stop the remaining BASE24-eps processes on SC67.

Transaction processing continues normally on SC48.

### **Recovery actions**

Use the UI to manually restart all BASE24-eps processes on SC67.

## **9.4.5 Application Management server failure**

This test details the results of an Application Management Server failure.

### **Objective and injection**

The objective of this test is to determine the effects of JMX being unavailable. To simulate, we stop the Application Management Server (JMX) process on one system, SC48 in this case.

### **Expected behavior**

If one of the JMX processes fails, the BASE24-eps IS processes continues to process transactions because the JMX process is not required for this.

### **Actual behavior**

Transaction processing continues, as expected.

### **Recovery actions**

After you fix the problem, you can restart JMX using the runacijmx script. An additional complication is the client UI, which requires connectivity to JMX to operate and administer the BASE24-eps application. The JMX processes on each system are also linked to provide a single view of our two BASE24-eps instances.

In our configuration, the Java Server component that is running on SC52 targets the BASE24-eps application on SC48. If JMX is unavailable on that system, or the system itself is down, the client UI cannot connect, which is fixed by updating the target DNS name in:

```
/usr/aci/epsui/eps1/ESWeb/ESConfig/config.properties.
```

An alternative to having to update this properties file is to use the DVIPA address and make use of sysplex distributor to connect you to the BASE24-eps application on the other system.

Following either of these changes, it is necessary to stop and start the Java application definition inside the WebSphere Application Server.

## **9.5 Hardware and z/OS system software**

In this section, we detail high availability tests that involve z/OS system software and hardware components.

## 9.5.1 Coupling facility failure

This test details the results of a coupling facility failure.

### Objective and injection

The objective of this test is to verify that the workload continues to flow and execute successfully in case of a coupling facility failure. A coupling facility failure could occur as a result of a hardware failure, such as a power failure.

Simulate failure and corresponding recovery of an active CF with BASE24-eps workload running. This scenario targets the CF that contains duplexed DB2 and MQ structures. The CF failure is simulated by deactivating CF02 from the HMC.

### Expected behavior

The duplexed structures will fail over to the remaining CF.

### Actual behavior

We displayed the CFs before the test to verify that the structures were duplexed on CF01 and CF02. After deactivating CF02 through the HMC, we verify that the structures in CF01 are in simplex mode, as expected. We observe that transaction processing continues as normal, and all BASE24-eps tasks stay up on both systems in the sysplex, as expected.

As a point of interest, we also ran a test where we used HMC to deactivate a CF when the structures were NOT duplexed. As expected, IBM WebSphere MQ failed on both systems in the sysplex with resulting failures of BASE24-eps processes, which demonstrates the importance of ensuring that the structures are duplexed.

### Recovery actions

Activate the failed CF (CF02) through the HMC. Verify that all structures are now duplexed.

## 9.5.2 LPAR failure

This test details the results of an LPAR failure.

### Objective and injection

The objective of this test is to verify that BASE24-eps transaction processing will move to other system(s) in a sysplex, if one system is lost.

We create this situation by using the HMC to reset one z/OS system image, SC67 in this case.

### Expected behavior

Processing continues on the remaining system.

### Actual behavior

After SC67 is reset, SC48 continues normal processing as expected. Connections on SC67 switch over to SC48 after SC67 is configured out of the sysplex. (Timing of this is dependent on the Interval value for the failing system, specified in the COUPLExx member of parmlib.) Sysplex distributor then directs transactions to SC48 to be processed as normal.

As is understood in a DB2 data sharing environment, you might see retained locks in the case of an LPAR failure:

**DSNT501I -D9G1 DSNILMCL RESOURCE UNAVAILABLE 879**

**CORRELATION-ID=is.exe#59116**

### **Recovery actions**

IPL failed system. Restart IBM WebSphere MQ, DB2, ICE-XS, and all BASE24-eps processes. You might want to take manual recovery actions to rebalance the network load.

## **9.5.3 IP virtualization failure**

This test details the results of an IP virtualization failure.

### **Objective and injection**

The objective of this test is to ensure that BASE24-eps processing will continue after a TCP/IP outage on the LPAR that is defined as the primary Sysplex Distributor(SC48). SC67 is defined as the backup Sysplex Distributor. On SC48, STOP TCP/IP(P TCP/IP) is issued to create the outage.

### **Expected behavior**

The backup Sysplex Distributor LPAR(SC67) will receive any new connections from the acquiring endpoints. These endpoints will reconnect to an ICE-XS in the surviving LPAR. Issuer stations in the failed LPAR are marked down. Processing continues in the remaining LPAR.

### **Actual behavior**

After the TCP/IP outage on SC48, the Sysplex Distributor moves to SC67 and receives the new connections from the acquiring endpoints. These endpoints reconnect to an ICE-XS on SC67. Connections with SC67 persist during the outage.

### **Recovery actions**

After manually restarting TCP/IP on SC48, the Sysplex Distributor moves back to SC48. The TCP/IP stack on SC67 resumes its role as backup Sysplex Distributor. New connections are now established with Sysplex Distributor on SC48 and distributed to ICE-XS listeners on SC48 and SC67.

## **9.5.4 ICSF failure**

This test details the results of an ICSF failure.

### **Objective and injection**

The objective of this test is to demonstrate BASE24-eps behavior if there is a loss of cryptographic services. To simulate this situation, we make the API unavailable by shutting down the Integrated Cryptographic Services Facility (ICSF) address space, effectively severing communication with the cryptographic function.

On system SC67, command P CSF is entered from SDSF, where CSF is the task name for the ICSF address space.

### **Expected behavior**

Processing will continue. Transactions on SC67 that require cryptographic services will receive return code 12s from their ICSF API calls. These transactions might or might not fail, depending on customer requirements. (Appropriate automated alerts should be in place.) Other transactions on SC67 will continue to process normally. All transactions on SC48 will be unaffected.

### **Actual behavior**

After stopping the ICSF address space, transaction traffic continued normally on both SC48 and SC67. In the event log, we see that some transactions on SC67 are getting return codes of 12, as expected.

### **Recovery actions**

Re-enable cryptographic services. Restart ICSF task.

## **9.5.5 z/OS maintenance**

This test details the results of applying z/OS maintenance.

### **Objective and injection**

The objective of this test is to verify that BASE24-eps processing will continue normally when we need to IPL one system in the sysplex for installation of z/OS maintenance. We simulate this scenario by performing orderly shutdowns of the application processes and all subsystems that run on one of the z/OS images (SC48). After the subsystems are stopped, we can re-IPL SC48 from a new System Resident (SYSRES) set that contains z/OS maintenance.

We create this situation by bringing down BASE24-eps application processes, and then bringing down all subsystems (which includes IBM WebSphere MQ and DB2). After the IPL for z/OS maintenance implementation, we restart JES, DB2, IBM WebSphere MQ, and then all BASE24-eps processes. We use the UI for the application processes and MVS commands for the subsystems. (In many installations, automated procedures are in place to facilitate the shutdown and post-IPL restart of subsystems.):

1. Shut down all BASE24-eps processes on SC48.
2. Stop subsystems on SC48.
3. Implement z/OS maintenance by IPLing SC48 from new SYSRES.
4. Restart subsystems on SC48.
5. Restart BASE24-eps processes on SC48.

### **Expected behavior**

Transactions will continue to be processed by BASE24-eps on SC67. After SC48 is IPLed, and subsystems and BASE24-eps are restarted, transactions will be processed by both SC48 and SC67, although all network traffic will still be going to SC67.

### **Actual behavior**

After SC48 is reset, SC67 continues normal processing, as expected. Connections on SC48 switch over to SC67 after SC48 is configured out of the sysplex. (Timing of this is dependent on the interval value for the failing system, specified in the COUPLExx member of parmlib.) Sysplex distributor then directs transactions to SC67 to be processed as normal.

### **Recovery actions**

You might want to manually re-balance the network workload to direct more traffic to SC48 after the application is again running there.

## **9.5.6 Coupling facility maintenance**

This test details the results of applying coupling facility maintenance.

### **Objective and injection**

The objective of this scenario is to demonstrate that we can make one CF available for maintenance purposes without disrupting BASE24-eps workload processing.

For the purpose of this test, we change the coupling facility policy to use non-duplexed structures. (It is not necessary to un-duplex the structures if a third coupling facility were available. It was only because of testing environment constraints that we took this step.) We then move the structures from CF02 to CF01 using the MVS command SETXCF START,REBUILD,CFNAME=CF02,LOCATION=OTHER. At this point, CF maintenance can be done without affecting the running workload.

### **Expected behavior**

Processing continues.

### **Actual behavior**

After CF02 structures are moved to CF01, we display CF02 to ensure that it contains no structures in use anywhere in the sysplex. No disruption to workload processing was observed.

### **Recovery actions**

Move structures back to CF02 using the MVS command SETXCF START,REBUILD,CFNAME=CF01,LOCATION=OTHER. In our test environment, we then reactivate the policy with duplexed structures.





## **ACI Simulation Services for Enterprise Testing**

In this appendix, we summarize our use of the ACI Simulation Services for Enterprise Testing (ASSET) software.

## ASSET simulator

ACI Simulation Services for Enterprise Testing (ASSET) provides a message generation system that allows customers to graphically define their own message formats and manipulate, switch, transmit, and receive these messages along communications channels of various protocols. ASSET provides facilities to both emulate and test all elements of the underlying payment system infrastructure.

We referenced the *ASSET Installation and Configuration Guide* for the installation procedures. Our ASSET environment was configured as a distributed test environment. ASSET was installed on three desktop machines, each with a 2.66 Ghz Intel® dual core processor, 3 Gb RAM, and 149 Gb hard drive running Windows® XP. One of the desktops was configured as a license server that required configuring the ASSET Manager service. The other two machines were clients that used the server license key.

The following endpoints were simulated using ASSET: VISA switch, VISA issuer, and NCR/NDC ATM devices. We used ASSET in both our installation verification testing and during our high availability testing. The ASSET scripts were configured to use Sysplex Distributor for distribution of traffic between LPARs SC48 and SC67.

The test configurations used by the ASSET simulator are created in Microsoft Excel workbooks providing a familiar environment that is already present on the majority of corporate computers. Macros are provided to save the data in a format that is suitable for ASSET at the click of a single button. Figure A-1 on page 165 shows the main worksheet for the tests that we used on the VISA stress workload.

Microsoft Excel - VisaNet\_Test\_Data.xls

FileEditViewInsertFormatToolsDataWindowHelp

Type a question for help

Tahoma10

**B***I*U

<

Figure A-1 ASSET

ASSET provides the option to run tests in single shot mode or in stress test mode. Messages for stress workloads are pre-generated from the test configurations in the Excel® workbook, ensuring that card numbers and other data are valid for the system being tested.

To begin a test:

We highlight the type of test, in this case it is the VisaNet\_Stress workload, and then we click **Start**, as shown in Figure A-2 on page 166.

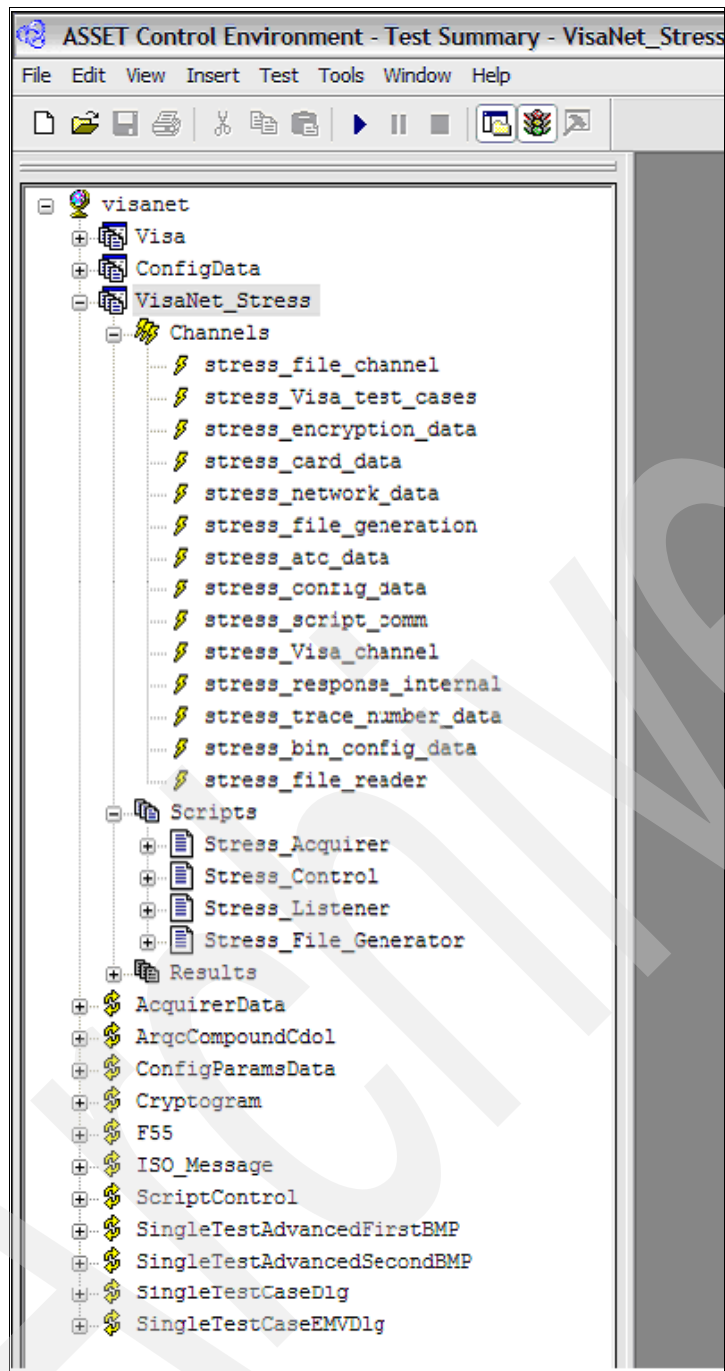


Figure A-2 VisaNet\_Stress

A pop-up window and dialog allows the user to specify whether to generate a new test data file. In this instance we will use an existing test data file, as shown in Figure A-3 on page 167. We specified 10 endpoints, which in this workload represents the number of VISA switches and an initial rate of 25 transactions per second. We specified 0 for the number of messages, which means ASSET will continue to execute the workload until the test is manually stopped.

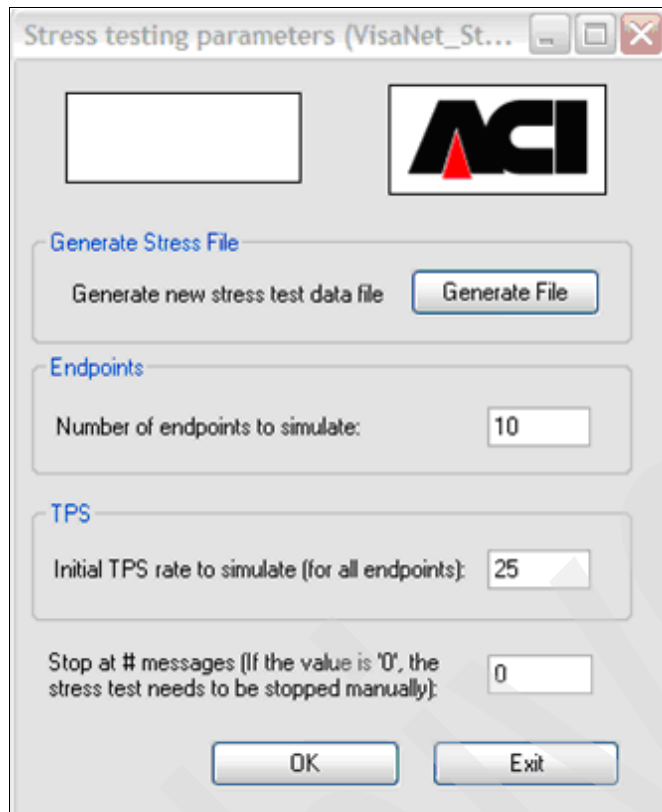


Figure A-3 Stress testing parameters

The Test Summary window, shown in Figure A-4 on page 168, provides important information, such as messages processed and the current messages per second.

Stress\_acquirer is the script that simulates the VISA switch that is sending outbound messages to BASE24-eps on the target system. In Figure A-4 on page 168, there are 10 active instances, which is what was requested on the start pop-up. See Figure A-3.

Stress\_listener is the corresponding listener script that processes the inbound messages from BASE24-eps.

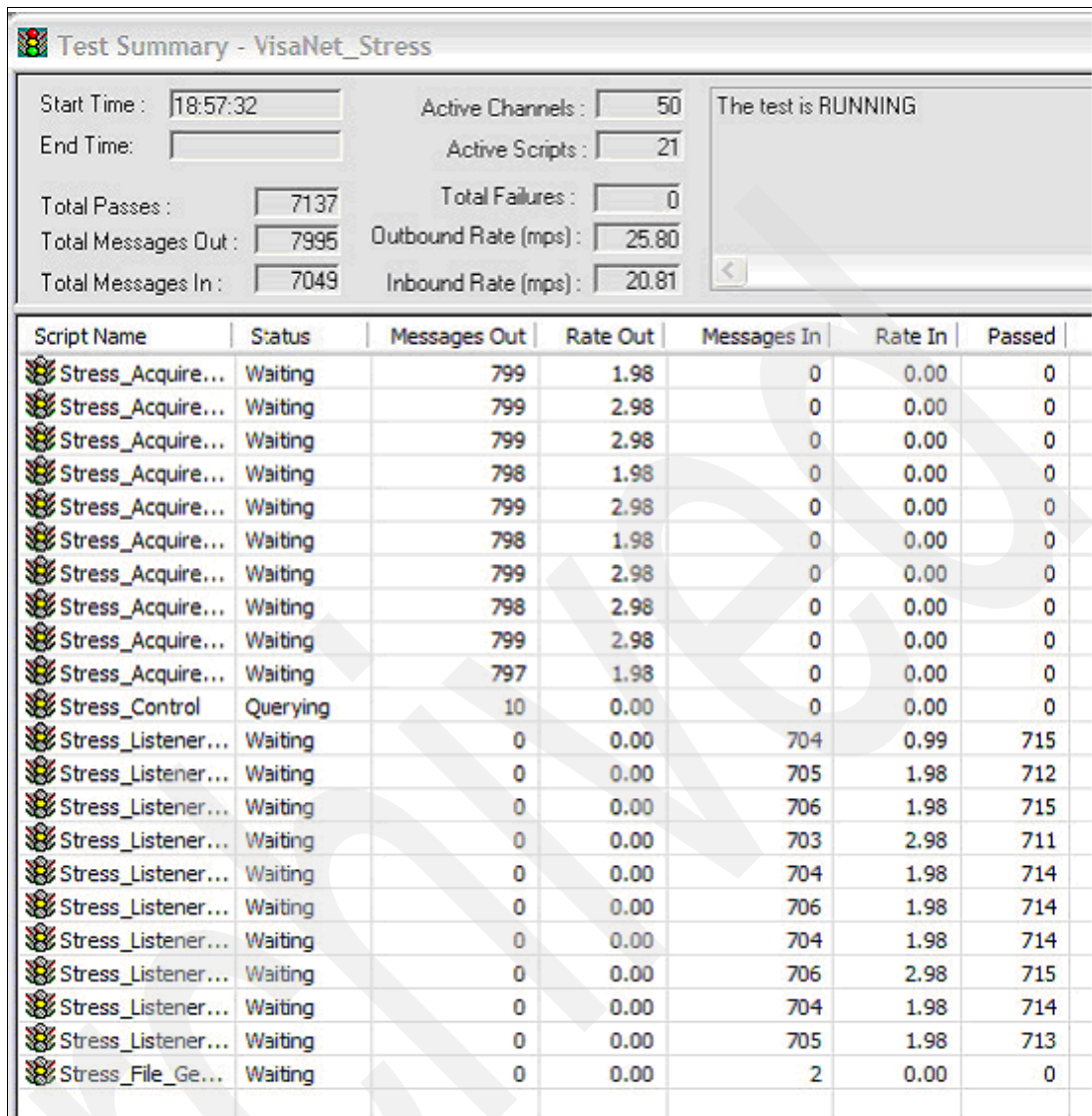


Figure A-4 Test summary window

The message contents can be verified using ASSET's built in message viewer. Figure A-5 on page 169 shows the message viewer displaying the fields of an outbound 0200 message.



Archived



# Glossary

**Application Programming Interface (API).** The interface by which an application program accesses operating system and other services. An API is defined at source code level and provides a level of abstraction between the application and the kernel (or other privileged utilities) to ensure the portability of the code.

**Automatic Restart Manager (ARM).** A z/OS recovery function that can automatically restart batch jobs and started tasks after they or the system on which they are running terminate unexpectedly.

**Asynchronous request.** A request that is issued without immediately awaiting a response. If a response is expected, the issuing application typically starts a timer and performs other work until a response is received or a timer expiration occurs.

**Coupling facility (CF).** A special LPAR that provides high-speed caching, list processing, and locking functions in Parallel Sysplex.

**CFRM policy.** The allocation rules for a coupling facility structure that are declared by a z/OS administrator.

**Coupling Facility Resource Management (CFRM).** A set of definitions for the use of a coupling facility. CFRM provides the services to manage coupling facility resources in a sysplex.

**Client (TCP/IP).** A TCP/IP client establishes a connection, typically using the connect() sockets API call. In the world of short-lived sockets, the client usually also speaks first, sending the initial application-level request, receiving a response, and disconnecting. In the world of long-lived sockets that are common in the world of financial online transactions, it is common for a TCP/IP client to be an application-level server. The BASE24-eps TCP/IP Client Communications Handler is a TCP/IP client. See *Server*, *Listener*.

**Central Processor (CP).** The part of the computer that contains the sequencing and processing facilities for instruction execution, initial program load, and other machine operations.

**Central Processor Complex (CPC).** In a z/OS or OS/390 environment, a physical collection of hardware that consists of main storage, one or more Central Processors, timers, and channels.

**Data Access layer (DAL).** A BASE24-eps System Interface Services component that provides the BASE24-eps application with a common API to access various databases and file systems, such as VSAM.

**Direct Access Storage Device (DASD).** A device that allows storage to be directly accessed, such as a disk drive.

**Dynamic Destination Map File (DDMF).** A routing configuration file used by the BASE24-eps System Interface Services Message Delivery Service to route asynchronous messages.

**Electronic Software Distribution (ESD).**

**Hierarchical File System (HFS).** A system for organizing files in a hierarchy, as in a UNIX system. z/OS UNIX System Services files are organized in an HFS.

**Hardware Security Module (HSM).** A secure Cryptographic facility for card holder PIN transaction, verification, and message authentication.

**Listener (TCP/IP).** A TCP/IP listener waits for a connection, typically using the bind(), listen(), and accept() sockets API calls. A TCP/IP server typically does not perform other productive work, handing the established socket off to another task to accept requests and send responses while the listener awaits more connections. The IBM-provided CSKL transaction is a TCP/IP listener, which hands the established socket off to another task, such as the BASE24-eps SIDC database server. See *Client*, *Server*.

**Logical partition (LPAR).** A subset of a single system that contains resources (processors, memory, and input/output devices). A logical partition operates as an independent system. If hardware requirements are met, multiple logical partitions can exist within a system.

**Not-on-us card.** A card that was not issued by the financial institution and that is usually routed to a public card network for authorization. See *on-us*.

**Offline authorization.** An authorization type wherein BASE24-eps is configured to authorize on-us transactions using its own database and scripts. See *on-us*, *online*, *online/offline*.

**Online Analytical Processing (OLAP).**

**Online Transaction Processing (OLTP).** A type of interactive application in which requests that a user submits are processed as soon as they are received. Results are returned to the requester in a relatively short period of time.

**Online authorization.** An authorization type wherein BASE24-eps is configured to route on-us authorization requests to a back end host authorization system. If the back end system is unavailable, requests might be configured to be routed to an alternate authorizer. See *on-us*, *offline*, *online/offline*.

**Online/offline authorization.** An authorization type wherein BASE24-eps is configured to route on-us authorization requests to a back-end host authorization system. If the back-end system is unavailable, requests may be configured to be routed to an alternate authorizer. If no alternate authorizer is configured, or if the alternate is unavailable, BASE24-eps may stand in and perform authorization using its own database and scripts. See *on-us*, *offline*, *online/offline*.

**On-us card.** A card that is issued by the financial institution and that is usually to be authorized locally. See *not on-us*.

**System Authorization Facility (SAF).** A z/OS facility through which programs communicate with an external security manager such as RACF.

**Static Destination Map File (SDMF).** A configuration file used by the BASE24-eps System Interface Services Message Delivery Service to define attributes of various endpoints.

**Server (TCP/IP).** A TCP/IP server waits for a connection, typically using the `bind()`, `listen()` and `accept()` sockets API calls. A TCP/IP server typically also performs productive work, accepting requests and sending responses. In the world of short-lived sockets, the client usually also speaks first. The server receives a request, sends a response, and awaits either a new request or a client disconnect. In the world of long-lived sockets common in the world of financial online transactions, the TCP/IP server may be an application-level client, and send the initial request. The BASE24-eps TCP/IP Server Communications Handler is a TCP/IP server. See *Client*, *Listener*.

**System Interface Services (SIS).** The group of BASE24-eps components that provide a platform-dependent implementation of a platform-independent system services API, allowing the rest of the BASE24-eps code to run without modification on various hardware and middleware configurations. SIS includes DAL, Time, Timer, Message Delivery, and similar services.

**System Initialization Table (SIT).** A table containing parameters that CICS uses upon startup.

**Synchronous Destination Map File (SYDMF).** A routing configuration file that the BASE24-eps System Interface Services Message Delivery Service uses to route synchronous messages.

**Synchronous request.** A request that is issued followed by an immediate wait for response. Generally acceptable only for lower-latency, highly-reliable servers.

**SYStem comPLEX (sysplex).** A set of z/OS systems communicating and cooperating with each other through certain multisystem hardware components and software services to process customer workloads.

**Virtual Storage Access Method (VSAM).** An access method for direct or sequential processing of fixed-length and varying-length records on direct access devices. The records in a VSAM data set or file can be organized in logical sequence by a key field (key sequence), in the physical sequence in which they are written on the data set or file (entry sequence), or by relative record number.

**VSAM Record-Level Sharing (VSAM RLS).** An extension to the Virtual Storage Access Method (VSAM) that provides direct record-level sharing of VSAM data sets from multiple address spaces across multiple systems. Record-level sharing uses the z/OS coupling facility (CF) to provide cross-system locking, local buffer invalidation, and cross-system data caching.

**Virtual Telecommunications Access Method (VTAM).** An IBM licensed program that controls communication and the flow of data in an SNA network. It provides single-domain, multiple-domain, and interconnected network capability.

**Workload Manager (WLM).** A z/OS construct that provides services to manage workload distribution, balance workload, and distribute resources.

# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

## IBM Redbooks

For information about ordering these publications, see “How to get Redbooks” on page 173. Note that some of the documents referenced here might be available in softcopy only:

- ▶ *A Guide to Using ACI Worldwide's BASE24-es on z/OS*, SG24-7268-00
- ▶ *ACI Worldwide's BASE24-eps V6.2: A Supplement to SG24-7268*, REDP-4338-00
- ▶ *Migration of Payments System to ACI BASE24-eps*, REDP-4337-00

## ACI Worldwide, Inc. publications

These publications are also relevant as further information sources:

- ▶ ACI BASE24-eps (v08.2) Hardware and Software Requirements
- ▶ ACI BASE24-eps (v08.2) z/OS Install Guide
- ▶ *ACI BASE24-eps z/OS Pre-Install Checklist and Worksheet*
- ▶ ACI BASE24-eps Transaction Security manual
- ▶ ACI System Object Security Standard BASE24-eps 8.2 Requirements

## Online resources

The ACI Worldwide, Inc. Web site is also relevant as an information source:

<http://www.aciworldwide.com/>

## How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks, at this Web site:

[ibm.com/redbooks](http://ibm.com/redbooks)

## Help from IBM

IBM Support and downloads

[ibm.com/support](http://ibm.com/support)

IBM Global Services

[ibm.com/services](http://ibm.com/services)

Archived

# Index

## A

- ACI Application Management
  - used to start BASE24-eps address spaces 37
- ACI Application Management Server 40
  - illustration 40
- ACI Application Management service 30
- ACI Application Management user interface 40
- ACI Communications Handlers 35
- ACI Desktop 15
  - User Interface 12
- ACI Enterprise Payments Solutions 9, 18
- ACI Event Adapter 41
- ACI Event Service 31
- ACI Metadata Utility 34
- ACI Simulation Services for Enterprise Testing (ASSET) 67, 164
- acquirer types 16, 18
  - device-acquired transactions 18
  - Host-acquired transactions 18
  - switch-acquired transactions 18
- acquiring device 5
- address space 32
- ADMF Table 34
- Application Management Agent 30
  - data available for external enterprise management platforms 30
- application notification 30
- APPN Topology Control File 37
- Asymmetric algorithms 103
- Asynchronous Destination Map File (ADMF) 34
- asynchronous message routing 42
- ATM or POS processing 3
- ATM processing 4
- authorization types 16
  - offline transactions 16
  - online transactions 16
  - online/offline authorization 17
  - switched transactions 16
- Automated Teller Machine (ATM) 3
  - availability requirement 2
  - reliability requirement 2
- Automated teller machine (ATM) 1
  - growth 2
- Automatic Restart Manager (ARM) 23

## B

- BASE24-eps
  - architecture
    - core components 30
      - application management 30
      - communications handlers 30
      - diagnostic events 31
      - Integrated Server 32
      - portable application architecture 31

- Platform independent Business services (BU) components 31
- Platform-independent Foundation (FN) components 31
- System Interface Services 31
- stateless application server processing 30
- user interface client 30
- user interface server 30
- core platform services
  - data communications 30
  - non-functional requirements 30
  - transaction management 29
- logical architecture diagram 28
- System z
  - application management 39
    - ACI Application Management Server 41
  - communications handling options 35
  - components 39
    - ACI desktop 39
    - agent 39
  - data communication 35
    - configuration files used by the ICE-XS communications handler 35
  - database 34
    - DB2 Data Sharing 34
  - files used by System Interface Services (SIS) Data Abstraction Layer (DAL 34
  - diagnostic event handling 41
    - appenders 41
    - illustration 41
    - modifiable template 41
    - Tivoli Enterprise Console adapter 41
    - use of persistent IBM WebSphere MQ queue 41
  - managed resources 39
    - data communication facilities 39
    - possible deployment illustration 39
    - processes 39
    - queue managers and queues 39
  - messaging middleware 33
    - queues
      - command queue 33
      - Definitions of files used by System Interface Services (SIS) for message routing 34
      - service queue 33
  - performance test 44
    - CPU cost per transaction 45
    - Total Cost of Ownership (TCO) 45
  - portable application architecture
    - message routing 42
    - symbolic routing configuration 42
  - stateless application server processing 37
  - transaction management 34

- DB2 transaction handling 34
- architecture overview 13
- consumer authentication and authorization 10
- Disaster Recovery 49
- flexibility in transaction routing 10
- goals 48
  - Architected High Availability 48
  - Data Integrity 48
  - Financial Integrity 48
  - Low Cost 48
- graphical user interface (UI) 12
- High availability 48
- interpreted scripting language for authorization logic 11
- introduction 9
- multiple platform configurations 13
- multiple platform support 15
- operational advantages 13
- routing structure for transactions 10
- scripting facility 14
- support for consumer e-payment transactions 10
- support for emerging delivery channels 11
- supports reporting and extracting of financial transaction data 12
- transaction processing customization 10
- transaction security support 12
- BASE24-eps ACI Desktop Client 30
- BASE24-eps ACI Desktop services 30
- BASE24-eps address space 31
  - boss thread 38
  - internal structure 31
  - logical worker thread 37
  - multiple pthreads 37
  - resource contention 37
  - statically-linked objects 38
  - threading model 37
  - threading model diagram 38
- BASE24-eps address spaces 37
- BASE24-eps card transactions 16
- BASE24-eps installation 64
  - batch jobs to bind DB2 plans and packages 77
  - CSUTIL job to define the MQ objects 78
  - DBRM for the BASE24-eps application 77
  - defining ICE-XS processes 95
  - electronic software distribution 64
  - ES\_Install 68
  - ES\_Install FTP process 71
  - ES\_Install program 68
  - essetup script 75
    - pause for execution of z/OS jobs 77
    - restarting 81
    - starts ACIJMX process 82
  - ICE-XS cold start 97
  - ICE-XS installation 72
  - ICE-XS license file 73
  - ICE-XS parameter file 73
  - IDBACK parameter tuning 66
  - JMX rebuild 97
  - OLTP warm booted 98
  - OMVS MAXCPU TIME tuning 66

- OMVS MAXMMAPAREA tuning 66
- Other software requirements 66
- Pre-Install Checklist and Worksheet 67
- reference documents 64
- starting base processes 98
- starting ICE-XS 75
- starting processes using the client UI 98
- stopping BASE24-eps processes 100
- test environment details 64
- test environment graphical depiction 64
- test installation software environment 65
- use of c89 C++ compiler required 66
- verify the installation 87
- view the MQ objects 84
- BASE24-eps Integrated Server
  - context free 38
- BASE24-eps manager processes 135
- BASE24-eps performance benchmark 44
- BASE24-eps process 37
- BASE24-eps software components 13
  - Adaptors 13
  - Business component 13
  - Foundation components 13
  - Platform-specific components 13
- BASE24-eps Transaction Security component 109

## C

- Capacity on Demand (COD) 24
  - functions 24
- Capacity Upgrade on Demand (CUOD) 8
- CEX2A 109
- CEX2C 108
- channel-to-channel (CTC) 25
- CICS-MQ Bridging facility 42
- clear key 102, 106
- collision 62
- Common shared networks 3
- config.csv 34
- continuous availability 6, 21–22, 25–26
  - popular definition 21
- continuous availability
  - building block 21
- continuous operations 22
- continuous reliable operation (CRO) 22
  - building blocks 22
- control vector 113
- CONTROL\_FILE 35
- cooperative networks 3
- Coupling Facility (CF) 7–8, 24
- CP Assist for Cryptographic Function (CPACF) 106, 108
- CPU cost 8
  - suitably reliable and low-latency local CICS authorization systems 8
- Crypto Express2 8, 25, 106, 108
- Crypto Express2 (CEX2) 107
- Crypto Express2-1P 108
- Cryptographic Coprocessor Facility 104
- cryptographic function 12, 102
- Cryptographic Key Data Set (CKDS) 110

- BASE24-eps connectivity to Crypto Express2 109
- clear key vs secure key 102
- Crypto Express2 as an accelerator 109
- cryptographic hardware connectivity 103
- dynamic keys 110
- how BASE24-eps derives key labels 111
- key block formats 110
- key entry 110
- key label 110
  - structure 112
- key token 110
- Key Token data source 110
- master key 110
- Transaction Security component functions supported with Crypto Express2 110
- Transaction Security component's use of key labels and tokens 110
- Trusted Key Entry Workstation to manage keys 109
- z10 features 105
- z9 features 108
- customer relationship management (CRM) 11

## D

- data access layer (DAL) 14, 29
- Data Encryption Standard (DES) 102
- data integrity 3
- Data Link Switching (DLSw) 36
- data sharing 26
- DB2 data sharing 128
- Dependent LUs 36
- Disaster Recovery
  - DB2 Queue Replication Active/Standby 58
    - availability characteristics 58
    - example 58
  - DB2 Queue Replication Dual Active
    - considerations 62
    - functionality 61
  - Global Mirror 57
    - availability characteristics 57
    - example 57
    - functionality 58
  - Metro Mirror
    - availability characteristics 56
    - example 56
    - functionality 56
  - Stretch Parallel Sysplex 59
    - availability characteristics 60
    - example 59
    - functionality 60
- Disaster Recovery (DR) 20, 49, 60
- DLSw 37
- dynamic keys 110

## E

- electronic funds transfer (EFT) 1
- encryption for avoiding data alteration 25
- encryption to avoid eavesdropping 25
- End Node (EN) 36
- Enterprise Services architecture of BASE24-eps 13

- ES\_Install 67
- esinfo 86
- essetup script 76
- esstatus 82, 96
- Extended Remote Copy (XRC) 57
- External CICS Interface (EXCI) 8, 42
- external security modules (HSMs) 8

## F

- fault tolerance 3
- finance industry
  - use of System z 21
- financial institution 3
- financial transaction 3

## G

- Geographically Dispersed Parallel Sysplex (GDPS) 26, 56–59

## H

- High availability 22
  - dual LPAR 51
    - Availability characteristics 51
    - example 51
  - Dual LPAR with Hyperswap 52
    - example 52
    - functionality 53
  - Single Site Dual Server with Hyperswap 54
    - example 54
    - functionality 55
  - Single-LPAR
    - characteristics 50
    - example 49
    - functionality 50
- High availability (HA) 25–26
- High availability (HA) test setup
  - Application Management Remote Agent configuration 137
  - configuration used for tests 129
  - DB2 subsystem 130
  - monitoring 150
  - MQ queue managers 130
  - multi-LPAR configuration 133
  - shared UNIX System Services file system 133
  - starting BASE24-eps processes 147
  - starting ICE-XS process 147
  - Sysplex Distributor 132
- High availability (HA) tests 152
  - BASE24-eps 156
    - Application Management server failure 158
    - ICE-XS maintenance 157
    - ICE-XS server failure 157
    - Integrated Server failure 156
    - maintenance 156
  - Data Environment
    - Data table unavailable at BASE24-eps initialization 154
    - DB2 catastrophic failure 154

- DB2 maintenance 155
- Hardware and z/OS system software 158
  - Coupling Facility failure 159
  - Coupling Facility maintenance 162
  - ICSF failure 160
  - IP virtualization failure 160
  - LPAR failure 159
  - z/OS maintenance 161
- IBM WebSphere MQ 152
  - adding a new MQ instance in a new LPAR 153
  - maintenance 153
- High Performance Routing (HPR) 36–37
- host-acquired transactions 16
- HTTP Server and Client 36
- Hyperswap 52

## I

- IBM System z
  - additional information 21
  - mainframe 20
- IBM WebSphere MQ 33, 43, 130
  - CICS Gateway 8
  - queue-sharing group instance 153
  - Shared Queue 33
- ICE-XS 35–36, 43–44
  - brief description of each component 36
- ICE-XS Communications Handler 35
- ICE-XS process 44
- icexs.param 35
- ICSF Application Programming Interfaces (APIs) 103
- IMSConnect 8
- Independent LUs 36
- InfoSphere Replication Server 58
- Integrated Cryptographic Service Facility (ICSF) 8, 25, 103
- Integrated Facility for Linux (IFL) 8
- Integrated Server 32
  - context-free 32
- Integrated Server (IS) 43, 49
  - parser components 18
- Intelligent Resource Director (IRD) 23
- Internet Communications for the Enterprise - Cross System (ICE-XS) 72
- issuing institution 5

## K

- key exchange 110
- Key Generation Utility Program (KGUP) 113
- key label 110
  - external format 111
- key token 110, 113
- Key Token data source 110

## L

- LICENCE\_FILE 35
- license file 35
- logical threads per BASE24-eps address space 37

## M

- master key 102, 112
- mdb.csv 34
- Message Authentication Code (MAC) 8, 25
- Message Delivery Service (MDS) 33
- Messaging Middleware 33
  - BASE24-eps process associated queue definitions 33
  - command queue 33
    - queue name 33
  - IBM WebSphere MQ queue definitions 33
  - memory-based queuing 33
  - MQSC file 33
  - persistent queuing 33
  - service queue 33
  - transactional queues 33
- Metro Mirror 55
- Million Instructions Per Second (MIPS) 45, 56
- mission-critical applications 20
- MQ queue sharing group 128
- Multi-Node Persistent Sessions 26

## N

- National Bridge Transactions 4
- NETSTAT display 95
- Network Node (NN) 36
- Network On Us 4
- NOF-XS 35–36
- NOF-XS Command files 35

## O

- object-oriented programming languages 13
- Online Transaction Processing tables 98
- on-us 4, 17
  - basic authorization method 17
  - Negative Authorization 17
  - Positive Authorization 17
  - Positive Balance 17

## P

- Parallel Sysplex 7, 20, 53, 128
  - architecture 25
  - cluster 6
  - clustering 25
  - clustering architecture 26
  - multiple LPARs 7
  - no single point of failure 6
  - parallel processing environment 6
  - scalability 8
  - workload balancing 8
- Parallel Sysplex cluster 7
- PCI (Peripheral Interconnect) 103
- Peer-to-Peer Remote Copy (PPRC) 53
- Personal Identification Number (PIN) 8, 12, 25
- Point-of-Sale (POS) 2, 3
  - availability requirement 2
  - growth 2
- POS device 4



- POS processing 4
- PU 2.0 36
- public key 112
- pull-based model 39

## Q

- Queue Replication 58

## R

- RAID technologies 25
- Reciprocal Transaction 4
- recovery action 152
- Recovery Point Objective (RPO) 49
- Recovery Time Objective (RTO) 49, 57, 59
- Redbooks Web site 173
  - Contact us xi
- regional network 4, 5
- reliability 2
  - requirements 2
- Resource Access Control Facility (RACF) 25
- Resource Control File (RCF) 37
- Resource Recovery Services (RRS) 23
- runevtparser 90

## S

- scripting engine 11
- scripting facility 14
  - uses 14
- secure hardware 102
- secure key 102
- Secure Sockets Layer (SSL) 25
- Security Control File 37
- SECURITY\_CONTROL\_FILE 35
- Self-Timed Interface (STI) 103
- Shared Queues
  - context data 39
- Single site
  - dual server Parallel Sysplex 55
  - Parallel Sysplex 51, 53
- SIS Message Delivery Service 33
- SOAP Server 36
- switching systems 5
- symbolic name routing 42
- symmetric algorithms 102
- Synchronous Destination Map File (SYDMF) 34
- synchronous message routing 42
- synchronous routing
  - CICS-based authorization system 42
- Sysplex Distributor 26, 128, 132
- System availability 2
- System Interface Services (SIS) 32
- System reliability 3
- System z
  - Application Assist Processor 24
  - architecture 20
  - Autonomic computing technologies 7
  - availability 6
  - backup and recovery 23

- BASE24-eps architecture 32
- BASE24-eps architecture implementation 32
- BASE24-eps Core Platform Services 32
- BASE24-eps performance benchmark 44
- centralized data storage 22
- compatibility 20
- Continuous availability 7
- Crypto Express2 card 12
- encryption 25
- features 22
- finance industry 21
- flexibility 7
- hardware 49
  - additional information 22
- hardware environment 7
- Integrated Cryptographic Service Facility 8
- Integrated Facility for Linux (IFLs) 24
- Integrated Information Processor 24
- multiple workloads 23
- Parallel Sysplex 6
- popularity 20
- recovery 23
- reliability, availability, and serviceability (RAS) 22
- scalability 23
- scalability 8
- securing BASE24-eps 62
- security 7, 24
- strategy 22
- value 6, 20, 22
- workload management 23
- System z Application Assist Processor (zAAP) 8
- Systemx z
  - performance management 23

## T

- TCP/IP connection 37
  - SNA networks 37
- transaction
  - acquisition 6
  - authentication 6
  - authorization 6
  - categories 3
- Transactions Per Second (TPS) 45
- Trusted Key Entry (TKE) 109

## U

- Unified Modeling Language (UML) 15
- User Defined Extensions (UDX) 104
- User Interface
  - Client 30, 39
  - Server 30, 39

## V

- Virtual IP Addressing, 26
- VTAM Generic Resource 26

## W

- Web Services Security (WSS) 36

Workload Manager 26

## **X**

X.25 over TCP/IP 36









# A Guide to the ACI Worldwide BASE24-eps on z/OS



## **BASE24-eps installation**

## **High-availability system designs**

## **High-availability tests**

In this IBM Redbooks publication, we explain how to use the ACI™ BASE24-eps™ product on z/OS. BASE24-eps is a payment engine that the financial payments industry uses. The combination of BASE24-eps and System z is a competitive and attractive end-to-end retail payments solution for the finance sector.

Failure in a financial payments environment is a high-visibility customer service issue, and outages at any level have debilitating effects on customer loyalty. The entire payments cycle must be conducted in near real-time. In such an environment, high availability is a mission-critical requirement. In this guide, we demonstrate how you can achieve a high availability configuration for BASE24-eps on z/OS.

We begin by outlining the requirements of a payments system, and then we introduce the structure and functionality that the BASE24-eps product offers. We also describe the strengths and abilities of System z and z/OS and explain the technical and physical architecture of BASE24-eps on z/OS.

We guide you in designing a system layout and in installing BASE24-eps. Finally, we detail the numerous failure scenarios that we tested to verify the robustness of the solution.

BASE24-eps is an evolving product. The information that we provide in this book is specific to BASE24-eps release 08.2 and is subject to change in subsequent releases of the product.

## **INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION**

## **BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:**  
**[ibm.com/redbooks](http://ibm.com/redbooks)**